Hacking Social Network Data Mining

Yasmeen Alufaisan, Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham
University of Texas at Dallas
{yxa130630,yan.zhou2,muratk,bxt043000}@utdallas.edu

***Abstract*—Over the years social network data has been mined to predict individuals' traits such as intelligence and sexual orientation. While mining social network data can provide many beneficial services to the user such as personalized experiences, it can also harm the user when used in making critical decisions such as employment. In this work, we investigate the reliability of applying data mining techniques on social network data to predict various individual traits. In spite of the preliminary success of such data mining applications, in this paper, we demonstrate the vulnerabilities of existing state of the art social network data mining techniques when they are facing malicious attacks. Our results indicate that making critical decisions, such as employment or credit approval, based solely on social network data mining results is still premature at this stage. Specifically, we explore *Facebook likes* data for predicting the traits of a Facebook user, including their political views and sexual orientation. We perform several types of malicious attacks on the predictive models to measure and understand their potential vulnerabilities. We find that existing predictive models built on social network data can be easily manipulated and suggest some countermeasures to prevent some of the proposed attacks.**

I. Introduction

Social networks provide a platform for collecting large amounts of user information, such as daily routines, pictures, locations, feelings, and many more. With existing data mining techniques, personal preference such as religious beliefs or political views can be easily predicted or inferred using these data collections. The substantial success reported in the research community has motivated many companies to make critical decisions, such as hiring future employees, based on their social network profiles. According to a survey conducted by CareerBuilder in 2014, 51% of employers who used social networks to find information about potential candidates discovered contents that caused them not to hire the candidate [1].

Despite their great success in research community, these data mining techniques may not be as reliable as has been portrayed. A simple attack could easily foil the trained data mining models and lead to false decisions. Consider *Facebook likes* as an example, when they are used in the prediction process, anyone can take a desired Facebook profile and customize it to be their own to imitate the desired traits (e.g., changing profile *likes* to look smarter). Both Facebook profiles would be equally appealing to the trained predictive models. We show that this kind of evasion attack can cause 100% error when accessing only 1% of the training data. Besides attacks at application/test time, data mining models built on social network data may also be influenced by fake profiles slipped into training data (i.e., poisoning attacks). We demonstrate in our experimental results that the existence of these fake profiles can significantly reduce the accuracy of the data mining models. We also consider different variations of poisoning attacks and show how they can weaken the prediction when adding only 5% noise to the training data.

In this paper, we comprehensively study the robustness of data mining models applied to social network data against different types of malicious attacks. We investigate the feasibility of predicting different classes of users using their features. We look closely at these user classes that can be successfully predicted using the features and investigate how easy it is to manipulate these features to fool the data mining models. We perform several different types of attacks to measure the robustness of existing state of the art data mining models. Our work demonstrates that mining social network data using statistical data mining models is vulnerable to these attacks as the predictive accuracy decreases with simple manipulations on the training or test data.

We study the impact of two different categories of attacks The first category is **Evasion Attacks** that is represented by a single user who edit their profile information to prevent any prediction that can expose their privacy. The impact of this attack is local to the individual user and it can result in less targeted advertising in exchange for having more privacy. The second category is **Poisoning Attacks** where the adversary manipulates data belonging to group of real or fake users to reduce the global classification accuracy of a prediction model. Poisoning attack can be used by adversaries to impact the data mining performance. For example, a terrorist group could launch such a coordinated attack to fool "is it a terrorist related profile" prediction model. This is a more serious attack with global effect on all users tested with the predictive model.

We examine different metrics to evaluate the importance of the features and match the best evaluation metric for each attack. We demonstrate our results using *Facebook likes* on two classification tasks: political view (democrat vs. republican) and sexual orientation (straight vs. gay). For each task, we consider one class to be the target class and the other to be the opposite class. For example, for the political view we consider "democrat" to be our target class and "republican" to be the opposite class. We choose the target and the opposite classes at random with no specific reasons for favoring one over the other. Finally, we propose various countermeasures to prevent some of the proposed attacks. We report the success of such countermeasures in reducing the severity of the attacks under various settings.

978-1-5090-6727-5/17/$31.00 ©2017 IEEE

II. Adversarial Attacks

In this section, we discuss several different types of adversarial attacks we consider in our study, as well as the metrics we use for feature selection. We model the input space as n-dimensional discrete value space $X = (X_1, ..., X_n)$ where X_i represents the i-th feature. For each feature with k possible values, we convert it to k binary features. Each instance x_i=($x_{i1}, .., x_{in}$), where $x_{ij} \in X_j$, is mapped to one class y. We assume binary class domain $y \in \{c_t, c_o\}$ where c_t represents the target class and c_o the opposite class.

We explore vulnerabilities in data mining models built on social networks data. We consider four lists of features: X_g^t, X_b^t, X_g^o and X_b^o to represent the good features for c_t, bad features for c_t, good features for c_o, and bad features for c_o respectively. These lists are discussed in details in Section II-C.

A. Evasion Attacks

An adversary here can be represented by a social network user who is willing to trade some potential benefits (e.g., viewing ads relevant to their real interests) for cloaking his personality traits for privacy reasons.

1) Good Feature\Bad Feature Attacks: In this attack we examine the effect of adding features from the list of good features for the target class users X_g^t to users in the opposite class c_o. In bad feature attack, we delete features in the list of bad features for the opposite class users X_b^t from users in in the opposite class c_o. Introducing false interest such as liking a Facebook page to receive a discount on some products could be seen as an example of adding features. On the other hand, deleting a feature represents hiding an interest. For example, a Twitter user might unfollow some accounts to keep their religion information private. In these attacks we assume the adversary has background knowledge about which features are considered good and which ones are considered bad. An adversary can easily have access to this information because of the fact that most social network features are publicly available and can be seen by anyone. If the adversary wishes to look like a certain class, all she has to do is to observe which features are associated with known users in this particular class. For example, if a Facebook user wants to look younger, they can simply search for known young users and like the same pages they like. We assume that the adversary has a limited budget on the number of features that she can manipulate.

2) Mimicry Attack: In mimicry attack, the adversarial goal is to make users in c_o look like users in c_t to manipulate their prediction. It is different than simply adding and deleting features from all the users as we discussed in the previous attacks. In this attack we assume access to some of the training data. Since it is difficult for the adversary in practice to have access to all the training data, we assume access to only 1% random sample of the training data. This attack works as follows: we start by randomly picking k users that are in the opposite class c_o from the test data. For each one of the k users, we search for the most similar user in the target class c_t from the training data sample. The most similar user is the one with shortest distance measured using L_1 norm. By using the most similar user for each user, we minimize the number of features manipulated. After that, we add\delete features to minimize the distance between these users. The distance is minimized to 75%, 50%, 25%, and 0% (that is, they become identical). We start by adding the best features until there are no more features to add and then we delete the worst features until we reach the desired distance.

B. Poisoning Attacks

In this section we present the different poisoning attacks we study in this paper.

1) Class Altering Attack: In this attack, we pick users from the training data that are in c_o and flip their labels. We assume the adversary has access to small percentage of the randomly chosen training data. The goal of this attack is to examine the ability of the data mining model in dealing with contradicting profiles. Contradicting profiles can be created due to users being dishonest about their true labels. For example, when a group of Facebook users are dishonest about their true age due to Facebook restriction on age [2], would the data mining model be able to detect such behavior and limit its impact?

2) Feature Altering Attack: In this attack, we poison the training data by altering the features of randomly chosen users. We alter users in c_o by adding good features for the target class users and deleting bad features for the opposite class users. By carrying this process, an adversary's goal would be to fool the prediction models into misclassifying users in c_o as c_t. We add good features chosen at random to the selected poisoned users to avoid detection by certain mechanisms employed to detect spam profiles in social networks. Consider Facebook as our social network and the liked pages as our features. Facebook can detect a group of users liking the same pages at about the same time using a mechanism named Copycatch [3]. The group of such users are considered suspicious and are penalized by restricting their *likes*. Therefore, adding features randomly is important to avoid detection by such mechanisms. When deleting bad features, we select features from X_b^t based on their ranks since removing features does not create spam related concerns (i.e., deleting features does not add fake information). In these attacks, we assume the adversary have access to a limited number of features.

3) Fake Users Addition Attack: Social networks contain a large number of fake users. In 2013, Facebook revealed in its annual report that about 5.5 to 11.2 percent of its monthly active users are fake [4]. Fake accounts can be created for different reasons such as spamming or marketing. A normal user might also be motivated to create a fake account. For example, since Facebook requires a user to be at least 13 years old in order to create Facebook account, a user who is under the age of 13 would violate Facebook policy by creating a profile with an older age. Having these fake users in the training data may affect the data mining process [2]. Therefore, in this attack, we shed some light on the influence of having fake accounts in the training data. When we poison the training data by injecting fake users, we create a fake account with class c_o then assign to it the top features in X_g^t.

TABLE I: Feature Selection Metrics

Metric	Formula
Score	$S_g^y(X_i) = \frac{N_{y\prime}N_y(X_i)}{N_{y\prime}N_y(X_i)+N_{y\prime}(X_i)N_y}$, where: N_y:number of users in class y, $N_{y\prime}$:number of users in class $y\prime$, $N_y(X_i)$:number of users in class y that have feature X_i, $N_{y\prime}(X_i)$:number of users in class $y\prime$ that have feature X_i
MostCommon	$Supp^y(X_i) = \frac{\sum_{k=1}^{n} I(x_k)}{n}$, where: n: number of users, $I(x_k) = \begin{cases} 1, & \text{if } x_{ki} = 1 \text{ and } x_k \text{ in class } y \\ 0, & \text{otherwise} \end{cases}$
Score_Support	$SS_g^y(X_i) = S_g^y(X_i) * 0.5 + Supp^y(X_i) * 0.5$

Each fake account is different in its vector of features in order to represent real world fake accounts. We start by choosing a random number of features that this account will contain. The range of this number depends on the average number of features for the real users. After that we randomly choose the features for this account from X_g^t. Furthermore, the fake injected users replace some of the real users in c_o to allow us to have the same number of users in the poisoned training data as in the original training data.

C. Feature Selection Metrics

Not all the features are equally powerful when it comes to predicting a specific user trait. Some features are strong indicators of age while others are better indicators of gender. We propose three metrics to evaluate the features and measure their strength in predicting the target trait.The formulas for the metrics to compute X_g are presented in Table I and X_b can be obtained similarly.

1) Score: The Score estimates how good a feature is in predicting a class y. A feature will have high goodness score if it occurs the most among users who are in y in contrast to users in y'. The score is inspired by token spam score computed for each word in an email to detect spams [5].

2) MostCommon: This metric simply considers one class at a time. It sorts the features based on the number of people who have those features and are in the same class y.

3) Score_Support: This metric is the result of combining the Score and the MostCommon metrics. We give both these metrics equal weights and then sum their weighted results. The reason we want to combine these two metrics is that Score metric by itself can score rare features (e.g., a misspelled word that is used rarely in a legitimate tweet) that are only seen in a few instances. At the same time, such rare features may have limited importance after feature selection/dimension reduction employed in practice. By adding the support to the Score metric, we overcome this weakness of the Score metric.

We use all the metrics for good feature\bad feature attacks. For the remaining evasion and poisoning attacks that require feature evaluation, we only consider the Score_Support metric for the features that we add unless indicated otherwise.

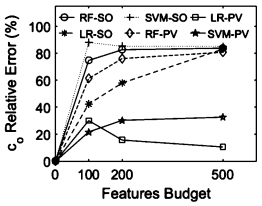
III. Case Study: Facebook Likes

In our work we use data collected from Facebook users by myPersonality project [6]. The databases used in our experiments are: anonymous *Facebook likes*, demographic data, religion and political views. The average number of *likes* per user in this dataset is 170. We consider a sample of the dataset that consists of 57,732 users from the US and a total of 55,041 *likes*. We remove from the sample any user with less than two *likes* and any *like* associated with less than 20 users. The resulting sample has 54,554 users and 54,986 *likes*. We follow *these preprocessing steps to replicate the methodology* in [6]. We also eliminate from the sample users who did not report their political view or gender information.

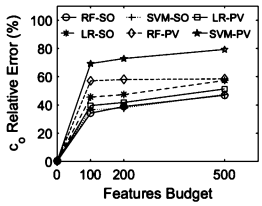
We use *Facebook likes* as our features in predicting political view and sexual orientation. For political view, we only consider users who indicated their political views as democrat or republican. For sexual orientation, we examine the demographic data and choose male users. A straight user is the one who chose "interested in females" and a gay user is the one who chose "interested in males". For political view, our sample contains 6,480 users and 53,845 *likes*. There are 57% of users identified as democrat and 43% of users as republican. For sexual orientation, the sample contains 7,493 users with this information available. However, only 5% of these users are indicated as gay. Therefore, we pick an equal number of straight users randomly and use them to avoid class imbalance. The number of users with sexual orientation information reduced to 868 with 50%-50% class distribution and a total of 30,144 *likes* associated with them.

We represent the users and their *likes* with a sparse matrix that has the user ID and 0/1 value for each *like* where 1 indicates that the user liked the page and 0 otherwise. Due to the large number of *likes*, we reduced the dimension of the sparse *user/like* matrix to 100 using Singular Value Decomposition (SVD) [7]. SVD is a factorization of a real matrix. The matrix M factorization would be of the form: $M = U \cdot S \cdot V^T$. SVD is used for dimensionality reduction by ignoring the smallest singular values. When choosing k dimensions, it keeps the k first columns of U and V and the top k singular values of S. The new reduced form of the matrix M is: $M_k = M \cdot V_k = U_k \cdot S_k$. The steps that we follow when performing SVD are: 1) Divide the data into 10 folds. For each fold we use $\frac{9}{10}$ of the data as our training data and the remaining $\frac{1}{10}$ as our test data. We refer to the original training data as D and the original test data as T and the training and test data with reduced dimensionality as D_k and T_k respectively; 2) Apply k-SVD on the training data and output: $U_k S_k V_k^T$ then reduce the dimensionality of the training data by performing the following step: $D_k = D \cdot V_k$; 3) Use V_k to project the test data onto the same reduced space as the training data: $T_k = T \cdot V_k$; 4) Build the classifier using D_k and test it using T_k.

For all the experiments we report the average results of 10-fold cross validation. If the attack is on the test data, we use V_k of the original training data on the attacked test data. If the attack is on the training data we use V_k of the poisoned training



(a) Good Feature Attack-Score_Support



(b) Bad Feature Attack-MostCommon

Fig. 1: Good\Bad Feature Attacks on Political View (PV) and Sexual Orientation (SO)

data on the original test data. We use Random Forest (RF) with 100 trees, Logistic Regression (LR), SVM with RBF kernel and logistic mode. All these models are available in Weka [8]. For all the baseline results, we report the accuracy represented by the Area Under the ROC Curve (AUC).

A. Baseline Results

In this section, we discuss our baseline results. The highest accuracy achieved for predicting political views is 0.87 using LR followed by an accuracy of 0.85 and 0.81 by RF and SVM respectively. With sexual orientation, RF, SVM and LR all have an accuracy of 0.9. These baseline experiments shows that *Facebook likes* are very powerful in predicting political view and sexual orientation. But since these experiments are based on users from the US, we wanted to test the bias of the predictive model when tested on users from other regions. The AUC of predicting sexual orientation when building a model on users from the US and test it on users from the Middle East becomes 0.557 with RF, 0.627 with LR, and 0.637 with SVM. The accuracy becomes comparable to random guess. We have different hypotheses for this significant drop in accuracy. The first one is that people in countries where being gay is not accepted will lie about their ground truth, therefore the accuracy will be low. Another reason could be cultural differences. It is well known that people in different cultures have different habits which can affect their *likes*.

B. Attacks Results

To measure the effectiveness of the proposed attacks, we use the relative error calculated as follows:

$$RelativeError = \frac{AttackError - BaselineError}{100 - BaselineError} \quad (1)$$

Intuitively, the relative error metric is measuring how well an attack can deceive the data mining model. The opposite class relative error (re_o) is our default metric to measure the attacks unless stated otherwise.

1) Evasion Attacks Results: In the good feature attack, the adversary adds the best *likes* with a budget of 100, 200, 500 good *likes*. Similarly, in bad feature attack the adversary deletes the worst *likes* using the top 100, 200, 500 bad *likes*. The *likes* budget represents a small percentage of the total features. A budget of 100, 200, and 500 *likes* represent only 0.2%, 0.4%, and 1% of the political view *likes* and 0.3%, 0.7%, and 2% of the *likes* in sexual orientation. Keeping a *like* budget is important because liking many pages in a short amount of time could be used to flag a user account as malicious and block them from further liking additional pages due to social networks internal controls [9]. For the good feature\bad feature attacks, we will only present the results for the most successful metric in increasing re_o due to page limitation. Figure 1(a) shows the results for the good feature attack with the Score_Support that caused the highest increase in re_o. We notice that with sexual orientation SVM had the most vulnerability in this attack followed by RF and LR. With political view the highest error was with RF. With LR and SVM, re_o did not increase above 32%. Figure 1(b) presents the results for bad feature attack with the MostCommon metric which had the highest re_o. With sexual orientation re_o was the highest with LR where it reached 57% when deleting with a budget of 500 *likes*. Using political view data, the highest re_o was with SVM where it reached 79% when deleting with a budget of 500 *likes*. Even though the adversary budget ranges from 100 to 500 *likes* for the bad feature attacks, the actual manipulated *likes* is significantly less since most users do not have these *likes* to delete. For example, the actual average of deleted *likes* using the MostCommon metric with political view data is 11, 18, and 33 when deleting with a budget of 100, 200, and 500 *likes*.

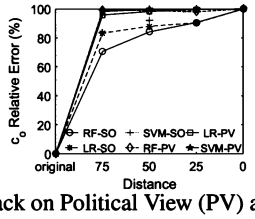
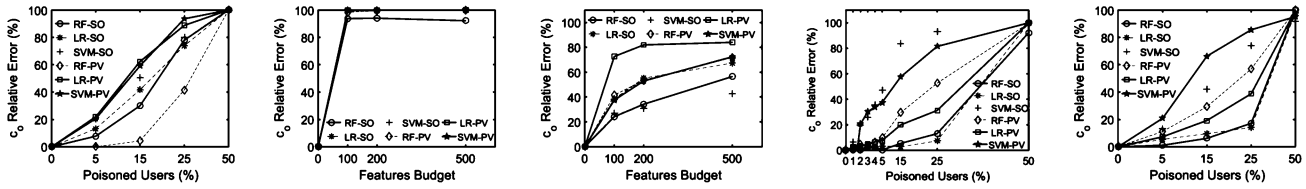


Fig. 2: Mimicry Attack on Political View (PV) and Sexual Orientation (SO)

Mimicry attack was performed on 20% of users in class c_o in the test data. Figure 2 shows the results for this attack. As the distance between the users under attack in the test data and their most similar users in the training data decreases, more users prediction changes from the class c_o to c_t. When the distance is 0%, all the attacked users are placed in the c_t class. When reducing the distance to 75%, political view relative error (re_o) reached a minimum of 96% with LR and sexual orientation reached a minimum of 71% with RF. These results demonstrate that mimicry attack has the highest success among all the evasion attacks.

2) Poisoning Attack Results: Figure 3(a) shows the results class altering attack when poisoning 5%, 15%, 25%, and 50% of the users. For political view, the relative error (re_o) reached more than 89% with LR and SVM when poisoning 25% of the users. When altering the class of sexual orientation users, re_o reached 74% and above with all the classifiers when poisoning 25% of the users. RF was the most robust classifier against this attack for both sexual orientation and political view data.

Figure 3(b) shows the features altering attacks where the adversary adds good random *likes* to the training data. Adding *likes* from X_g^t to users in c_o class is the most effective poisoning attack. When having a budget of only 100 *likes*, both attributes re_o reached 94% and above. When performing bad



(a) Class Altering (b) Good Feature (c) Bad Feature (d) 500 Good Features (e) Fake User Addition

Fig. 3: Poisoning Attacks on Political View (PV) and Sexual Orientation (SO)

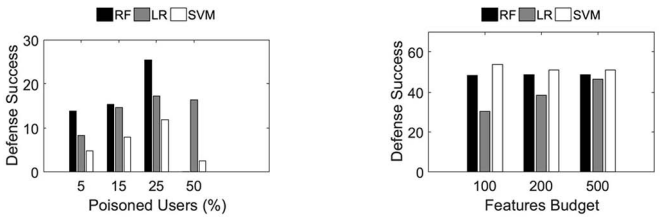
feature evasion attack we noticed that using the MostCommon metric caused the highest error. Therefore, we also use the MostCommon *likes* when deleting features from the training data. We can see in Figure 3(c) that deleting bad *likes* for c_t from users in class c_o is not as effective as adding *likes*. The reason for the lower success results in this attack is similar to the bad feature evasion attack. Basically, with 100 *likes* budget, the actual deleted *likes* is 11 for political view and 12 for sexual orientation.

Since it is impractical to assume access to all the training data, we test having 500 good *likes* budget when assuming access to different percentages of the training data in Figure 3(d). When accessing 25% of the political view training data, re_o showed an increase of 53% with RF, 31% with LR, and 82% with SVM. For sexual orientation, when accessing 25% of the training data, RF showed re_o increase of 13% while LR showed 3% of re_o increase. SVM had the highest vulnerability where re_o reached 93% when accessing 25% of the training data. In this particular experiment, we also show the effect of poisoning less than 5% of the training data. With a budget of 500 *likes* to add to only 3% of the training users, re_o increased to 26% with sexual orientation and 30% with political view using SVM. This is quite significant given the small size of the poisoned data.

Fake user addition poisoning attack results are shown in Figure 3(e) where we add crafted fake users. The results of this attack is very similar to the results of adding good *likes* with a budget of 500 shown in Figure 3(d). The only exception is for SVM with sexual orientation where SVM was more vulnerable when we added *likes* with 500 random *likes* budget. The difference in re_o increase reached more than 40% when we poisoned 15% of the training data.

IV. Countermeasures

In this section we discuss different countermeasures on sexual orientation data only due to page limitations. We propose *Flip on Negative Impact (FONI)* as a defense against the various poisoning attacks inspired by [5]. FONI, evaluates the influence of an instance x_i on the classification model by building the model on both the training set with x_i and without x_i. If the addition of x_i decreases the performance of the classifier, x_i label is flipped. As shown in Figure 4(a), RF has the highest success in decreasing re_o when poisoning less than 50% of the users. LR shows a significant reduction in the re_o when poisoning 50% of the users compared to RF and SVM. For evasion attacks, we propose *Strong Ties Detection* defense. With this defense, we aim to catch fake target class



(a) FONI Defense Against Class Altering Poisoning Attack

(b) Strong Ties Detection Defense Against Good Feature Evasion Attack

Fig. 4: Countermeasures

users (opposite class users with fake features). We search for users placed in c_t class by the ML model but with strong ties to the c_o class. The strong ties is detected by having a number of c_o top *likes*, *likes* in X_g^o, above a threshold δ. We set δ=1% of the top 500 *likes* in X_g^o. The result of applying this defense against good feature evasion attack is shown in Figure 4(b). SVM has the highest defense success where re_o is reduced by more than 50% with the different features budget.

V. Discussion and Related Work

Many prior research studied the effect of different attacks on the prediction models. Most of these research were applied in the context of security such as attacks on spam filtering and intrusion detection systems [10–14]. In the context of social networks, Wang et al. investigated the robustness of machine learning techniques against adversarial attacks in China's Twitter network [15]. To our knowledge, [16] is the only work on attacking data mining models built on Facebook data. The work in [16] studied a problem similar to our work where the goal is to hide Facebook user personality by manipulating their *likes*. The only attack performed is deleting features based on their coefficient in LR model at test time. Compared to [16], we consider many more attack classes for different types of data mining models.

Different mechanisms have been introduced to protect against attacks in social networks. Previous techniques (e.g., [17–19]) are designed to detect fake accounts that spread malicious content and to detect compromised accounts [20]. All these mechanisms do not apply to our attacks since our attacks do not have malicious content and only modify the user profiles with legitimate content (e.g., legitimate *likes*). Beutel et al. offer a mechanism named Copycatch to detect lockstep behavior where group of Facebook users like the same pages at about the same time [3]. Copycatch main goal is to detect spammers trying to increase a page *like* count. This mechanism can not easily detect any of our poisoning attacks that add

features since we randomize feature addition per profile to prevent group behavior. In addition, in good feature evasion attack, we investigate an individual's ability in evading prediction. Therefore, this attack does not create a group behavior that can be detected by CopyCatch. SynchroTrap is another mechanism that is used for malicious accounts detection in social networks that has similar design as CopyCatch [21]. In addition to the features offered by CopyCatch, SynchroTrap can also uncover repeated spam-behavior from the same user or IP address such as page *likes* or photo uploads. The same argument of how our attacks can evade CopyCatch can be applied to SynchroTrap. Our attacks can easily evade such mechanism since we do not manipulate posts in the normal users' profiles and only the *likes* are manipulated. Moreover, when we create fake users in the training data, we do not create any posts for them which is allowed in Facebook.

Finally, we would like to stress that our work can be easily extended to different social networks. For example, we can consider Twitter as our social network and the features could be created using the following list and tweeted messages.

VI. Conclusion and Future Work

In this paper we analyzed *Facebook likes* and their usefulness in predicting different users in different classes. We showed that using *likes* alone, we can predict political view and sexual orientation with high accuracy. We performed several attacks on these predictable users to measure the vulnerability of the prediction models. Our results show that these models are vulnerable to the different types of attacks. Therefore, data mining models built on social network data could be easily manipulated by malicious attackers. We also presented countermeasures to defend against both evasion and poisoning attacks. The countermeasures successfully reduced the severity of the discussed attacks. A possible future direction is to study the impact of users' social network. Even if a user has no *likes*, they can be profiled by who they connect to [22, 23]. Using these social links and neighbors can make the prediction model more robust since it is difficult for the adversary to control their neighbors.

Acknowledgement

The authors would like to thank Michal Kosinski and David Stillwell for sharing their data.

The research reported herein was supported in part by NIH awards 1R01HG006844,2R01HG006844 NSF CNS-1111529, CNS-1228198, CICI-1547324, and IIS-1633331.

References

[1] CareerBuilder. (2014) Number of employers passing on applicants due to social media posts continues to rise. http://www.careerbuilder.com/.

[2] Facebook help center. https://www.facebook.com/help/.

[3] A. Beute, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: Stopping group attacks by spotting lockstep behavior in social networks," in *WWW*, 2013.

[4] (2014) Facebook form 10-k annual report. http://investor.fb.com/secfiling.cfm?filingid=1326801-14-7\&CIK=1326801.

[5] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. S. J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter." Usenix Workshop, 2008.

[6] M. Kosinski, D. Stillwell, and T. Graepel, "Private traits and attributes are predictable from digital records of human behavior," in *PNAS*, 2013.

[7] G. H. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix," *SIAM Journal on Numerical Analysis*, vol. 2, no. 2, pp. 205–224, 1965.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, Nov 2009.

[9] (2011) Can pressing the "like" button too many times get you in trouble with facebook? http://www.webpronews.com/facebook-like-button-mike-doughty-2011-06/.

[10] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna, "Automating mimicry attacks using static binary analysis," in *USENIX Security Symposium*, 2005.

[11] D. Wagner, C. Kruegel, and P. Soto, "Mimicry attacks on host-based intrusion detection systems," in *CCS*, 2002.

[12] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," in *CEAS*, 2005.

[13] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *ICML*, 2012.

[14] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines," in *ECAI*, 2012.

[15] G. Wang, T. Wang, H. Zheng, and B. Y. Zhao, "Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers," in *23rd USENIX Security Symposium*, 2014, pp. 239–254.

[16] D. Chen, S. P. Fraiberger, R. Moakler, and F. Provost, "Enhancing transparency and control when drawing data-driven inferences about individuals." NYU Working Paper No. 2451/33969, 2015.

[17] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on twitter," in *CEAS*, 2010.

[18] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: Social honeypots machine learning," in *SIGIR*. ACM, 2010, pp. 435–442.

[19] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *ACSAC*. ACM, 2010.

[20] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, "Compa: Detecting compromised accounts on social networks," in *Proceedings of NDSS*, 2013.

[21] Q. Cao, X. Yang, J. Yu, and C. Palow, "Uncovering large groups of active malicious accounts in online social networks," in *Proceedings of CCS*, 2014, pp. 477–488.

[22] F. A. Zamal, W. Liu, and D. Ruths, "Homophily and latent attribute inference: Inferring latent attributes of twitter users from neighbors," in *ICWSM*, 2012.

[23] N. Z. Gong and B. Liu, "You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors," in *25th USENIX Security Symposium*, 2016.