technologies

MDPI

Article

Exploring Adversarial Attacks and Defences for Fake Twitter Account Detection

Panagiotis Kantartopoulos [1], Nikolaos Pitropakis [1,2,*], Alexios Mylonas [3,*] and Nicolas Kylilis [2]

[1] School of Computing Edinburgh Napier University, Edinburgh EH11 4DY, UK; 40220561@live.napier.ac.uk
[2] Eight Bells LTD, Nicosia 2002, Cyprus; nicolas.kylilis@8bellsresearch.com
[3] Department of Computer Science, University of Hertfordshire, Hatfield AL10 9AB, UK
* Correspondence: nikolaos.pitropakis@8bellsresearch.com or N.Pitropakis@napier.ac.uk (N.P.); a.mylonas@herts.ac.uk (A.M.)

Received: 9 October 2020; Accepted: 2 November 2020; Published: 6 November 2020

Abstract: Social media has become very popular and important in people's lives, as personal ideas, beliefs and opinions are expressed and shared through them. Unfortunately, social networks, and specifically Twitter, suffer from massive existence and perpetual creation of fake users. Their goal is to deceive other users employing various methods, or even create a stream of fake news and opinions in order to influence an idea upon a specific subject, thus impairing the platform's integrity. As such, machine learning techniques have been widely used in social networks to address this type of threat by automatically identifying fake accounts. Nonetheless, threat actors update their arsenal and launch a range of sophisticated attacks to undermine this detection procedure, either during the training or test phase, rendering machine learning algorithms vulnerable to adversarial attacks. Our work examines the propagation of adversarial attacks in machine learning based detection for fake Twitter accounts, which is based on AdaBoost. Moreover, we propose and evaluate the use of k-NN as a countermeasure to remedy the effects of the adversarial attacks that we have implemented.

Keywords: adversarial attacks; poisoning; social media; machine learning; Twitter

1. Introduction

Social media occupies a part in everyday peoples' life, allowing their users to express and share their views regarding life and timelines. They have also become a key medium that people use on a daily basis to indite their opinion in public matters or to access information. As a result, multiple online social networks are constructed comprising of users with common interests. Unfortunately, in the current threat landscape malicious social media users exist who wish to take advantage of those social networks and use them as a tool to achieve their objectives. One notable example is the USA presidential election in 2016, where Russian users used fake identities to populate Twitter in order to affect the users' opinion regarding the appropriate candidate. A year later, fake news was spread about the French presidential candidate in the elections of Europe, creating a cyber attack framework of dividing, destabilizing and deceiving users and public opinion [1].

Nowadays online social media suffers from the proliferation of fake accounts, which are either manually or automatically created. Fake accounts are used as the vector to interact with a population in a social network, aiming to deceive or mislead them by posting either malicious or misleading content. This content includes (i) using shortened URLs that contain malware and tricking innocent users to click them, (ii) identity theft, (iii) data leakage, and (iv) generic social engineering techniques. Additionally, malicious social media users even form and spread rumors, which may affect businesses and society in a large scale. To ensure that fraudulent accounts stay for a longer period on the

social network, fake users constantly pursue a diligent strategy based on legitimate profiles, i.e., by mimicking real users. As a result, this impairs the capability of social media to identify and stop the distribution of malicious content from those accounts.

Undoubtedly, identifying and eliminating fake accounts preserves the integrity of social media and can lead to the reduction of malicious behavior. As such, a variety of techniques have been proposed in the literature to detect fake accounts on Twitter [2,3]. Amongst them, machine learning has been used in order to optimize and strengthen the detection processes of fake accounts. Considering that the data volume generated by social media constantly grows, there is a need for efficient and robust detection of malicious behavior on these platforms, which will allow them to be more secure and reliable for users. Nonetheless, threat actors have updated their tactics and they have started to target machine learning algorithms themselves with adversarial attacks [4]. Adversarial attacks, which could facilitate poisoning, evasion or mixed mode attacks, weaken the reliability and robustness of the fake profile detection. Thereat, there is a perpetual need for an in-depth understanding of the effect of adversarial attacks to this machine learning based detection.

To this end, this work studies the feasibility of adversarial attacks on fake Twitter account detection systems that are based on supervised machine learning. We propose and implement two adversarial attacks that aim to poison a classifier that is used to detect fake accounts on Twitter. We evaluate our attacks with two different scenarios where the attacks are used against a classifier, which provides fake account detection and has been trained with a real dataset. We also propose and evaluate a countermeasure that mitigates the adversarial attacks that we propose herein.

The contributions of our work can be summarised as follows:

- We mount adversarial attacks against a supervised machine learning classifier that detects fake accounts on Twitter, which has been trained with a real dataset. We explore two types of attacks in which existing entries in the training dataset are poisoned as well as new malicious entries are appended in the dataset.
- We study the effect of the aforementioned adversarial attacks in the classification accuracy of the detection system with two scenarios, i.e., static adversarial attacks and dynamic adversarial attacks.
- We propose and evaluate the use of a countermeasure to protect from the aforementioned attacks, by detecting potential poisoning that has taken place in the training dataset. We demonstrate that our countermeasure is able to mitigate the type of adversarial attacks that we have focused in this work.

The rest of the paper is organized as follows: Section 2 includes related work. Section 3 describes our methodology and Section 4 provides our experimental results. Section 5 concludes the paper and provides pointers for future work.

2. Related Work

Profiling a Twitter user is possible by analyzing the tweets and feeds that the user is subscribed to, however this process is not trivial [5]. Past literature has focused on profiling users based on their social traffic, comparing their profile description, or their demographic data with the topic, or even following frequent patterns along with term frequency [6], a process that needs to be completed in a timely manner [7]. Other studies focused on detecting malicious Twitter content arisen from the malicious URLs used by non-legitimate users. As Twitter had a limitation of 140 characters per tweet message (the current limit is 280 characters), according to [8] malicious users shortened URLs that in turn mask the URL, thus facilitating phishing links using a popular hashtag or responding with interesting lexical content.

Machine learning has been used in the past to detect fake accounts on social media. On Twitter, detecting fake accounts consists of building models based on a user profile and behavior as well as language processing. Fake accounts can be spotted in various ways, such as finding posts

with malicious content, having an unbalanced set of more followers than users following them, observing their social behavior in posting new information or reproducing posted content, and their social network interactions. These data can be used as features in machine learning to train and test models that identify fake accounts on Twitter.

In this regard, the majority of past literature analyzes user activities referring to attributes based on the user profile, such as friends and followers, region and language features. Related work also follows an elaborated investigation of posts' content, the users' creation time and the users' responsiveness, aiming to register a timeline of users that will endeavor and fortify the detection process. Stringhini et al. [9] analyzed how spammers operate on social media platforms, reporting more than 15,000 suspicious accounts. Yang et al. [10] proposed taxonomy criteria for detecting Twitter spammers, providing experiments on how designed measures have higher detection rates. Liu et al. [11] developed a hybrid model for detecting spammers in Weibo, a Chinese online platform resembling Twitter, by combining behavioral features, network features, and content-based features.

Moreover, different algorithms have been proposed to identify malicious Twitter accounts as quickly as possible. As Ensemble Learning [12] enables predictions based on different models it became popular. One of the most popular choices is Random Forest [13] despite the fact that it is sensitive to noise [14]. One other popular option is Adaboost, which uses an iterative approach [15] to learn from the mistakes of weak classifiers, and strengthen them [16]. Miller et al. [17] used anomaly detection techniques in spam detection on Twitter, by using clustering on tweet text and a real-time tweet stream. Similarly, Song et al. [18] applied clustering [19] by grouping accounts sharing similar name-based features, identifying malicious accounts when users were created within a short period of time. Finally, the goal of Jane Im et al. [20] was to automatically create a machine learning model for the detection of fake Twitter accounts, presuming upon features of users' profiles, such as language and region, pieced together with accounts' relationship on following and followers, in addition to the tweet or retweet feature of Twitter.

Detection of fake accounts with machine learning algorithms is nonetheless prone to adversarial attacks, where carefully crafted input data, either at training or at test time, can subvert the predictions of the models [21]. There are various types of attacks against machine learning, with evasion attacks being more popular, where the attacker uses malicious samples during the test time in order to evade detection, and poisoning attacks where the attacker injects carefully crafted data in the training dataset and reduces its performance with false negatives [4]. The selected attack type depends on the knowledge of the system and its data. Nonetheless, poisoning attacks often become more efficient as injecting misleading samples directly into the training data can cause the system to produce more classification errors. In addition, poisoning attacks target training data and due to the need of periodic retraining of a machine learning model this allows poisoning attacks to be used more often and in multiple steps [21].

Prior work on data poisoning has mostly focused on attacking spam filters, malware detection, collaborative filtering, and sentiment analysis. Shafahi et al. [22] proposed an attack during training time to manipulate test-time behavior by crafting poison images colliding with a target image, stating the power of these attacks in transfer learning scenarios. Wang et al. [23] performed a systematic investigation of data poisoning attacks on online learning, claiming that Incremental and Interval attacks have higher performance than Label Flip attacks. Brendel et al. [24] introduced the Boundary Attack, which does not require hyperparameter tuning and does not rely on substitute models towards decision-based attacks, suggesting that these types of attacks are highly relevant for many real-world deployed machine learning systems. Furthermore, Zhang et al. [25] proposed a poisoning attack against the federated learning system, showing that it can be launched by any internal participant mimicking other participant's training samples.

Chen et al. [26] explored an automated approach for the construction of malware samples with three different threat models, significantly reducing machine learning classifiers' detection accuracy. With regards to social media, Yu et al. [27] conducted adversarial poisoning attacks in the training data

using two methods, namely: (i) a relabeling technique of existing data and (ii) with the use of crafted data as additional entries. The crafted data can resemble a legitimate user having a decent number of followers and friends; concerning the tweet that can mimic a legitimate tweet by avoiding the inclusion of any sensitive words or hashtags. Moreover, the crafted data can be marked as available for future retraining use for the machine learning model, enforcing the poisoning procedure of the dataset.

A starting point defending against adversarial attacks is to hide important information about the system from the adversary. Although determining the features used by the classifier is not difficult, manipulating all those features may be impossible. Another defending strategy is randomizing the system's feedback, as it makes it harder for the adversary to gain information, especially on Twitter where the adversary and the benign users use the same channel [21]. Another strategy is to create an adversary classifier and train it with adversary attack patterns. In that case, as with the use of any classifier, there is a constant need for retraining to avoid poisoning attacks as well as adjust to new adversarial attack patterns.

Laishram et al. [28] examined poisoning attacks against Support Vector Machine algorithms, proposing a method to protect the classifier from such attacks. Their method, i.e., Curie, works as a filter before the buffered data are used to retrain the classifier removing most of the malicious data points from the training data. Nevertheless an attacker might bypass this detection system using a combination of poison and evasion attack methods. Biggio et al. [29] experimentally proved that bagging is a general and effective technique to address the problem of poisoning attacks on specific applications, such as spam filtering and web-based intrusion detection systems. They noted that using a good kernel density estimator requires more effort from the adversary, who potentially needs to handle a much larger percentage of training data to enhance his attack. Paudice et al. [30] proposed an outlier detection based scheme capable of detecting the attack points against linear classifiers. However, in less aggressive attacks, like label flipping, detection can become infeasible since the attack points are on average closer to the genuine points.

3. Methodology

In this section we present our approach to impair the performance of machine learning based detection of fake accounts on Twitter. In doing so we present different scenarios, with the use of dynamic and static thresholds, where adversarial machine learning attacks are mounted against a classifier that is trained on a real dataset in order to be used for the fake account detection.

This work assumes that the attacker has the ability to poison the training set of the machine learning classifier, which a defender uses in order to identify and block fake accounts from Twitter. The goal of the defender is to achieve the highest accuracy in detecting fake accounts on Twitter with the use of machine learning. The goal of the attacker is to: (i) introduce confusion to the classifier, thus decreasing its performance (i.e., increasing false positives and false negatives) and (ii) introduce the minimum possible modifications in the training set.

We assume that the attacker poisons the training dataset by: (i) randomly relabeling data, i.e., altering the label of an entry from the fake account class to otherwise, or (ii) adding new crafted entries following the format of the tweets in the dataset. We also assume that the attacker has to meet a threshold of records, which refers to the percentage of randomly poisoned records. The attacker's goal is to adhere to this poisoning threshold in order to remain undetected and at the same time reduce the classifier accuracy as much as possible. Finally, we assume that the detection mechanism is capable of detecting if the training dataset has been poisoned more than a specific threshold.

3.1. Adversarial Attacks

The first type of adversarial attack that we examine in this work is a relabeling attack based on the attacker's pursuit to relabel entries that belong to fake identities in the training set [31]. The outcome of the attack is a training set poisoned such that it impairs the classification accuracy of the classifier. As per Algorithm 1, the poisoning function aims to lower the classifier's accuracy to less than 60% in

every data frame, whilst the threshold is initialized with the value of 1. Then, the threshold is adjusted until the poisoning target is reached. Specifically, the threshold is increased by three if the classifier accuracy is a lot higher than 60% and by one if the threshold accuracy is higher but close to 60% and the threshold has a value higher than six. In both scenarios our work considers this to be a white box attack, therefore the attacker has knowledge on the targeted system and its data.

Algorithm 1: Poisoning Relabeling Algorithm.

```
Parameters: threshold, cleanLabels, i=1;
Input: training set T = xi,yi , 1 <= i <= m;
do
    if Ti = fake_identity then
        random_clean_label = rand(cleanLabels)
        Ti <- random_clean_label
        accuracy<-classify(T)
        if accuracy > 60% AND threshold>=6 then
            i++
        else if accuracy > 60% AND threshold<6 then
            i<- i+3
    end
while i<= threshold;
Output: Ti;
```

The second adversarial attack that is proposed herein is the addition of the new entries attack. In this attack, we assume that the attacker adds new entries in the dataset without disrupting its pre-existing structure, with the aim to decrease the classifier's performance. To this end, the attacker copies a subset of the entire dataset based on the poisoning threshold and modifies the author, account type, region, updates, and retweet status accordingly. Then, the subset of those newly crafted data entries is appended to the original dataset, before the classifier's training.

More specifically, the poisoning is semi-random and follows the approach that is described in Algorithm 2. A random row pointer is selected in which the name of the account is replaced with a random author derived from another row. The region is set to Unknown, the account type is replaced with a legitimate one, the updates field is set to the value of the randomly selected row and the retweet field is set to zero.

Algorithm 2: Poisoning New Entries Algorithm.

```
Parameters: threshold, cleanAccountsType, i=1, newEntries[];
Input: Dataset D = xi,yi , 1 <= i <= m;
for ( i = 1; i <= threshold; i = i + 1 ) {
    newEntries[i].append(Di)
    random_author=rand(authors)
    random_clean_account_type = rand(cleanAccountsType)
    newEntries[i].author = random_author
    newEntries[i].account_type = random_clean_account_type
    newEntries[i].region = 'Unknown'
    newEntries[i].updates = random_row
    newEntries[i].retweet = 0;
}
Output: newEntries;
```

Finally, in this work we study adversarial attacks that are mounted with the use of static and dynamic thresholds. In the static scenarios, the training dataset is available as a whole and therefore the training and poisoning takes place with a static threshold. In contrast, in the dynamic scenario we assume that the data appear in chunks and as a result the classifier is periodically, independently trained, and as a result the threshold used is dynamic. This could happen when the defender cannot process all the available Twitter data as they appear therefore the detection engine is trained with the latest chunks of data. This would allow the defender to periodically adjust the detection engine to match the trends of the Tweets from the malicious and benign users.

3.2. Defensive Mechanisms

In this work we use AdaBoost for the detection of fake Twitter accounts. AdaBoost is a decision tree algorithm combining multiple weak classifiers, forming shorter decision trees and thus increasing the accuracy of classifiers. Predictions are made from individual classifiers by incrementing the weights of the incorrectly classified observations in the training dataset. AdaBoost was used in this work as it is, in general, not prone to overfitting and noise [16,32], which is present in our dataset.

To train the classifier we used the Twitter dataset from [33]. We used an 80–20 split (i.e., using 80% of data for training and 20% for testing), which is a commonly used split in machine learning. Upon normalization, the classifier uses the following features, which correspond to metadata of the Twitter accounts that are contained in the dataset, namely: (i) region, (ii) language, (iii) number of accounts following, (iv) number of followers, (v) updates, (vi) type (tweet or retweet), and (vii) account type (i.e., malicious or benign). In our experiments we used the default parameters for AdaBoost from Scikit-learn, namely DecisionTreeClassifier with max_depth = 1; maximum number of estimators n_estimators = 50; linear loss function; and learning_rate = 1.

Even without any tuning and feature engineering activities and by solely using the account metadata that have been mentioned beforehand, AdaBoost achieves an accuracy of over 80% in detecting fake Twitter accounts. It worth noting that optimizing the classification performance in identifying fake accounts is not the scope of this work.

Moreover, in this work we propose a defensive algorithm that mitigates the adversarial attacks that are studied in this work. We assume that during the above-mentioned adversarial attacks the defender facilitates Algorithm 3 as a detection mechanism, which scans and reports if the classifier's training set is poisoned, alongside a poisoning threshold.

Algorithm 3: Defence Algorithm.

```
Parameters: k=5, malicious_samples = 0;
Input: training set T = xi,yi , 1 <= i <= m;
for ( i = 1; i <= m; i = i + 1 ) {
    Di = k-NN (Ti ,T)
    vote = label_voting (Di)
    confidence = vote_confidence (Di,vote)
    if confidence > 0.60 and vote != Ti then
        malicious_samples++
    end
}
Output: malicious_samples;
```

The algorithm uses k-NN to find the closest neighbors of each sample in the training dataset, with the aim of spotting outliers that might be malicious, and computing the poisoning percentage. For each sample in a potentially poisoned training dataset, the closest neighbors are found using the Euclidean distance, which are then used for majority voting. If the confidence of voting is over 60%

and the label differs from the training set's label, the instance is marked as poisoned. Then, a mitigation percentage of those records is calculated informing the mitigation function for the amount of relabeling that needs to be set, as per Algorithm 3. Afterward, the output of the defense algorithm is passed to the mitigation function for implementing relabeling consequently. It is worth noting that for k-NN we are using the default parameters from Scikit-learn library (namely: n = 5; algorithm = auto; leaf_size = 30; p = 2, euclidean distance; and metric = minkowski), without optimizing the classification performance of the classifier, which we consider out of the scope of this work.

4. Implementation of Adversarial Attack Strategies

In this section we describe four scenarios where we mount static and dynamic adversarial attacks against a classifier that detects fake accounts on Twitter.

4.1. Test Environment

With regards to our test environment, we used a 64-bit Windows 10 Operating System, with an i5-5200 CPU @ 2.20 GHz Processor and 8GB memory (RAM). For our experiments we used python and the scikit-learn machine learning library.

This work uses the Twitter dataset from [33]. It consists of approximately three million tweets from the 2016 United States Elections, which includes: (i) the account name, (ii) language, (iii) region, (iv) number of followers and following, (v) information about the tweet whether it is original or a retweet, and (vi) the content of the tweet. From the aforementioned data our analysis focuses on all the data apart from the content of the tweet. It is worth noting that manually identifying fake accounts is not a trivial task. To this end, an exploration of the dataset with the use of statistics allowed us to understand the data and contributed to a solid strategy creating a more accurate algorithm.

In this regard, we started with an analysis of the relationship between the number of friends and followers for each user to identify popular and unpopular accounts (Figure 1). Figure 1 illustrates that the majority of users have a linear relationship with followers and friends, with the exception of outliers found in the top left corner and in the middle right. Inspecting the accounts, 28% has at least 2-fold more followers than following, whilst a fair amount (over 27%) of the popular and unpopular accounts are Russian language accounts.

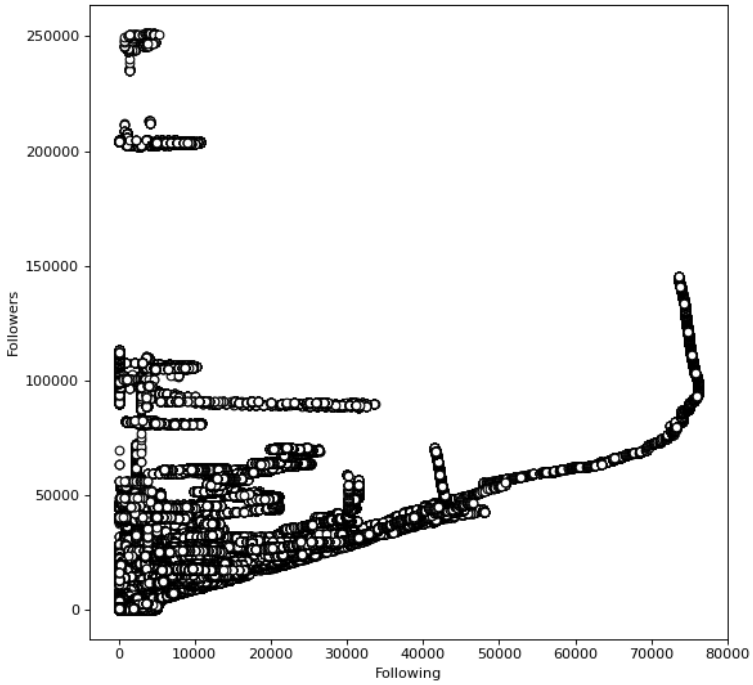
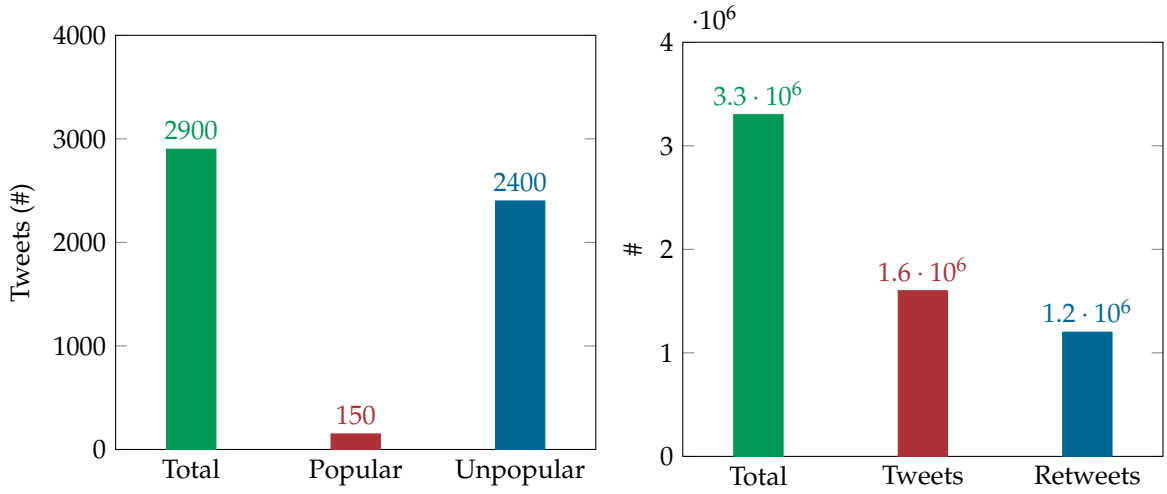


Figure 1. Number of Followers and Following.

It is worth noting that one would anticipate tweets from a popular account to have a high influence on the network, as they will be viewed by many followers, opposed to a tweet from an unpopular account, which has a lower influence on the network. Nonetheless, as depicted in Figure 2a, our exploratory analysis suggests that unpopular accounts were more active in the dataset than popular ones, having a high percentage of influence amongst the network. Further inspection of the unpopular tweets and retweets activity in Figure 2b, suggests a retweeting pattern between the fake profiles network, setting a structure on retweeting techniques to increase the chances of those tweets getting seen by multiple users out of their network. This was a valid strategy for adding more legitimate users to their social network thus blending fake and non-fake profiles.



(**a**) Popular and Unpopular Users (**b**) Tweets and Retweets

Figure 2. Users and Tweets statistics.

This indicates that a strategic plan has been followed by users trying to alter the results of the 2016 election by influencing the perceptions of real users. This tweet amplification was achieved through a social network of fake profiles following a precise activity, taking time to gather followers by making a equal number of friends posing like real profiles [31]. After gathering followers, they could then spread their content and opinions amongst a huge network consisting of fake and real users, thus influencing real users.

Moreover, one would consider that expressing an opinion during a specific country's election event when the users are not part of this country might need to be investigated. Even though in some cases, as in the one in this dataset, the country's election can be a worldwide concern, the analyst needs to consider the political context. In the current dataset, however, it is evident that a social network with fake identities is constructed to participate in the social media battle regarding the subject of the elections.

4.2. Adversarial Attacks

In the following subsections we refer to the static and dynamic adversarial attack scenarios that we have studied in this work. The adversarial attacks aim to impair the classification performance of AdaBoost, which has been trained with a clean dataset. The model achieves 81% accuracy, having more than 200,000 records successfully identified as malicious actions; although approximately 40,000 entries were marked as fake despite being authentic. Prior to the poisoning, the AdaBoost model classified all the malicious records successfully, but an error rate of 13% regarding False Positives is noticed, as original samples were marked wrongly as fake.

4.2.1. Static Relabeling Attack

In the static relabeling scenario, the attacker aims to increase the false negative rate of the classifier to hinder the detection of fake accounts. Our results suggest that by using a random relabeling procedure in the training dataset a classifier can be confused thus decreasing its performance. Specifically, performing the static adversarial attack scenario results in reducing the classifier's accuracy to 48% from 81%, by poisoning only 7% of records. It is worth noting that the accuracy of the classifier can be further impaired at a desirable percentage, by adjusting the poisoning threshold accordingly.

As depicted in Figure 3, the mitigation plan managed to restore the classifier accuracy on detecting fake identities, having the same false positive error, i.e., 12.9%. The low poisoning threshold indicates the effectiveness of the adversarial attack in this scenario in confusing the classifier. In other words, our results indicated that recovering from the adversarial attack is more expensive than poisoning the dataset, as reflected by the percentage of entries that need to be poisoned or recovered, i.e., 7% (poisoning threshold) and 10% (mitigation threshold), respectively.

Finally, our results suggest a 6% false positive rate. It is worth noting that even though the classifier's goal is to detect fake identities, thus minimizing false negatives, false positives should also be considered by the defender. This holds true as, while false positives might not lead to the realization of a threat, they affect the accuracy of the classifier, as well as might irritate the benign Twitter users if Twitter decides to adopt the suggested detection mechanism.

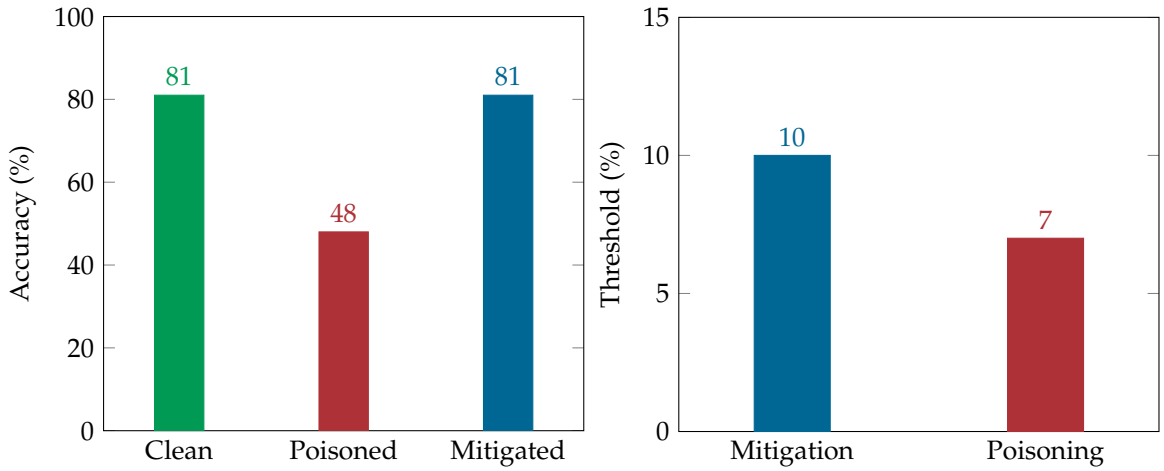


Figure 3. Static Relabeling Results.

4.2.2. Dynamic Relabeling

Figure 4 and Table 1 depict the accuracy of the classifier during the dynamic relabeling attack scenario for each of the 12 frames, summarizing the results of our experiments: (i) before poisoning (i.e., clean classifier green line), (ii) after poisoning (poisoned classifier, red line), and (iii) upon the use of the detection mechanism upon poisoning (mitigation, blue line). It also shows the variance of poisoning and mitigation threshold during our dynamic relabeling attack scenario. Overall, the accuracy of the clean, poisoned and mitigated classifier is 80%, 44% and 71%, respectively. This suggests that this poisoning attack is quite successful, as it limits the performance of the classifier almost in half, thus changing the assumption of the system about benign and malicious entities, with a low poisoning threshold (i.e., 5% on average), which minimizes the computational effort and resources of the adversary.

More specifically, in the first four frames, the accuracy of the classifier before poisoning (hereto referred to as *clean classifier*) starts from 80% and slightly increases in the next three frames, whereas the classifier upon the mitigation (hereto referred to as *mitigated classifier*) decreases to 51%. The poisoned classifier begins with a very low accuracy, slightly increasing and reaching approximately 48% in the first two frames, but its accuracy decreases after the threshold's value

increases to 10%. In the next frames, as Figure 4 suggests, the clean and mitigated classifiers show a sudden fluctuation in frame 5, as the former drops to 37% accuracy, which is the lowest accuracy in this scenario, while the latter reaches 92% respectively. In the 6th frame the accuracy for both classifiers is 82%. Nonetheless the accuracy of the mitigated classifier keeps showing sudden fluctuations in the next two frames, reaching its lowest accuracy (i.e., 36%) in frame 7 and rising to 74% in frame 8. Finally, the accuracy of the two classifiers becomes similar in the last frames.

As shown in Figure 4, the poisoned classifier's accuracy remains between 37% and 54%, hindering the identification of fake accounts and has the same accuracy in frame 7 with the clean classifier, as discussed earlier. From frame 8, the poisoned classifier has its highest accuracy, which ranges from 50% to 56%. Except from frame 5 where the difference between the accuracy of the mitigated and clean classifier is 55%, the mitigation accuracy is always lower than that of the clean classifier, but always higher or equal than poisoned classifier's.

Regarding the effects of the poisoning and mitigation thresholds to the respective classifiers' accuracy the results showed that in frames 3, 4 and 7 the greater the value of the poisoning threshold, the lower the accuracy of the poisoned classifier. However, this behavior is not observed in frames 9 and 10, where the poisoning threshold is increased without decreasing the accuracy of the poisoned classifier. On the other hand, the mitigation threshold is stable in the first 3 frames, nonetheless the accuracy of the mitigated classifier decreases. In frames 4 to 6 a mitigation threshold increase leads to an increase in the accuracy of the classifier. However, this is not the case in the following frames, as even though the threshold remains high, the accuracy decreases. Moreover, in the last 4 frames the mitigated classifier's accuracy is high, with similar values to that of the clean classifier, having a threshold between 8% and 10%. One should note an abnormality of the detection mechanism in the static and dynamic relabeling procedure. While similar ranges of poisoning and mitigation thresholds were expected in the two scenarios, nonetheless in frames 3 to 6 the thresholds are diametrically opposite. Finally, these fluctuations in the accuracy of the classifier (clean, poisoned, mitigated), as summarized in Figure 4, are due to the fact that: (i) the training data are not available as a whole but they appear in chunks (e.g., they are being collected periodically, for instance every 2 h), (ii) the contents of every frame, which although have the same data volume, consist of different distributions of benign and malicious data.

Table 1. Results of Dynamic Relabeling Scenario.

Data Frame	Original Accuracy	Poisoning Accuracy	Poisoning Threshold
1	0.78	0.19	1
2	0.87	0.49	1
3	0.88	0.43	10
4	0.98	0.36	10
5	0.37	0.37	1
6	0.84	0.44	1
7	0.81	0.36	9
8	0.85	0.54	1
9	0.98	0.51	10
10	0.96	0.56	10
11	0.72	0.56	1
12	0.98	0.51	5
Average	0.83	0.44	5.0

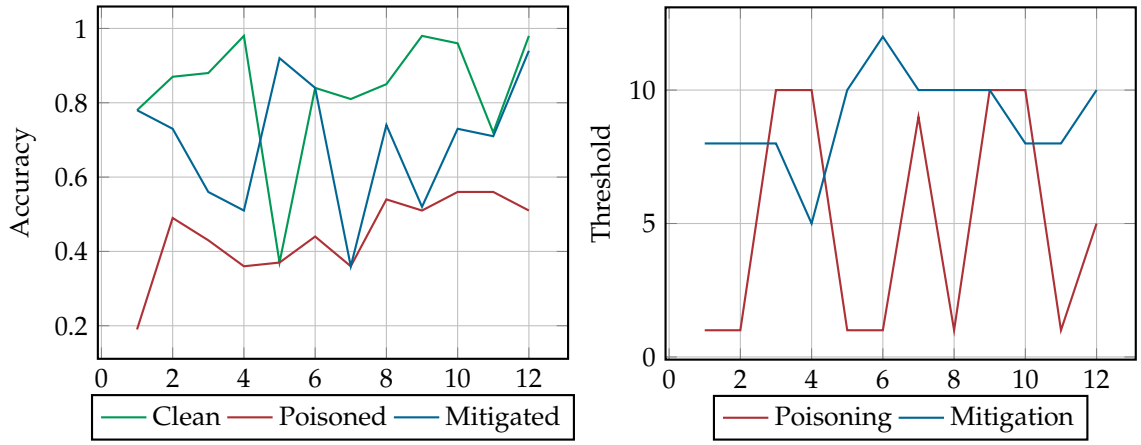


Figure 4. Dynamic Relabeling Results.

4.2.3. Static Addition of New Entries

In the static scenario of the new entries adversarial attack, all malicious entries were detected by both the clean and mitigated classifier. However, the clean classifier has 16% false positives, i.e., original entries that are classified as fake new entries. The mitigated classifier has a much lower false positive, i.e., 2%. The poisoned classifier had 50% false negatives and 2% false positives. As Figure 5 suggests the remediation of this adversarial attack scenario demands a higher threshold compared to the poisoning threshold, namely 5% and 10%. Nonetheless while the poisoning attack achieves half the accuracy of the classifier (i.e., from 81% to 38%), the mitigation plan causes a higher accuracy of the classifier, i.e., 88%. In this case the adversary, using once more minimal computational resources (ranging from 5% and 10%) compared to the ones spent by the defensive mechanism, manages to change the perception of the system about benign and malicious instances. At the same time it significantly increased the resources needed for securing our system against the specific attack.

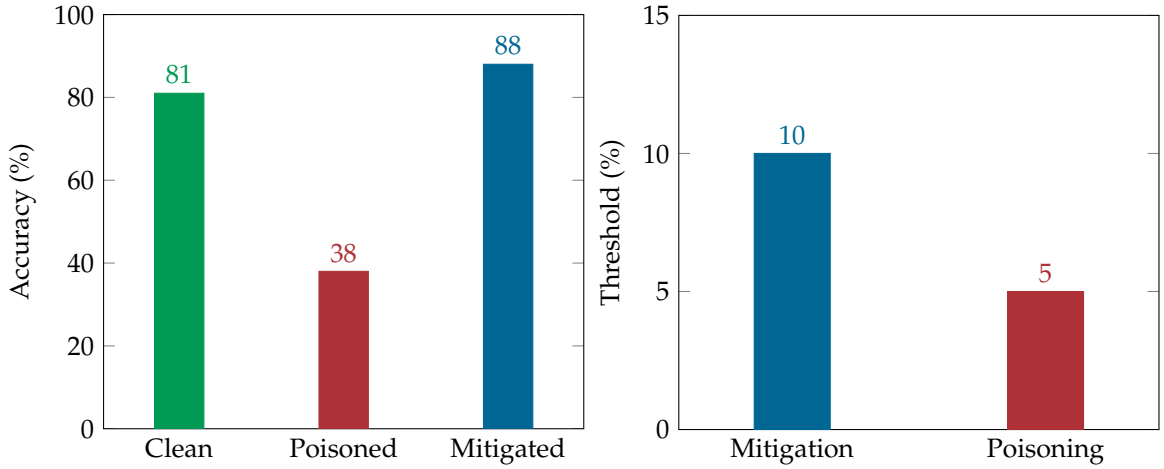


Figure 5. Static New Entries Results.

4.2.4. Dynamic Addition of New Entries

Regarding the dynamic scenario of the new entries adversarial attack, the results considering the accuracy within the 12 frames for the: (i) clean classifier (green line), (ii) poisoned classifier (red line), and (iii) mitigated classifier (blue line) are provided in Figure 6 and Table 2. The average accuracy of the classifier is 67% with a 10% poisoning threshold. This suggests that, assuming that the defender uses this training dataset, this strategy should be the least preferable of the attacker as it achieves the least reduction in the classifier's accuracy.

The results (see Figure 6) revealed steep fluctuations in the accuracy of the clean and mitigation classifier. Initially, the accuracy of the clean and poisoned classifiers are increasing, contrary to the mitigation classifier, which has notable fluctuations. The latter has its lowest accuracy in frame 2, which is lower than the accuracy of the poisoned classifier. In the next two frames (i.e., 4 and 5) the accuracy of both the clean and mitigated classifier drops again, being less than the accuracy of the poisoned classifier. Nonetheless, in frames 6 and 7 their accuracy is increasing significantly, with the accuracy of the mitigated classifier scoring 99%. At the same time, the poisoning attack manages the highest drop in the classifier's accuracy in these two frames, where the classifier has the lowest accuracy in frame 7.

In the last frames, the mitigation classifier drops significantly, reaching 30% in frame 9, and then achieves an increment of 66% in frame 10, where thereafter achieves an almost steady accuracy. In contrast, the clean classifier achieves high accuracies in these frames, except for frame 11, while the poisoned classifier increases its accuracy from 47% in frame 9 to 92% in frame 12.

As summarized in Figure 6, our results indicate a considerable difference in the values of the two thresholds. In frame 2 the mitigation threshold value reaches 25%, but nonetheless the mitigated classifier achieves only 45% accuracy. Moreover, when the poisoning procedure reaches a limit of approximately 86% the threshold increases, successfully reducing the accuracy of the classifier. Similarly, in the last four frames the accuracy of the poisoning classifier increases as the threshold decreases.

The results in frames 2, 5, and 9 suggest that the mitigation strategy fails to mitigate the adversarial attack of this scenario. This holds true as the accuracy of the mitigated classifier is not only lower than the clean classifier, but also lower than the poisoned one. Although both thresholds' values show stability of the detection mechanism in frames 3 to 5 and 7 to 10, they do not correspond to the mitigated classifier's accuracy in frames 5 and 9. In addition, the curve of the poisoned classifier has lower accuracy reducing the classifier's overall accuracy in detecting fake samples. Furthermore, the mitigation classifier is unstable compared to the clean classifier, as suggested by the fluctuations in curve of their accuracy. This suggests that the contribution of the mitigation plan to the classifier's accuracy is not consistent throughout the dynamic new entries scenario. As a result, a different approach or different tuning of the classifier used could be investigated for this particular scenario with the use of imbalanced datasets, derived from this or other datasets. We leave this for our future work.

Table 2. Results of Dynamic New Entries Scenario.

Data Frame	Original Accuracy	Poisoning Accuracy	Poisoning Threshold
1	0.78	0.63	10
2	0.87	0.63	10
3	0.88	0.72	10
4	0.98	0.87	10
5	0.37	0.83	5
6	0.84	0.59	12
7	0.81	0.40	15
8	0.85	0.52	12
9	0.98	0.47	12
10	0.96	0.73	10
11	0.72	0.74	10
12	0.98	0.92	5
Average	0.83	0.67	10.08

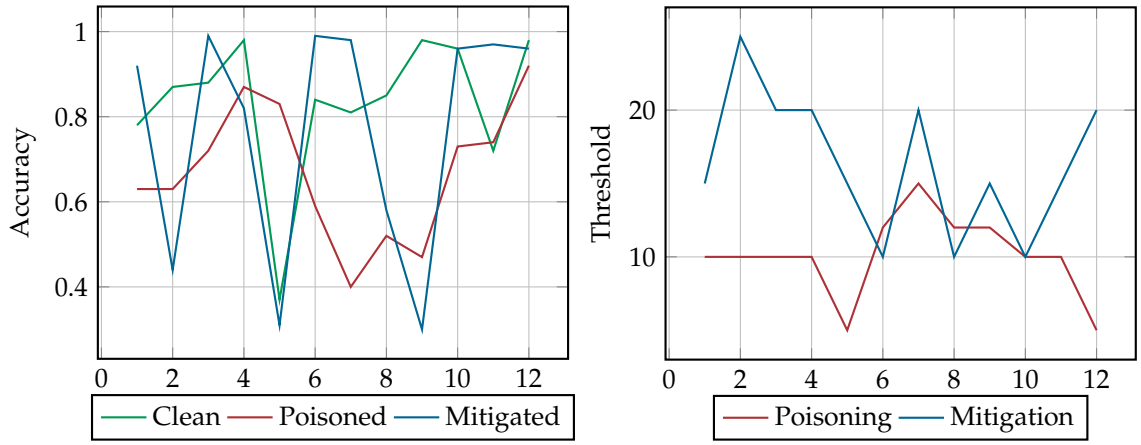


Figure 6. Dynamic New Entries Results.

4.3. Poisoning Detection

Figure 7 summarizes the accuracy of the poisoned and mitigated classifier as well as the mitigation threshold during our experiments. The mitigation approach during the relabeling poisoning attack, in both static and dynamic scenarios, performed well restoring the classifier's accuracy close to its initial values. An interesting fact here is that the mitigated classifier exceeds the accuracy of the clean classifier in the new entries static scenario, upon the execution of the mitigation plan in the new crafted entries. The mitigation performed well in the new entries dynamic scenario as well, even on a smaller scale. Nonetheless, the overall poisoned accuracy was not low enough to show whether the countermeasure is indeed appropriate against this dynamic poisoning attack.

In specific, as summarized in Table 3, in the static scenario the above-mentioned mitigation plan restores the classifier's accuracy back to 81%, after relabeling 10% of the total records. In the dynamic scenario however, the classifier accuracy is restored to 71%, which is 9% below the initial (i.e., before poisoning) accuracy of the classifier, having a mitigation threshold of 8.9%. Table 4 summarizes the results of our experiments, providing details for every frame regarding the mitigation relabeling, classifier's accuracy and mitigation threshold.

The relabeling is propagated based on the non-fake labels changing them into fake with a threshold of 10%. As shown in Table 5 in the new entries static scenario, the results suggest a 5% poisoning threshold, a 10% mitigation threshold, resulting in an 88% classifier's accuracy.

Moreover, in the dynamic new entries scenario the mitigation plan achieved a 10% increase of classifier accuracy compared to the poisoned classifier, which is still less than the initial classifier's accuracy. As summarized in Table 6, in the majority of the frames the mitigation strategy achieves restoring the classifier's accuracy.

Our experiments show, with regards to the mitigation of the adversarial attacks that have been described in this work, that our mitigation approach restores the classifier's accuracy from 10% to 50% showing a 32% average improvement. This comes with the cost of training the classifier, with this time summarized in each scenario in Tables 7–9. It is not surprising that training with the whole dataset is more expensive than training the classifier with smaller data frames.

Table 3. Adversarial Attacks and Mitigation Relabeling.

Scenario	Classifier Ac.	Poisoning Ac.	Poisoning Threshold	Mitigation Ac.	Mitigation Threshold
Static	0.81	0.48	0.07	0.81	0.1
Dynamic	0.83	0.44	0.05	0.71	0.089

Table 4. Dynamic Adversarial Attacks and Mitigation Relabeling.

Data Frame	Original Ac.	Poisoning Ac.	Poisoning Threshold	Mitigation Ac.	Mitigation Threshold
1	0.78	0.19	1	0.78	8
2	0.87	0.49	1	0.73	8
3	0.88	0.43	10	0.56	8
4	0.98	0.36	10	0.51	8
5	0.37	0.37	1	0.92	10
6	0.84	0.44	1	0.92	10
7	0.81	0.36	9	0.36	10
8	0.85	0.54	1	0.74	10
9	0.98	0.51	10	0.52	10
10	0.96	0.56	10	0.73	8
11	0.72	0.56	1	0.71	8
12	0.98	0.51	5	0.94	10
Average	0.83	0.44	5.0	0.71	8.9

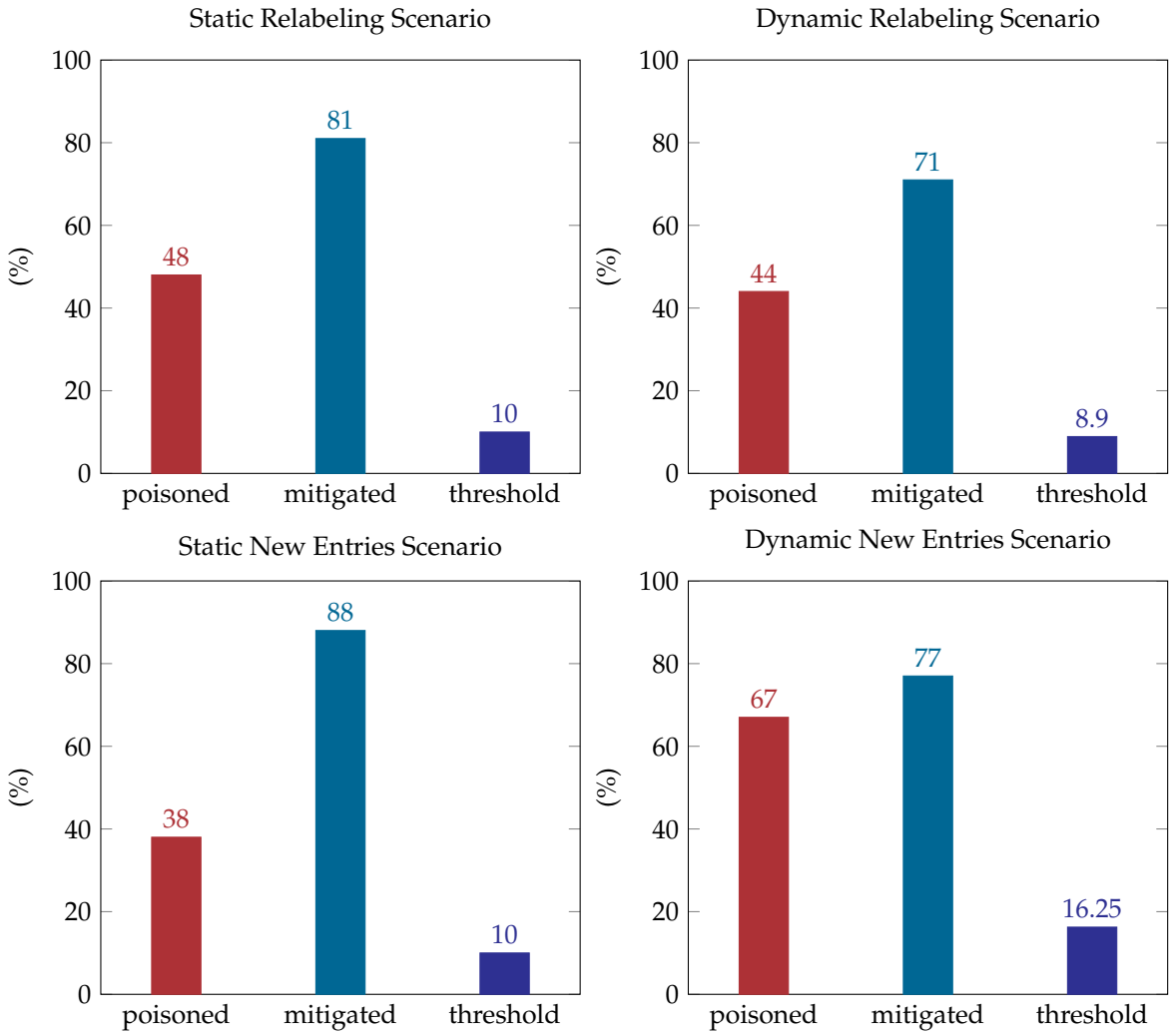


Figure 7. Results.

Table 5. Adversarial Attacks and Mitigation New Entries.

Scenario	Classifier Ac.	Poisoning Ac.	Poisoning Threshold	Mitigation Ac.	Mitigation Threshold
Static	0.81	0.38	0.05	0.88	0.10
Dynamic	0.83	0.67	0.1008	0.77	0.1625

Table 6. Dynamic Adversarial Attacks and Mitigation New Entries.

Data Frame	Original Ac.	Poisoning Ac.	Poisoning Threshold	Mitigation Ac.	Mitigation Threshold
1	0.78	0.63	10	0.92	15
2	0.87	0.63	10	0.44	25
3	0.88	0.72	10	0.99	20
4	0.98	0.87	10	0.82	20
5	0.37	0.83	5	0.31	15
6	0.84	0.59	12	0.99	10
7	0.81	0.40	15	0.98	20
8	0.85	0.52	12	0.58	10
9	0.98	0.47	12	0.30	15
10	0.96	0.73	10	0.96	10
11	0.72	0.74	10	0.97	15
12	0.98	0.92	5	0.96	20
Average	0.83	0.67	10.08	0.77	16.25

Table 7. Summary of Training Time (Seconds) for Proposed Static Adversarial Scenarios.

Scenario	Training Time
Clean Classsifier	624.45
Static Relabeling Poisoned Classifier	515.37
Static Relabeling Mitigated Classifier	514.95
Static New Entries Poisoned Classifier	773.77
Static New Entries Mitigated Classifier	1009.57

Table 8. Classifiers' Training Time (Seconds) for Dynamic New Entries Adversarial Scenario.

Frame	Clean Classifier	Poisoned Classifier	Mitigated Classifier
1	44.64	32.87	43.21
2	47.27	36.66	45.16
3	51.83	37.42	47.13
4	43.09	26.80	38.61
5	37.53	25.69	40.47
6	60.79	34.66	47.17
7	32.77	22.58	26.08
8	25.67	17.63	25.36
9	27.71	24.71	22.84
10	41.50	26.92	34.71
11	49.93	23.92	39.57
12	41.63	27.20	36.70
Total	504.36	337.06	447.01

Table 9. Classifiers' Training Time (Seconds) for Dynamic Relabeling Adversarial Scenario.

Frame	Clean Classifier	Poisoned Classifier	Mitigated Classifier
1	44.64	30.80	32.35
2	47.27	29.84	33.13
3	51.83	31.66	32.79
4	43.09	27.17	33.89
5	37.53	21.64	23.14
6	60.79	31.39	39.65
7	32.77	21.87	25.99
8	25.67	16.39	19.95
9	27.71	14.46	15.24
10	41.50	22.39	25.34
11	49.93	27.97	27.66
12	41.63	30.84	29.87
Total	504.36	306.42	339

5. Conclusions

During the past decade, social media has become a vital service for people commenting on news and significant global events. Even politicians use them to campaign for elections, communicating directly to voters, thus providing a sense to social media users of direct interaction with society's prominent personalities about current affairs. As social media interconnects users across the world, this introduces threats to their security and privacy. This holds true as data for companies and individuals can be found on online social networking platforms, alongside the fact that those platforms are widely unable to distinguish fake and legitimate accounts. The popularity of social media, such as Twitter, allows users to disseminate incorrect information through fake accounts, which results in the spread of malicious or fake content as well as the realization of various methods of cyberattacks. The fact that everyone can create an account and interact with others creates a thin line questioning legitimate use. This has attracted the attention of cybercriminals, who use social media as a vector to perform a variety of actions to serve their objectives, thus threatening users or even national security.

In this paper we have conducted adversarial attacks against a detection engine, which uses machine learning in order to uncover fake accounts on Twitter. To this end, a label flipping poisoning attack strategy was proposed and implemented, effectively impairing the accuracy of the detection engine. Also, a new entries poisoning attack was also proposed and implemented, being almost equally efficient in decreasing the classifier's accuracy. To mitigate these attacks a defense mechanism was recommended based on a k-NN algorithm and relabeling techniques aiming to defend against our poisoning attacks. Our work empirically showed the significant degradation of the classification accuracy as well as the effectiveness of the proposed countermeasure to successfully mitigate the effect of such attacks. While these results are relevant to the classifiers that we have used to implement the defensive mechanisms (i.e., AdaBoost and k-NN), we consider that they are applicable to other boosting algorithms or their respective versions.

More specifically, considering the implication of the adversarial attacks to the detection system, our results have shown that in every scenario (static or dynamic) that was explored herein the poisoning decreases the classifier's accuracy from 13% to 53%, having a 30% reduction on average. Specifically, in the static relabeling attack scenario, the poisoning managed to decrease the accuracy of the classifier under 50% with a low poisoning threshold. Moreover, in the new entries static scenario the accuracy reached a value under 40%, with a lower poisoning threshold. This proved that the attacker will be more successful in impairing the classification accuracy by adding new crafted data

to the training dataset than relabeling the existing entries in the static scenario. On the other hand, in the dynamic scenarios, poisoning the classifier with relabeling impairs the classifier's accuracy more (namely to 47% on average) compared to the new entries scenario, where the poisoning decreases its accuracy only 13% on average with a double poisoning threshold.

Our plans for future work include the evaluation of the adversarial attacks and the mitigation strategy to additional datasets, as they become available. In addition, we will investigate other potential scenarios that could be used by the defender or the attacker (such as adding new entries with fake profiles having features that resemble those of real users or with legitimate users having features that normally appear in entries created by fake users). We also plan to explore the effectiveness of our attacking and defending strategies in other social media datasets. Finally, we plan to investigate the effects of our proposed adversarial attacks using other similar supervised machine learning algorithms.

Author Contributions: All authors contributed to the conceptualization and methodology of the manuscript. P.K. performed the data preparation and assisted N.P. with writing, A.M. and N.K. reviewed and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: The research leading to these results has been partially supported by the H2020 project CARAMEL, (GA 833611).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Polyakova, A.; Boyer, S.P. The future of political warfare: Russia, the West, and the coming age of global digital competition. EUROPE 2018. Available online: http://www.assetallocation.org/resources/Research-Materials/Russia/Russia_Digital_Hybrid_Warfare.pdf (accessed on 30 September 2020).
2. Çıtlak, O.; Dörterler, M.; Doğru, İ.A. A survey on detecting spam accounts on Twitter network. *Soc. Netw. Anal. Min.* **2019**, *9*, 35. [CrossRef]
3. Wu, T.; Wen, S.; Xiang, Y.; Zhou, W. Twitter spam detection: Survey of new approaches and comparative study. *Comput. Secur.* **2018**, *76*, 265–284. [CrossRef]
4. Pitropakis, N.; Panaousis, E.; Giannetsos, T.; Anastasiadis, E.; Loukas, G. A taxonomy and survey of attacks against machine learning. *Comput. Sci. Rev.* **2019**, *34*, 100199. [CrossRef]
5. Pitropakis, N.; Kokot, K.; Gkatzia, D.; Ludwiniak, R.; Mylonas, A.; Kandias, M. Monitoring Users' Behavior: Anti-Immigration Speech Detection on Twitter. *Mach. Learn. Knowl. Extr.* **2020**, *2*, 192–215. [CrossRef]
6. Rao, P.; Kamhoua, C.; Njilla, L.; Kwiat, K. Methods to Detect Cyberthreats on Twitter. In *Surveillance in Action: Technologies for Civilian, Military and Cyber Surveillance*; Springer International Publishing: Cham, Switzerland, 2018; pp. 333–350, [CrossRef]
7. Kejriwal, M.; Gu, Y. A Pipeline for Rapid Post-Crisis Twitter Data Acquisition, Filtering and Visualization. *Technologies* **2019**, *7*, 33. [CrossRef]
8. Venkatesh, R.; Rout, J.K.; Jena, S.K. Malicious Account Detection Based on Short URLs in Twitter. In *Proceedings of the International Conference on Signal, Networks, Computing, and Systems*; Lobiyal, D.K., Mohapatra, D.P., Nagar, A., Sahoo, M.N., Eds.; Springer: New Delhi, India, 2017; pp. 243–251.
9. Stringhini, G.; Kruegel, C.; Vigna, G. Detecting spammers on social networks. In Proceedings of the 26th Annual Computer Security Applications Conference, Austin, TX, USA, 6–10 December 2010; pp. 1–9.
10. Yang, C.; Harkreader, R.; Zhang, J.; Shin, S.; Gu, G. Analyzing Spammers' Social Networks for Fun and Profit: A Case Study of Cyber Criminal Ecosystem on Twitter. In Proceedings of the 21st International Conference on World Wide Web (WWW'12), Lyon, France, 16–20 April 2012; Association for Computing Machinery: New York, NY, USA, 2012; pp. 71–80, [CrossRef]
11. Liu, Y.; Wu, B.; Wang, B.; Li, G. SDHM: A hybrid model for spammer detection in Weibo. In Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014), Beijing, China, 17–20 August 2014; pp. 942–947.
12. Tkachenko, R.; Izonin, I.; Kryvinska, N.; Dronyuk, I.; Zub, K. An Approach towards Increasing Prediction Accuracy for the Recovery of Missing IoT Data based on the GRNN-SGTM Ensemble. *Sensors* **2020**, *20*, 2625. [CrossRef] [PubMed]

13. ElAzab, A. Fake accounts detection in twitter based on minimum weighted feature. *Int. J. Comput. Inf. Eng.* **2016**, *10*, 13–18.
14. Tkachenko, R.; Duriagina, Z.; Lemishka, I.; Izonin, I.; Trostianchyn, A. Development of machine learning method of titanium alloy properties identification in additive technologies. *East.-Eur. J. Enterp. Technol.* **2018**, *3*, 23–31. [CrossRef]
15. Hörtenhuemer, C.; Zangerle, E. A Multi-Aspect Classification Ensemble Approach for Profiling Fake News Spreaders on Twitter. In Proceedings of the International Conference and Labs of the Evaluation Forum (CLEF), Thessaloniki, Greece, 22–25 September 2020.
16. Izonin, I.; Kryvinska, N.; Tkachenko, R.; Zub, K. An approach towards missing data recovery within IoT smart system. *Procedia Comput. Sci.* **2019**, *155*, 11–18. [CrossRef]
17. Miller, Z.; Dickinson, B.; Deitrick, W.; Hu, W.; Wang, A.H. Twitter spammer detection using data stream clustering. *Inf. Sci.* **2014**, *260*, 64–73. [CrossRef]
18. Song, J.; Lee, S.; Kim, J. Spam Filtering in Twitter Using Sender-Receiver Relationship. In *Recent Advances in Intrusion Detection*; Sommer, R., Balzarotti, D., Maier, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 301–317.
19. Oujezsky, V.; Horvath, T. Traffic similarity observation using a genetic algorithm and clustering. *Technologies* **2018**, *6*, 103. [CrossRef]
20. Im, J.; Chandrasekharan, E.; Sargent, J.; Lighthammer, P.; Denby, T.; Bhargava, A.; Hemphill, L.; Jurgens, D.; Gilbert, E. Still out There: Modeling and Identifying Russian Troll Accounts on Twitter. In *12th ACM Conference on Web Science (WebSci'20)*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1–10. [CrossRef]
21. Imam, N.H.; Vassilakis, V.G. A Survey of Attacks Against Twitter Spam Detectors in an Adversarial Environment. *Robotics* **2019**, *8*, 50. [CrossRef]
22. Shafahi, A.; Huang, W.R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; Goldstein, T. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Advances in Neural Information Processing Systems 31*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2018; pp. 6103–6113.
23. Wang, Y.; Chaudhuri, K. Data Poisoning Attacks against Online Learning. *arXiv* **2018**, arXiv:1808.08994.
24. Brendel, W.; Rauber, J.; Bethge, M. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. *arXiv* **2017**, arXiv:1712.04248.
25. Zhang, J.; Chen, J.; Wu, D.; Chen, B.; Yu, S. Poisoning Attack in Federated Learning using Generative Adversarial Nets. In Proceedings of the 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, 5–8 August 2019; pp. 374–380.
26. Chen, S.; Xue, M.; Fan, L.; Hao, S.; Xu, L.; Zhu, H.; Li, B. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. *Comput. Secur.* **2018**, *73*, 326–344. [CrossRef]
27. Yu, S.; Vorobeychik, Y.; Alfeld, S. Adversarial Classification on Social Networks. *arXiv* **2018**, arXiv:1801.08159.
28. Laishram, R.; Phoha, V.V. Curie: A method for protecting SVM Classifier from Poisoning Attack. *arXiv* **2016**, arXiv:1606.01584.
29. Biggio, B.; Corona, I.; Fumera, G.; Giacinto, G.; Roli, F. Bagging Classifiers for Fighting Poisoning Attacks in Adversarial Classification Tasks. In *Multiple Classifier Systems*; Sansone, C., Kittler, J., Roli, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 350–359.
30. Paudice, A.; Muñoz-González, L.; Lupu, E.C. Label Sanitization Against Label Flipping Poisoning Attacks. In *ECML PKDD 2018 Workshops*; Alzate, C., Monreale, A., Assem, H., Bifet, A., Buda, T.S., Caglayan, B., Drury, B., García-Martín, E., Gavaldà, R., Koprinska, I., et al., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 5–15.
31. Boatwright, B.C.; Linvill, D.L.; Warren, P.L. Troll factories: The internet research agency and state-sponsored agenda building. *Resour. Cent. Media Freedom Eur.* **2018**. Available online: https://www.rcmediafreedom.eu/Publications/Academic-sources/Troll-Factories-The-Internet-Research-Agency-and-State-Sponsored-Agenda-Building (accessed on 30 September 2020).

32. Wyner, A.J.; Olson, M.; Bleich, J.; Mease, D. Explaining the success of adaboost and random forests as interpolating classifiers. *J. Mach. Learn. Res.* **2017**, *18*, 1558–1590.
33. Roeder, O. Why We're Sharing 3 Million Russian Troll Tweets. FiveThirtyEight. 2018. Available online: https://fivethirtyeight.com/features/why-were-sharing-3-million-russian-troll-tweets/ (accessed on 30 September 2020).