TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Kristopher Ryan Price 184562IVCM

Analysis of the Impact of Poisoned Data within Twitter Classification Models

Master's Thesis

Supervisors: Sven Nomm
Ph.D., Applied Mathematics

Jaan Priisalu
MSc., Automatics

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristopher Ryan Price 184562IVCM

MÜRGITATUD ANDMETE MÕJU ANALÜÜS TWITTERI KLASSIFIKATSIOONIMUDELITELE

Magistritöö

Juhendaja: Sven Nomm

Ph.D., Rakenduslik Matemaatika

Jaan Priisalu

MSc., Automaatika

Tallinn 2018

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Kristopher Ryan Price

11.05.2019

Abstract

Many online communities today face growing problems of group polarization, radicalization, and fake news. These issues are being exacerbated by the phenomenon of bots – automated accounts that are becoming less and less distinguishable from real people. While methods exist to detect these bots, they are not perfect and may be vulnerable to manipulation. The field of data-science concerned with how this might be done is known as adversarial machine-learning. In the past researchers have drawn out frameworks for various adversarial machine-learning attacks involving altering the data that models are trained and tested on. However, there has been little research on applying these techniques to models for detecting automated social-media accounts, specifically on Twitter. Research to date has focused on how bots have changed their appearance or behavior in order to avoid detection. These kinds of 'evasion attacks' involve altering the data (the Twitter accounts) after a model has been trained. This research focuses on how attackers might carry out a 'poisoning attack' – altering the data before a model has been trained, with the goal of reducing that model's accuracy in correctly identifying bots on Twitter. The results show that by introducing mislabeled data-points into a such a model's training data, attackers can reduce its accuracy by up to twenty percent. In spite of these positive results, this research is still limited by the data available for testing and the attack methods capable of being tested on it. It is hoped that this paper will raise awareness of the potential for such data manipulation and encourage future studies to build off it, with the ultimate goal of mitigating bots and their ability to influence people on social-media.

This thesis is written in English and is 72 pages long, including 6 chapters, 20 figures and 10 tables.

Annotatsioon

Mürgitatud andmete mõjuanalüüs Twitteri andmete turvaliigituse mudelite põhjal

Täna seisavad paljud online-kommuunid vastamisi järgnevate probleemidega : gruppide polariseerumine, radikaliseerumine ja võltsuudised. Välja toodud probleeme võimendab võrgurobotite fenomen - automaatika põhiselt käituvad kontod, mis muutuvad järjest vähem äratuntavaks võrreldes pärisinimestega (päriskasutajatega). Võrgurobotite avastamiseks on olemas meetodid, mis aga ei ole täiuslikud ja mis võivad olla kaitsetud erinevate manipulatsioonide eest.

Andmeteaduse uurimisvaldkond, mis käsitleb selliste meetodite väljatöötamist on tuntud kui vastaseõpe. Minevikus on uurijad loonud mitmeid raamistikke erinevate vastaseõppe rünnete kohta, mis põhinevad andmemudelite muutmistega, mille peal masinõppe andmemudelid on treenitud ja testitud. Vähe on sarnaseid uurimustöid tehtud selle kohta, kuidas neid raamistikke ja tehnikaid rakendada mudelite peal, mis on mõeldud automaatsete sotsiaalmeedia kontode tuvastamiseks, eriti Twitteri sotsiaalvõrgustiku kohta. Uurimustööd, mis on olemas, on keskendunud sellele, kuidas võrgurobotid on muutnud oma väljanägemist (visuaalset poolt) või käitumist, eesmärgiga vältida avastamist eelpool mainitud andmemudelite poolt. Sellised põikeründed põhinevad sisendandmete muutmises (selles kontekstis Twitteri kontod), peale seda, kui masinõppe andmemudelit on juba treenitud.

Käesolev uurimustöö keskendub sellele, kuidas ründajad võivad kasutada mürgitamisrünnet - ehk muuta andmeid enne, kui andmemudel on välja treenitud, eesmärgiga, vähendada selle andmemudeli täpsust, kui hästi ta suudab Twitteri võrguroboteid identifitseerida. Uurimustöö tulemused näitavad, et tutvustades andmemudeli treenimissüsteemi andmetesse valesti markeeritud andmepunkte, suudab ründaja vähendada selle andmemudeli täpsust kuni 20 protsenti. Sellistest positiivsetest tulemutest sõltumatult, on see uurimustöö limiteeritud kättesaadava testitava andmestiku tõttu ja ründemeetodite jaoks vajaoleva riistvara ja tarkvara millega selliseid testitavad andmestikke rünnata puudumise tõttu,. Autor loodab, et see uurimustöö tõstab

teadlikkust selliste andmete manipulatsiooni võimalikkuse ja ohtude eest ning julgustab tulevikus tehtavaid töid olemasolevat aluseks võtma ja seda edasi arendama. Autori enda peamine eesmärk on võidelda vastu võrgurobotite olemasolule ja nende võimele mõjutada inimesi online sotsialmeedias.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 72 leheküljel, 6 peatükki, 20 joonist, 10 tabelit.

List of abbreviations and terms

Misinformation	False or misleading information that may be partially wrong or taken out of context
Disinformation	False information intentionally spread to deceive people
Fake News	Misinformation or Disinformation that mimics actual news media content in form but not in organizational process or intent
Selective Exposure	Phenomenon where people tend to prefer information that is consistent with their pre-existing attitudes
Confirmation Bias	Phenomenon where people view information that confirms their pre-existing beliefs as more persuasive than dissonant information that does not confirm them
Desirability Bias	Phenomenon where people are inclined to accept information that pleases them
Group Polarization	Phenomenon where biases within a group of like-minded people are shifted in an extreme direction due to Selective Exposure, Confirmation Bias, and Desirability Bias
Supervised Machine Learning	A type of machine-learning in which an algorithm generates a model based on labeled ‘training’ data which classifies unlabeled ‘test’ data.
Adversarial Machine Learning	A field of data-science concerned with how machine-learning algorithms may be vulnerable to deception and manipulation in an uncontrolled setting.
Evasion Attack	A type of adversarial machine-learning where the attacker modifies the test data to increase a model’s error rate
Poisoning Attack	A type of adversarial machine-learning where the attacker modifies the training data to increase a model’s error rate
KNN	Short for K-Nearest Neighbor. An algorithm in which an object is classified based on its position relative to a majority of some number K of its closest neighbors [citation-needed]
GLM	Short for Generalized Linear Model; predicts the value of a class based on a linear combination of observed values; allows predicted values to vary linearly with observed values
SVM	Short for Support Vector Machine; a type of supervised-learning algorithm that is trained to sort data into two categories with margins between the two as wide as possible

Social-Bot	An automated social-media account that effectively mimics the behavior of a real human user
Spambot	An automated social-media account that frequently tweets or retweets spam
Fake-Follower	A social-media account that exists to artificially inflate a user's follower-count and make them appear more popular
Favourites_count	The number of tweets a Twitter user has liked in their account's lifetime
Friends_count	A variable in the Twitter account data that denotes how many Twitter accounts a user is following or 'friends with'.
Followers_count	The number of Twitter accounts that are following a user
Statuses_count	The number of tweets (including retweets) issued by the user of a Twitter account
Listed_count	The number of public lists a user is a member of
Geo_enabled	When true, indicates the user has enabled the possibility of geotagging their tweets
Protected	Indicates whether or not a user has chosen to protect their tweets. Protected tweets can only be seen by a user's followers.
Verified	Indicates whether or not a user has a verified account. Verified accounts are determined to be associated with public individuals, ie. diplomats, journalists, actors, etc. This helps distinguish authentic accounts from ones merely using that individual's name.
Follow_request_sent	Indicates the authenticating user has sent a follow request to this account in order to see their tweets
Following	Indicates the authenticating user is following this account. This feature has 100% correlation with Follow_request_sent in the Cresci-2017 data-set.
Notifications	If true, indicates that the authenticating user has chosen to receive Tweets from this account by SMS. This feature has 100% correlation with Follow_request_sent in the Cresci-2017 data-set.
Contributors_enabled	If true, allows for Tweets issued by the user to be co-authored by another account. This feature has 100% correlation with Follow_request_sent in the Cresci-2017 data-set.
Geo_follow_protect_verify	A feature created to represent eight different combinations of values for four different binary features – Geo_enabled, Follow_request_sent, Protected, and Verified
Lang	The language the user's account is in. This feature only pertains to the account's user interface – the Tweets may be in a different language.

Time_zone	The time zone that a user declares themselves to be in. This feature is no longer available in the Twitter API due to GDPR compliance.
Utc_offset	Number of hours subtracted from or added to Coordinated Universal Time for the user's time zone This feature is no longer available in the Twitter API due to GDPR compliance.
Created_at	UTC Date and time a Twitter account was created at
Authenticating user	The user account that a Twitter API request is sent from. These API requests may be for automating an account or collecting data on other, public accounts.
Mahalanobis Distance	Measures the distance between a given data-point and the mean of every other point in the data-set using the values of two given features

Table of contents

Contents

List of figures

List of tables

1 Introduction

Today, many people communicate through social media. This medium has become a ubiquitous part of society that is used by many to read the news, contact friends, and plan events. Social networks such as Facebook tend to recommend users to connect with people already within their social circle rather than with complete strangers. While on the surface this seems intuitive and convenient, this feature leads to a systemic problem in social-media, the effects of which are just now beginning to have effects in real-life spaces. The social structure resulting from this feature, where everyone knows one another and shares similar beliefs, can create a homogeneous social network. In such a setting, people's tolerance of alternate views and their willingness to accept information that contradicts what they already believe are reduced. This is referred to as selective exposure. Research has shown that people find information that agrees with what they already think more persuasive, and that they are more inclined to accept information that pleases them. These are referred to as confirmation bias and desirability bias, respectively. According to David M.J. Lazer, it is within this context that fake news has been able to find a mass audience [1].

It is important to clarify what is meant when discussing 'fake news' in the context of this paper. Originally the term was referred to misleading information or news stories spread online. However, in recent years many political figures have begun using the term to dismiss or cast doubt on stories that do not support their positions. Because of this, many researchers and media companies have come to believe that the term has lost all academic meaning. These people have instead decided to use the term 'false news' [2]. However, the term 'fake news' is far more familiar to the average person and helps draw attention to the political factors behind its use. This word will therefore be used throughout this paper as an all-encompassing term for misinformation or disinformation that "mimics actual news media content in form but not in organizational process or intent". Misinformation is defined as false or misleading information that may be partially wrong or taken out of context. Disinformation is defined as "false information intentionally spread to deceive people". These definitions are taken from the research overview "The Science of Fake News", written by David M.J. Lazer [1].

Thanks to selective exposure, confirmation bias, and desirability bias, fake news not only confirms but also reinforces and strengthens people's beliefs. According to J. Ratkiewicz et al., "when politically active individuals can avoid people and information they would not have chosen in advance, their opinions are likely to become increasingly extreme as a result of being exposed to more homogeneous viewpoints and fewer credible opposing opinions" [3]. Political researcher Cass Sunstein refers to this phenomenon, where existing biases in a group are shifted in an extreme direction, as group polarization [4]. There can and have been grave consequences to this kind of group polarization. Research has shown that there is a strong link between the spread of fake news stories and the radicalization of terrorists [5]. Examples of terrorist acts that resulted from online radicalization include the Christchurch mosque shooting in March 2019 as well as the 'Pizza-gate' shooting in Washington, D.C. in December 2016 [6], [7].

Worryingly, research suggests that polarizing fake news can be spread faster by social bots. In one experiment, researcher Luca Maria Aiello created a bot in an online forum. According to Aiello, "our bot was widely mistaken as a human user and … triggered the polarization of opinions of community members". Aiello believes that because of how often social bots post content they may be viewed as more trustworthy and be better at influencing groups of people [8]. This is supported in a more recent study by Shao et. al., in which the authors analyzed 14 million messages spreading 400 articles on Twitter from 2016 to 2017. The authors found that bots played a large role in amplifying the presences low-credibility content shortly before it went 'viral' [9].

A large part of this problem is the increasing difficulty online users have in distinguishing bots and people. Many data-scientists have dealt with this by using machine-learning to detect social-media bots. However, there is a dearth of research on the opposite: how bots might avoid detection. According to Zhouhan et. al., there is an ongoing virtual arms race between creators of malicious bots on Twitter and the detection systems used by Twitter and data scientists. Many methods for detecting bots have become less effective as bot-programmers catch on and refactor them [10]. Rather than study how the behavior of bots may change over time, as Cresci et. al. did in "The Paradigm Shift of Social Spambots", the goal of this research is to ascertain how effective a poisoning attack may be on reducing the effectiveness of existing methodologies in detecting bots on Twitter [11].

2 Background

2.1 Using Machine Learning to detect Bots on Twitter

When classifying social media accounts as genuine or automated, many data scientists use supervised machine learning algorithms. These algorithms are trained on data about social media accounts that have already been classified. By analyzing which account features strongly correlate with how social media accounts have been categorized in the past, these algorithms can predict how social media accounts in the future may be classified. For example, say that a model is given training data from several hundred Twitter user accounts. These accounts are all given a label of either 'bot', or 'human'. After being given the training data, the machine-learning algorithms 'learn' that accounts with a high number of tweets are much more likely to be labelled as 'bots'. These algorithms then create a predictive model based on this, which scientists run on a test data-set. The predictive accuracy of these models is given by the ratio of test data that it correctly labels as 'bot' or 'human' to the data it misclassifies. There are many different algorithms that scientists used to distinguish between genuine and automated accounts, several of which served as a foundation for the methodologies used in this thesis.

In their paper "Don't @ Me", researchers Jordan Wright and Olabode Anise designed a supervised machine-learning algorithm that could classify individual accounts on Twitter as bots. Using Twitter's streaming API, Wright and Anise were able to collect data on 19 million different accounts. Their data consisted of the last 200 tweets of each account. The researchers noted that bot accounts tended to have a higher average number of active hours per day, and that they may frequently change their screen-name, geolocation, and profile image to try to avoid detection. They found that a high number of bots were only active in liking or following other accounts, to make them look legitimate. The researchers dubbed these accounts 'amplification-bots'. Other researchers have instead used the term 'fake-followers' to describe these bots [11]. By following reply-threads and chains of bots following other accounts, the researchers ultimately discovered a massive botnet of Twitter accounts that were scamming users into buying fake crypto-currencies [12].

Efthimion et. al. also created their own supervised machine learning algorithm for classifying bots on Twitter. Their algorithm was trained using the Cresci-2017 dataset. They also trained their algorithm with an archive of 200,000 tweets and 400 user profiles confirmed to have engaged in malicious activity to influence the 2016 US Election [13]. One way that bots may avoid detection is by posting the same messages repeatably, but slightly reworded. Efthimion et. al. measured the Levenshtein distance between the text of different tweets made by the same account to determine how similar they were. Their algorithm had a 96.81% success rate in identifying social bots. For their data, the best indicators that an account was a human were if it had geo-location enabled and had less than 30 followers. The best indicators that an account was a bot were if the language was set to English, it didn't have a profile picture, and it followed over 1000 people. A very similar 97.13% success rate was achieved in identifying traditional spambots based on similar factors. The algorithm was 100% successful at identifying amplification bots, with heaviest weighted factors being if the account had a profile picture, had at least 30 followers, 50 tweets, and twice as many followers as friends. There was a high success rate of 99.87% at identifying bots from the NBC dataset, with the most important indicator again being whether or not an account had a profile picture [14].

The studies outlined above involve supervised-machine learning. However, a supervised machine-learning algorithm is limited by the data used to train it. Wright and Anise's algorithm, for example, was trained on crypto-currency Twitter profiles, and was less accurate at classifying non-crypto bots [12]. In contrast, Zhouhan et. al. presented an unsupervised learning algorithm which did not require any training data and was not biased toward any language, keyword, or topic. Their approach was focused on detecting bots based on duplicate content over time and the use of shortened URL's, such as TinyURL. Zhouhan et. al. used the supervised algorithm BotOrNot and the unsupervised algorithm DeBot as benchmarks. BotOrNot focused on identifying individual bots, but it misclassified 40% of the bots identified by the researcher's own algorithm. The unsupervised learning algorithm DeBot only had a 7% overlap of identified bots with Zhouhan et. al.'s algorithm. DeBot mostly identified news-organization bots, based on the synchronicity of Twitter posts. While their algorithm was able to identify more diverse groups of bots, it failed to detect more sophisticated bots, especially on the individual level [10].

Freitas et. al.'s motivation was to see what kinds of social bots were most likely to succeed on Twitter. To that end, they created 120 social-bot accounts on Twitter. These bots were implemented using the open-source 'Real-boy' project and used one of 12 distinct IP addresses. They were each given a custom profile, including a screen-name, biography, and background image. The bots initially followed a pre-selected set of Twitter profiles. To avoid following real spam-bots, these social bots only followed back users who had a friend-to-follower ratio of 1:2. These bots either reposted or retweeted already existing content, or synthetically generated tweets using a Markov generator. At the end of the study, Twitter suspended 38 of the 120 social-bots. Most of these were the last ones that had been created, likely because by that point, Twitter's systems had become suspicious of several IP addresses used. Bots that tweeted synthetically-generated text were also more likely to be detected, whereas bots that retweeted or reposted content were more successful [15].

2.2 Adversarial Machine Learning

Adversarial Machine Learning is the idea that machine-learning algorithms that are tested and trained in a controlled environment may not perform as well 'in the wild' and may in fact be more liable to deception and manipulation. Researchers Xiao et. al. developed a framework for describing this type of setting. This framework consists of two main types of attacks: evasion and poisoning. During an evasion attack, attackers try to cause the algorithms to misclassify data after they have already been deployed [16].

One example of this kind of attack was demonstrated by researchers Hosseini et. al. against the Perspective API. The Perspective API is a project recently started by Google and Jigsaw API that uses machine learning to detect bullying, harassment, and abusive speech. This API assigns a toxicity score to strings of text. Perspective was trained by taking millions of comments from different publishers and asking panels of ten people to rate the comments on a scale from 'very toxic' to 'very healthy'. The researchers tested this API by modifying the example comments used to demonstrate it and measuring how different modifications altered the toxicity score. They showed that an adversary can subtly alter a toxic phrase so that the API will assign a lower toxicity score to it. For example, the comment "*Climate change is happening and it's not*

changing in our favor. If you think differently, you're an idiot." has a toxicity score of 84%, but the comment "*Climate change is happening and it's not changing in our favor. If you think differently, you're an idiiot."* has a score of 20%. Through repeating this pattern of modifying specific words, Hosseini et. al. observed that the Perspective API wrongly assigns high toxicity scores to apparently benign phrases. For example, the phrase "*they are not stupid and wrong*" receives a score of 83%. They also found that the API tended to give higher scores to misspelled words. Hosseini et. al. concluded that to improve the Perspective API and make it less susceptible to adversarial machine learning, it should be trained on modified versions of toxic words, as well as implement a spell-checker [17].

Xiao et. al. described a poisoning attack as a type of adversarial machine learning where adversaries are aware of what kind of data is being collected and are able to introduce maliciously crafted samples. Xiao et. al. simulated a poisoning attack on algorithms meant to detect malware in PDFs, and found that by introducing only a few malicious data-points, they were able to increase the algorithm's misclassification rate by ten times [16].

Another example of how data may be poisoned was provided by researchers Morstatter et. al. Their research suggests that Twitter's Sample API that many data scientists use to stream tweets and collect data may also be vulnerable to manipulation. Twitter's Sample API returns a pseudo-random 1% sample of all public tweets. Each tweet has a unique ID number, which contains a millisecond level timestamp for when it was generated. The Sample API selects all tweets with a timestamp between 657 and 666 milliseconds. While a human user would be unable to time their tweet to the exact millisecond, a bot would.

Morstatter et. al. designed an experiment to see if the millisecond timestamp of a tweet can be predicted. They sent tweets every ten milliseconds and observed the resulting timestamps. After repeating this process 6 times, they found a strong linear correlation between the time the tweet was sent and the tweet's millisecond timestamp. Based on this evidence, they were able to design an algorithm for a Twitter Bot to time their tweets to maximize the number that appeared in the Sample API. Attackers may try to abuse this to introduce their own poisoned samples to a classification algorithm at training time [8].

Research has been done, however, in detecting these kinds of attacks so that data can be pre-filtered. In 2018 Paudice et. al. proposed a defense mechanism to mitigate optimal poisoning attacks based on outlier detection. Paudice et. al. tested multiple outlier detection algorithms, based on distance between datapoints and distribution. In their methodology, they first curate a small fraction of trusted datapoints. These are split into several classes, which are used to train different outlier detectors. They computed the threshold to detect outliers based on the *outlier-ness score* of each trained algorithm using the *Empirical Cumulative Distribution Function* (ECDF). They then collect a new untrusted dataset and remove all samples with an outlier-ness score over the threshold. It should be noted that this methodology fails if the data used to train the outlier detectors is also poisoned.

This methodology was tested on the Spam-base dataset, consisting of multiple spam emails. They showed that with 20% poisoned samples and without outlier detection, the machine learning classification error increases from 0.112 to 0.195. After pre-filtering the data with Outlier detection, however, the classification error went back down to 0.112. Similar results were achieved when testing outlier detection defense on the MNIST dataset, which is used for training algorithms to recognize handwritten digits. When no defense was applied, the classification error is 0.391, but when defense was applied, the error drops to 0.07 [19]. Based on these results, the potential use of outlier filters to remove poisoned data is promising.

3 Methodology

This research is focused on experimenting with and determining the effectiveness of several poisoning attack methods. This research mainly makes use of the Cresci-2017 dataset, consisting of 'genuine' accounts as well as 'social-spambots' and 'traditional spambots' [11], [20]. This data is publicly available in the Bot Repository, an online data repository maintained by the Network Science Institute of Indiana University [21]. The analysis of the Cresci-2017 data as well as the training and testing of models based on this data is performed using R-Studio.

Initial analysis of the data consists of establishing a ground-truth for the accuracy, precision, recall, and F1 Score of models trained on the Cresci-2017 data. These same metrics will be measured after several different poisoning attacks, in order to gauge their effectiveness. This process is repeated for models trained by several different algorithms, including K-Nearest-Neighbor, Generalized Linear Models, and Support Vector Machines. One method for mitigating a poisoning attack that has been proposed is using outlier detection algorithms to filter out poisoned samples [19]. After measuring the impact of several different poisoning attacks, the effectiveness of a distance-based outlier-detector in mitigating these attacks is tested. The most optimal attack method is determined based on the results of these methods, and is analyzed to determine how it would be implemented in a real-life scenario. These methods are explained in more detail below.

3.1 Data Preparation

The Cresci-2017 data-set was initially collected from eight different sources. Three of these sources included a social media campaign for an Italian political candidate, promotions for a mobile phone app, and advertisements for products on Amazon. In this research, the accounts from these sources were joined into a single group labeled 'social-bot'. Accounts labeled by Cresci et. al. as "traditional spambots" were collected from four different sources. The first set of spambots was used as a training-set in research by Yang et. al. [22]. The second source was accounts the spammed scam URLs in 2014. The third and fourth source for these kinds of bots were automated accounts spamming job offers in 2013 and 2009, respectively. In this study, the spambots from these sources were joined into a single group labeled 'spambot'. In their paper "Fame for Sale", Cresci et. al. studied in-depth the phenomenon of fake-followers on Twitter. They used three of the online markets covered in this study - *fastfollowerz.com*, *intertwitter.com*, and *twittertechnology.com* - to buy 3351 fake accounts [23]. For this research, these accounts are also labelled as 'fake-followers'. Cresci et. al. obtained a set of accounts confirmed to be human by randomly contacting 3474 different accounts. These accounts have been labeled 'genuine' [11].

For this study, models are trained to distinguish between each different type of bot (spambot, social-bot, fake-follower) and genuine accounts. Because of this, a common

set of features is required. There were initially 37 different features within the Cresci-2017 data-set. Before determining which features to use, the accounts were labeled and merged into a single file. The data was then further prepared by removing several features deemed unimportant or that might cause the models to become overfit.

Of the initial 37 features in the Cresci-2017 data-set, twelve were binary, having values of true or false. Six of these binary features mostly had null values, neither positive or negative. The binary features *following, notifications, contributors_enabled,* and *follow_request_sent* were found to have 100% correlation with each-other – meaning only one of them was needed to obtain information on the other three. Because of this, most of these binary features were dropped. The three remaining binary features were *follow_request_sent, geo_enabled, protected,* and *verified*.

A dummy-variable was created to collectively represent the four remaining binary features. The true (1) or false (0) value for each variable was multiplied by a power of ten until all four could be added up into a four-digit number of zeroes and ones. This dummy-variable was named *geo_follow_protect_verify*. Two examples of how the value for this feature is obtained are shown in Table 1.

Table 1. Examples of Creating Dummy-Variable.

geo_enabled	follow_request_sent	protected	verified	geo_follow_protect_verify
1	1	0	0	1100
0	1	0	1	0101

Features that are unique to every Twitter account, such as *id* or *screen_name,* and features that have thousands of unique values, such as *url* or *profile_banner*, can cause the models to become overfit. This means that models will be trained to recognize specific Twitter accounts as genuine or automated, and may not be so effective when looking at a different set of accounts. Such features were also dropped from the data-set.

This leaves ten features to determine one class. The ten remaining features are *statuses_count*, *followers_count*, *friends_count*, *favourites_count*, *listed_count*, *lang*, *time_zone,* the dummy-variable *geo_follow_protect_verify*, *utc_offset,* and *created_at*. As of May 23rd, 2018, the *time_zone* and *utc_offset* features are no longer available through the Twitter API user endpoint, in order to be GDPR compliant [24]. More details

about each of these features are given in the list of abbreviations and terms, and are taken from Twitter's online documentation [25], [26], [27].

3.2 Initial Data Analysis

Before experimenting with poisoning the data, an initial data analysis must be performed. This means establishing baselines for how accurately the class of each account (Genuine, Spambot, Social-bot, Fake-follower) can be predicted when the data is not poisoned. To do this, the Cresci-2017 dataset was imported and split into training and testing subsets. These were used to train and test three different models – one based on a Generalized Linear Model, one based on K-Nearest-Neighbor, and one based on a Support Vector Machine. After training these models, a confusion matrix was used to check their accuracy, recall, precision, and F1 Score. This process was repeated several times to obtain the metrics for the models in distinguishing genuine human operated Twitter accounts from social-bots, fake-followers, and spambots, respectively. The effectiveness of the experimental poisoning attacks is measured against these baselines.

When designing an optimal poisoning attack, the features with the most impact on the classification of an account must be considered. Knowing what these features are is also useful when filtering outliers to try to mitigate a poisoning attack. To this end, several methods were used to determine which features most heavily influenced the classification of an account. The first method was simply obtaining the Fisher scores for every feature. Features with a higher Fisher score tend to more heavily predict the classification outcome of a model. The second method was measuring the impact on the F1-Score when a feature was excluded from data when training and testing a model. Features were ranked from most to least important based on which exclusions lowered the F1-Score the most. The average rank for each feature was calculated based on its order placement in both tables. A series of decision trees were used to visualize the decision boundary for the top two features between Genuine accounts and Spambots, Social-bots, and Fake-followers.

3.3 Data Poisoning

After successfully establishing baselines to measure against, the next step is to measure the effectiveness of a poisoning attack. Several different methods of poisoning the data were used, based on previous research.

3.3.1 Label-Flipping

The first and simplest method used was a label-flipping attack based on research by Biggio et. al. [28], [29]. In the label-flipping attack, the data was poisoned by changing the class of a random row from *bot* to *genuine* or vice versa. Random samples of N% of rows were taken from the Cresci-2017 data-set, from N=1 to N=20. These rows were copied and had their labels flipped before being inserted back into the data-set. The Support Vector Machine (SVM) , Generalized Linear Model (GLM), and K-Nearest Neighbor (KNN) algorithms were used to fit this poisoned data to several models, with a cross-validation of ten. After generating these models, the accuracy, recall, precision, and F1-score were measured. In order to avoid bias toward a specific set of random samples, this process was repeated three times with different random samples taken to train and poison the data.

3.3.2 Feature Poisoning

While a simple label flipping attack may be effective in raising the classification error of the data, it may be more optimal to poison the data by altering the values of one of the features. The effectiveness of this method will be tested by selecting one feature determined by the initial data analysis to have the most impact on the classification of Twitter accounts as genuine or as automated. Several metrics will be used to measure the effectiveness of this method. The first metric will be the percentage of points that must be poisoned, ranging from 0 to 30%. The second metric will be how much the targeted feature must be altered. For example, the value of feature X may be incremented by a number from 1 to 2500 for each poisoned row.

The accuracy and F1-score will be used to measure each combination of percentage of poisoned rows and how much they are altered by. The accuracy plainly represents how many accounts were correctly identified as either class. The F1-Score is used to make the results easier to understand, since it gives the harmonic average of both a model's recall and precision.

This method will be repeated for nine models using three different algorithms to detect three different kinds of bots – Spambots, Fake-followers, and Social-bots. Intense computation was needed to generate these models. Therefore, in order to put less strain on the available hardware, this method was limited to using the KNN, GLM, and SVM

algorithms to train the models. The results for each model will be visualized using a series of 3D surfaces.

3.4 Outlier Detection as a Method for filtering Poisoned Data Rows

Paudice et. al. described a method of filtering out poisoned data points by using outlier detection algorithms [19]. In this paper, the effectiveness outlier detection method based on the Mahalanobis distance function is tested. While many distance functions such as Manhattan distance or Euclidean distance measure the distance between two distinct data-points, the Mahalanobis distance instead measures the distance between one point in a data-set and the mean of every other point. This effectively compares a data-point to the rest of the data-set, and therefore makes this distance function very useful for detecting outlier data-points that stand out from the rest of the data. Data-points are marked as an outlier based on the two features determined to have the most impact on classification. In the next section, we outline the two most important features for detecting each kind of bot. By obtaining the Mahalanobis distance of a data point based on the values of these two features, we may determine whether or not a data point is an outlier.

After poisoning the data, we use this method to mark outlier points. Data-points above a threshold for 80% of the data are marked as outliers and removed from the data-set before the poisoned data is used to train the models. The accuracy and F1-Score of the filtered data models are measured against that of the unfiltered models in the results below.

4 Results

4.1 Initial Data Analysis

In Table 2, three different models have been trained using the GLM, SVM, and KNN algorithms to classify Twitter accounts from the Cresci-2017 data-set as either Genuine human users or Social Bots mimicking human users. The metrics for accuracy, recall, precision and the F1-Score are given. This process was repeated in Table 3 to classify accounts as Genuine human users or Fake-Followers, and in Table 4 to classify accounts as Genuine or as Spambots.

Table 2. Results for Genuine/Social-Bot Classifier.

Model	accuracy	recall	precision	F1 Score
GLM	93%	91%	97%	94%
KNN	98%	99%	98%	98%
SVM	96%	96%	98%	97%

Table 3. Results for Genuine/Fake-follower Classifier.

Model	accuracy	recall	precision	F1 Score
GLM	98%	96%	99%	98%
KNN	95%	95%	95%	95%
SVM	95%	95%	95%	95%

Table 4. Results for Genuine/Spambot Classifier.

Model	accuracy	recall	precision	F1 Score
GLM	95%	91%	99%	95%
KNN	98%	97%	98%	97%
SVM	93%	94%	91%	92%

The tables below show the results for the two different feature selection methods. In the first method, each F1-Score is calculated by excluding the respective feature from the training data. This indicates the negative impact of the exclusion of that feature on the model, and therefore its importance. Table 5 shows the results for calculating the F1-Score using this method on the data for detecting Social Bots.

Table 5. Calculated F1 Score for Excluding Each Feature, for Detecting Social Bots.

Feature	F1 Score without
favourites_count	0.9448047
geo_follow_protect_verify	0.9488607
created_at	0.9589940
time_zone	0.9597032
friends_count	0.9607935
utc_offset	0.9614000
lang	0.9621306
listed_count	0.9624317
statuses_count	0.9626558
followers_count	0.9637003

Table 6. Fisher Score of Each Feature, for Detecting Social Bots.

Feature	Fisher Score
Geo_follow_protect_verify	1.6
Time_zone	0.73
created_at	0.52
statuses_count	0.24
favourites_count	0.16
lang	0.057
utc_offset	0.04
Listed_count	0.012
Followers_count	0.0018
friends_count	0.00025

Table 6 shows the Fisher Scores for each feature of that data. Table 7 shows the different features used to classify Twitter accounts as genuine or social bots in descending order, based on the mean order of importance of features in Table 5 and Table 6.

Table 7. Features in Descending Order of Importance, for Detecting Social Bots.

Feature
Geo_follow_protect_verify
favourites_count
time_zone
created_at
statuses_count
Lang
utc_offset
friends_count
listed_count
followers_count

Based on the results in Table 7, the two most important features for classifying a Twitter account as a Social-bot are the dummy-variable *geo_follow_protect_verify* and *favourites_count*.

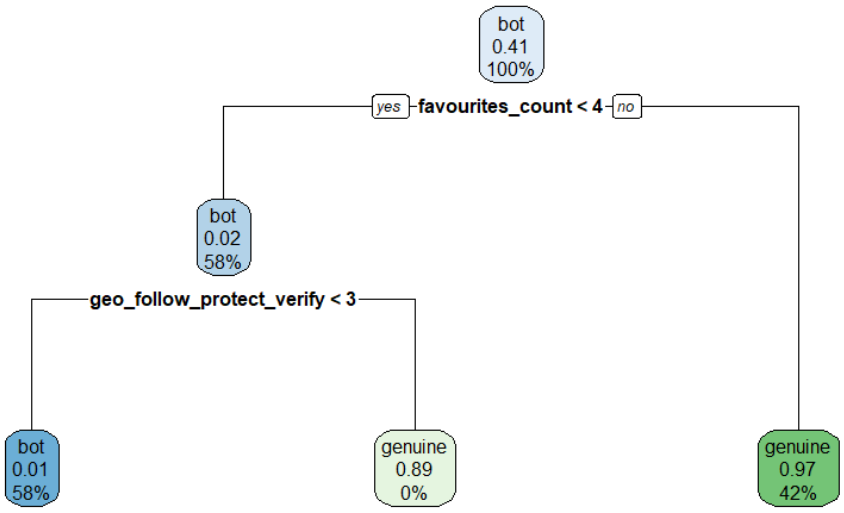

Figure 1. Decision Tree for Genuine vs Social-bot Classifier.

The decision tree for classifying accounts as genuine or as social bots is illustrated in Figure 1. The feature *Favourites_count* indicates how many tweets a Twitter user has liked. *Geo_follow_protect_verify* represents four different binary features: *geo_enabled*, *follow_request_sent*, *protected*, and *verified*. Looking at this, most Twitter accounts that have liked less than four tweets are classified as bots. At the next branch, Twitter accounts that have a factor level for *Geo_follow_protect_verify* less than three are overwhelmingly likely to be bots. The two lowest factor levels for *Geo_follow_protect_verify* are 1 and 0. These correspond to a negative value for *follow_request_sent*, *geo_enabled*, and *protected*, and either a positive or negative value for *verified*. This means that being geo-enabled or protected are good indicators that an account is genuine. Interestingly, it also means that being verified does not necessarily indicate that an account is genuine.

Table 8. Features in Descending Order of Importance, for Detecting Fake-followers.

Feature
favourites_count
friends_count
statuses_count
time_zone
Geo_follow_protect_verify
created_at
Lang
listed_count
utc_offset
followers_count

The feature selection methods are repeated for the data used to train models to detect Fake-followers. Table 8 gives the final order of importance, akin to Table 7. The results for the Fisher Score and F1 Score methods can be seen in Appendix 1. Likewise, the decision tree for classifying accounts as genuine or as fake-followers is illustrated in Figure 2.

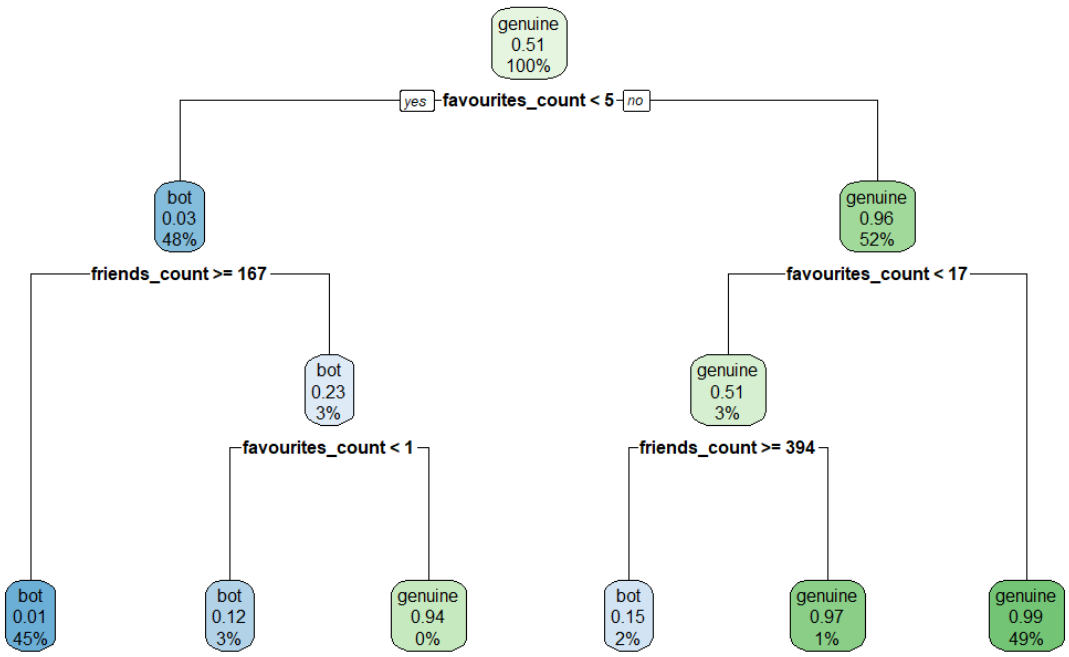

Figure 2. Decision Tree for Genuine vs Fake-Follower Classifier.

Based on Table 8 and Figure 2, the two most important features for classifying a Twitter account as a Fake-follower are the *favourites_count* and the *friends_count*. The *friends_count* is the number of Twitter accounts an account is following. There seems to be an inverse correlation between the number of accounts followed and the number of tweets liked. Specifically, Fake-followers tend to follow more Twitter accounts but like few to zero tweets. Conversely, Genuine accounts tend to like more tweets but follow less accounts. This can be summarized as a favourite/friend ratio, where accounts with a ratio less than 0.03 are Fake Followers. This type of behavior is reflected by name of this bot, Fake-followers. Their sole purpose is to follow other Twitter accounts to make them seem more popular. Many Fake-followers can be bought or sold in online markets [30].

The methods outlined above are repeated again for the data used to train models to detect Spambots. Table 9 gives the final order of importance. The results for the Fisher Score and F1-Score methods are in Appendix 1. The decision tree for classifying accounts as genuine or as spambots is shown in Figure 3.

Table 9. Features in Descending Order of Importance, for Detecting Spambots.

Feature
favourites_count
statuses_count
Lang
utc_offset
created_at
friends_count
time_zone
followers_count
Geo_follow_protect_verify
listed_count

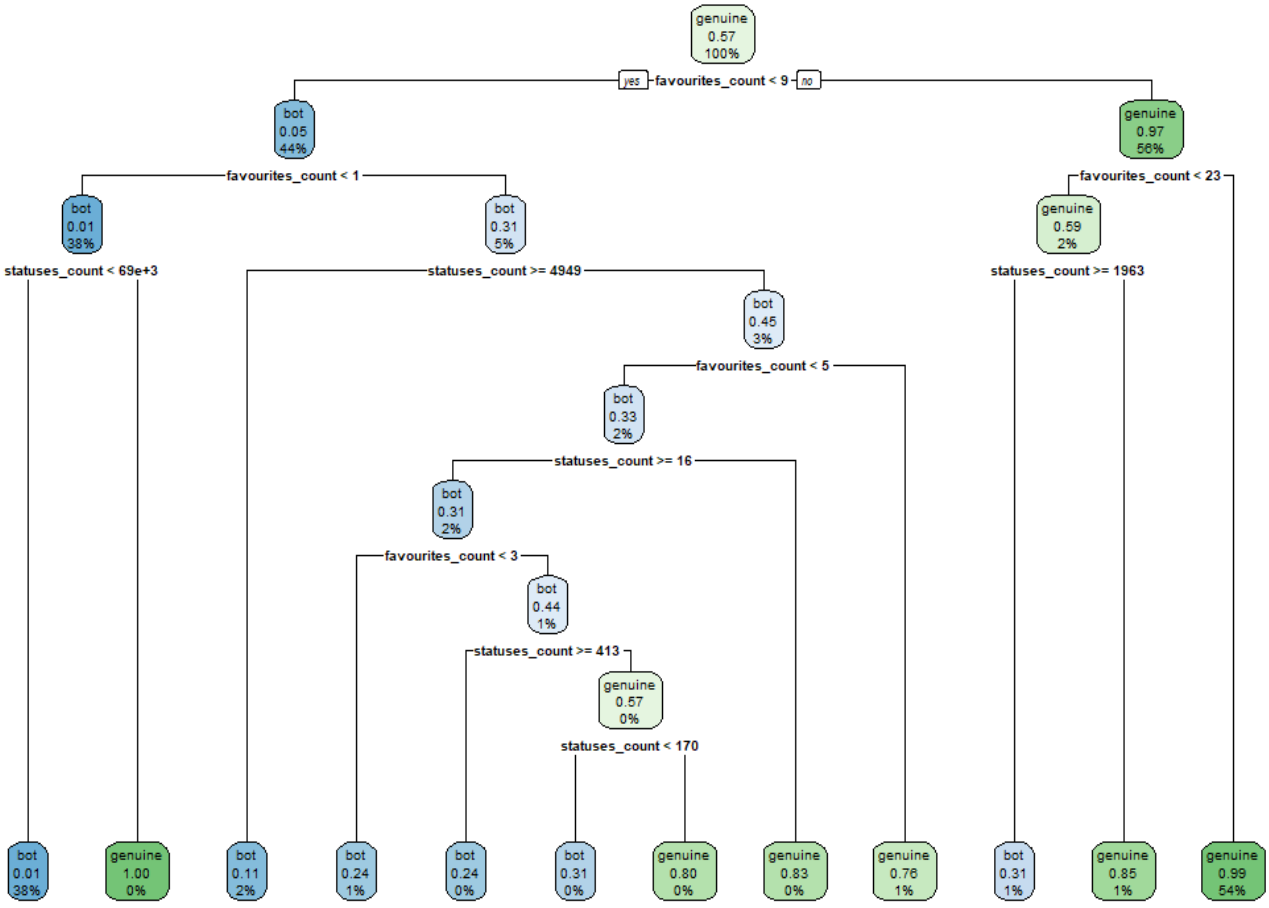

Figure 3. Decision Tree for Genuine vs Spambot Classifier.

Based on the results in Table 9 and Figure 3, the two most important features for classifying a Twitter account as a Spambot are *favourites_count* and *statuses_count*. The feature *statuses_count* gives the number of tweets or retweets made by an account. Similar to Fake-followers, there appears to be an inverse correlation between *favourites_count* and *statuses_count*. Specifically, Spambots tend to make a lot of tweets, but like few to zero tweets. Conversely, genuine accounts tend to make far fewer tweets but like more tweets made by others. Summarized, accounts with a favourite/statuses ratio less than 0.000876 are classified as Spambots.

It is very interesting to note that a common sign that an account is any of the three types of bots in the Cresci-2017 data-set is if it has liked a very low number of tweets.

4.2 Random Label Flipping

The figures below represent a Random-Label-Flipping-Attack on Data used to classify Twitter Accounts as Genuine or as Social Bots. Figures 4 through 6 show the results on models trained using GLM, KNN, and SVM algorithms, respectively.

In the random-label-flipping attack, the models that were most sensitive to this type of data-poisoning were trained using a Generalized Linear Model algorithm. The models that were most resilient were trained using the K-Nearest-Neighbor algorithm.

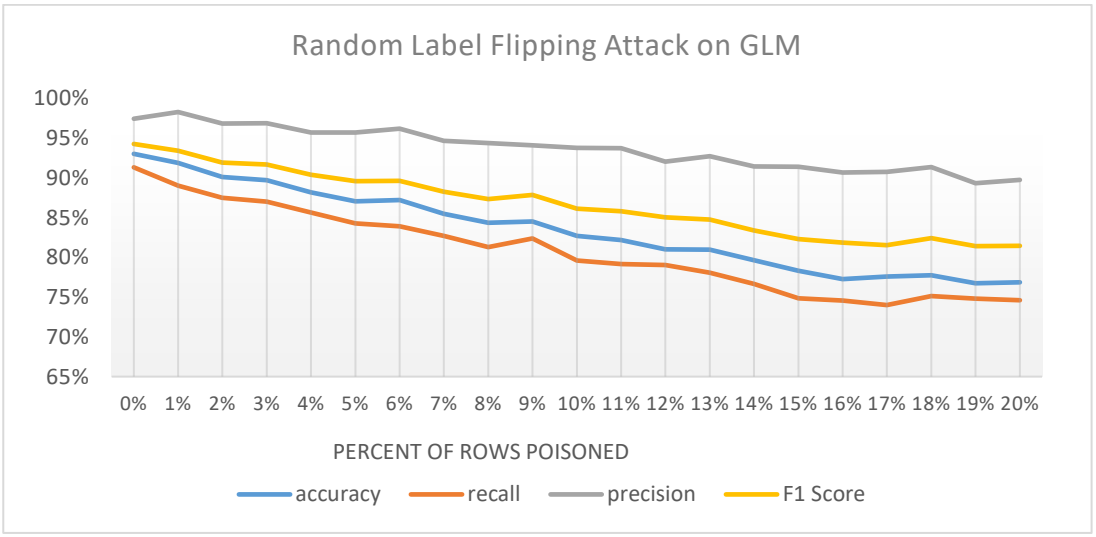

Figure 4. Results of Random Label Flipping Attack on GLM-trained Model for Detecting Social-Bots.

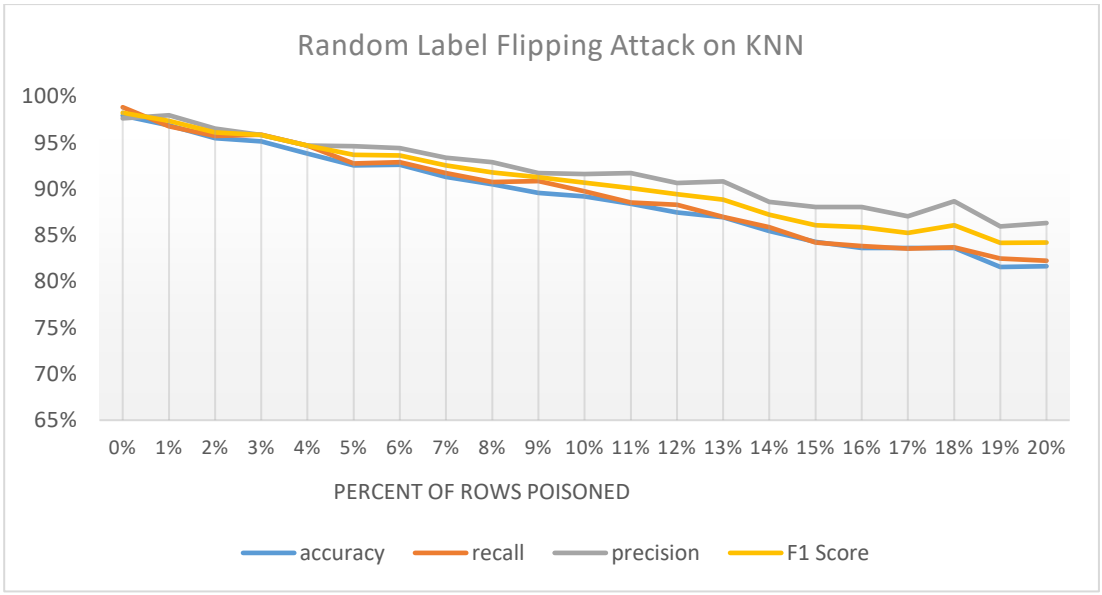

Figure 5. Results of Random Label Flipping Attack on KNN-trained Model for Detecting Social-Bots.

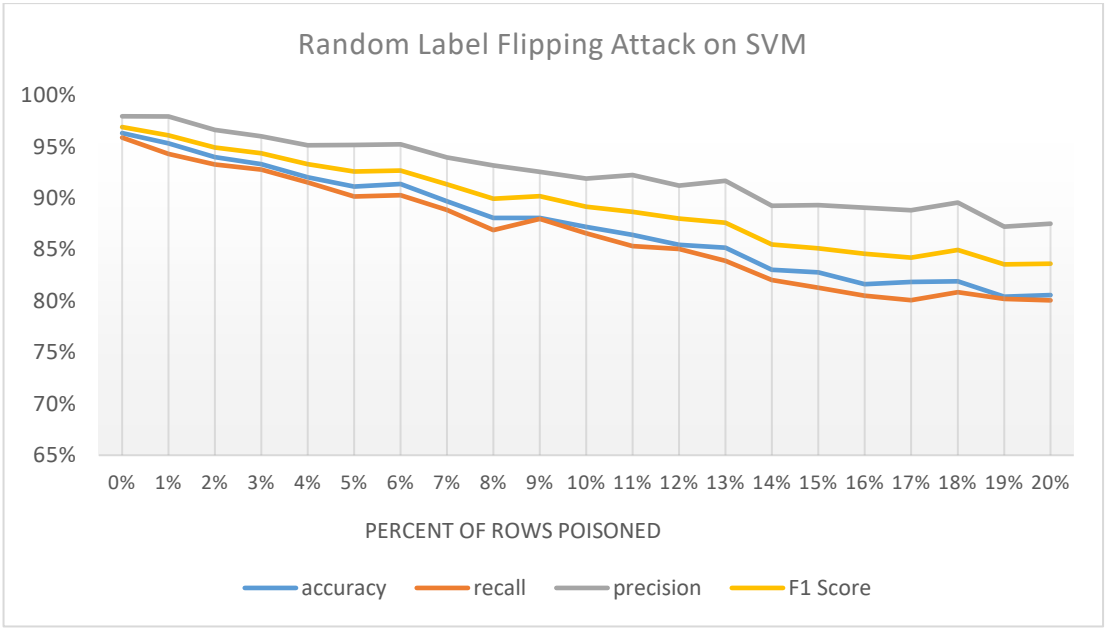

Figure 6. Results of Random Label Flipping Attack on SVM-trained Model for Detecting Social-Bots.

Figures 7 and 8 show the results of Random Label Flipping Attack on models trained by a GLM algorithm to detect Fake-Followers and Spambots, respectively. The results for the corresponding attacks on models trained using KNN and SVM are overall very similar to Figure 5 and Figure 6, but can be seen in Appendix 2.

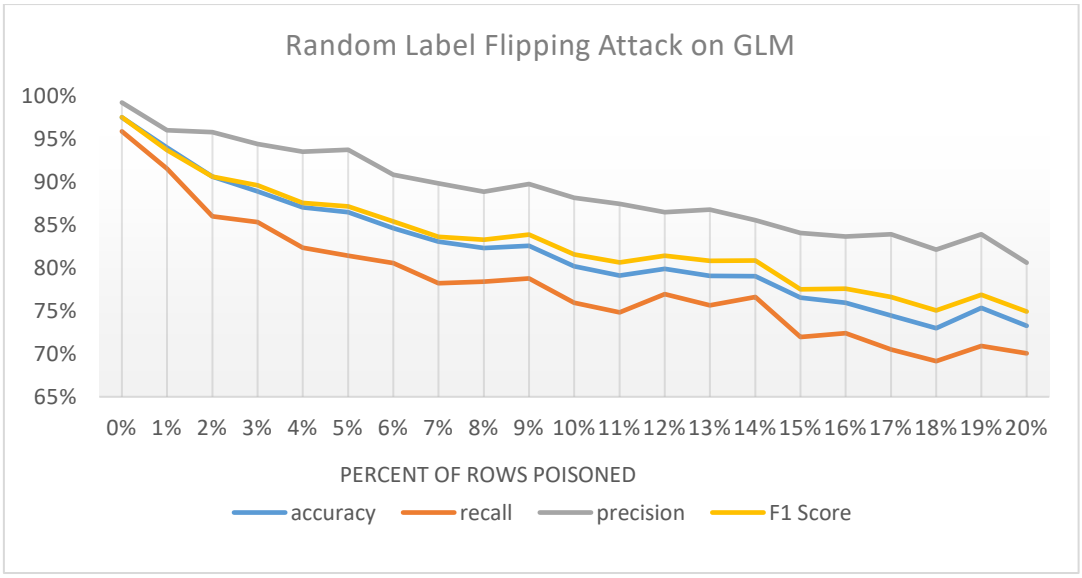

Figure 7. Results of Random Label Flipping Attack on GLM-trained Model for Detecting Fake-Followers.

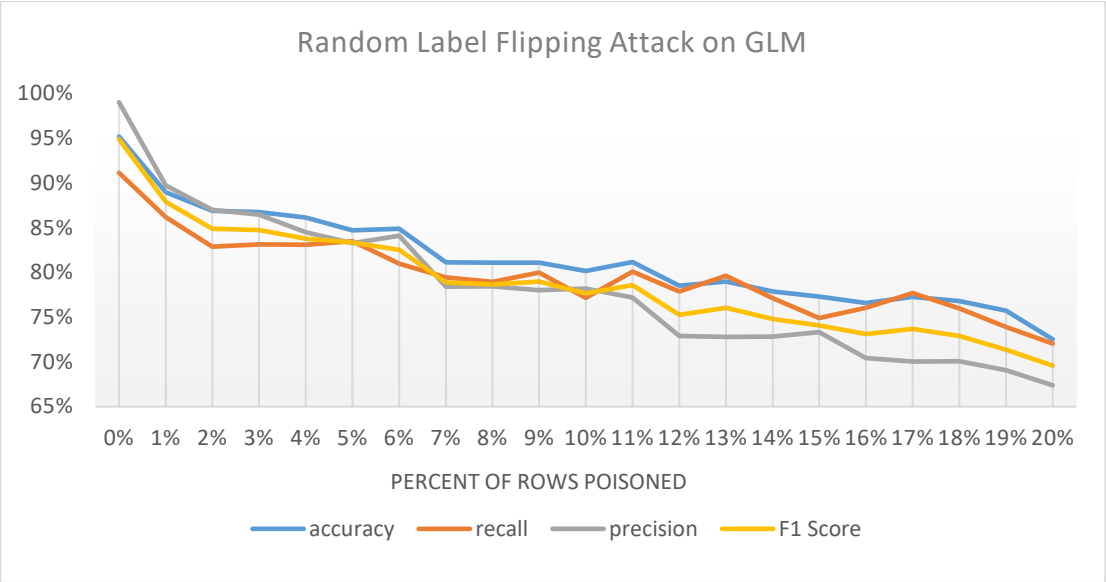

Figure 8. Results of Random Label Flipping Attack on GLM-trained Model for Detecting Spambots.

4.3 Direct Feature Poisoning Attack

Based on the initial data analysis, the most influential feature for detecting each different kind of bot appears to be *favourites_count*. The one exception to this is social-bots, the most influential feature for which is *geo_follow_protect_verify*. However, since this feature only has discrete values, the way it is altered cannot be measured the same way as altering a continuous feature can. Because of this, the results for this method of attack are measured based on how much the *favourites_count* of each data-set is altered.

Similar to the random-label-flipping attack, a random 1-30% of account data-points are cloned. For every bot in these cloned accounts, the *favourites_count* is incremented by a random number. This number is taken from a range of 50 integers, starting at 1-50, then 50-100, and continuing up to a range from 2450-2500. The *favourites_count* for genuine accounts is not altered. This is because to achieve the same effect, the *favourites_account* would have to be decremented, which runs the risk of giving a negative value for the feature.

Three different models were trained using GLM, KNN, and SVM to detect spambots, social-bots, and fake-followers, with a total of nine models overall. For eight out of these nine models, no significant drop in accuracy, recall, precision, or F1 score was

recorded. The sole exception was the model trained using GLM to classify accounts as spambots or genuine users.

The 3D surface representing the impact on this model's accuracy is shown in Figure 8. For this surface, the Y-axis represents the percent accuracy, the X-axis represents the percent of rows poisoned, and the Z-axis represents the value that *favourites_count* is incremented by.

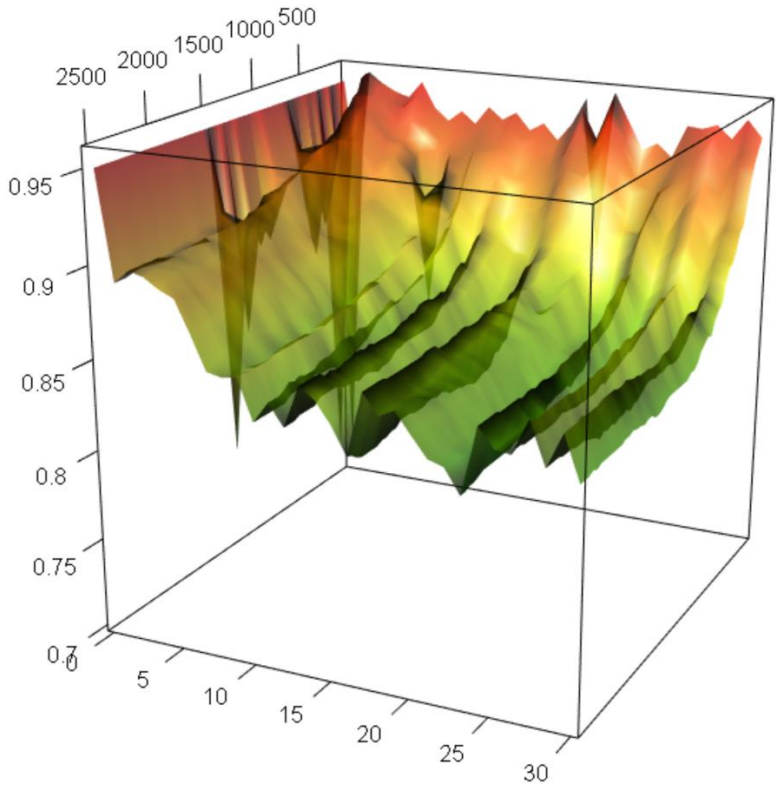

Figure 8: Surface representing Accuracy of a single GLM-trained model for detecting Spambots.

At first glance the surface in Figure 8 looks very unusual. What it shows is that there is an inverse correlation between the accuracy of the model and how much the accounts are poisoned. As a greater percent of accounts are poisoned, and as the value *favourites_count* is incremented by goes up, accuracy goes down. However, there are diminishing returns. Accuracy tends to stop decreasing after dropping to around 80%. There are some unusually sharp drops in accuracy, however, which will be explained in the section below.

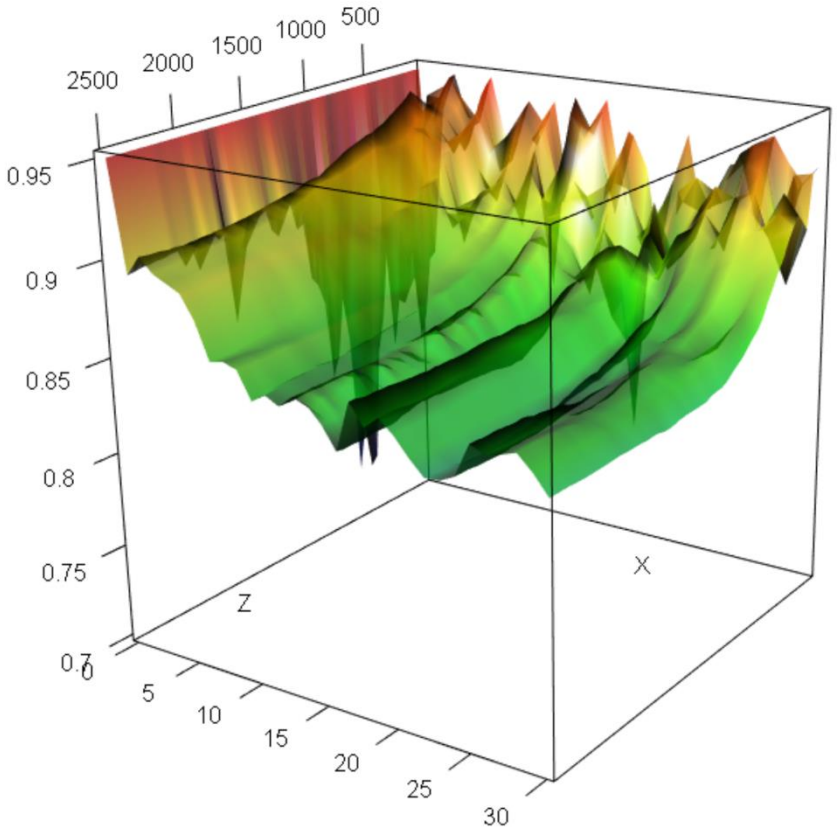

Figure 9. Average Accuracy of Ten Different GLM-trained models for detecting Spambots.

The surface in Figure 8 is based on poisoning a specific set of data-points. The surface in Figure 9 gives a more general idea of how many points need to be altered and by how much. This surface was generated from the results of poisoning ten different sets of data. The 3D surfaces for the models trained using different algorithms to detect different bots, as well as the surfaces that Figure 9 was averaged from, can be viewed in Appendix 4.

There is still the possibility that the surface in Figure 9 is biased towards a specific data-set. Focus is put on three different minima within the surface. In the table below, the accuracy of these points is given, as well as the percent of rows poisoned and how much the *favourites_count* of each row was poisoned. To test whether these averaged minima are truly representative or biased, corresponding results are taken from models trained on two different poisoned data-sets. The results can be seen in Table 10.

Table 10. Local Minima for 3D Surface for the Accuracy a Poisoned GLM model for detecting Spambots.

Favourites_count incremented by number between	Percent of rows poisoned	Accuracy (Average of 1st ten models)	Accuracy (11th model)	Accuracy (12th model)
1700:1750	22%	82%	82.6%	83%
250:300	19%	77%	88%	87%
150:200	3%	79.5%	93%	94%

The first minimum is the most consistent no matter which data-points are poisoned. If the *favourites_count* of 22% of the rows is incremented by a number between 1700 and 1750, the accuracy of a model trained on that data in detecting bots will decrease to roughly 83%.

The second minimum is less consistent. While the average results of the first ten poisoned models show that the accuracy will decrease to 77% if the *favourites_count* of 19% of the rows is increased by a number between 250 and 300, these results diverge for the 11th and 12th models trained on different poisoned data-sets. For these models, poisoning that many rows by that much has less negative impact on their accuracy in detecting bots. This inconsistency between models is also true for the third minimum.

Although the results for the third minimum are not consistent between different sets of rows being poisoned, they illustrate something important. The results of an algorithm trained to detect Twitter spambots can be drastically reduced to 79.5% by increasing the *favourites_count* of less than 3% of the rows (140 spambots) by as little as 150. Because these results are inconsistent with those of the 11th and 12th models, there must exist a specific set of data-points that the model is sensitive to being altered. A more sophisticated poisoning method is needed to determine what these data-points are. Unfortunately, solving this problem requires far more computing power than is available for this study. Future researchers with better resources may have an easier time determining this.

4.4 Outlier Detection as a Method for filtering Poisoned Data Rows

4.4.1 Results of filtering Outliers from Random Label Flipping Attack

Figure 10 represents Random Label Flipping Attack on a model trained using a GLM algorithm to detect social-bots; the same as Figure 4. However, in this attack, data-points with the greatest Mahalanobis-distance are removed before the data is used to train the model. Figure 11 directly compares the accuracy of the results in Figure 4 and Figure 10.

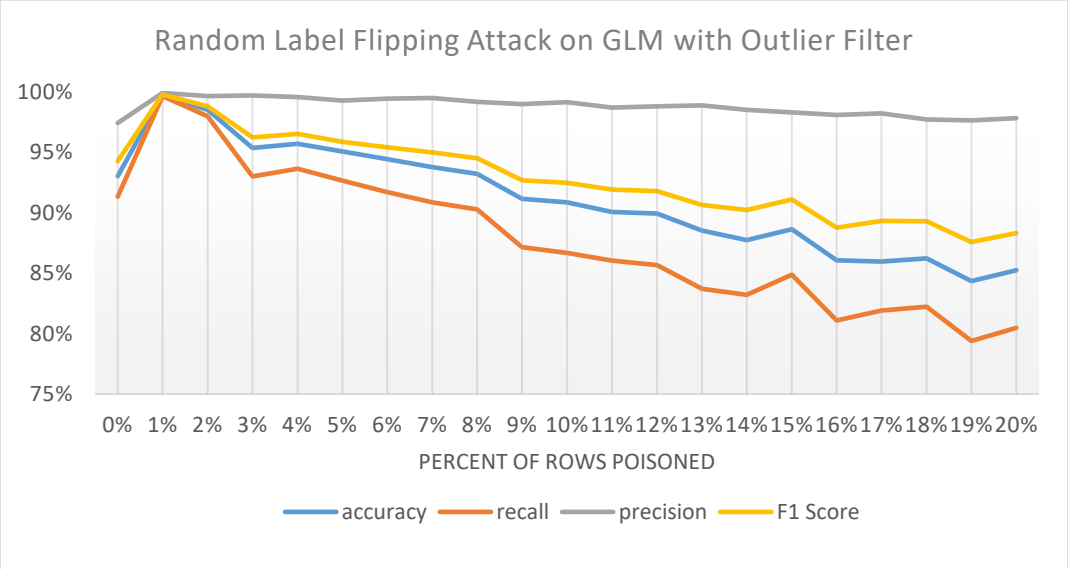

Figure 10. Results of Random Label Flipping Attack on GLM-trained model for detecting Social-bots with Outlier Filter.

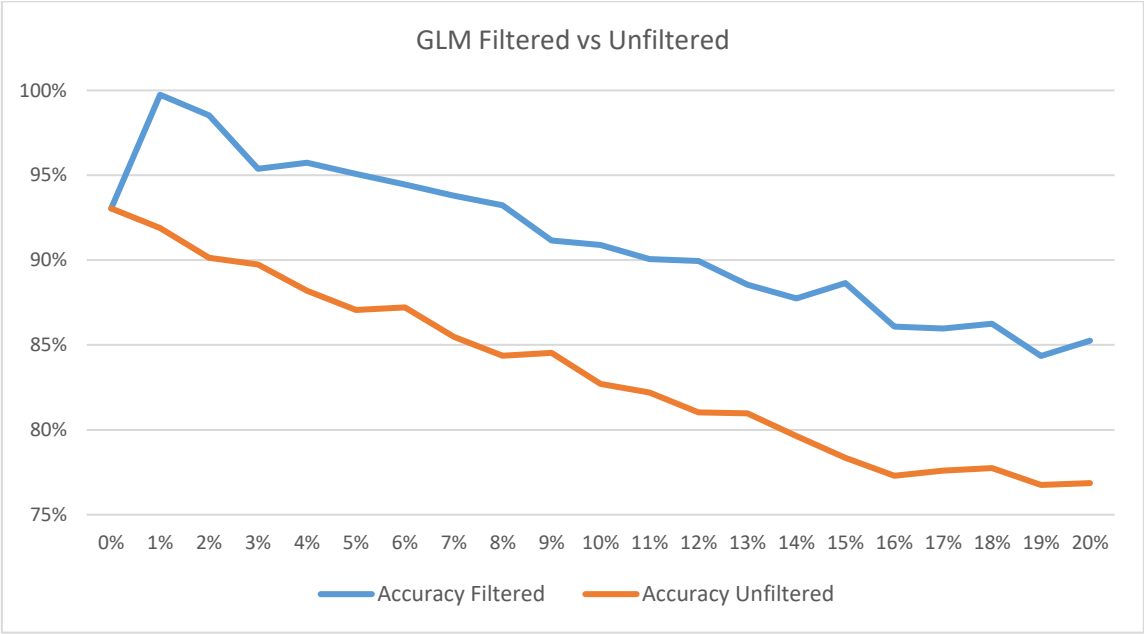

Figure 11. Accuracy of GLM-trained Models for detecting social-bots, poisoned by Random Label Flipping Attack, With & Without Outlier Filter.

Figure 12 and Figure 13 correspond to the results of a Random Label Flipping attack on a KNN-trained model in Figure 5. The same distance-based outlier method as before is used to mitigate the impact of the poisoned data.

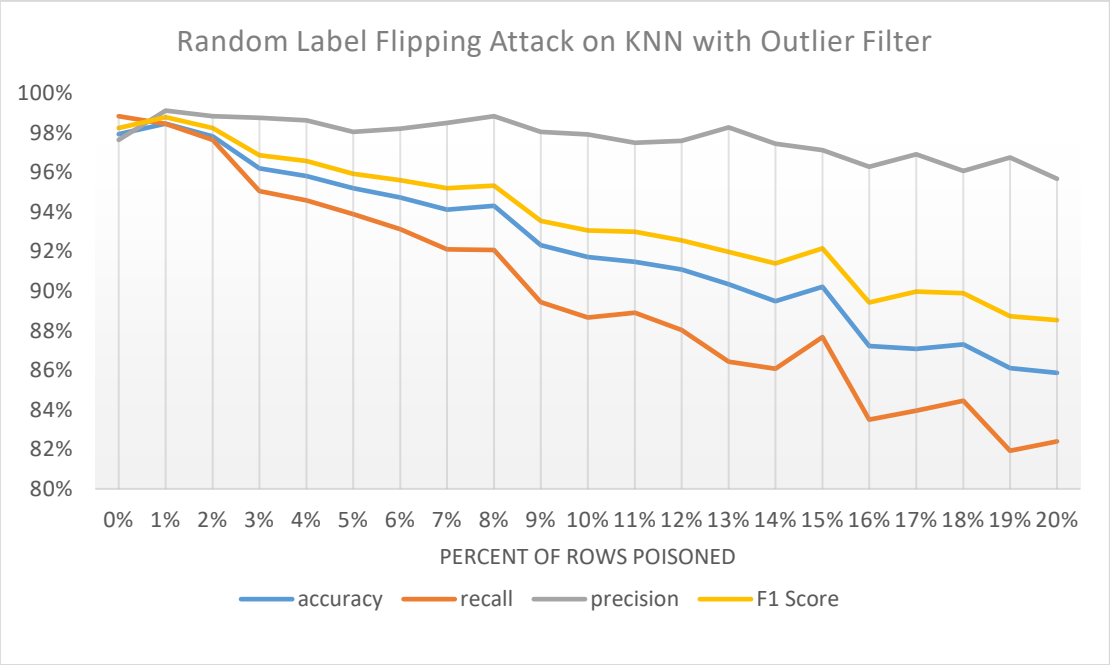

Figure 12. Results of Random Label Flipping Attack on KNN-trained model for detecting Social-bots with Outlier Filter.

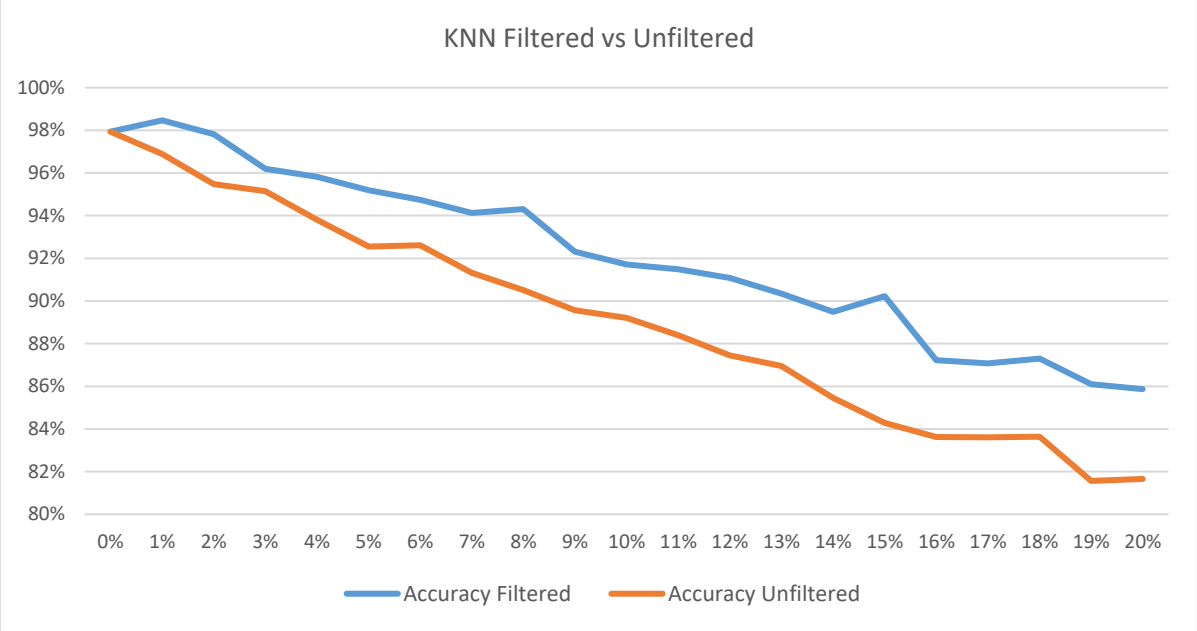

Figure 13. Accuracy of Poisoned Social-Bot KNN-trained Models for detecting social-bots, poisoned by Random Label Flipping Attack, With & Without Outlier Filter.

Figure 14 and Figure 15 correspond to Figure 6, which shows the results of Random Label Flipping Attacks on an SVM-trained model for detecting social-bots. Again, Mahalanobis-distance is used to filter out poisoned data.

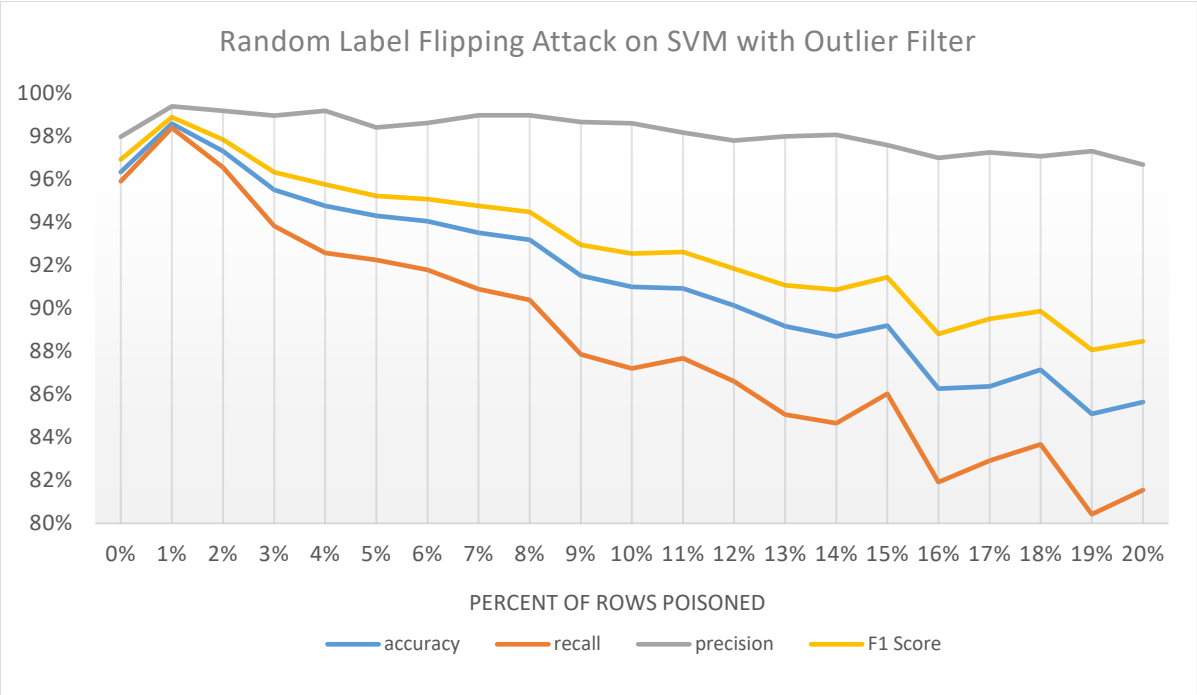

Figure 14. Results of Random Label Flipping Attack on SVM-trained model for detecting Social-bots with Outlier Filter.

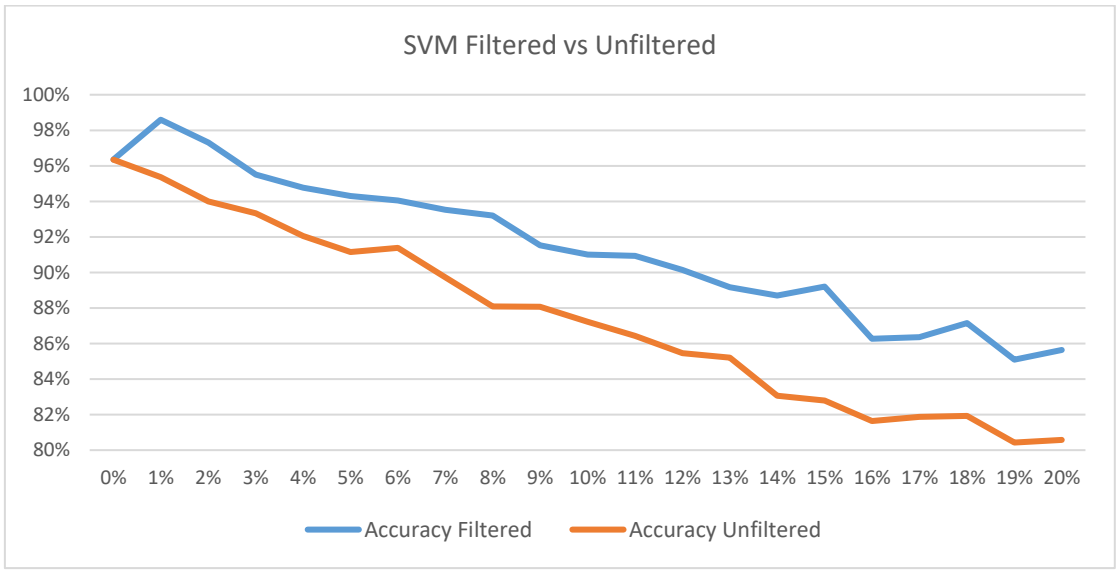

Figure 15. Accuracy of SVM-trained Models for detecting Social-bots, poisoned by Random Label Flipping Attack, With & Without Outlier Filter.

Figure 16 corresponds to Figure 7 and gives the accuracy of a model trained to detect Fake-followers with and without an outlier filter. Figure 17 corresponds to Figure 8 in the same way, for models trained to detect Spambots. More detailed results for models trained using a Mahalanobis-distance-based outlier filter can be found in Appendix 3.

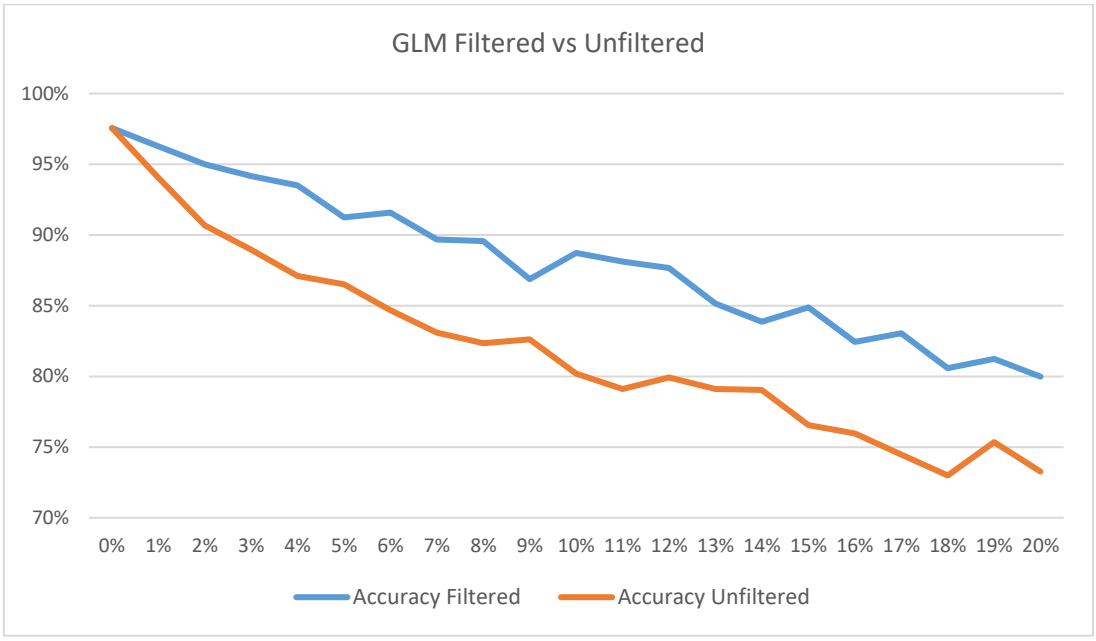

Figure 16. Accuracy of Poisoned Fake-Follower GLM Models With & Without Outlier Filter.

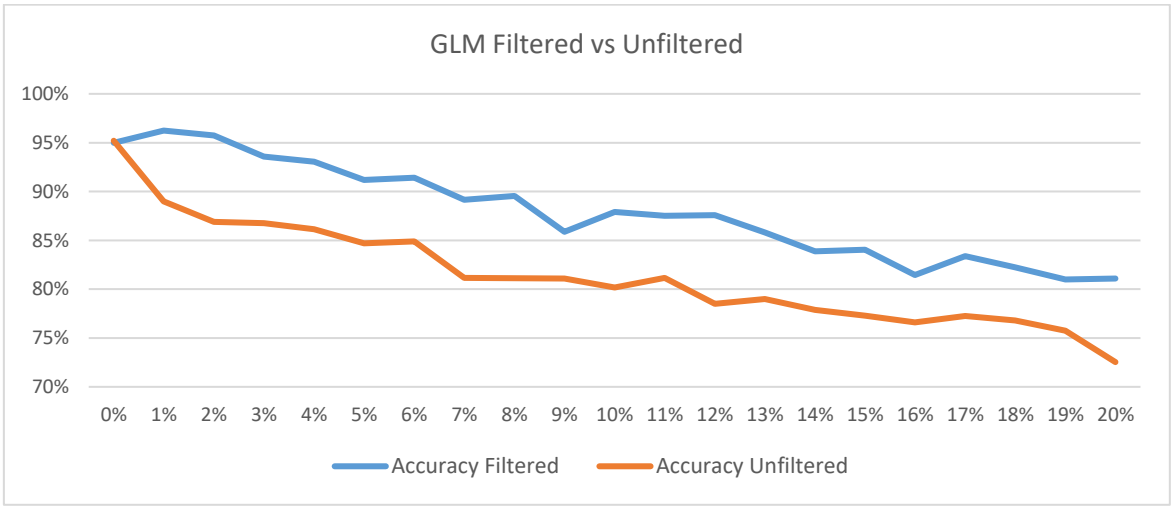

Figure 17. Accuracy of Poisoned Spambot GLM Models With & Without Outlier Filter.

4.4.2 Results of filtering Outliers from Direct Feature Poisoning Attack

The same distance-based outlier method using Mahalanobis-distance was applied to the GLM model used to detect Spambots. Figure 18 represents the impact on accuracy a direct-feature poisoning attack has after filtering outliers. Figure 19 similarly shows the impact on the model's F1 Score. The Z axis of both figures represents the number, from 1 to 2500, by which the *favourites_count* of the Spambot accounts is incremented. The X axis represents the percent, from 0 to 30, of accounts targeted by the poisoning attack. the Y axis gives the accuracy.

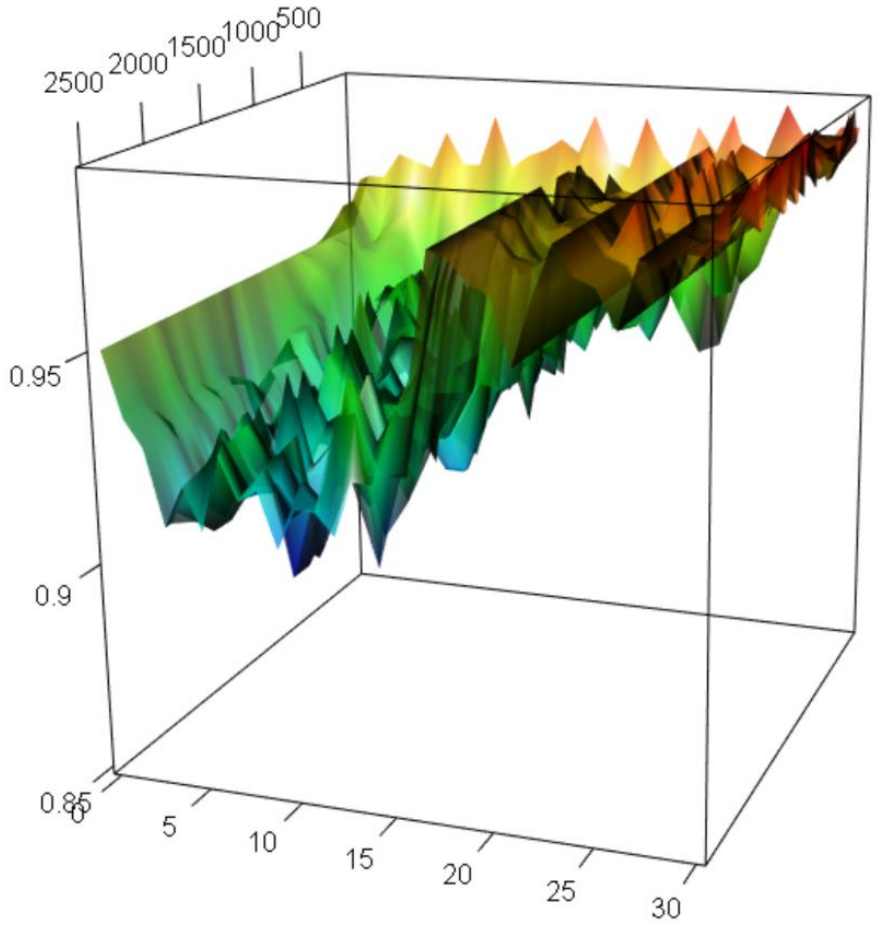

Figure 18. Surface of the Accuracy of a GLM-trained model for detecting Spambots, with Outliers filtered.

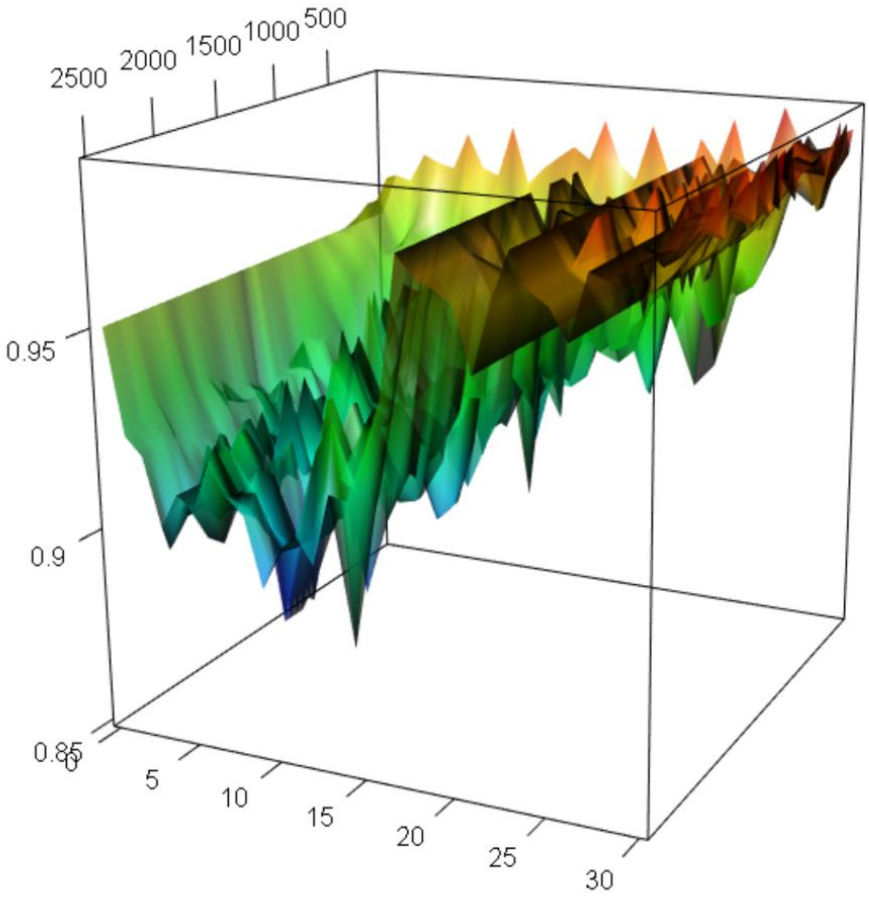

Figure 19. Surface of the F1-score of a GLM-trained model for detecting Spambots, with Outliers filtered.

Figure 18 and Figure 19 show that there is a drastic reduction in the impact of poisoned data after using a Mahalanobis-distance-based outlier filter. The models even appear to become more effective at correctly identifying Spambots then before their training data was poisoned.

5 Analysis

After analyzing the results, the next step is to determine an optimal attack strategy based on them. When designing an attack strategy, it is important to keep in mind the two

different types of adversarial machine learning attacks. According to the framework developed by Xiao et. al., a poisoning attack occurs when an attacker is able to manipulate the data used to train an algorithm. According to the same framework, an evasion attack occurs when an attacker manipulates the test data [16]. In the experiments for this research, focus was put on altering the training data before it was used to create a model for classifying accounts as bots or humans. Experimenting with altering the test data would be interesting, but any information gleaned from modeling an evasion attack would likely be out of date. As mentioned before by Zhouhan et. al., data scientists who study automated accounts on social media and the people who create those accounts are engaging in a 'virtual arms race' [10]. The behavior of bots on Twitter and other social media sites changes almost as quickly as researchers learn how to detect them. Automated accounts are already changing their behavior to evade detection. This research is more concerned with how a poisoning attack may affect bot-detection models.

The results of the experiments show that a Direct Feature Poisoning attack outlined in Section 3.3.2 can be very effective if it targets specific data-points determined to have a high impact on the model. However, such an attack would be very computationally intensive in order to determine what the optimal data-points to target are. Even if an optimal set of points is found, it has been shown that using a simple outlier filter based on Mahalanobis-distance is sufficient to negate the impact of the attack. Based on the results, the optimal attack method that is the least computer-intensive and takes the least time is the Random Label Flipping attack.

In order to implement this method, an attacker must have knowledge of what training data is being collected, when it is being collected, and by whom. Since this attack method explicitly targets supervised-learning algorithms, the attacker must also know who is labeling the data. Many researchers use crowdsourcing from sites such as Amazon Mechanical Turk or Figure-Eight to create a 'ground truth' and efficiently label large data-sets [11]. By targeting and compromising this process, an attacker can potentially reduce the ultimate accuracy of a Twitter-bot detector to 75%. This attack is illustrated in Figure 20.

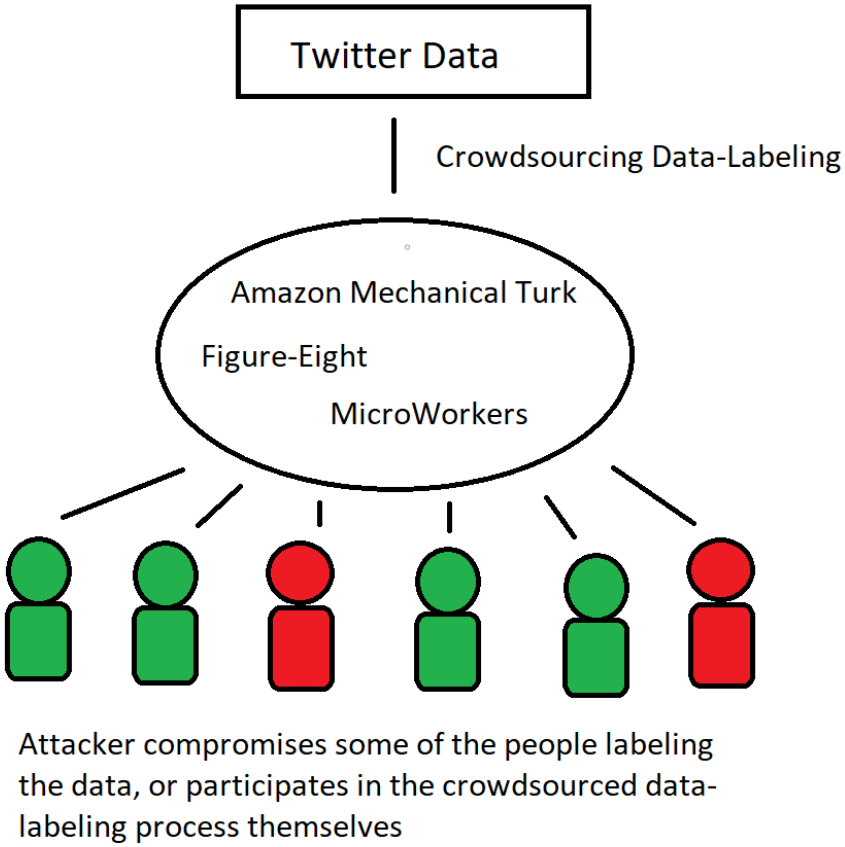

Figure 20. Illustration of Proposed Attack Method for Targeting Crowdsourcing Process.

6 Conclusion

The possibility of a poisoning attack against training data for Twitter Bot detection models deserves attention. Right now, these models are regularly retrained on new data as bots change their behavior to avoid detection. This training data is already difficult to collect. While obtaining Twitter account data using the Twitter API is relatively simple, finding pre-labeled data-sets of accounts confirmed to be human or bots is very difficult. The Cresci-2017 data-set used in this study contained a substantial set of Twitter accounts, but many of the automated accounts in the data-set were very old, some having been created as early as 2009 [11]. If someone influenced or altered this training data maliciously, it would make it even harder to train models to accurately spot automated accounts.

Analyzing the results of this research shows that such bot-detection models are indeed very vulnerable to poisoning attacks. By simply introducing mis-labeled data-points into the Cresci-2017 data-set, the accuracy, recall and precision of the models were lowered by as much as 20%. This label-flipping attack was determined to be the most optimal attack, since the impact was mostly consistent regardless of what data-points were mis-labeled or what algorithm was being trained on them. One way an attacker could implement this method in real life is by participating directly in the crowdsourcing process that many researchers use to label Twitter accounts as humans or bots.

Depending on what tools they have, attackers may be able to have an even greater negative impact on the accuracy of these models than the aforementioned label-flipping attack. Future researchers must investigate what these more optimal poisoning attack methods might be. Evidence for one such method was found in this study. By targeting a specific set of data-points in a poisoning attack instead of randomly-selected ones, an attacker could maximize their negative impact on the accuracy of a model in detecting bots. Hardware limitations prevented further investigation, but data scientists with more resources may find that the accuracy of a model for detecting bots may be lowered by as much as 50% after only poisoning less than a hundred data-points.

Other avenues for future research include testing the resilience of different models besides the KNN, GLM, and SVM-trained models used in this study. While the Mahalanobis-distance-based outlier filter was found to be very effective in removing poisoned data, it may not be as effective against more sophisticated poisoning attacks. Researchers should explore different kinds of methods for removing or mitigating the impact of poisoned data as well. This phenomenon is not limited to Twitter either. Bots have a presence across all forms of social-media, meaning there is the potential for similar poisoning attacks against models for detecting bots on Facebook, Reddit, or Instagram.

By better understanding this form of adversarial-machine-learning, data scientists can make their own algorithms for detecting bots more resilient. In doing so, they can reduce the impact of these poisoning attacks, and ultimately mitigate the negative impact that malicious bots have on social-media.

Bibliography

[1] D. M. J. Lazer *et al.*, "The science of fake news: Addressing fake news requires a multidisciplinary effort," *Science (80-.).*, vol. 359, no. 6380, pp. 1094–1096, 2018.

[2] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science (80-.).*, vol. 359, no. 6380, pp. 1146–1151, 2018.

[3] M. D. Conover, J. Ratkiewicz, M. Francisco, B. Gonçalves, A. Flammini, and F. Menczer, "Political Polarization on Twitter," in *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[4] C. R. Sunstein, "The Law of Group Polarization," *J. Polit. Philos.*, vol. 10, no. 2, pp. 175–195, 2002.

[5] J. Johnson, "The Self-Radicalization of White Men: 'Fake News' and the Affective Networking of Paranoia," *Commun. Cult. Crit.*, vol. 11, no. 1, 2018.

[6] B. Bostock, K. Corcoran, and B. Logan, "This timeline of the Christchurch mosque terror attacks shows how New Zealand's deadliest shooting unfolded," *Insider*, 2019. [Online]. Available: https://www.thisisinsider.com/christchurch-shooting-timeline-49-killed-new-zealand-mosques-2019-3. [Accessed: 28-Mar-2019].

[7] M. Haag and M. Salam, "Gunman in 'Pizzagate' Shooting Is Sentenced to 4 Years in Prison," *The New York Times*, 2017. [Online]. Available: https://www.nytimes.com/2017/06/22/us/pizzagate-attack-sentence.html. [Accessed: 28-Mar-2019].

[8] L. M. Aiello, M. Deplano, R. Schifanella, and G. Ruffo, "People are Strange when you're a Stranger: Impact and Influence of Bots on Social Networks," in *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media*, 2014, pp. 10–17.

[9] C. Shao, G. Luca Ciampaglia, O. Varol, K. Yang, A. Flammini, and F. Menczer, "The spread of low-credibility content by social bots," *Nat. Commun.*, vol. 9, 2018.

[10] C. Zhouhan, R. S. Tanash, R. Stoll, and D. Sabramanian, "Hunting Malicious Bots on Twitter: An Unsupervised Approach," *Lect. Notes Comput. Sci. Soc. Informatics*, pp. 501–510, 2017.

[11] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. e," in *26th International Conference on World Wide Web*, 2017, pp. 963–972.

[12] J. Wright and O. Anise, “Don’t @ Me: Hunting Twitter Bots at Scale,” in *Blackhat USA 2018*, 2018.

[13] Ben Popken, “Twitter deleted 200,000 Russian troll tweets. Read them here.,” *NBC News*, 2018. [Online]. Available: https://www.nbcnews.com/tech/social-media/now-available-more-200-000-deleted-russian-troll-tweets-n844731.

[14] P. G. Efthimion, S. Payne, and N. Proferes, “Supervised Machine Learning Bot Detection Techniques to Identify Social Twitter Bots,” *SMU Data Sci. Rev.*, vol. 1, no. 2, 2018.

[15] C. A. Freitas, F. Benevenuto, S. Ghosh, and A. Veloso, “Reverse Engineering Socialbot Infiltration Strategies in Twitter,” in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2015, pp. 25–32.

[16] H. Xiao *et al.*, “Is Feature Selection Secure against Training Data Poisoning ?,” *J. Mach. Learn. Res.*, vol. 37, 2015.

[17] H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran, “Deceiving Google’s Perspective API Built for Detecting Toxic Comments,” 2017. [Online]. Available: http://arxiv.org/abs/1702.08138.

[18] F. Morstatter, H. Dani, J. Sampson, and H. Liu, “Can One Tamper with the Sample API ? - Toward Neutralizing Bias from Spam and Bot Content,” *WWW’16 Companion*, pp. 81–82, 2016.

[19] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu, “Detection of Adversarial Training Examples in Poisoning Attacks through Anomaly Detection,” pp. 1–10, 2018.

[20] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “Social Fingerprinting: detection of spambot groups through DNA-inspired behavioral modeling.,” *IEEE Trans. Dependable Secur. Comput.*, 2017.

[21] O. Varol, “Bot Repository,” 2018. [Online]. Available: https://botometer.iuni.iu.edu/bot-repository/datasets.html.

[22] C. Yang, R. C. Harkreader, and G. Gu, “Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers,” *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 8, pp. 1280–1293, 2013.

[23] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “Fame for sale: efficient detection of fake Twitter followers.,” *Decis. Support Syst.*, vol. 80, pp. 56–71, 2015.

[24] Twitter Staff, “Upcoming changes to the developer platform,” 2018. [Online]. Available: https://twittercommunity.com/t/upcoming-changes-to-the-developer-platform/104603.

[25] “User object - Twitter Developer Platform.” [Online]. Available: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/user-object.html. [Accessed: 17-Mar-2019].

[26] “About Twitter Verified Accounts.” [Online]. Available: https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts. [Accessed: 17-Mar-2019].

[27] “About public and protected Tweets.” [Online]. Available: https://help.twitter.com/en/safety-and-security/public-and-protected-tweets. [Accessed: 17-Mar-2019].

[28] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, “Support vector machines under adversarial label contamination,” *Neurocomputing*, vol. 160, pp. 53–62, 2015.

[29] B. Biggio, B. Nelson, and P. Laskov, “Poisoning Attacks against SVMs,” in *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[30] A. Gupta, H. Lamba, and P. Kumaraguru, “$1.00 per RT #BostonMarathon #PrayForBoston: Analyzing fake content on twitter,” in *eCrime Researchers Summit, eCrime*, 2013.

Appendix 1 – Data Analysis

Table 11. Calculated F1 Score for Excluding Each Feature, for Detecting Fake-Followers.

Feature	F1 Score without
Friends_count	0.9584701
favourites_count	0.9610283
Lang	0.9612841
Created_at	0.9613791
Statuses_count	0.9627992
Listed_count	0.9642820
Time_zone	0.9651803
Followers_count	0.9653676
Geo_follow_protect_verify	0.9673579
Utc_offset	0.9700938

Table 12. Fisher Score of Each Feature, for Detecting Fake-Followers.

Feature	Fisher Score
Geo_follow_protect_verify	1.2
Time_zone	0.62
statuses_count	0.3
favourites_count	0.16
friends_count	0.027
Created_at	0.024
Listed_count	0.015
Utc_offset	0.013
lang	0.0078
Followers_count	0.0064

Table 13. Calculated F1 Score for Excluding Each Feature, for Detecting Spambots.

Feature	F1 Score without
favourites_count	0.8945241
Utc_offset	0.9224079
Statuses_count	0.9261423
friends_count	0.9352764
Followers_count	0.9357285
lang	0.9362008
Listed_count	0.9362119
Created_at	0.9365591
Geo_follow_protect_verify	0.9371561
Time_zone	0.9481067

Table 14. Fisher Score of Each Feature, for Detecting Spambots.

Feature	Fisher Score
Created_at	0.69
Lang	0.42
Statuses_count	0.2
Favourites_count	0.16
Time_zone	0.14
Utc_offset	0.069
Geo_follow_protect_verify	0.058
Friends_count	0.0011
Listed_count	0.00035
Followers_count	0.000032

Appendix 2 – Label Flipping

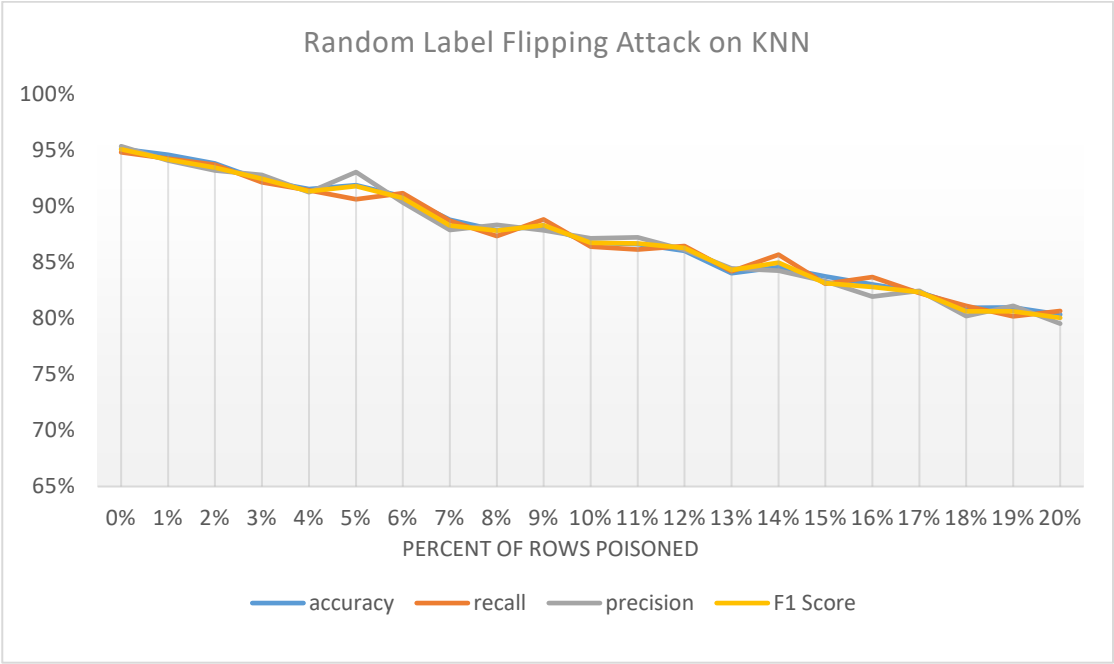

Figure 21. Results of Random Label Flipping Attack on KNN-trained Model for Detecting Fake-Followers.

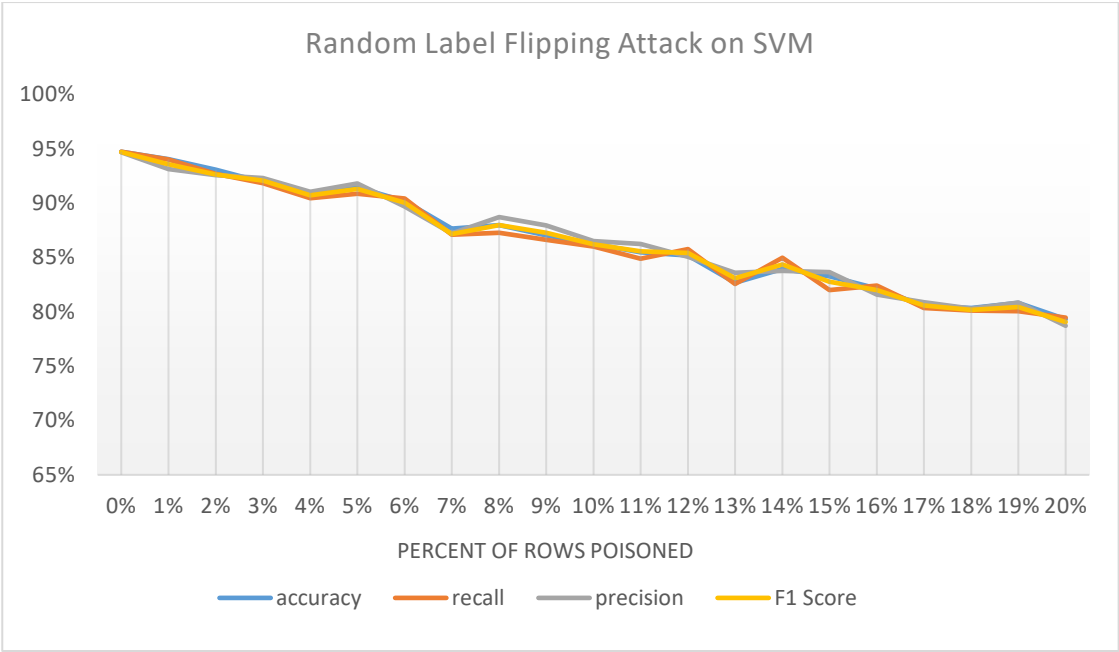

Figure 22. Results of Random Label Flipping Attack on SVM-trained Model for Detecting Fake-Followers.

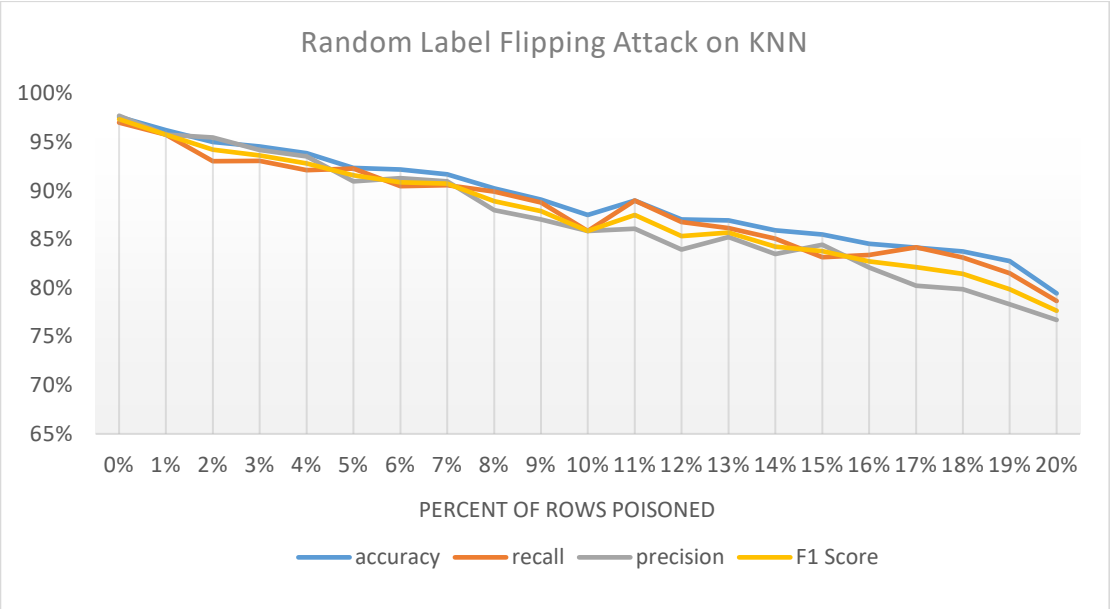

Figure 23. Results of Random Label Flipping Attack on KNN-trainede Model for Detecting Spambots.

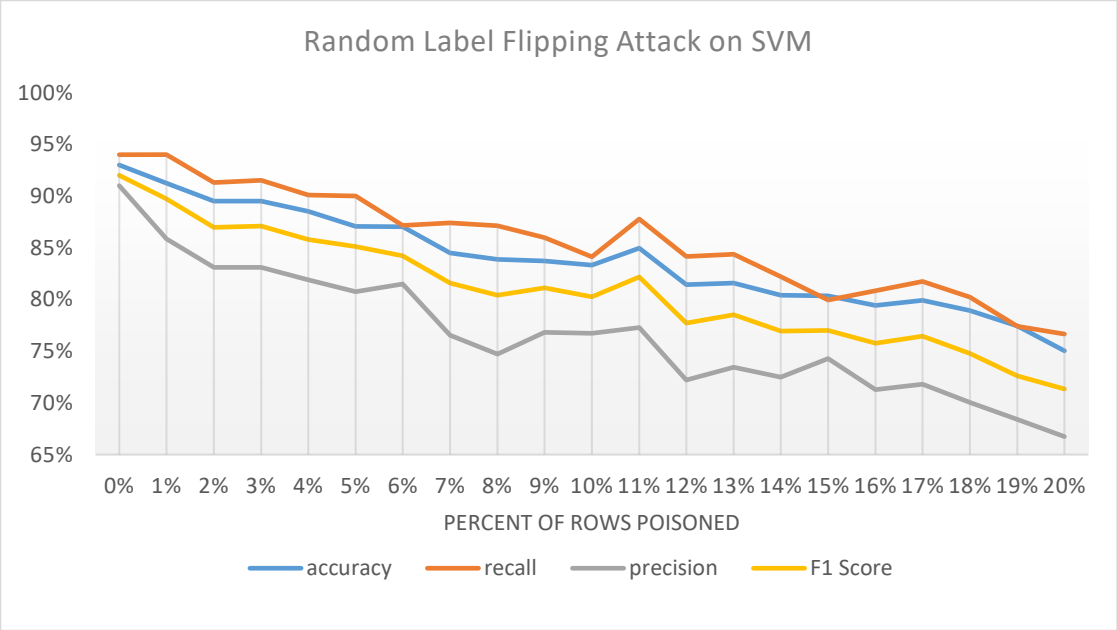

Figure 24. Results of Random Label Flipping Attack on SVM-trained Model for Detecting Spambots.

Appendix 3 – Comparison Charts

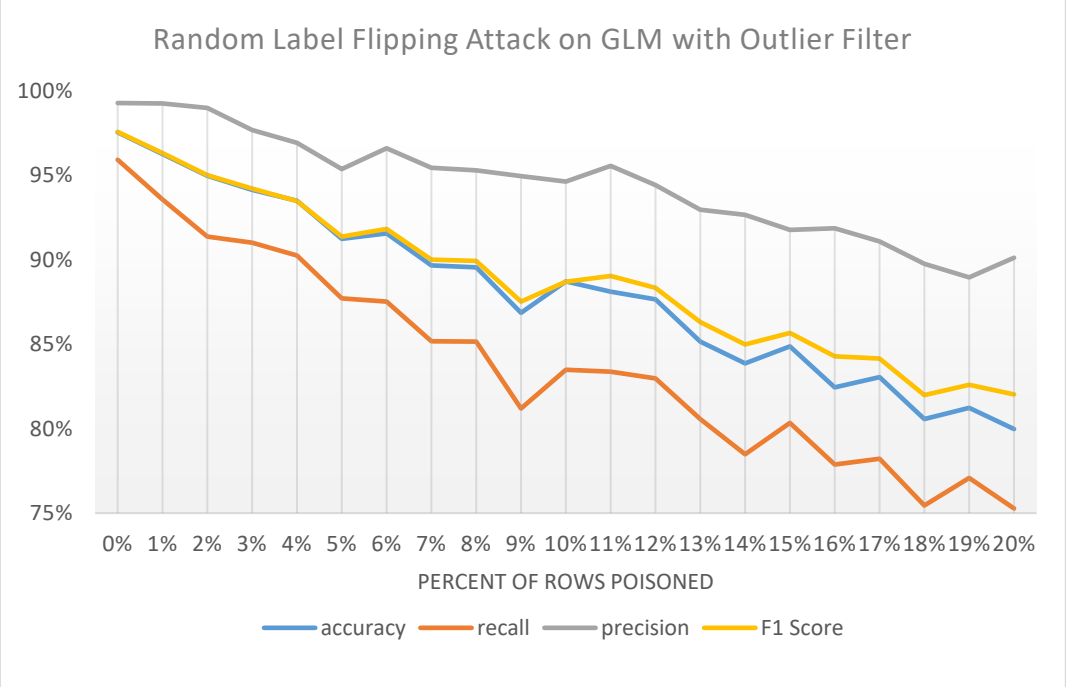

Figure 25. Results of Random Label Flipping Attack with Outlier Filter on GLM-trained Model for Detecting Fake-followers

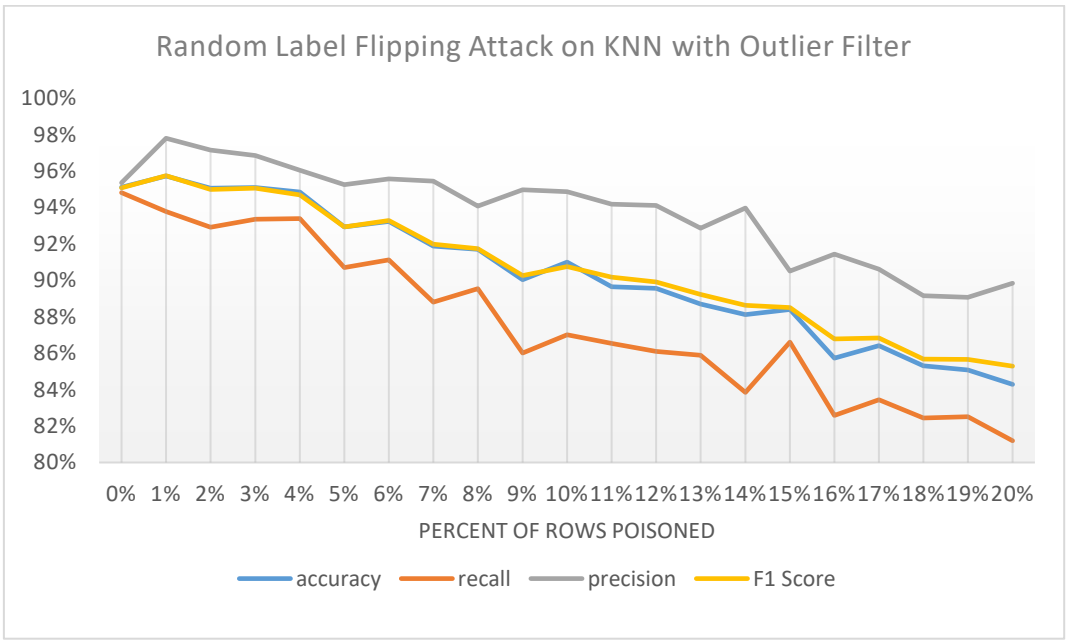

Figure 26. Results of Random Label Flipping Attack with Outlier Filter on KNN-trained Model for Detecting Fake-followers

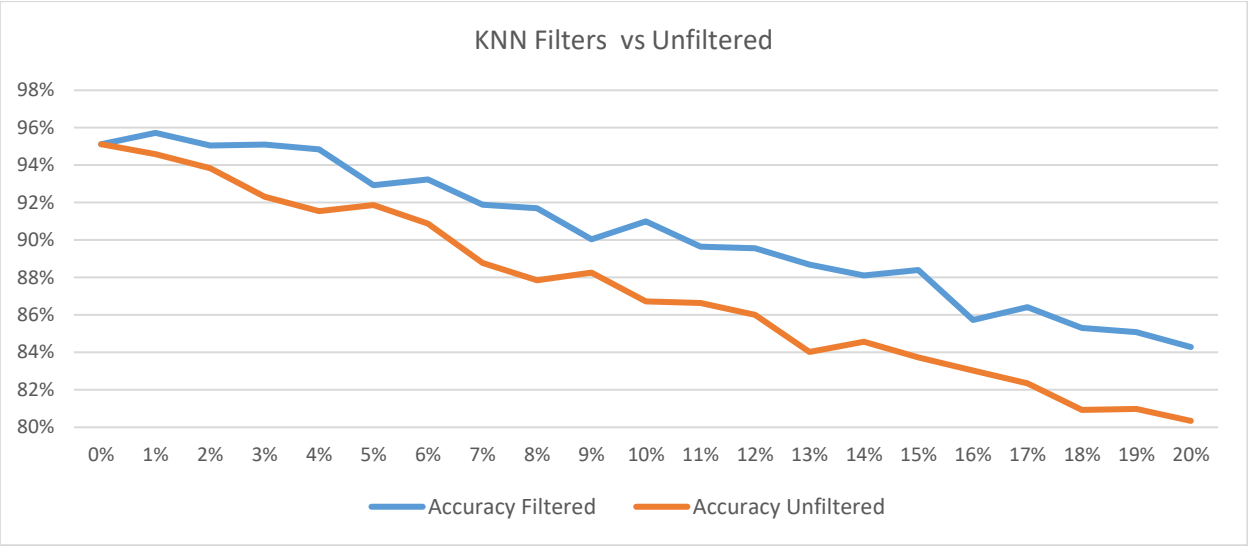

Figure 27. Accuracy of KNN-trained Models for detecting Fake-followers, poisoned by Random Label Flipping Attack, With & Without Outlier Filter.

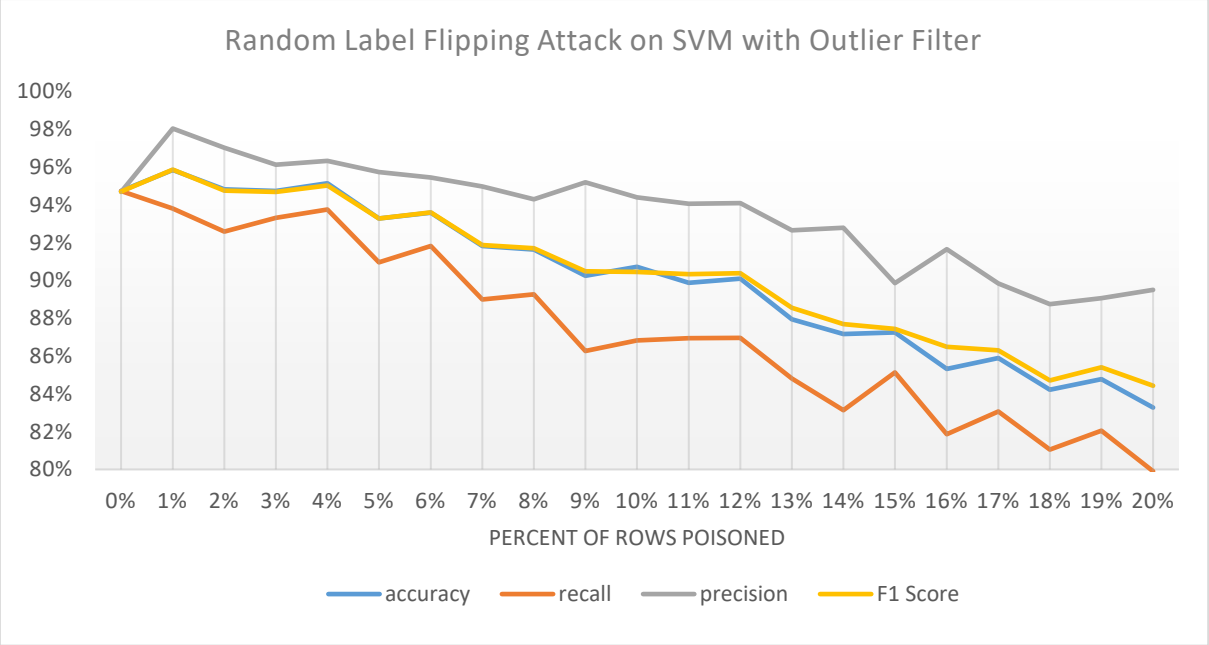

Figure 28. Results of Random Label Flipping Attack with Outlier Filter on SVM-trained model for detecting Fake Followers

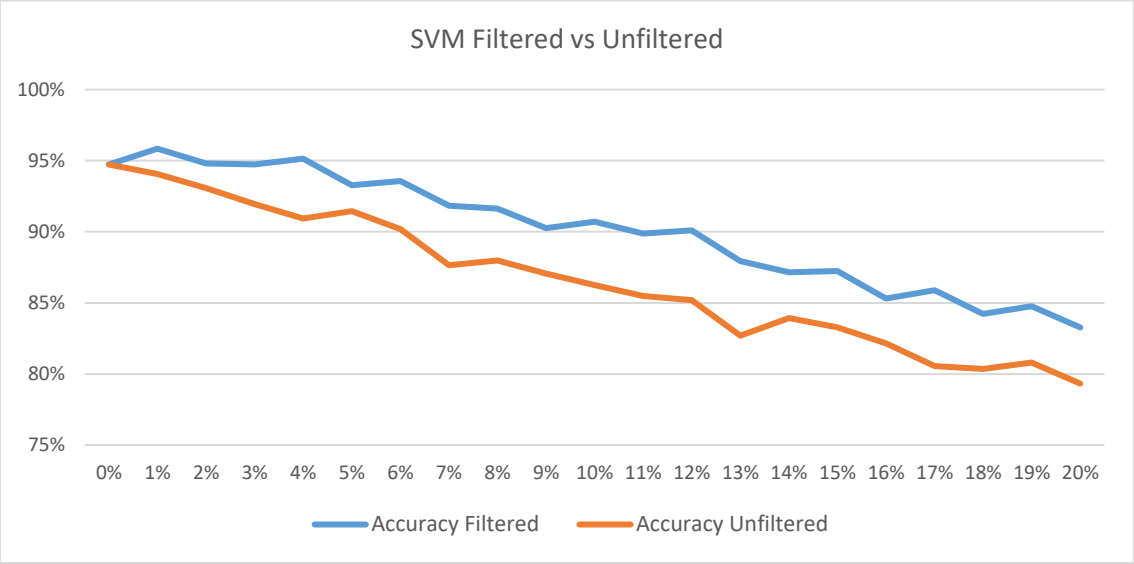

Figure 29. Accuracy of SVM-trained Models for detecting Fake-followers, poisoned by Random Label Flipping Attack, With & Without Outlier Filter.

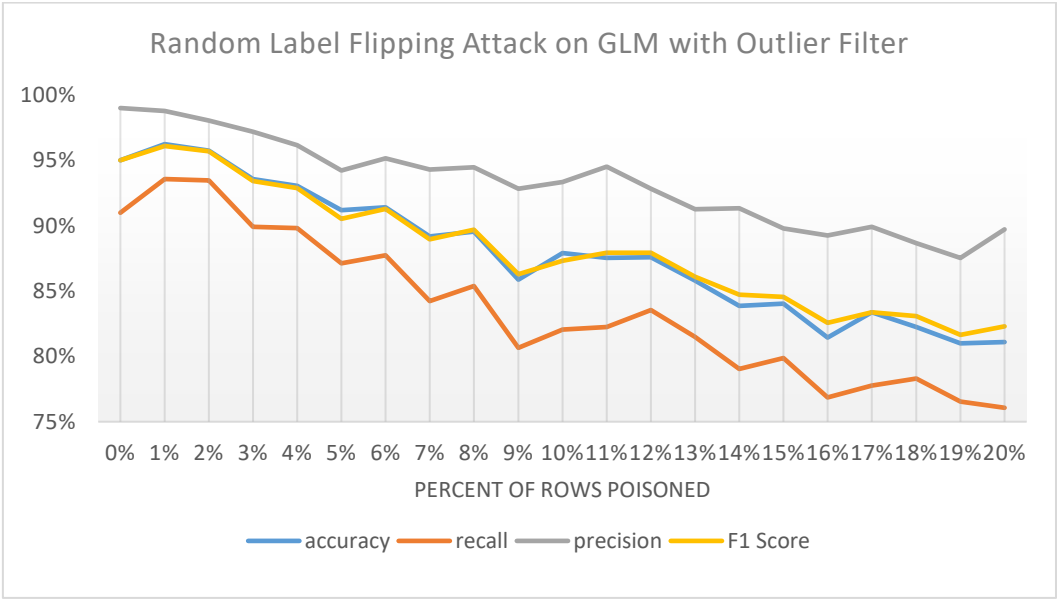

Figure 30. Results of Random Label Flipping Attack with Outlier Filter on GLM-trained model for detecting Spambots

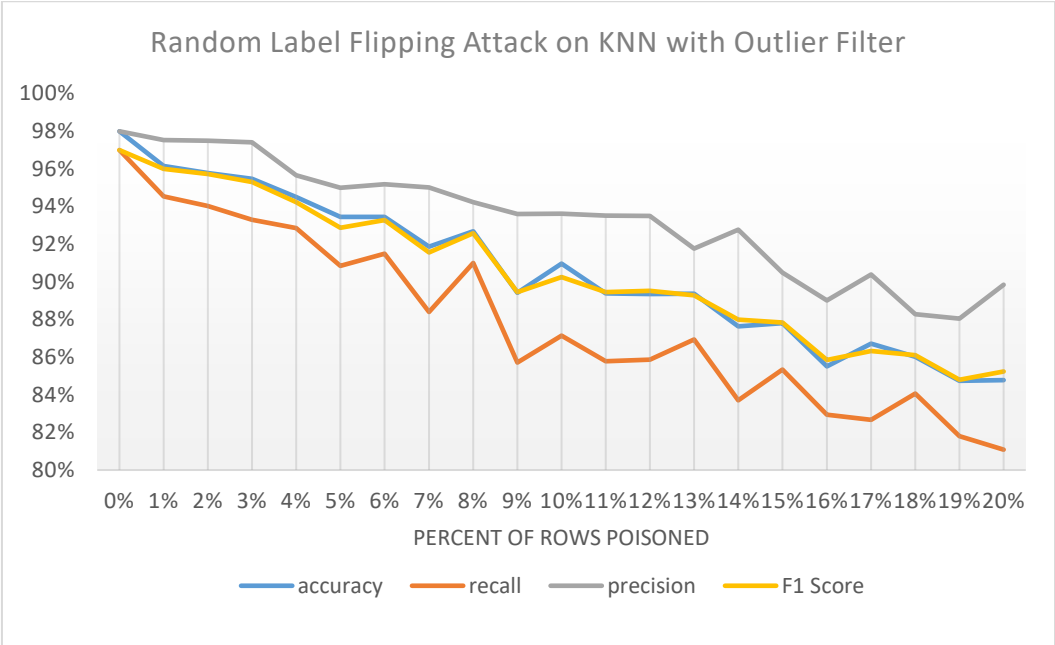

Figure 31. Results of Random Label Flipping Attack with Outlier Filter on KNN-trained model for detecting Spambots

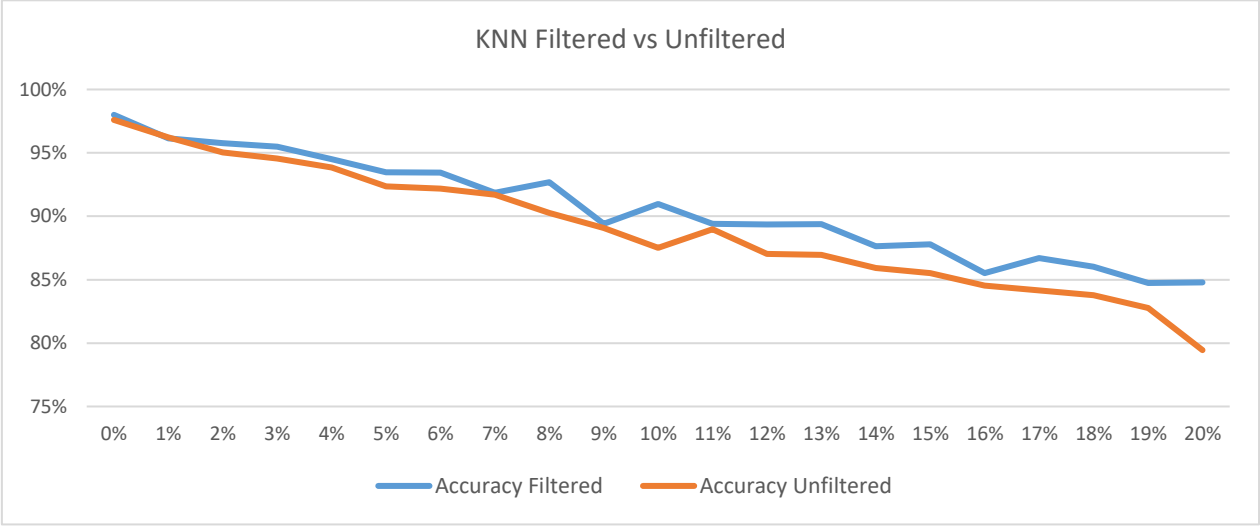

Figure 32. Accuracy of KNN-trained Models for detecting Spambots, poisoned by Random Label Flipping Attack, With & Without Outlier Filter.

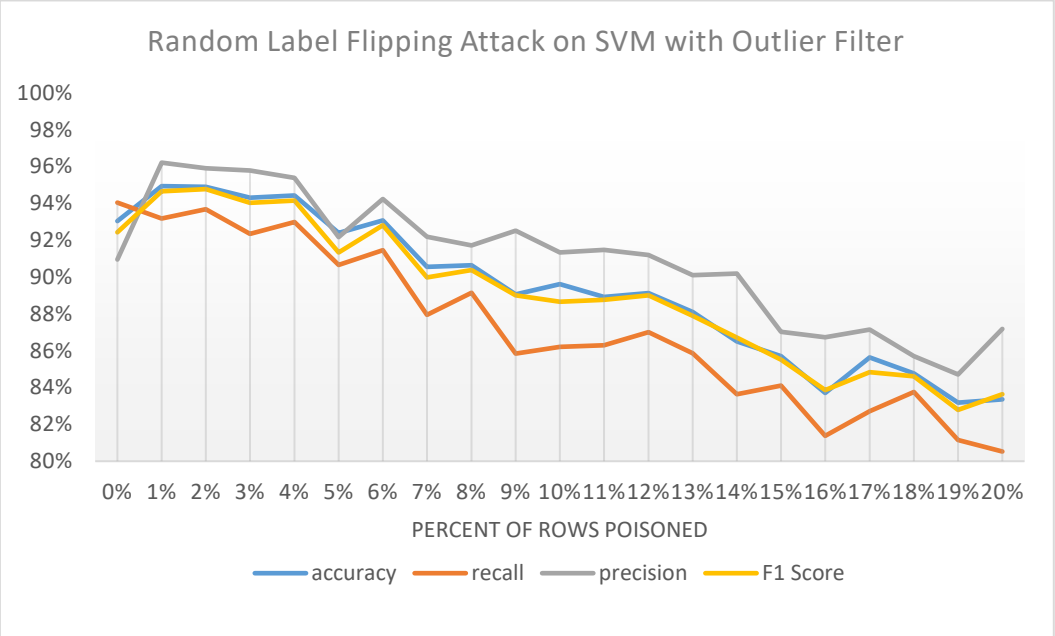

Figure 33. Results of Random Label Flipping Attack with Outlier Filter on SVM-trained model for detecting Spambots

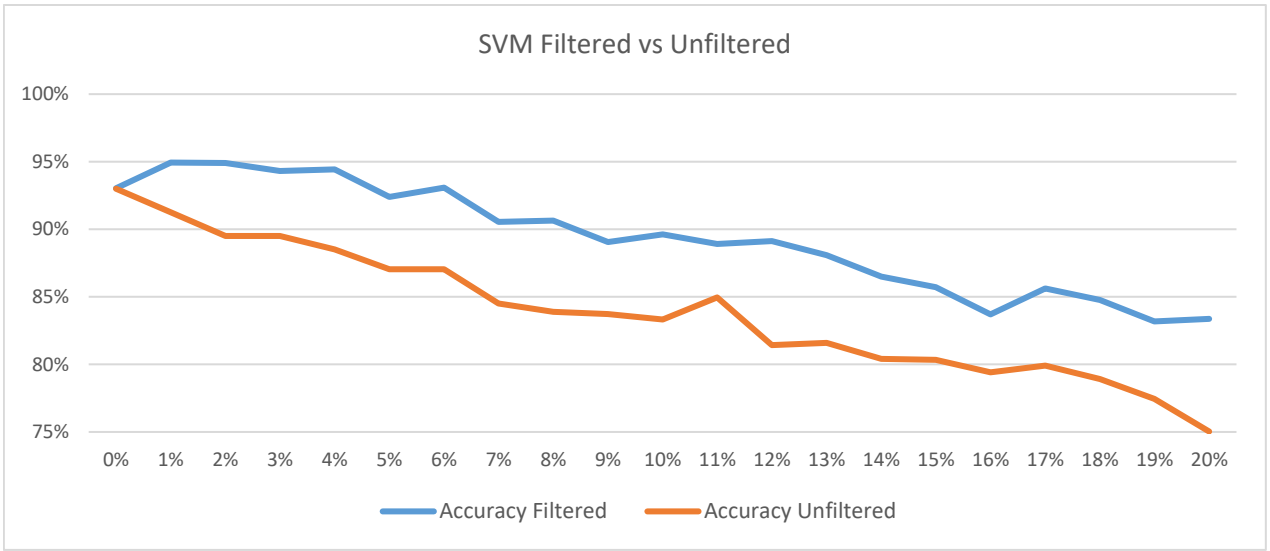

Figure 34. Accuracy of SVM-trained Models for detecting Spambots, poisoned by Random Label Flipping Attack, With & Without Outlier Filter.

Appendix 4 – Poisoned Surfaces

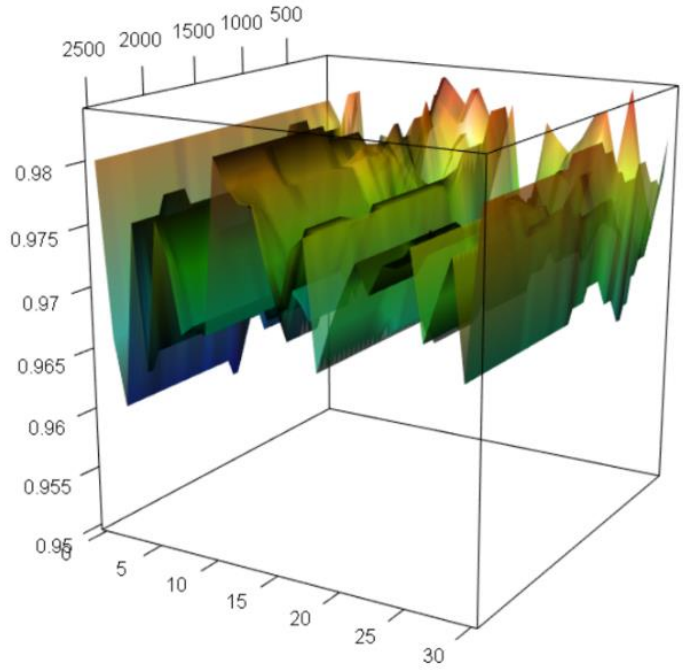

Figure 35. F1 Score of GLM model trained to detect Fake-followers after Favourites_count is poisoned.

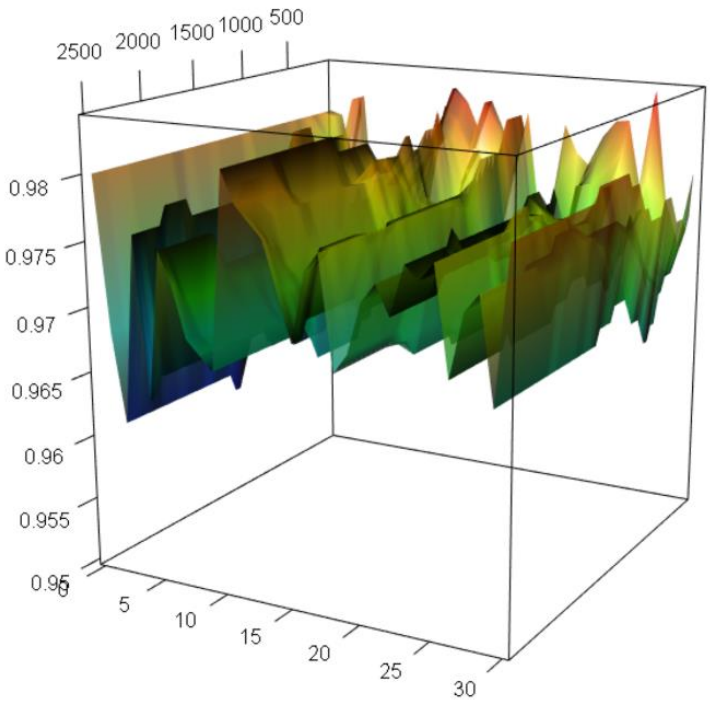

Figure 36. Accuracy of GLM model trained to detect Fake-followers after Favourites_count is poisoned.

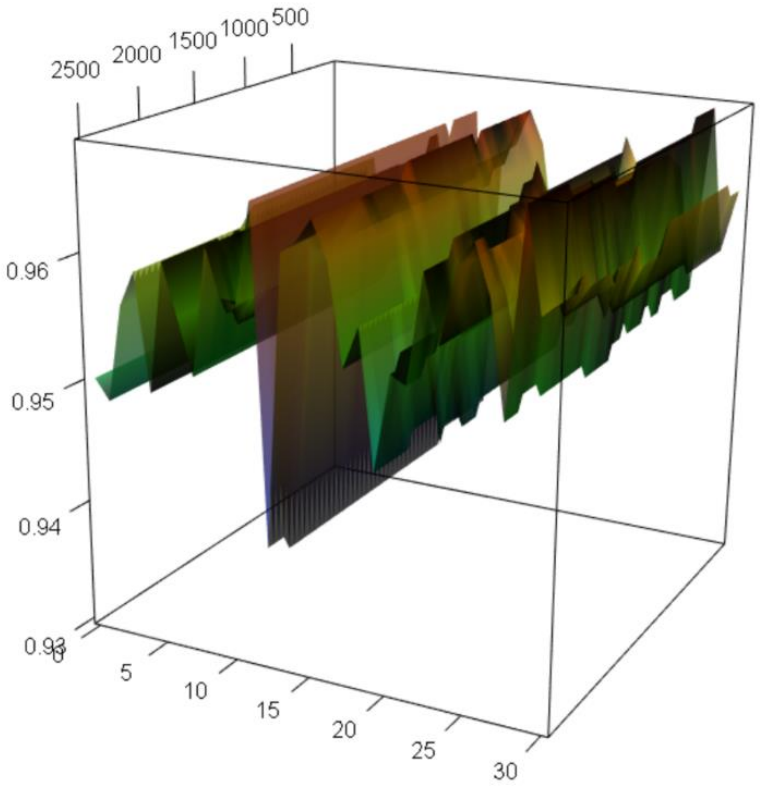

Figure 37. F1 Score of KNN Model trained to detect Fake-followers after Favourites_count is poisoned.

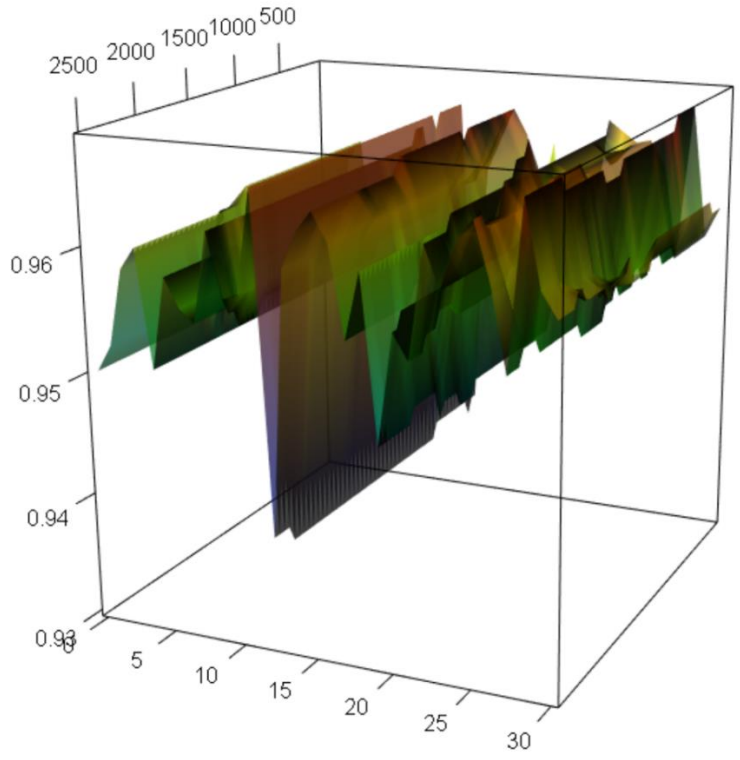

Figure 38. Accuracy of KNN model trained to detect Fake-followers after Favourites_count is poisoned.

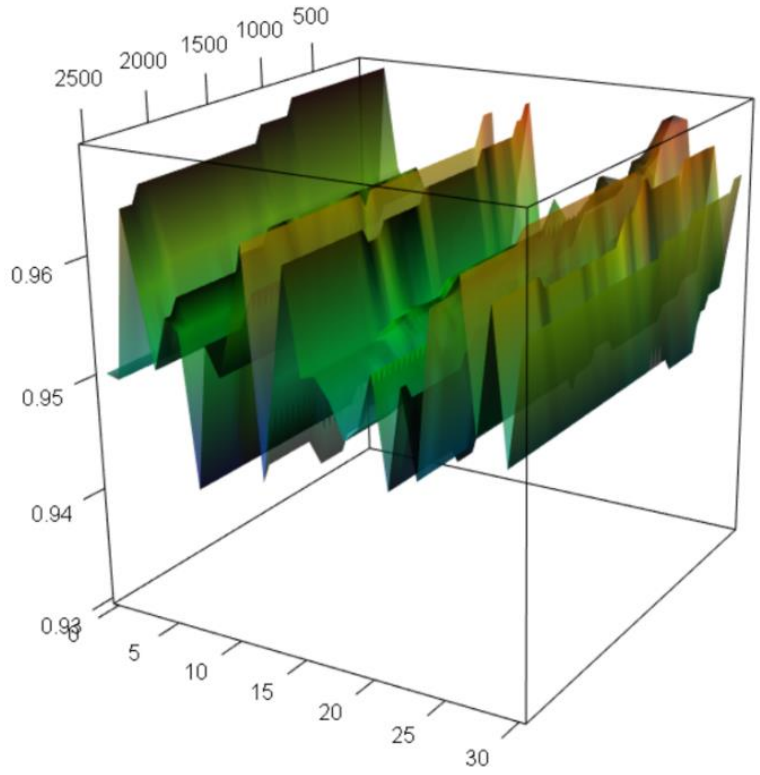

Figure 39. F1 Score of SVM Model trained to detect Fake Followers after Favourites_count is poisoned

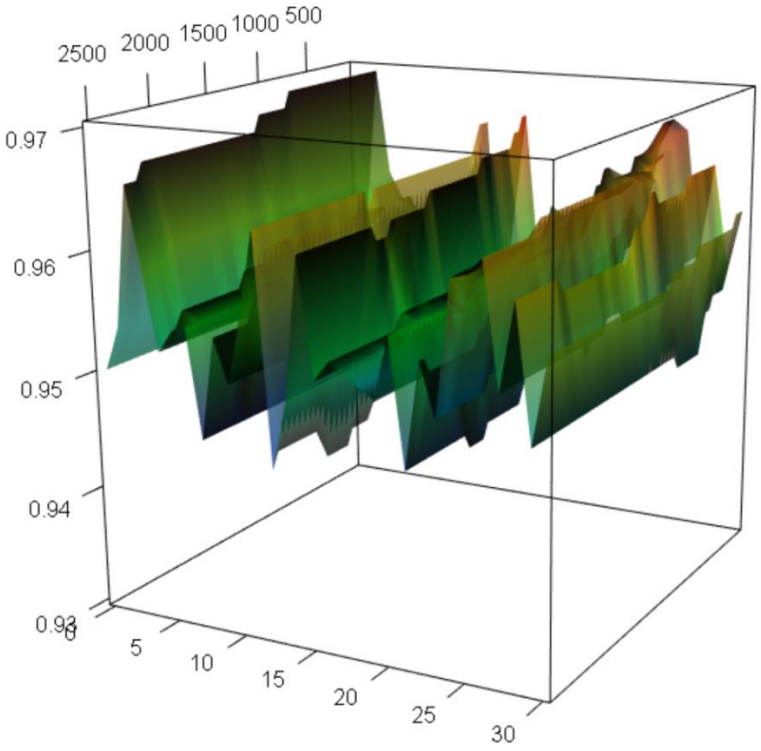

Figure 40. Accuracy of SVM Model trained to detect Fake-followers after Favourites_count is poisoned.

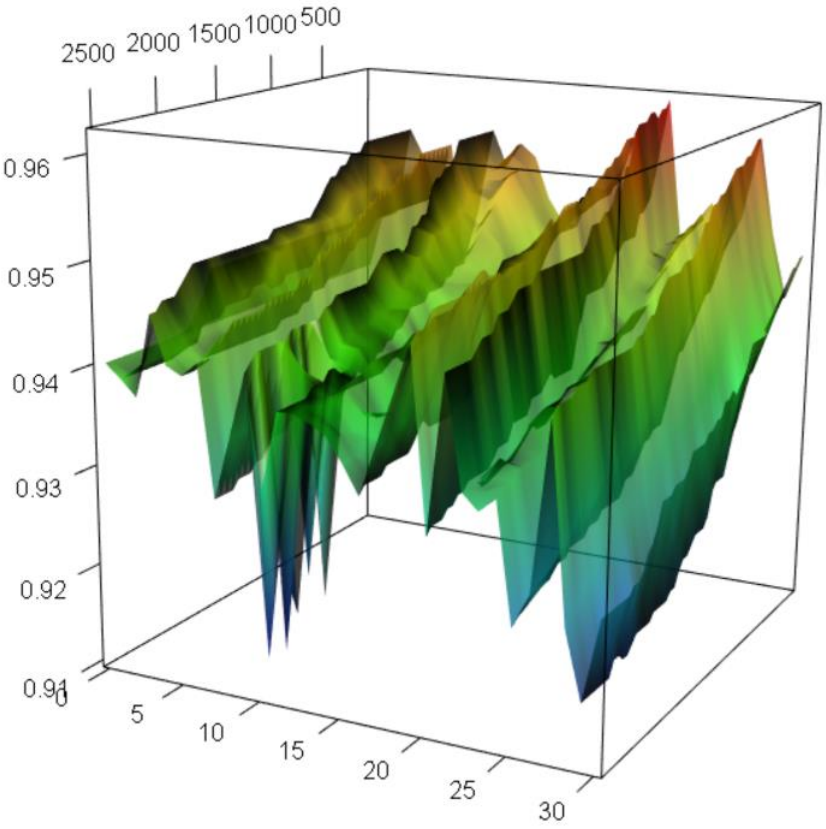

Figure 41. F1 Score of GLM Model trained to detect Social-bots after Favourites_count is poisoned.

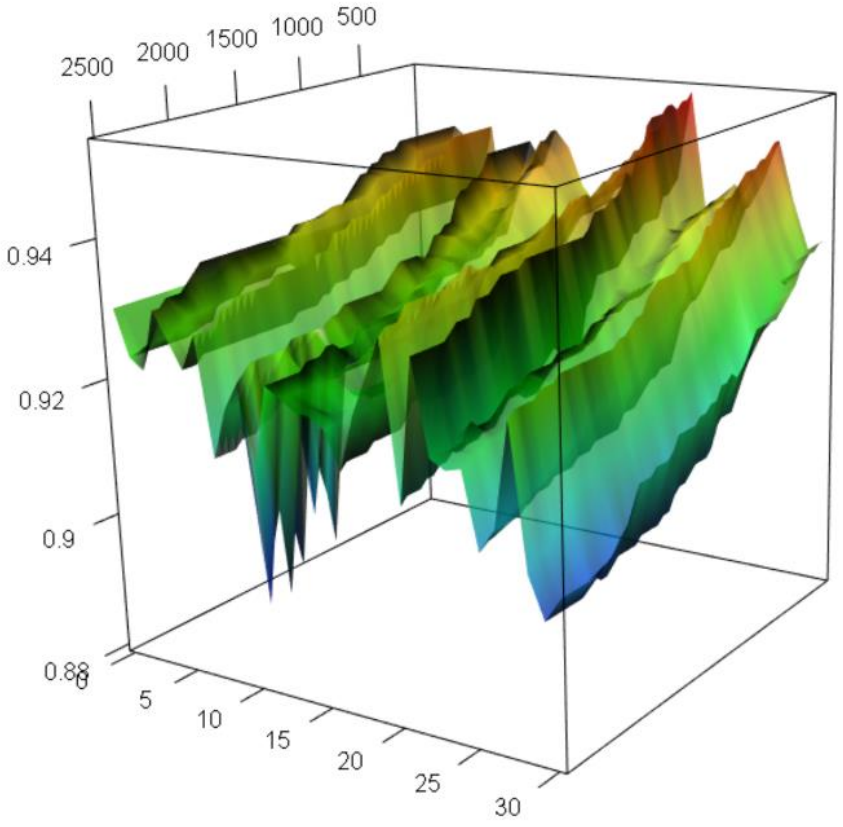

Figure 42. Accuracy of GLM model trained to detect Social-bots after Favourites_count is poisoned.

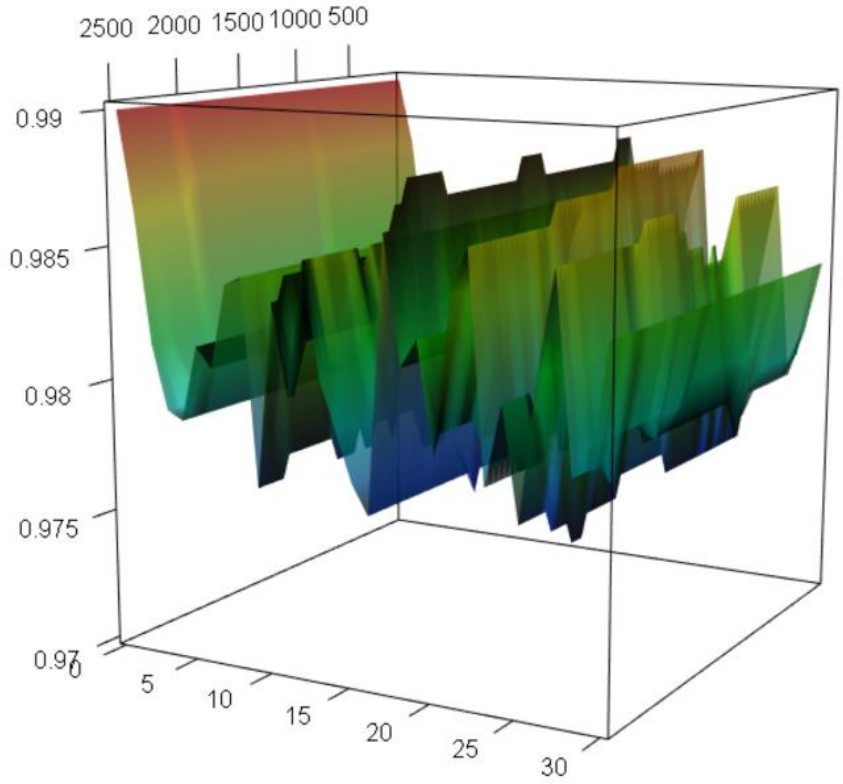

Figure 43. F1 Score of KNN Model trained to detect Social-bots after Favourites_count is poisoned.

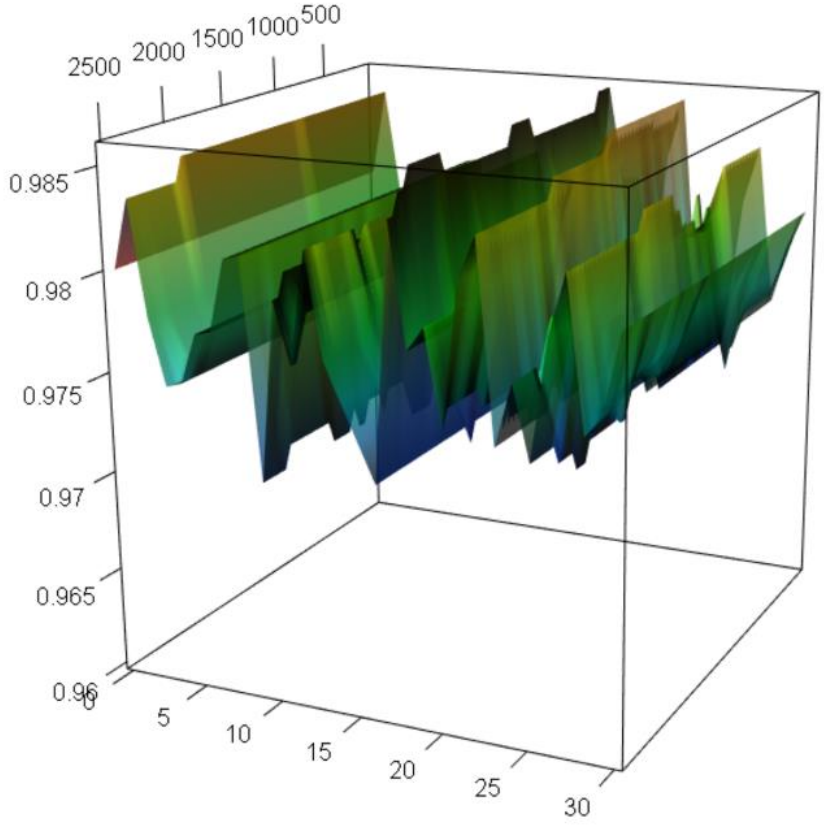

Figure 44. Accuracy of KNN model trained to detect Social-bots after Favourites_count is poisoned.

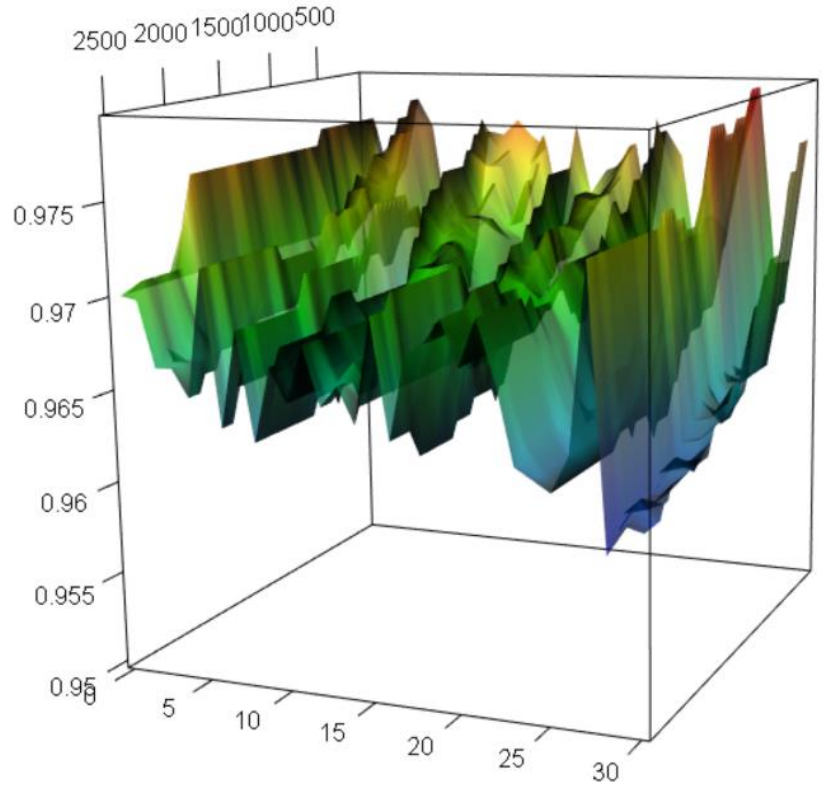

Figure 45. F1 Score of SVM Model trained to detect Social-bots after Favourites_count is poisoned.

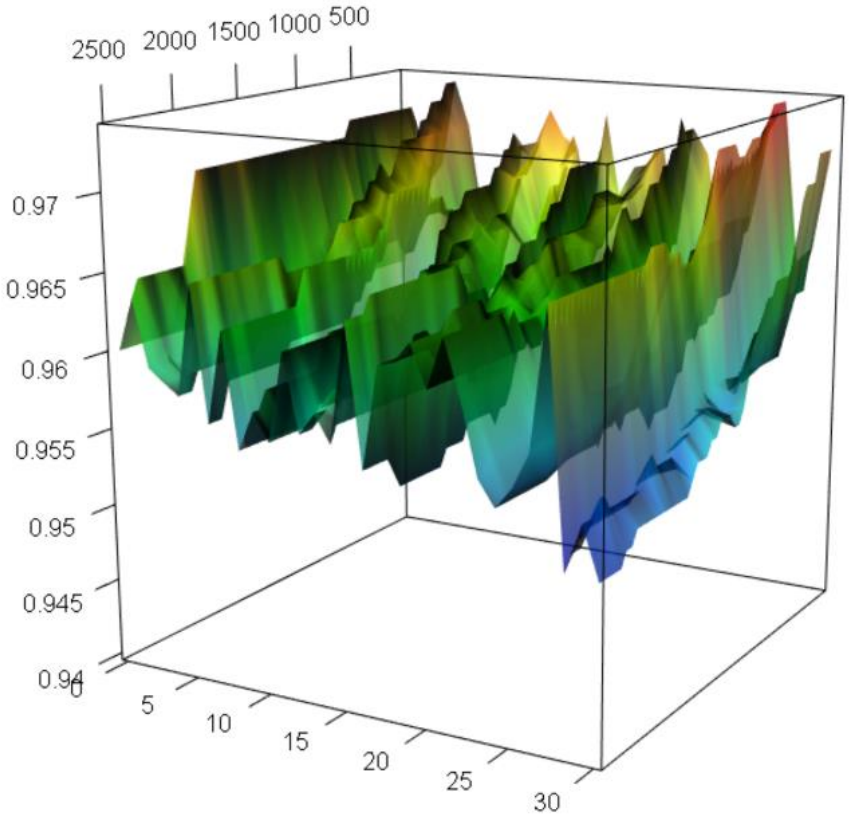

Figure 46. Accuracy of SVM model trained to detect Social-bots after Favourites_count is poisoned.

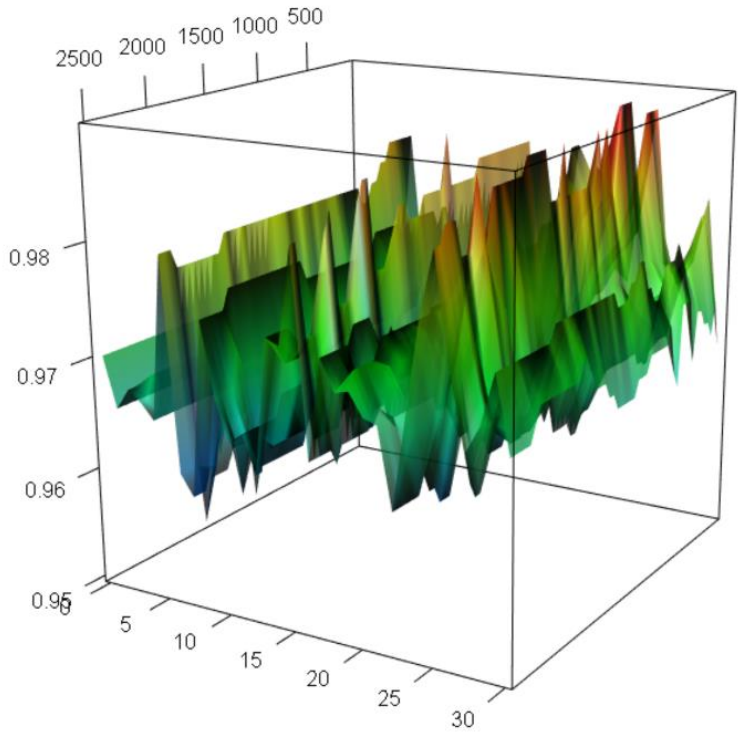

Figure 47. F1 Score of KNN Model trained to detect Spambots after Favourites_count is poisoned.

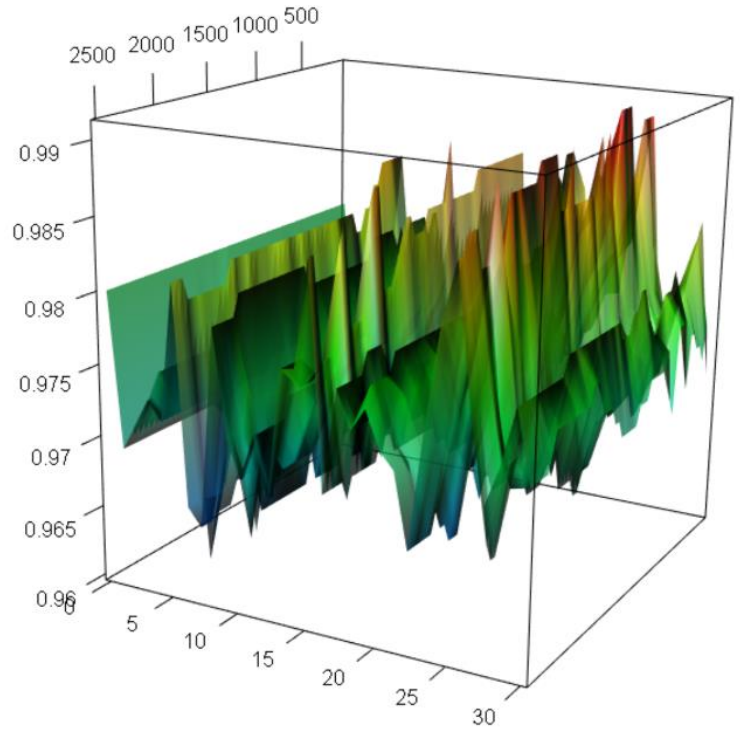

Figure 48. Accuracy of KNN model trained to detect Spambots after Favourites_count is poisoned.

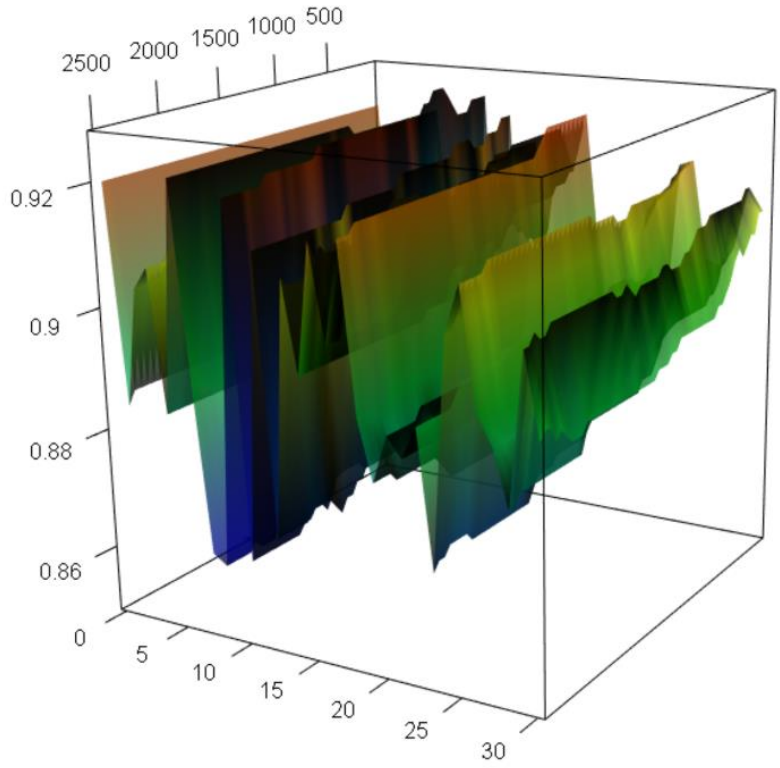

Figure 49. F1 Score of SVM Model trained to detect Spambots after Favourites_count is poisoned.

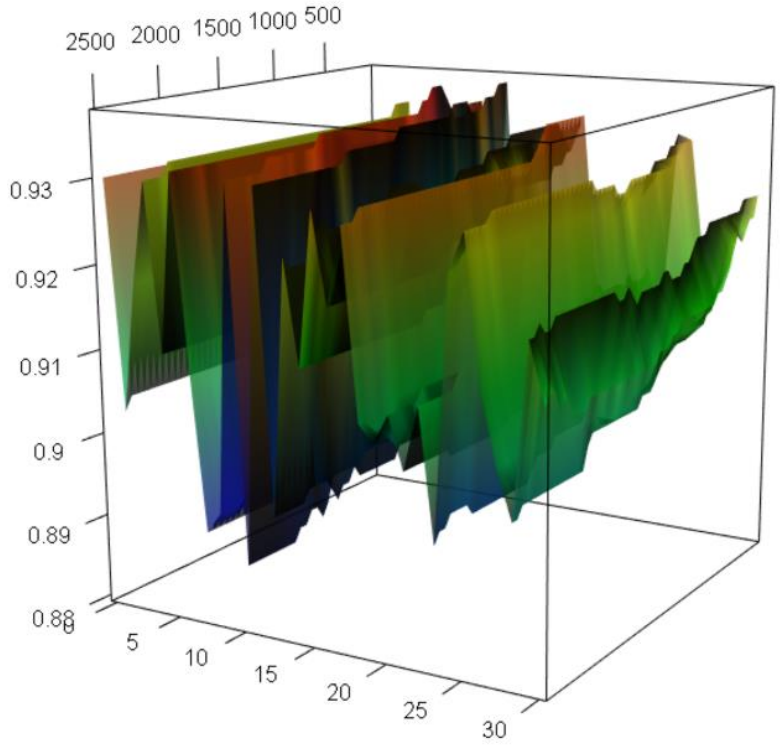

Figure 50. Accuracy of SVM model trained to detect Spambots after Favourites_count is poisoned.

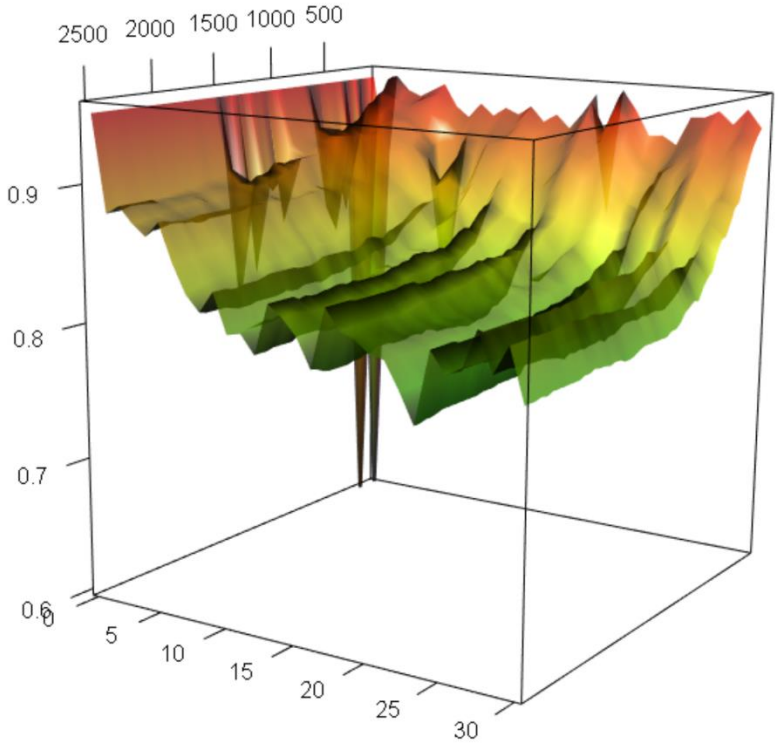

Figure 51. F1 Score of GLM model trained to detect Spambots after favourites_count is poisoned

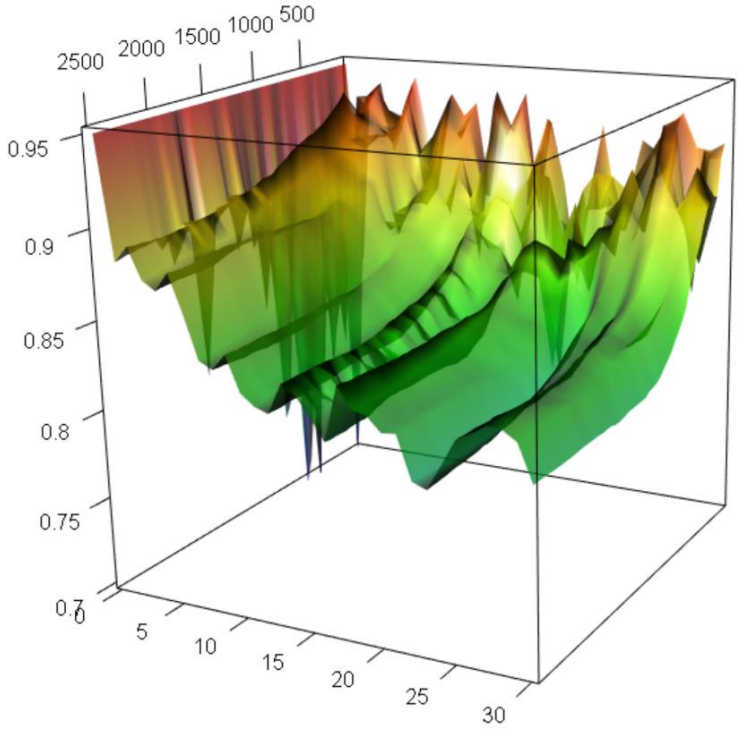

Figure 52. Average F1 Score of Ten GLM Models trained to detect Spambots

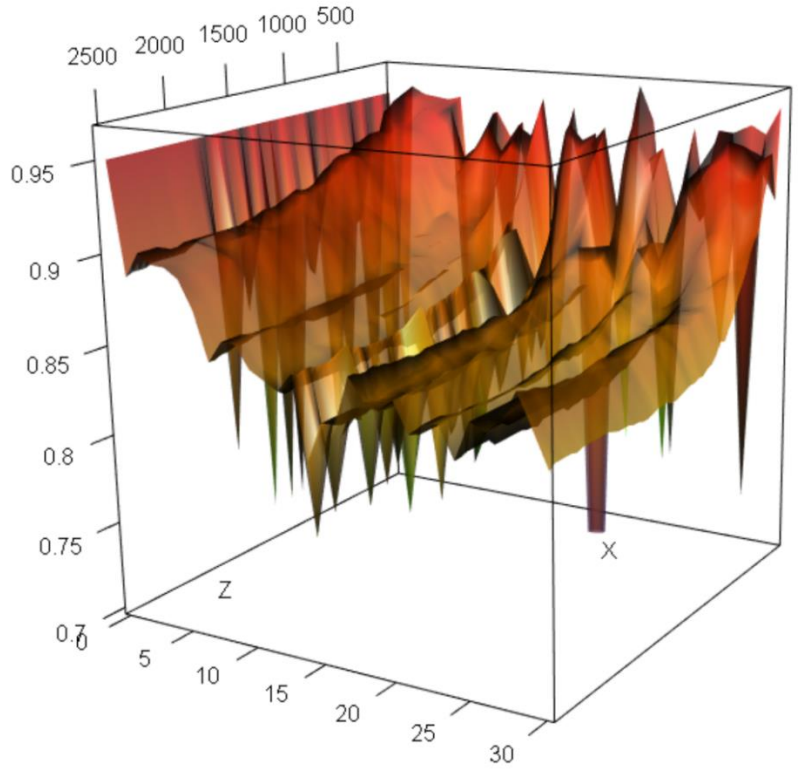

Figure 53. Surface representing Accuracy of a GLM-trained model for detecting Spambots with the 1st alternate set of accounts poisoned.

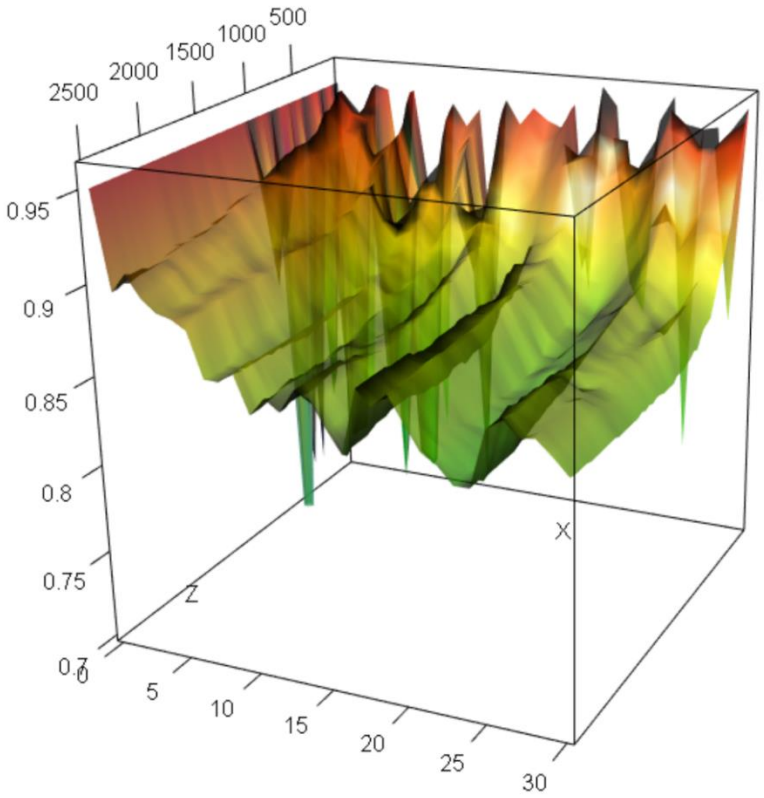

Figure 54. Surface representing Accuracy of a GLM-trained model for detecting Spambots with the 2nd alternate set of accounts poisoned.

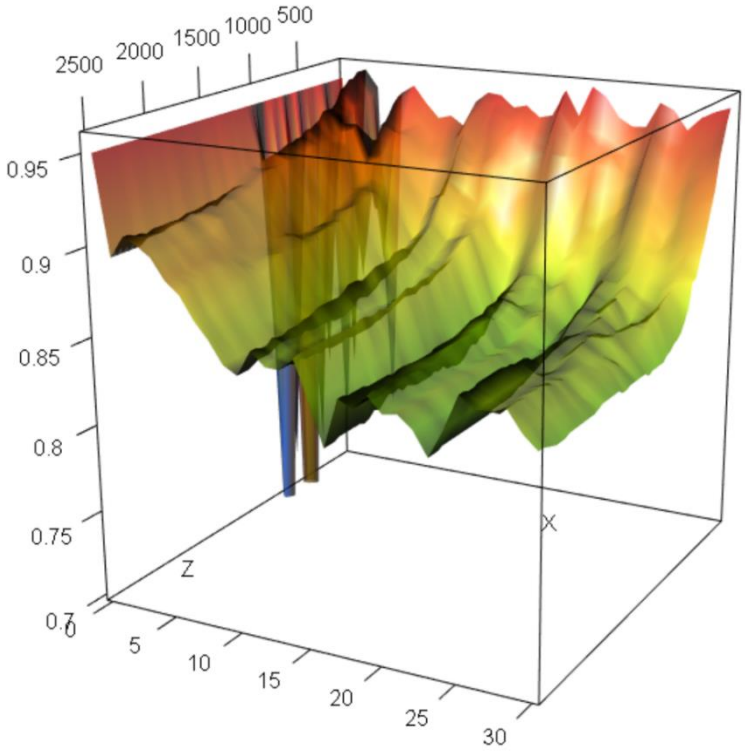

Figure 55. Surface representing Accuracy of a GLM-trained model for detecting Spambots with the 3rd alternate set of accounts poisoned.

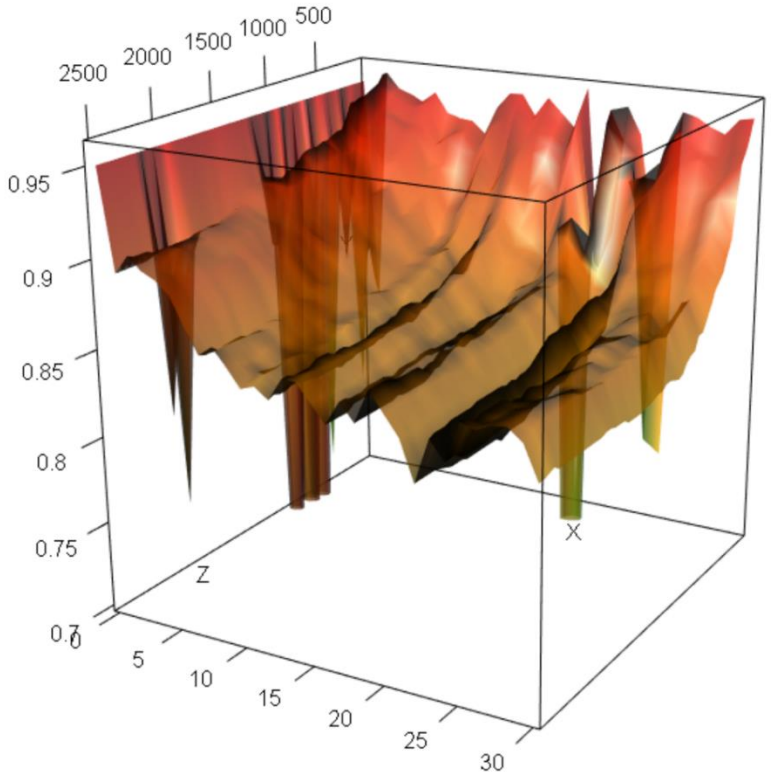

Figure 56. Surface representing Accuracy of a GLM-trained model for detecting Spambots with the 4th alternate set of accounts poisoned.

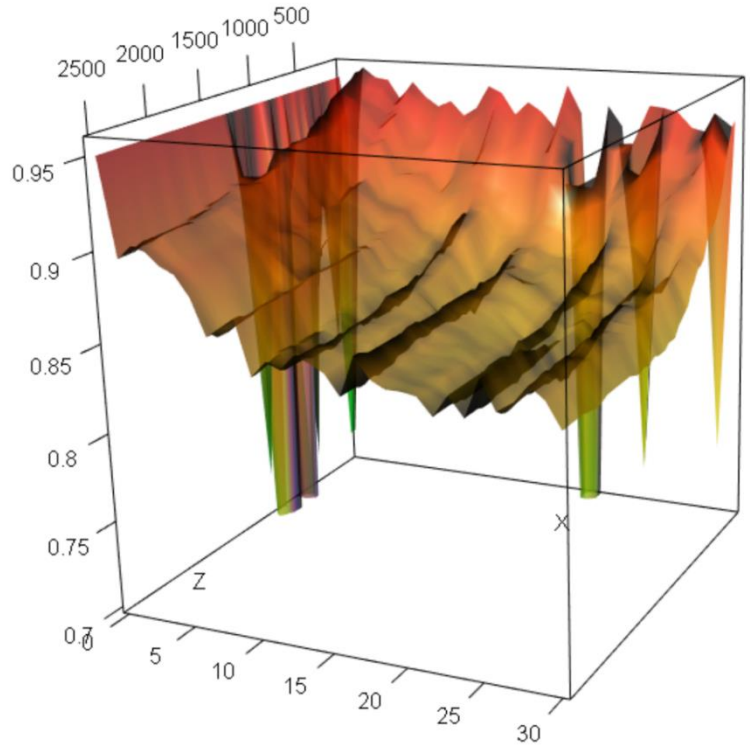

Figure 57. Surface representing Accuracy of a GLM-trained model for detecting Spambots with the 5th alternate set of accounts poisoned.

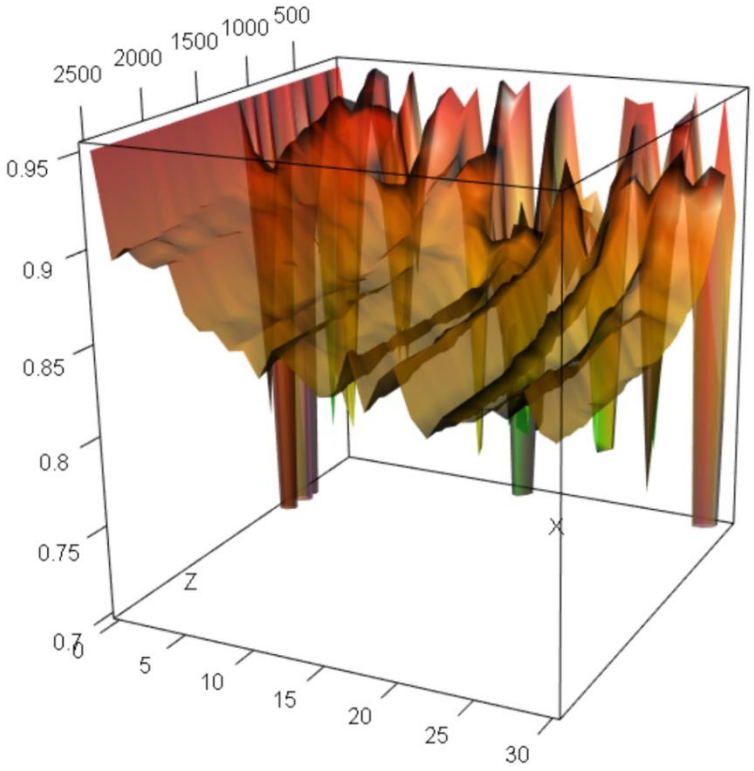

Figure 58. Surface representing Accuracy of a GLM-trained model for detecting Spambots with the 6th alternate set of accounts poisoned.

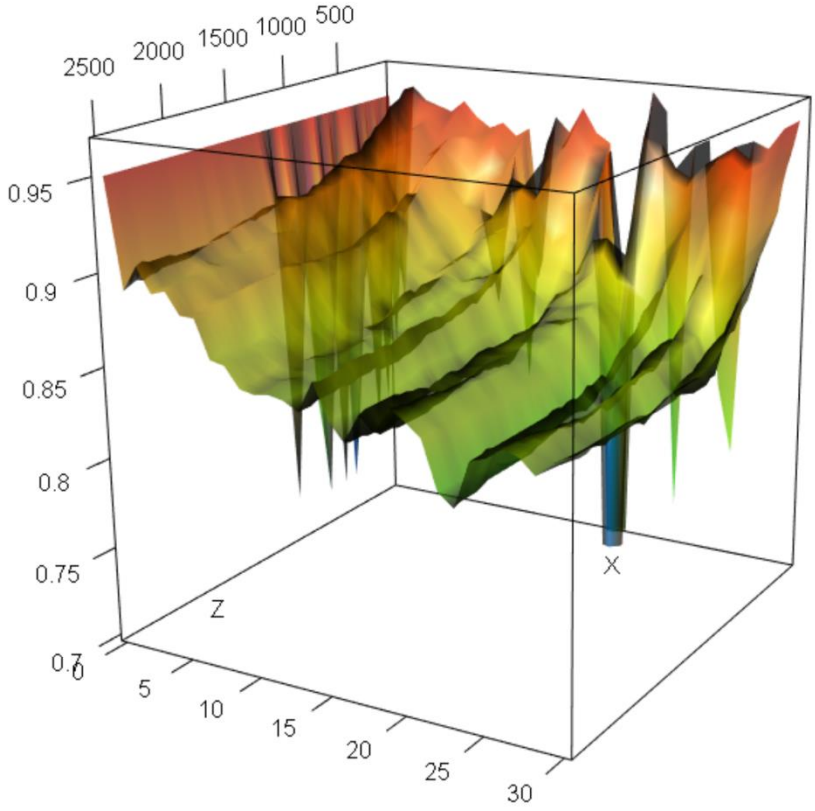

Figure 59. Surface representing Accuracy of a GLM-trained model for detecting Spambots with the 7th alternate set of accounts poisoned.

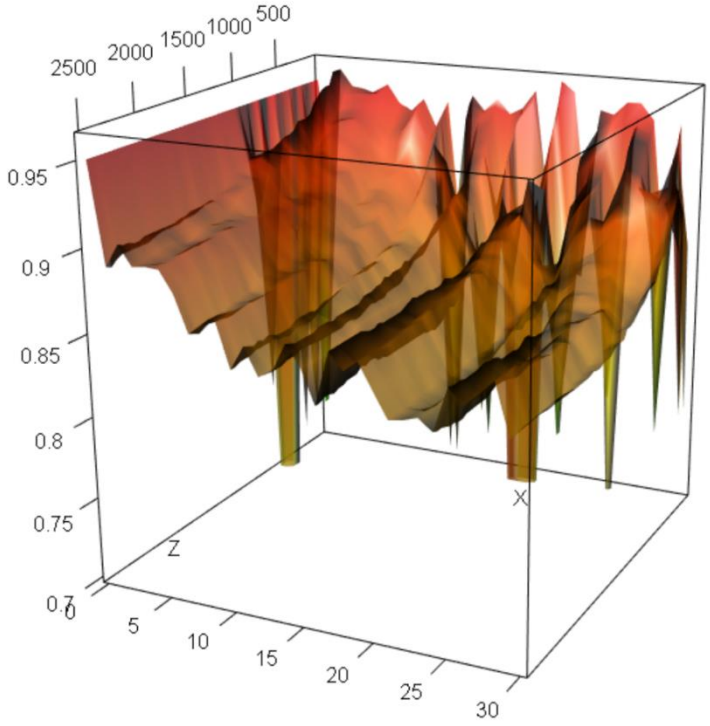

Figure 60. Surface representing Accuracy of a GLM-trained model for detecting Spambots with the 8th alternate set of accounts poisoned.

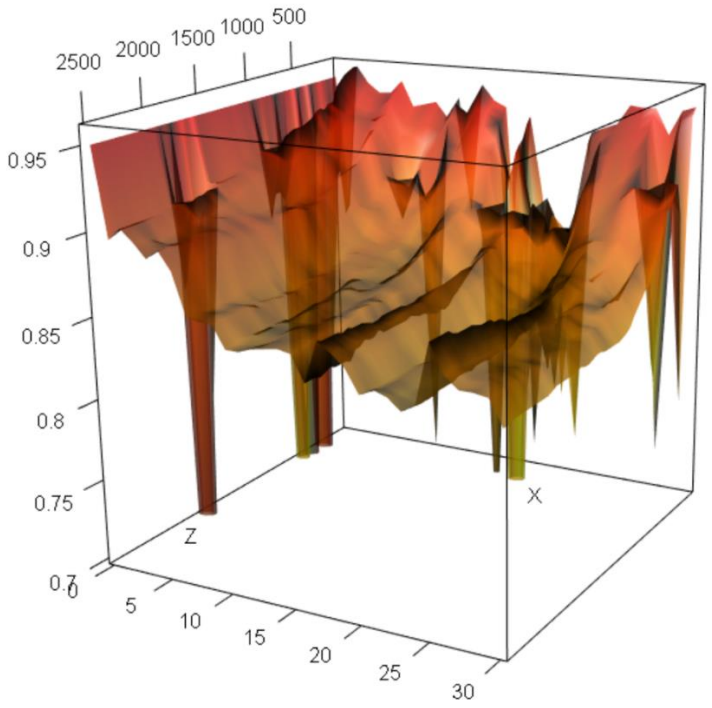

Figure 61. Surface representing Accuracy of a GLM-trained model for detecting Spambots with the 9th alternate set of accounts poisoned.