Anonymization of Data MSDS 7349 Project Final Draft

Alex Frye, Michael Smith, and Lindsay Vitovsky

Abstract—In the name of scientific progress, an amazing number of data sets have been released as a way to promote collaboration and development of the data science community. This paper critically reviews an array of anonymization theories and techniques and applying them to a fake data set, with the goal of helping others weigh the cost, benefit, and, most importantly, the ethics of anonymizing data. We test these techniques on a fake dataset, taking an in-depth look at with what ease someone can be armed with a small amount of data and yet cause serious ramifications.

Index Terms—micro-data, k-anonymization, quasi-identifiers, unique identifiers, personal ly identifiable information (PII), high-dimensional data sets, sparcity

1 Introduction

On September 18, 2009, Netflix announced the winner of its Netflix Prize competition, a contest that asked contributors to improve upon its existing algorithm that predicted subscribers' future movie ratings. With a prize of $1,000,000, the competition certainly attracted many groups from all over the globe. In the end, Netflix received 44,014 submissions from 41,305 teams, all who had analyzed the seemingly anonymized dataset of over 100,000,000 movie ratings by Netflix subscribers.

According to the Netflix Prize website, which was still live at the time of this paper, "The data were collected between October, 1998 and December, 2005 and reflect the distribution of all ratings received by Netflix during this period. The ratings are on a scale from 1 to 5 (integral) stars. To protect customer privacy, all personal information identifying individual customers has been removed and all customer ids have been replaced by randomly-assigned ids. The date of each rating and the title and year of release for each movie are provided. No other customer or movie information is provided."

Unfortunately for Netflix, resulting from this contest was a lawsuit for releasing this data, and an investigation by the Federal Trade Commission. Apparently, one actually could identify unique users from this data set, as proven by two University of Texas researchers in 2008 [1]. To make matters worse, Netflix did not end up using the winning algorithm, citing "engineering" challenges and the fact that their model was rapidly changing from DVDs to streaming [2].

The repercussions from this anticlimax caused Netflix to cancel its Netflix Prize 2, drawing both praise and criticism. Simply reading the comments section of various online articles covering this news reveals the various viewpoints of the importance (or non-importance) of privacy in todays cyber world. [3], [4], [5]

Even if Netflix had obtained user permission to release these ratings (which they did not), those who opted in most likely would not have surmised that they could have been identified personally by a complete stranger, unaffiliated with Netflix. Certainly, Netflix would have assured that their ratings were anonymized, since that is what it believed to be the case as well. The trust that the subscribers would have placed in Netflix to better their experience, without compromising their privacy, would have been misplaced.

Today, there are several popular techniques for anonymizing data sets, but none are perfect. Certain methods may work for a particular sized set of records, only to lose its credibility as that set scales over time. Others may change the data so much that its value dramatically declines to a data scientist. Is true anonymization even possible in this cyber world? The ability to pull public information from outside sources, to then match it with private data sets, has proven to be a very dangerous risk for institutions and the individuals that have entrusted them.

After an investigation of current, popular anonymization techniques, we will then review our own attempts to anonymize a data set we created, investigating how easy or difficult it is to garner identifying information from something seemingly innocuous. Our entire experiment can be found on our project github repository [12] Ultimately, we make final recommendations to the reader as to what we believe are essential questions a company should be reflecting upon as they consider their privacy policies and procedures.

1.1 Background

1.1.1 Existing Research / Techniques

Before traveling too much further into these methods, it warrants mentioning that privacy itself has yet to be universally defined. One could say that privacy is when non-public information is made available to a third party, such as a social security number or the results from a medical test. This insinuates that the nature of the data is what determines the need for privacy. But what about those pesky ads that pop up on a smart phone? Ads are to be expected,

Alex Frye, amfrye@smu.edu
Michael Smith, michaelsmith@smu.edu
Lindsay Vitovsky, lvitovsky@smu.edu

but what if they directly reflect the search history a user performed on a completely different device? The discussion of what privacy means, particularly what a user authorizes others to collect, store and share, has yet to be answered in a world where mobile devices outnumber humans [6]. Certainly, there are individuals who would this collection of their data feels, at the very least, intrusive.

Of equal concern to the authors is the other end of the spectrum, which is the evidently voluntary surrender of the collection and use of one's data. There are a large number of users who do not seem to take issue with their non-personal data being stored or sold to other companies. The task of maintaining privacy is certainly overwhelming, and perhaps this is why so many users throw up a white flag and say, "Who cares?" Perhaps, they simply do not want to think about the amount of their data that is currently available, since there is presumably nothing that can be done about it.

Adding on to this acceptance of the release of "unimportant" data is the common misconception that less privacy means more security. When the FBI very publicly condemned Apple for not helping it unlock an iPhone that belonged to a gunman who helped kill fourteen people in San Bernardino, CA in 2015, there were many people who joined the FBI in the criticism. The reality is that these keys to our privacy are not held behind a locked safe, guarded by trustworthy individuals. Often our privacy is protected simply on an agreement of trust. Not only would the FBI now have known how to unlock an iPhone, but there would have been employees of Apple that would have then been privy to the information as well. Our message here is that the bounds around privacy should not be thought of as well-defined. The questions must be asked so that those involved can properly reflect on the danger of using anonymized data and weigh it against any supposed benefit.

1.1.2 Analysis of Existing Techniques

Of particular interest for our paper were some of the more frequently employed anonymization techniques. To be clear, our definition for these techniques fall more into the realm of "attempts" as it has been well documented that anonymization has continued to elude the cyber community. We review the methods of K-anonymization, Sparse Data Sets, Perturbation, and Deletion of Data.

K-anonymization

K-anonymization is a popular anonymization technique, most widely used as a method when maintaining statistical integrity is important. By removing instances with less than k number of matching records, the data set seeks to protect the ease of finding "loners" by taking them out altogether. For example, if you have 30,000 surgery patients, and all personal information has been removed, then it is possible to have multiple instances of the same data (i.e. twelve gallbladder removal surgeries). However, as documented by Greg Loukides' team, the efficacy of this method on large sets of "anonymized data" has its challenges [7]. K-anonymization acknowledges that those records that do not appear as often are at increased risk of being re-identified, so unless that record has k number of other records like it, it is removed. While theoretically sound, the disadvantages are the computational power required on large data sets to find these counts, and the assumption that the remaining tuples are measurably more difficult to match to additional data.

Perhaps domain comes into play here. For medical information, physical attributes and maladies can often be accompanied by other instances of care. It would be wise for those storing this data to understand the overall web of available data on someone who might only be treated for an infection. This patient will most likely be prescribed antibiotics, which means there might be pharmacy information to match. Or, consider a more multi-symptomatic issue high blood pressure. Suppose a patient was being treated for this, but by his or her psychotherapist? Now we have potentially devastating information that could alert someone in this person's circle of friends, family, and work colleagues of a mental illness.

The benefit of k-anonymization is not to be ignored, however. It certainly helps to avoid making the lesser occurring tuples easy to identify like low hanging fruit. The benefit to statistical analysis is also to be given credit. While one would not have the nuanced meaning that outliers can bring, with data sets being so large, the argument stands that these one offs can be ignored with the same level of confidence.

Sparse Data Sets

Heavily used by many large corporations, including Amazon, sparse data sets have become a very prevalent way to store tuples [1], [8], [9]. Sparse data is kept in a much less comprehensive form, only confessing a small number of attributes per record. This results in many more records, perhaps duplicate records, and a harder time for adversaries to draw connections among the many tuples.

As tested by Narayanan and Shmatikov, this method has its drawbacks as well. As the data set grows, it is easier for an adversary to find patterns and relationships in sparse data. We liken it to the experience of purchasing groceries. The cashier (or you, if you find yourself at a self-checkout kiosk) can either place many items in few bags, or only a few items inside of many bags. Say you wanted to pull out all of the fruit. If there are only a few items per bag, a literal GROUPBY function is easier with smaller bags as you can quickly see what is in each bag. Conversely, if you were to have the bags with many items in them, the more bags you have, the longer it is going to take you to go through each one. If there were only two or three bags with many items, it would not be a large task compared to the other method. However, in the same way that large data sets continue to grow, if you now have to look through fifteen or twenty bags, it might take significantly longer just in processing time.

Is this a textbook case of *a priori*? It seems absolutely foolish to contain too much information in a record. One would be giving adversaries much more data to work off of without even making them look for it. But is this a real world scenario? Perhaps the answer is in the nature of the data. Sparse data has been adopted as much safer than comprehensive, and to us, it makes sense. Data engineers are then tasked with comprehending 1) what they know about the data relationships, and 2) what an adversary might know. If relationships are hard to draw conceptually,

sparse data is more robust to attacks. However, if each tuple is simple in nature, such as purchases for an online retailer, a company would just need to employ additional measures to further mask the data. Knowing that an anonymous user's high blood pressure diagnosis is linked to depression is perhaps a harder reach than, say, knowing someone who purchased milk most likely also bought cereal, childrens foods, or beer on the same visit.

Of course, armed with additional, outside data, such as credit card transactions for a certain zip code, this situation become even more vulnerable. Regarding the blood pressure scenario from above, perhaps an adversary knows from this outside data, the purchase amount at a pharmacy. He or she sees that this amount is very large, and the patient most likely picked up additional items. Even armed with this limited information, an adversary could begin formalizing guesses at to other needs this person had. Indeed, Narayanan and Shmatikov proved how adversaries knowing very little of the data, or even what they were looking for, could use these generalities to completely re-identify a user [1].

Perturbation

Another method used to anonymize data is to actually change the attributes enough to create further distance from a given source, but not so much as to statistically alter data science efforts. Common examples are to create ranges and categories out of numerical fields, or to use internal identifiers instead of more publicly recognizable attributes. In fact, perturbation is also used as a way to actually detect data leaks, based on the theory that if a particular recipient is given a data set that was perturbed with a certain dictionary of identifiers, and this information is found somewhere it should not have been, you can determine which party released the data since there is no other recipient that had that particular set of identifiers.

Common examples of perturbation include:

* Removing Personal Identifiable Information (PIIs) and replacing with an ID number.

* Taking a scalar value and applying ranges. Instead of ages, the owner of the data could use age ranges that mask the exact answer.

* Selectively altering or even deleting data to create doubt in a bad actor. If a malicious party has, say, matched up a perturbed data set with another set found online, the principle here is that they have a reduced chance of correctly determining a subject's attribute values.

* Changing the definition of certain attributes. For example, if the need for a data set is to review density of certain attributes, then it does not matter if the values reflect the original values. As long as the correct scale is used, one could change, say, zip codes to names of fruits and vegetables. Of course, this might cause more confusion than value depending on the purpose of the data mining activity, so the owner of the data would need to know the purpose of a data scientist to correctly perform this technique.

* Using a "watermark" to discourage data leaks. Watermarking involves altering the data to embed a secret code per data set or subset that is released to a user. This helps a data source know who leaked a data set because each recipient received a different watermark.

Again, no method is perfect, and neither is perturbation. If one does not consider *why* a data set is needed, perturbing certain fields could make the data unusable. The number of fields released directly affects the strength of a perturbed attribute. 500,000 records with their ages changed to age ranges is very different from 100 records. The more a malicious attacker has to work with, the greater the chance that he or she can figure out a "secret" alteration method, especially if that field is now a category where there are not as many "wrong answers". Remember, an attacker does not necessarily have to be exactly right in their guess. Even having a broad understanding of where someone might fall has been proven dangerous to the confidentiality of a data set, as evidenced by the Netflix prize fiasco [1]. Finally, watermarks are not effective if an attacker has discovered how to delete it.

Perturbation is certainly a way to cause doubt for an attacker, but it must be taken into consideration with other factors. The size of the data set, distribution of the various attribute values, and the confusion it could cause for a data scientist indicate that additional techniques and factors need to be considered before this method is held up as the solution to a problem.

Deletion of Data

Similar to perturbation, deletion of data alters a data set, if not completely eliminates it. One might wonder what the point is, since deleting a data set would effectively make it anonymous, but there are interesting results from this method that warrant further explanation.

Firstly, virtual deletion of part of a data set is routinely performed by creating subsets. Here you limit what is given out to reduce the chances of identifying a subject. This is, of course, a sparse data set technique. However, what if an entity actually truly deleted attributes permanently in the interest of protecting more sensitive fields?

For example, say there is a medical data set that held information showing subjects' zip codes, ages, names of hospitals visited, number of days they were in the hospital, and the diagnostic codes of whatever they were were in the hospital for. Is is imperative that we have a subject's zip code? Is knowing the hospital visited essentially the same thing and would this be enough for the question at hand? And is knowing the hospital *likely* to give away a zip code anyway, in which case perturbation of the hospital name would be necessary?

To make matters more challenging, permanent deletion of data, especially partial, can create a new host of problems for databases. Indexing can be disturbed if too many records are deleted, resulting in energy waste as a database interface searches through empty locations. Keys (or lack thereof) also present a problem. Deleting a field such as a social security number might seem like a reasonable idea, but if that field serves as at least one primary key in any

of the database records, the results could be catastrophic.

That being said, in an age where data leaks have put consumers at risk and hurt various companies' bottom lines (think Yahoo¡s reduced purchase price following a data breach [11]) , is deletion of data actually a good move to create good will? While "Big Data" seems to be at the forefront of many companies' minds, many entities seem to have an inability to handle it against malicious attacks. Perhaps a company could receive more value from promoting that they *delete* data instead of store it. This would take a commitment and understanding well in advance of a database design to avoid operational impediments, or simply enough flexibility that a corporation could actually do this without putting themselves at more risk of net lost sales or decreased operational efficiency.

1.1.3 De-anonymization

To gain a better understanding of how effective various anonymization techniques are, our team explored de-anonymization methods as well. We did this by reading existing papers and articles, and by actually testing certain de-anonymization processes on our sample data set. The following points are worth noting as background knowledge before delving into our de-anonymization experiment.

*Assumptions of an adversary's knowledge: a bad actor might have only general knowledge about the fields in a data set. It is not impossible for someone to explore a data set and find likely identifiers with what they deem as a satisfactory level of confidence. Therefor, just because an exact value cannot be determined by an attacker, a general idea might be enough for their uses.

*Acceptance of the power of data: a simple gut check of "what could one possibly get out of this data?" is not a valid test against the vulnerability of a data set. As evidenced by the Netflix prize de-anonymization, Narayanan and Shmatikov were able to take seemingly innocuous information and tie it back to names individuals. [1] To assume that data is not "important enough" or too "vague" could be considered gross negligence in this age of digital security enlightenment.

*Acknowledgment of the repercussions of shared data: it is important to know that implications exist for subjects whose data is known by others. Being able to identify who someone is based on their movie reviews may seem harmless, but in the wrong hands, it could be used to punish someone for their political or religious views. For those who would balk at the weight of such a discovery, the ability to then match up names with another publicly available data set, could truly make this situation a nightmare for a company.

Additionally, once an attacker has at least one nugget to work with, even just a name, there is a myriad of other activities one could perform to gain additional knowledge on a data set. For example, an attacker can make physical phone calls or send electronic messages to various parties to retrieve additional knowledge on a very small piece of information they garnered from a dataset. They can then return to that same data set or even to another data set, armed with a more complete picture of a subject's life. In other words, an attacker's knowledge of a data set is not necessarily limited to one data set.

*Algorithms can be your enemy: for an adversary with limited knowledge about the subjects of a data set, an algorithm can speed up their investigation and narrow down the list of possible values. In fact, it was an algorithm that helped Narayanan and Shmatikov gain significant progress in connecting subjects with their data. For those adversaries who are not working with much information to start with, but who have an effective algorithm, they are no longer looking for a needle in a haystack.

2 Testing of a Data Set

To experience an anonymization as well as de-anonymization for ourselves, we created a fake data set of 10,000 records. The attributes included are:

TABLE 1
Dataset Attributes

Name	string divided into first name, middle name, and last name
First Name	string
Middle Name	string
Last Name	string
Sex	categorical of male or female
Age	numerical
Ethnicity / race	categorical
Hispanic / Latino	binary categorical yes / no
Blood Type	categorical
Hair color	categorical
Eye Color	categorical
Street Address	string
City	String
State	String
Zip Code	numerical
Phone number	string
Social Security Number	string
Z-Virus	categorical, denotes immune, carrier, or infected

This data set was generated randomly, and provided us with several opportunities to try common anonymization techniques discussed in this paper. The attributes were chosen to reflect common member records that would go under de-identification methods, including 1) changing addresses to zip codes, 2) changing specific ages to an age range, 3) keeping only the area code of a phone number, 4) removing all names and replacing with one field of an assigned identification number, and 5) dividing the columns into less comprehensive tuples.

2.1 Anonymizing Data

Creating the Dataset

In order to illustrate the process of anonymizing and deanonymizing data, we needed a dataset that had a variety of features that ranged from categorical to quantitative data. As most of our research involved health care or demographically inclined data, we felt this was a good starting point, hence our appropriation of the "Z-virus" trend in pop culture.

Initially, we had originally sought out different datasets with the intention of either deanonymizing an already existing anonymous dataset, or anonymizing and then deanonymizing a publicly available dataset. Unfortunately, this led to two problems in attempting to use either option as an exercise. Firstly, with an already anonymized dataset, we would have no way of verifying our results. Moreover, as we explain later in this example, properly anonymized data is virtually impossible to deanonymize.

Secondly, there are some aspects of anonymization that are important to the process, but any publicly available datasets would have already gone through those processes (such as removal of PII). In order to make this example as complete as possible, it was necessary to build our own dataset from the ground up.

Creating a dataset is not difficult. Most of us have some familiarity with either bootstrapping data or generating random bits and bytes. To that end, our first version of the dataset was remarkably simple. We randomly pulled from attribute lists, resulting in an even distribution of all feature values across all observations. However...

This is a dataset that is meant to be anonymized and then deanonymized. This required thinking backwards, not from the idea of plaintext, but rather from the idea of what the resulting encrypted output might look like. And from there, understanding what techniques are necessary to conceal, hide, perturb, or manipulate data to prevent the ability of an attacker identifying a person or group of people. A set of completely random features and values with no correlated or causal relationships is incredibly effective against cryptanalysis attacks. While some datasets might truly be random, once our ”version 1.0” dataset was anonymized, we would be unable to deanonymize it. But this led us to a realization about one of the weaknesses of anonymization in general: it is only as strong as how little the attacker knows about the dataset.

To that end, we relied heavily on demographic data to produce a somewhat statistically accurate selection of certain features such as ethnicity/race, hispanic/latino, age, blood type, hair color and eye color. If we were to attempt to perform a logistic regression on the anonymized dataset, we would have also included controls for determining which features were selected for to produce our z-virus classifications, but we were primarily concerned with producing a dataset for which an attacker might know some information on what the dataset contained.

Anonymizing with K-anonymization

The attributes “Age“, “Ethnicity“, “Blood Type“, “State“, “Hair Color“, “Eye Color“, and “Hispanic Latino“ were all great candidates for examination with K-anonymization, and we removed those elements that have a low repetition count. The chosen number K used as a threshold typically depends on the sample size of the original dataset, and contextual knowledge of the domain. We chose to set our K threshold for K-Anonymization to 5% of the original sample size (i.e. 500 observations). This means that for every attribute “class“, we analyzed the frequency of that class, and for every class with fewer than 500 observations in the original dataset, we removed all observations in that class. During the K-Anonymization process, we went from a sample size of 10,000 down to 7,662, removing classes such as:

*Age(after converting to age ranges): ¿=80

*Ethnicity: ”American Indian or Alaskan Native”, ”Asian American”, ”Native Hawaiian or Other Pacific Islander”

*Blood Type: AB+, AB-, B-

*Hair Color: Grey, Light Blond, Light Brown

*Eye Color: Black, Grey, Light Blue, Light Brown, Other

Attributes “State“ and “Hispanic Latino“ were untouched during this process due to equal distribution among classes or the K threshold was met for all classes. One notable disadvantage of this process is that insights made on this dataset may not be extended to these demographics, as models created on the shared data may not hold true to the extended population.

Anonymizing with Stratification of the Response

We were concerned about the distributions of each classifier matching publicly available data. For example, the American Red Cross shares publicly blood type distributions by ethnicity type . We feared that information like this, and similar information for other attributes, may make it possible to identify data even after perturbation (as discussed in the next section). As one can see by viewing our complete iPython Notebook, by pulling equal distributions of each ZVirus type (a fake virus we randomly assigned to observations in the data set), we slightly skewed the predictable distributions based on available population data. This also helps with prediction for our open source contributors, as we help eliminate bias towards dominating classes.

When identifying how many records are remaining in each ZVirus category, we found that within the remaining dataset, our least frequent ZVirus type was ”Infected” with a frequency of 1528. In order to stratify, this means we could not have a sample size larger than 1528 * 3 = 4584. Our chosen stratification was random samples without replacement taking 1,250 samples within each ZVirus type for a total sample size of 3,750.

Anonymizing with Perturbation

With our dataset stratified and properly K-Anonymized we are left with only highly occurring observations distributed among equally sized classification groups. This is the bare minimum of what should ever be released. Now, occasionally, datasets will have slightly different requirements, but in order to reduce the probability of identifying an individual, PII removal and K-Anonymization are necessary. However, there is an additional step that can be taken to further secure the data: Perturbation.

In its simplest form (like below), the perturbation of data is simply running the table through a hashing algorithm to produce a one-way encrypted dataset. The best part about machine training/learning is that a computer has no cognitive understanding of the data. It simply identifies the string as a string and goes from there. While a random collection of characters and numbers might mean nothing to you, to the computer it will have value in its relation to the other strings of random characters. As such, our dataset containing entirely classificatory data can be run through a

simple SHA256 hash (with a salt) to produce an encrypted dataset with no obvious way to decode the value of each feature for every observation.

For quantitative or linear features, a hashing algorithm will not work. For "Age", one of our original quantitative features, we broke it up into classes and assigned each observation to a class. This is one way of anonymizing that data and did ultimately work with the method of hashing the data. Another way is to restrict extreme values, by setting a cap on maximums and minimums. This requires sufficient knowledge of the dataset to identify local and global centers for the feature to best decide how to cap or control those extremes. Additionally, some fuzzing can be done to the values within a feature so no individual feature is accurate, but when taken on the aggregate the sample as whole still represents the population.

2.2 Deanonymizing Our DataSet

Say an attacker, Bob, wants to identify individuals who participated in this study, in order to identify Blood Type and ZVirus Type information. Bob was informed on the date / time / location the Texas study took place for data capture. Bob sat on a bench outside the facility, taking pictures of individuals as they walked in/out of the facility. Bob logged timestamps from each photo identifying when individuals walked into the facility, and out of the facility so he could remove those individuals he perceived as employees due to long time durations. Since Texas is not the first state in which this study has taken place, Bob is familiar with the data shared by the CDC and their descriptions of the dataset values. Using pictures, he visually logged his perceived record value for the following attributes:

*"Sex" (0, 1) (i.e Female, Male)

*"Ethnicity" (Black American, White American, Other Race)

*"Hispanic Latino" (0, 1) (i.e. No, Yes)

*"HairColor"(i.e. Black, Blond, Brown, Light Brown)

*"State"

To produce an example dataset of this visual attack, we randomly selected 50 records from Texas in our original Dataset - storing only these attributes. In practice, there would be an error margin reducing the effectiveness of this data gathering technique, but for the purpose of this exercise we selected straight from the original data.

Is Stratification and K-Anonymization after removing PII sufficient?

Pertubation on a dataset creates a number of problems when it comes to interpreting and utilizing the data, thus many times you may be inclined to avoid doing so. Given our dataset, shared after removing PII / K-Anonymization records and stratifying the result set, Bob can attempt to detect a given individual from his 50 observations. Bob performed the following steps to identify two individuals with nearly complete confidence, and many more with less confidence within the shared dataset:

1) Aggregate the shared dataset on attributes Bob was able to capture visually from his photographs ("Sex", "Ethnicity", "Hispanic Latino", "HairColor", and "State"), computing the frequency count for each. This allows him to exploit the more unique characteristic of each observation by the combination of all values captured vs looking at one.

2) Merge frequencies on the shared dataset with Bobs data captured, finding those records with demographic matches.

3) Filter merged data to those with a frequency count of 1.

4) Merge filtered data with the original shared dataset. These results, post step 4, provide Bob with those individuals from the shared dataset he observed at the facility and have photos of that have the highest confidence in a match. Further insights may be drawn with less confidence, but for this report, only highest confidence matches were chosen for identification. Notice, we do not consider results with 100% confidence, even though these observations were classified as stand-alone in the shared dataset. Because we do not know if and which observations were removed from the original study, it is very possible that there was another observation that we are unaware of in the study that was removed from the shared sample set. Ultimately, the data before perturbation, in our case study was extremely vulnerable even after removing PII / K-Anonymization records and stratifying the result set.

Is Perturbation sufficient?

Given that Bob was ultimately able to identify two individuals even if he was using an associated dataset, perturbation certainly should be considered for all datasets and weighed against the difficulties of ultimately interpreting the final model. But is a perturbed dataset the end-all-be-all of anonymization? Could individuals still be identified?

Let us start over, but this time Bob has only a set of the perturbed data. If he can successfully reverse engineer the datset into human readable terms, then it's possible for him to identify an individual in the same method as above. The first step, of course, is seeing if Bob can decode the hash/encryption.

A note on this process. We used a weak hashing method on purpose to illustrate a possible method of attack. A cryptographically secure method would not use a dictionary based salt, nor would that salt be static and independent of both the generator of the dataset and the recipient of the downloaded dataset.

Essentially, what we attempted to do is create a forced hash collision by using what we know about the data. For example, both the "Hispanic Latino" and the "Sex" columns appear to be binary, however, they have different values. This means context neutral classifier values such as 0 or 1 are unlikely for either one column or both. As such, it will be simpler to start with the sex column as values within that feature would be limited to "Male" or "Female", "m" or "f", and other variations.

But before we even break out our coding IDE, we need to use a hacker's best friend: Google. All we have to do is copy and paste our given hashes into the search bar. This will let us know if the hashes have already been calculated before, and if they have, what the value is.

A quick search of either of the unique values in the "Sex" column tells us that either they aren't common values or a salt has been used (non exclusive or). If they were common values without a salt, then we would have turned

up a few pages with the hash and probably its equivalency. For example, searching for a particular hash value gave us "dog" using SHA256.

This means we have to attempt a bruteforce. Now we know based on the length of the hash, that it's most likely SHA256. What we don't know is how or if a salt was used. Normally in a better brute force function, there'd be a greater variety of attempts for mixing the salt with the hash function. There would also likely be a larger dictionary and a few other tricks, but for the purposes of this exercise, using a little bit of code was sufficient [12].

In this instance, the binary classification meant it was almost trivial to guess at values. This means the security rests entirely on the salt. In less than one second, we were able to evaluate a list of 10,000 passwords and discover which one was used in the hash algorithm.

With the Password and Hash function now known, we can begin our attempts to guess the remaining data set. While hash functions are normally one way, as long as we know some basic meta information about the dataset, it becomes possible to decode other attributes as it's much easier to guess the values of classifiers than it is to guess the salt for a hash.

In the case of ZVirus classification, we know that the CDC has been labeling their publicly available statistics and graphs with Carrier, Immune and Infected. So it's likely that these three values were used in this dataset especially as it contains three unique classifiers in the ZVirus column. Of course it's possible they used non-contextual based classifiers such as 0, 1, 2, etc. Or even used partial contextual classifiers like C, Im, and In. As always, we started with the obvious and worked our way down. Using our found salt and a list of possible plaintext values, we were able to easily identify which hash value resulted from which classifier.

Each other feature follows the same path, taking educated guesses as to the values present, and running them against the given hashes. Once a pattern is identified, it becomes even easier. For example, the hardest feature to guess is probably the Ethnicity/Race as those terms are somewhat inconsistent across all reporters, but if you knew the CDC used the census classifications (as they should) then it becomes all the easier to crack.

But let us step back and assume that Bob could not crack the password and that whoever perturbed the data used a cryptographically strong method. Let us take a look at what he could do instead.

Metacontextual Analysis

Taking another look at our dataset, in particular hair and eye color, we noticed a strange coincidence. Within this dataset, because all values for all features were applied the same salts, any features that use the same values will have the same hashes. In this case, a particular hash is repeated for both hair color and eye color.

Going through a list of hair colors and eye colors, very few match up naturally on a person. Realistically speaking, the only color that appears in both hair and eyes (using common parlance) is brown in some fashion. Whether it's light brown, dark brown, or just brown, we can assume that this particular hash probably means brown in some way. This is an example of inferring meaning based on context and using any metaknowledge known about the dataset. Without ever having to know what that hash translates directly into, we still have transformed it from incomprehensible into a usable bit of information. Similarly, it is possible to decode the ethnicity feature column as well.

Now, Bob, like us, has access to the US Census data and would know that there are not just three ethnicities or races as our anonymized dataset might suggest. A quick search of our dataset gives us the following percentages for each unique hash string:

TABLE 2
% Frequency of Unique Hashes

41a0446	77.7%
239bfd6	14.2%
f2c2851	8.1%

Either the data was prepared in such a way that ethnicities were grouped together or only the most common ethnicities were selected for representation. While it is possible that certain groups were combined, such as maybe Native Americans and other indigenous people, it's unlikely that, say, White Americans and Black Americans were grouped together. If Bob knows that White Americans make up around 73% of the population, Black Americans about 13%, and Asian Americans around 8%, those values are close enough that Bob could, in fact, assume that those hashes equate in some way to those demographic populations.

This demonstrates a very important aspect to perturbance. Even with encrypted values, analysis can still be performed to assume a value or values for a feature if the feature is known and has associated data that is available to the public. In this case a quick search of census data gives us near equivalent demographic statistics which makes it easy to draw a conclusion. We can easily perform this same analysis to uncover the meanings for the "Hispanic Latino" feature set as well by looking at the relative population of each value to the presumed White American hash.

Not every feature will be able to be inferred in this fashion. "Bloodtype" will do so (based on how we built our dataset), but for features like "Sex" or "State", simple probability or statistical demographic analysis probably will not be enough. However, much like the popular number game Sudoku, once you start filling in the easy blanks, the harder blanks will follow. For example, we can verify our Ethnicity selections by looking at "Hair" and "Eye color" demographics and comparing them to our assumptions. Further, this is based entirely on this dataset alone. If Bob has access to any additional data, it just becomes easier and easier to make those metacontextual connections necessary to identify feature values.

Stopping Metacontextual Analysis

Interestingly, after playing with the data, "Age Class" appeared to be the most difficult to predict and crack. Because the groupings didn't follow census data, but rather an internal logic, it would have been almost impossible for Bob to put together which class/group equated to which hash, or even how to group ages together to begin with. Certainly, census data would be easy enough to deanonymize,

but the only reason it worked so well with this dataset is because our original dataset proportionally represented a real population. Most datasets of this nature probably will not actually represent a real population, but it is important to remember for those datasets that do, especially for an extremely large number of observations. And, in actuality, the easiest data to deanonymize were for features that had easily predictable values.

3 Conclusion

It should be said that any anonymization approach should not resemble ”either K-Anonymization or Perturbance” as a sufficient thought process (although one should always go by the mantra ”remove PII”). Both are methods of anonymizing datasets and both are equally valuable in hindering malicious analysis. And while it would be an easy conclusion to say that it all depends on the dataset as to which methods should be employed, that is ultimately the predicated conclusion. Every dataset has a purpose, as well as features and qualities that would make some methods more viable than others. In the case of perturbance, a purely quantitative and dense dataset would probably not be a very good candidate for most forms of perturbance - especially if precision is important to the final model. Each method has its own weaknesses and strengths which should be considered whenever anonymization is performed on a dataset.

Much of the strength behind K-Anonymization lies behind the principle of finding a needle in a needle-stack. Small datasets by their nature are susceptible to de-anonymization because there so few individuals that have been captured, that distinguishing between them becomes relatively simple. Differences, again, are not necessarily between individual features, but combinations of features as well. K-Anonymization can be employed to overcome this weakness by identifying those more unique individuals and removing them from the final dataset. This comes at a cost, however, as by removing individuals and observations from a dataset ultimately impacts the accuracy, precision and effectiveness of any models derived. But a sufficiently large dataset or a dataset with low variance would be perfect candidates for K-Anonymization without impacting the overall performance of any algorithms run on the dataset.

While machines have very little problem using values that are 64 characters long and anonymous feature names to develop models for classification or prediction, it can lead to an issue of readability for the human aspect of the team. A truly anonymous set might in fact be a list of features x1, x2, x3... with a collection of values that have been encrypted or fuzzed to such a degree that any attempt to apply any discovered models would be largely theoretical and probably practically useless. Any attempt at perturbance then weighs readibility and usability against confidentiality and anonymity. Largely, most concerns about the weaknesses of perturbance come down to the concerns related to most forms of data security. For example, using a proper salt or some form of cryptographically secure methods of encrypting/perturbing the data is a must. Whether or not to perturb and which methods to employ then depend on what is wanted out of a shared dataset. Are contributors expected to publish and interpret their own results, or are they to simply return them to the owners of the dataset?

It is not reasonable to avoid data collection and sharing altogether. Thus, we have compiled a series of recommendations that we would advise to those interested in having a better realization of the remaining vulnerability of their anonymization techniques.

1. Actively attempt to de-anonymize data sets. Depending on the sensitivity of the data, this might need to be performed by individuals not privvy to the anonymization processes used.

2. Treat normally innocuous data as weaknesses for an attacker to exploit.

3. Consider using multiple anonymization techniques. Test along the way to see how this affects statistical analysis.

4. Work to understand the goal of a data mining activity before providing data sets. Consider using subsets, and weigh the the value of data mining activities with the vulnerability of the subjects information.

5. Consider the long range goals of an organization before employing the use of public contests. Will you even need this winning algorithm or model in the long term?

REFERENCES

[1] Narayanan, Arvind and Vitaly Shmatikov *Robust De-anonymization of Large Sparse Datasets*. SP 08 Proceedings of the 2008 IEEE Symposium on Security and Privacy. 2008.

[2] Johnston, Casy. *Netflix Never Used Its $1 Million Algorithm Due to Engineering Costs*. Wired Magazine. https://www.wired.com/2012/04/netflix-prize-costs/. April 2012.

[3] Anderson, Nate. *Neflix Prize 2 (Privacy) Apocalpyse Now?* ars Technica. https://arstechnica.com/tech-policy/2009/09/netflix-prize-2-privacy-apocalypse-now/?comments=1. September 2009.

[4] Ohm, Paul. *Netflix Cancels the Netflix Prize 2*. Freedom to Tinker.com. https://freedom-to-tinker.com/2010/03/12/netflix-cancels-netflix-prize-2/. March 2010.

[5] Cheng, Jacqui. *Netflix settles privacy lawsuit, ditches $1 million contest*. ars Technica. https://arstechnica.com/tech-policy/2010/03/netflix-ditches-1-million-contest-in-wake-of-privacy-suit/?comments=1. March 2010.

[6] Borne, Zachary Davies. *There Are Officially More Mobile Devices Than People in the World*. Independent.co.uk. http://www.independent.co.uk/life-style/gadgets-and-tech/news/there-are-officially-more-mobile-devices-than-people-in-the-world-9780518.html. October 2014.

[7] Loukides, Grigorios Aris Gkoulalas-Divanis, and Bradley Mali. *Anonymization of electronic medical records for validating genome-wide association studies*. Proceedings of the National Academy of Sciences of the United States of America. Vol. 107, No. 17, pp 7898-7903. April 2010.

[8] Gentry, Jerry. *Big Data vs. Sparse Data*. Data Center Knowledge. http://www.datacenterknowledge.com/archives/2012/10/11/big-data-versus-sparse-data/. October 2012.

[9] Rimes. *The difference between "dense" and "sparse" data*. . Rimes.com. https://www.rimes.com/insights/the-difference-between-dense-and-sparse-data/. February 2012.

[10] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[11] Albanesius, Chloe. *After Breach, Verizon Drops Yahoo Purchase Price by $350M*. PC News. 1em plus 0.5em minus 0.4em. February 2017.

[12] Frye, Alex and M. Smith and L. Vitovsky, *Anonymizing Data Notebook*. github.com/amfrye777/MSDS7349_AnonymizingData/Dataset/. Apr 2017.