Ensuring Participant Deniability in Public Datasets: An Alternative Approach

Riddho Ridwanul Haque[1] and Redwan Ahmed Rizvee[1]

Masters in Science, Department of Computer Science and Engineering, University of Dhaka

Abstract. Collecting individual information for building large datasets is an integral part of various modern data scientific applications. While survey data and various records help us in gaining a data-oriented view of the world around us, they are prone to collecting sensitive, but potentially useful, personal information. Preserving the privacy of individuals in large datasets is thus of utmost importance. Traditionally, anonymizing the datasets by simply blurring out the names of the participants was thought to be sufficient. However, applications of linkage attacks in recent times provide ample evidence to the contrary. In simpler terms, anonymizing data is not enough. In this work, we introduce a novel tree-based approach that *changes* the dataset in a controlled way, ensuring that the participants of a survey can *deny* having replied to a query in the way that their responses are shown in the dataset. Furthermore, we experiment on a real-life public dataset to find out how the changes we make affects its usability, and explore the usability-deniability tradeoff in light of these experimental results.

Keywords: Deniability in Public Datasets · Differential Privacy · Decision Tree Variants

1 Introduction

Public data repositories are quickly becoming common, with the burgeoning fields of data mining and machine learning continuously contributing new methods to make the data around us more useful. Analyzing public survey data provides us with an opportunity to understand the impact of various events and policies and enables us to make more well-informed policies in the future. Medical data can be used to detect diseases at an early stage based on the symptoms that are observed. Survey data can be used to mine opinions and understand public reactions. Ratings and reviews we drop on various online platforms can be used to build better recommender systems. The bottom line is, our personal information can be harnessed by data science to make wonderful things happen.

On the flip side of the coin, the collection of personal information raises valid concerns regarding the protection of our individual privacy. The issue becomes even more important when we are dealing with sensitive data such as medical and survey records.

Traditionally, it was believed that anonymizing the identifying attributes of individuals (such as blurring out their names and addresses) in a public dataset suffices for ensuring their privacy. Apparently, that is what Netflix thought as well when they started a competition in 2006 known as the “Netflix Prize”, where the competitors had to write an algorithm to predict how a user would rate a movie. The dataset Netflix released contained around 100 million ratings submitted by 480,000 users on over 17,000 movies. The usernames were of course removed, making it a fully anonymized dataset. However, Narayan et al. showed in [2] that they could successfully identify many of the users by combining the data present in the Netflix Prize dataset with a dataset of a similar nature provided by IMDB. Such attacks, commonly termed as “linkage attacks”, occur when pieces of seemingly anonymous data in different, and often unrelated, datasets are combined to discover real identities. A different example of linkage attacks is presented in [4] where Sweeney et al. show how anonymized data from hospital visits by state officials in Massachusetts can be de-anonymized using public voter registration records. Interestingly, Sweeney et al. show that 87 percent of all Americans can be successfully identified using only their gender, zip code and date of birth.

All this shows us that any efforts we take to anonymize entries in a public dataset are in vain. This leads to a rather troublesome situation, where companies and data analysts want more personal information while at the same time participants demand more privacy.

This problem sets up the backdrop for the area of study known as differential privacy, which explores the concept of ensuring privacy in a much more different way than the philosophy adapted by cryptographers. Differential privacy acknowledges that all statistical and data-scientific applications are interested not in an individual entry in a dataset, but in the *story* that is revealed by the dataset as a whole. Have enough data, and it should not even matter if certain entries in the dataset are *altered* or *changed* from their original values. Differential privacy explores the question of whether we can willingly add noise to the dataset in a way that preserves the usability of the dataset, while ensuring an individual’s actual response stays *undetectable.*

State-of-the-art approaches in differential privacy involve various forms of query perturbation, where noise is added to aggregate queries to ensure a certain degree of privacy. However, such approaches require the data to stay beneath a hidden layer of a privacy-preserving mechanism, such as those mentioned in [1]. The problem with such an architecture is that it restricts the type of queries that a user is allowed to make on a dataset, such as disallowing cross-table queries. Furthermore, it does not truly allow an organization to publicize its data in the way that is done by many companies during kaggle competitions, or contributed to various websites by different researchers. Publicizing data has an important place in the modern world, being a medium of outsourcing, and public data being used for academic and research purposes.

In this work, we propose a novel structure which we call the Deniability Ensuring Tree, which provides a mechanism to ensure that a user has sufficient deniability, meaning that even if his/her true identity is revealed, he/she will have sufficient grounds to deny ever having responded to the queries as they are shown in the dataset. In this work, we give deniability a mathematical definition that is modelled upon entropy. We allow the user to set their minimum required guaranteed deniability value as a parameter to the model, and propose a method to change the dataset in a certain way so as to ensure that the user has a mathematical ground for denying his/her answer while making an effort to ensure that the changed version of the dataset still remains usable from a data scientific perspective. We also explore ways on how we can possibly predict the usability of a changed dataset for classification purposes based on a validation set that closely resembles the test data, giving the user pointers as to the sort of performance we can expect from a classification model trained on the changed data.

The remainder of this report is organized as follows. Section 2 contains a brief overview of existing approaches in the field of differential privacy. Section 3 describes our proposed approach. Section 4 explores the deniability vs usability tradeoff based on experimental results from the "Nursery" dataset, collected from the UCI Machine Learning Repository. Section 5 provides a discussion on concession points as to how various aspects of the Deniability-Ensuring Tree could be modified in order to ensure a gentler usability-deniability tradeoff.Section 6 provides a conclusion of our findings, and implications this project might hold for future work.

2 Related Works

As stated earlier, existing approaches on differential privacy focus on adding noise to responses made by a database to a query. For completion, we describe the mechanism here with a bit more detail. A database can be thought of as a set of rows. Let two databases D_1 and D_2 differ at atmost one element if they both have the same rows, with the exception of at most one row, which is present in only the larger of the two databases. A randomized function K is said to provide ϵ-differential privacy if and only if for all datasets D_1 and D_2 differing at at most one element and for all S $\subseteq$ Range(K),

$$Pr[K(D_1) \in S] \leq exp(\epsilon) \times Pr[K(D_2) \in S] \tag{1}$$

While Dwork et. al. prove and provide a more robust version of the following equation, an ϵ-differential privacy preserving mechanism K which responds to queries on a single attribute, for a query function f whose output can vary with a value of Δf, K responds with:

$$f(x) + lap(\Delta f/\epsilon) \tag{2}$$

A scaled symmetric exponential distribution with standard deviation $\sqrt{2}\Delta f/\epsilon$ is denoted by $lap(\Delta f/\epsilon)$. Here, x is a subset of rows specified by a query. [1] provides more details and mathematical proofs for the above formulae. Here, the ensuing discussion will focus on the implications of such a mechanism, and its limitations.

The mechanism works on a selected subset of rows in the dataset which is specific to the query itself. This forces the mechanism to act as a sort of middleman in between the analyst making the query and the data itself, making the data *not visible* to the one making the queries. While the non-transparency of the data may not be a problem for many applications, it still implies the fact that the existing mechanisms identified in [1] do not necessarily solve the problem of how can we make datasets containing sensitive personal data public without compromising individual privacy.

The mechanisms in [1] also require the user to enter the parameter value of ϵ. However, setting the value of ϵ is difficult for the user, since it is difficult for a human to relate the value of the parameter to a real life situation.

The mechanisms shall have considerable difficulty while adapting to multi-table queries. For an example, let us suppose a dataset contains a response to the survey question "Are you happy with your government?" in one table, with the response to the queries being differentially-privatized, while a different table in the same dataset contains responses to the question "Are you happy with your work?" in another table. Making queries such as "What percentage of people not happy at work are not happy with the government as well?" shall not be possible, since we cannot involve more than one differentially privatized responses in a single query.

In the following section, we explain our proposed approach and see how it attempts to solve the problems mentioned above.

3 Our Proposed Approach

To begin proceedings, we first address the issue of *deniability* for responses in a dataset, and relate it to entropy. Our approach is based on constructing a decision-tree like structure from the given dataset. Much like the ID3 decision tree [3], the deniability tree makes a greedy choice of selecting the attribute which gives the most favorable value for an attribute selection criteria that is tailored to the needs of our problem at hand. We then introduce the attribute selection criteria for discrete valued features first and for continuous valued features afterwards. Afterwards, we show our algorithm for constructing a Deniability Ensuring Tree and how it is used to change the original dataset into one that preserves the user-defined minimum deniability value. The last subsection provides a small extension of our approach, that in the presence of a validation set, can predict the performance of existing classification approaches when trained on the changed dataset, thereby estimating the changed dataset's usability.

3.1 Defining Deniability:

The deniability of an individual in a dataset is the degree *uncertainity* that exists to his actual response to a query. For example, if an individual is a member of a demographic of survey participants in which it is known that 100 percent of all survey participants answered affirmatively in response to a query, then the individual has *no* deniability at all, since it is implied that he too responded with a yes. Similarly, if 0 percent participants respond with a "yes", each individual participant has *no* deniability, with the direct implication that he too answered with a no. If the participant belongs to a demographic where half of all participants responded with a yes and the other half with a no, and if it is known that the responses have been distributed randomly among the participants keeping the original distribution of "yes"-es and "no"-s unchanged, then each participant has as much deniability as possible. From this discussion, it can be intuitively argued that the Deniability vs Probability graph for a yes/no (binary) variable shall look as follows:

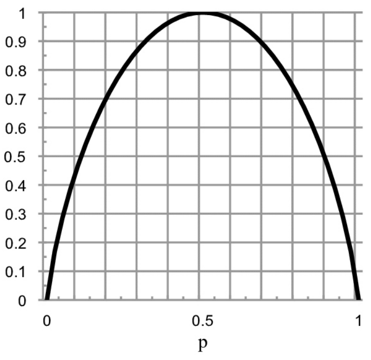

Fig. 1. Deniability vs Probability graph for binary variables

The graph indeed resembles that of entropy, the measure of uncertainity. Indeed, it is easy to see the connection between uncertainity and deniability. Assume that a mechanism exists to randomly distribute the responses to a query among individuals in a dataset while maintaining the original probability distribution of the responses. The more uncertain the answer to a response is, the harder it gets to determine the actual answer by any certain individual. In case of an evenly-distributed response, the uncertainity is the maximum, and so is the deniability. Thus, we define deniability of a participant in a dataset partition D, where the probabilities of the response to a query having n possible outcomes are p_1, p_2, p_3, ... , p_n respectively, as being the ratio of the entropy of the partition and the maximum possible entropy with respect to a variable having n possible outcomes.

$$Deniability(D) = \frac{-\sum_{i=1}^{n} p_i log(p_i)}{-log(\frac{1}{n})} \tag{3}$$

Eq. 3 considers 2 to be the base of all log values used in it. In case of a binary variable, n=2, and $-log(\frac{1}{2}) = -(-1) = 1$, thus $Deniability(D) = Entropy(D)$.

A certain advantage to defining deniability as a ratio is that it allows a user to intuitively define the value of the minimum deniability required in a dataset. A survey question, for an example, might ask participants the following question: “Have you ever stolen a car in your life?”

In order to ensure that the participants do not face legal action due to their responses in an anonymous survey, the survey organizer might look at the judicial process of the country and find that in order to convict a person of stealing cars, the prosecution must convince all 12 jury members that the accused is truly guilty. In such a case, the participant might intuitively want at least $1/12 = 8.33\%$ deniability. The ease of parameter setting sharply contrasts with the difficulty of truly interpreting the value of the parameter ϵ which is commonly used in existing approaches, as shown in Eq 1.

3.2 Attribute Selection Criteria

Let us take a moment to consider the mechanism mentioned in Eq 3. We could device a random function that initially learns from the original dataset, to find out the probabilities of all outcomes to a response. The probabilities to n possible outcomes to a query may be p_1, p_2, p_3, ... , p_n. We could then sequentially input each tuple of the original training data to the random function, whose output will be 1 with probability p_1, 2 with probability p_2, ... , n with probability p_n. While the original probability distribution will most likely be restored in the original database, this mechanism risks muddying up any and all attribute correlations that used to exist in the original dataset. For an example, perhaps it was the case that the probability distribution of the question “Do you eat ice-cream at least once a week?” is 50-50 in a survey questionnaire. However, it may well be the case that the percentage is 75-25 in favour of yes for people aged below

40, and 25-75 in favour of no for people aged above or equal to 40. This would suggest that the probability distribution radically changes if we partition the current dataset with respect to age. Thus, in our mechanism, we search for the attribute that changes the probability distribution of the outcome by the highest margin, and then partition the dataset according to that attribute, and then recursively continue this process upto the point when partitioning the dataset further according to none of the features causes a large enough difference in the probability distribution. This subsection details how we choose our attributes. The first subsection shows how we find out the difference in two different probability distributions. The second subsection shows the attribute selection metric for discrete valued variables. The third subsection extends this problem to show how we determine the splitting points for continuous variables.

Distance between Probability Distributions Given two dataset partitions D_1 and D_2, whose probability distributions are p_{11}, p_{12}, p_{13}, ... , p_{1n} and p_{21}, p_{22}, p_{23}, ... , p_{2n} respectively, we choose to define the distance between the probability distributions as the euclidean distance between the two probability distribution vectors, divided by the normalizing factor n.

$$Distance(D_1, D_2) = \frac{\sum_{i=1}^{n}(p_{1i} - p_{2i})^2}{n} \tag{4}$$

Attribute Selection Metric for Categorical Attributes For a categorical feature F, with possible values 1, 2, 3, ... v, let a dataset D is divided into v partitions D_1, D_2, D_3, ... , D_v and the attribute selection metric is defined as the weighted sum of the distance of the probability distributions of each of the prospective partitions with the current dataset. The weights are proportional to the size (number of rows) of the partitions, divided by the number of rows in D. The attribute selection metric is as follows:

$$ASM(D, F) = \sum_{i=1}^{v}(\frac{|D_i|}{|D|} \times Distance(D, D_i)) \tag{5}$$

The equation implicitly assumes that for all D_i, Deniability(D_i) satisfies the minimum required deniability.

Attribute Selection Metric for Continuous Valued Attributes For continuous valued attributes, there exists an added challenge of having to determine the best splitting points. To solve this problem, we employ a dynamic programming approach which works in a time complexity of $O(n^2)$ and a memory complexity of $O(n)$. The approach is best described by the following pseudocode:

Algorithm 1 ASM For Continuous Values Attributes

```
    Input A Data Partition D, A continuous feature C, min. deniability minD
    Output ASM(D, C)
 1: Sort rows in D according to values of C
 2: Initialize dp[] with size $|D| + 1$
 3: $dp[|D| + 1] \leftarrow 0$
 4: for each $i = |D|, |D| - 1, ...1$ do
 5:     $D_{temp} \leftarrow \phi$
 6:     for each $j = i, i + 1, ..., |D|$ do
 7:         $D_{temp} \leftarrow D_{temp} \cup D_j$
 8:         if $Deniability(D_{temp}) \geq minD$ then
 9:             dp[i]=max(dp[i], $\frac{|D_{temp}|}{|D|} \times Distance(D, D_{temp}) + dp[j + 1]$)
10: return dp[1]
```

The splitting points can be found by simply running a path finding procedure on the dp array. This procedure finds the best possible way to split the data partition so as to ensure the maximum attribute selection metric allowed by the minimum deniability is found.

3.3 Construction of the Deniability Ensuring Tree

The construction procedure of the deniability ensuring tree heavily resembles that of the ID3 decision tree. However, while the leaves of an ID3 decision tree point to a discrete value, the leaves of a deniability ensuring tree point to random functions whose output is probabilistic, with the probability value of each determined by the frequency of that output value in the data partition corresponding to that leaf. The Deniability tree makes the greedy choice of choosing the attribute which best suits its purpose for now. It chooses the attribute with the highest attribute selection metric and recursively extends itself, deeper and deeper into the depths. As a deniability tree recursively extends downward, it captures the probability distribution of the actual dataset more and more accurately. If at any moment, the addition of any unselected attribute does not change the distribution of the output by a value of more than ϵ, the tree does not go recursively deeper. Here, ϵ is a parameter, low values of which help render high dataset usability, but high values of which help keep the size of the tree limited and avoid memory and scalability issues. The recursive partitioning also terminates if dividing the current data partition with any unselected attribute results in a partition that does not have the user-specified minimum deniability, *minD*.

In general, the deeper a tree goes, the better it approximates the hidden statistical distributions in a dataset, thus making the changed values more usable. However, this can also lead to privacy concerns, as partitions that go deep are usually faulty of being too pure and having low deniability values. Thus, higher values of *minD* indicate high amounts of ensured deniability, but potentially low

Algorithm 2 ConstructTree

Input A node nd, A Data Partition D, A feature mask m, minimum deniability minD and minimum change ϵ

Output A tree rooted at nd, whose leaves point to random functions

```
1: if D is empty then
2:     Get Probability distribution of parent as leaf's pointed function
3:     return
4: if all bits of m are set then
5:     Get Probability distribution of D as leaf's pointed function
6:     return
7: for each All features f for which corresponding bit in m is off do
8:     Find feature mf with max. asm value satisfying minD
9: if ASM(D, mf) $\leq \epsilon$ then
10:     Get Probability distribution of parent as leaf's pointed function
11:     return
12: Partition D according to mf
13: for each Partition $D_p$ of D do
14:     Add child newNode to nd
15:     ConstructTree(newNode, $D_p$, setBit(m, mf), minD, $\epsilon$)
```

usability of the dataset, while low *minD* indicates a low amount of ensured deniability, but potentially higher usability. This trade-off between the deniability and usability is explored further in Section 4.

In order to create a new dataset, we simply re-enter the rows of the original dataset onto the tree. However, instead of using the actual output values in them, we output a new dataset, whose output values are set according to the output of the random function which is pointed to by the leaves that each row is directed to by the tree.

3.4 Predicting the Accuracy of Deniability Ensuring Tree Induced Datasets

In this section, we seek to find out, if given a validation set that closely resembles that nature of the test data, can we find out a way to roughly predict the accuracy of different classifiers when trained using this dataset. Assuming the training data follows the same sort of distribution as the training data, we propose a mock classifier, known as the DenETree Classifier.

This classifier works in a very simple way. It takes the tuples in the validation set as input, passes them on to the constructed deniability tree, follows the path that leads them to their respected leaves, and simply adds the probability of the output of the leaf being equal to the output in the validation data to the accuracy score. The final estimated accuracy is the mean of all the accuracy values across the tuples in the validation data.

In the following section, we show that this mock classifier actually gives a rough value of the expected accuracy of other well-known classifiers, helping the user guess the usability of the generated dataset beforehand, and allowing him to smartly set the parameters.

4 Experimental Results

In order to gain experimental results, we used the "Nursery" dataset from the UCI machine learning data repository. The main objective of the dataset was to predict if a child's application for admission into nursery schools in Ljubljana, Slovenia will be accepted provided his background information. The attributes provided for 12960 children provided in the dataset were as follows:

1. Parents' nature (Usual/Pretentious/Greatly pretentious)
2. Child's nursing environment (Proper/Less Proper/Improper/Critical/Very Critical)
3. Number of children in family (1/2/3/more)
4. Housing condition (Convenient/Less convenient/Critical)
5. Financial condition (Convenient/Inconvenient)
6. Social condition (Non-problematic/Slightly problematic/Problematic)
7. Health condition (Recommended/Priority/Not recommended)

The outcome could be 5 values, depending on the child's acceptance status (Not recommended/Recommended/Very recommended/Priority/Special priority).

The dataset was first shuffled randomly before being partitioned into 10 folds, with each fold marking 1/10-th of the data as the test data, and the other part as the training data. Each duplicate training data was created with a value of ϵ=1.5 and with minimum deniability values 1%, 5%, 10%,15%, 20%, 25%, 30%, 35% and 40% respectively. For each fold, for each value of minimum deniability, 5 different input files were produced to account for normalizing the randomization errors, making a total of 460 versions of training data.

In order to test the performance of contemporary classification algorithms after training them with the duplicated training data instead, we used three different well known classifiers- the ID3 Decision Tree, the Random Forest and the K Nearest Neighbour Classifiers. We also compared the performance of these classifiers with that which was predicted by our mock classifier, the DenETree classifier.

Table 1. Accuracy of different classifiers on the original training data

Classifier	Accuracy(In Percentage)
ID3 Decision Trees	99.51
Random Forests	98.3
KNN	98.52
DenETree Classifier	87.47

Table 1 shows the original accuracy of each of these classifiers after being trained by the original training set. Fig 2 shows how this accuracy changes with more restricting limits in the minimum deniability value. As discussed in the earlier section, greater values of minimum deniability required make the tree much shallower than it ought to be, to accurately detect all the inherent probability distributions in the dataset. This results in a lack of accuracy for different training models when the value of minimum required deniability is raised.

It is to be remembered however, that only a small amount of guaranteed deniability is necessary for most applications as indeed it is seen that most classifiers fare fairly well in low values of guaranteed deniability.

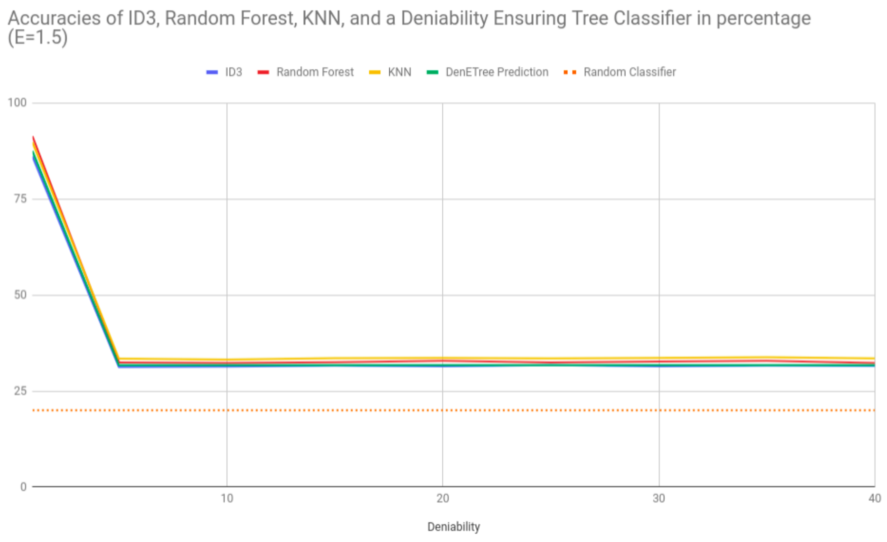

Fig. 2. Accuracy vs Deniability Measure for various classifiers

Another interesting facet of the findings found from Fig 2 is that the mock DenETree classifier that we have described in the earlier section, can actually give a rather accurate prediction of the overall accuracy of the model. As a consequence, a user can actually use this classifier to understand the potential usability of a dataset, and set the parameters for the model accordingly.

The DenETree classifier can also be used to design non-greedy backtracking approaches in the future. Since before publicizing a dataset, any organization would provide precedence to issues such as deniability and usability over efficiency of a duplicating model, such approaches are worth looking at, with the DenETree classifier providing a perfect medium for evaluation of the current model's usefulness during backtracking.

5 Discussions and Possible Alternation Approaches

With the slope for the deniability-usability tradeoff being steeper than one would like, as presented in Fig 2, this section discusses some key points and points out alternative approaches that may be tested in the future to improve upon our current solution for the construction of a deniability ensuring tree.

1. For our experiments, we utilized a personal laptop with 8 GB of RAM, which forced us to keep the value of the scalability parameter ϵ as 1.5, which is a rather high value. As discussed earlier, lower values of ϵ allows the tree to go further down and capture the dataset distribution even better, at the cost of requiring more memory. Conducting experiments with more RAM in our hands would have allowed us to set a lower value for ϵ, which would have improved results, particularly at lower values of minimum deniability.

   A possible alternate approach could be to save partitions of the tree as files in the hard disk instead of storing the entire tree in the main memory. Despite some performance loss, this alternative would allow the user to set extremely low values for ϵ, ensuring greater usability.

2. Revisions could be made by carefully analyzing the attribute selection metrics we have chosen in this project to ensure better experimental results. Possible alternatives include, but are in no way limited to, using cosine similarity for finding distance between probability distribution vectors, using statistical significance test and affinity-based measures. Such modifications could be tested to figure out what could have been done differently in order to improve the outcome.

3. The deniability ensuring tree, akin to the ID3 decision tree, uses a greedy top-down approach. However, as demonstrated by the usefulness of the DenETree Classifier, the performance of a dataset generated by a deniability

ensuring tree can be approximated beforehand using a validation set. This provides a platform to ponder on more advanced backtracking techniques. While no amount of branching and bounding can make a backtracking algorithm as fast as a greedy approach, it is to be remembered here, that given the application, many companies would be willing to allow the algorithm to take weeks to prepare a dataset that ensures the highest level of usability, while maintaining the specified deniability.

These are but some of the possible alleyways that future initiatives based on this work can follow, and this project provides a solid base for all such works, with its novel proposals and alternative concepts.

6 Conclusions

In this project, we introduce a novel approach to determine how to publicize dataset containing sensitive information, while at the same time, providing individuals mathematical grounds for having plautible deniability. While this project as of yet, does not give satisfactory results for relatively high values of guaranteed deniability, we still take some massive strides such as introducing the concepts of deniability in a dataset, constructing a deniability-ensuring tree and setting up a mock classifier to predict the usability of a dataset when it is created by our approach. All these ideas can be used later on to produce better, non-greedy approaches that result in even better usability statistics for higher values of guaranteed deniability.

References

1. Dwork, C.: Differential privacy. Encyclopedia of Cryptography and Security pp. 338–340 (2011)
2. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large datasets (how to break anonymity of the netflix prize dataset). University of Texas at Austin (2008)
3. Quinlan, J.R.: Induction of decision trees. Machine learning **1**(1), 81–106 (1986)
4. Sweeney, L.: Guaranteeing anonymity when sharing medical data, the datafly system. In: Proceedings of the AMIA Annual Fall Symposium. p. 51. American Medical Informatics Association (1997)