Comparing K-Anonymity and ε-Differential Privacy Effectiveness For Social Media

Andres Mauricio Calle

Yu-Cheng Chen

Zhiqiang Hao

Acknowledgements

The authors of this paper extend their thanks to Professor Ming-Hwa Wang for providing the opportunity to practice paper writing skills as well as his support in determining the nature of this project. Additionally, it was due to the Professor's teaching of the fundamentals of data mining that provided us with the ability to understand and construct this comparative paper in the first place.

Abstract

For this project proposal, this team will present our plan to conduct a comparative analysis of two different privacy frameworks, k-anonymity and ε-differential privacy. We will present the algorithms we will use to implement these two concepts, Modrian and Laplacian respectively. Furthermore, we will explain the metrics, such as hidden failure and misclassification error, with which we will evaluate the results in order to determine their respective efficiency and effectiveness at maintaining privacy. These algorithms will be run on a dataset of pseudo-Facebook users with over 90,000 members in order to ascertain their ability to protect the privacy of social media users when data mining popular social media sites.

Table of contents

Introduction

Objectives

In 2018, the General Data Protection Regulation went into enforcement across the European Union hailing the end of the early, minimal regulation era of data collection across the internet. Behind the legislation were many valid concerns on the extent to which companies, hungry for more data to conduct data mining on, would invade privacy to get that data.

As companies are now being regulated into maintaining a minimum level of privacy for their users, they must first define what privacy is. This paper will aim to judge and compare two common frameworks of privacy against each other from a data mining viewpoint. The purpose of which being to determine which framework is more useful, with utility being determined by a mixture of factors including efficiency, general applicability, and effectiveness at maintaining privacy.

The two frameworks will be applied to a selected dataset of social media data, in order to test the frameworks on one of the most common and sensitive data sets that it will need to maintain privacy for. While privacy frameworks can be implemented with different algorithms, this paper will not seek to study each implementation and will instead take use a single, general implementation for each framework.

What is the problem

Before privacy can be applied to a dataset, there needs to be a definition of what it means for a dataset to be private while still being able to conduct useful data mining on it. From this question have arisen

various answers, including the concepts of k-anonymity and ε-differential privacy. The two are different frameworks that determine whether or not results from a dataset can maintain the privacy of the members of the dataset. The problem then becomes determining which is a more useful framework for maintaining privacy, as it would be wasteful to split under the two should one prove to be the superior framework.

Why this is a project related to this class

While we have learned many data mining techniques in our class, ways to ensure the privacy of those we are data mining is not a topic that received much attention. With the rising prominence of data mining across every major industry comes rising concern over its potential for misuse. As such, in the future data miners will have to address these concerns, and if the field will not self-regulate then governments will do so instead. Furthermore, if we can learn ways to introduce security to our data mining techniques, we will be better data miners for it.

Why other approach is no good

As not implementing privacy on datasets can carry a host of ethical and legal issues, data miners must ensure that the results they publish are secure. This team chose to conduct this comparative study after searching for, and failing to find, a conclusive paper that informed which privacy framework would be preferable to implement while data mining. Rather, previous comparative papers had instead overviewd a large number of techniques across a broad variety of scenarios, ultimately casting their net too wide on their search and obtaining no definitive answer.

Why you think your approach is better

By narrowing down the scope of the investigation to a particular use case, the number of factors that must be considered to determine which framework would be the best is reduced. This produces a less complicated outcome of the comparison, avoiding being unable to compare models which might not even be applicable to certain forms of data mining, and produces a clearer and more definitive answer.

Area or scope of investigation

This comparison will be limited specifically to two frameworks, k-anonymity and ε-differential privacy, and two implementations of them, Mondrian and Laplace mechanism respectively. The two frameworks will only be applied to a social media dataset, and will be evaluated based on their efficiency, ability to extend privacy to different forms of data mining, and the level of privacy that they provide the members of the dataset with. These evaluations will be measured by a number of metrics including but not limited to, hidden failure, misses cost, and misclassification error.

Theoretical bases and literature review

Related research to solve the problem

Privacy-Preserving Data Mining (PPDM)[1] techniques have been developed to allow for the extraction of knowledge from large datasets while preventing the disclosure of sensitive information. The vast majority of the PPDM techniques modify or even remove some of the original data in order to preserve privacy.

One of the most known privacy models is the k-anonymity model[2], proposed by Samarati and Sweeny. This model's key concept is that of k-anonymity: if the identifiable attributes of any database record are indistinguishable from at least other k − 1 records, then the dataset is said to be k-anonymous. In other words, with a k-anonymized dataset, an attacker could not identify the identity of a single record since other k −1 similar records exist. The set of k records is known as equivalence class. In the k-anonymity model, the value k may be used as a measure of privacy: the higher the value of k, the harder it is to de-anonymize records.

Currently, there are many algorithms to implement k-anonymity. Mondrian is a Top-down greedy data anonymization algorithm for relational dataset. A k-d tree, or k-dimensional tree, is a data structure used in computer science for organizing some number of points in a space with k dimensions. It is a binary search tree with other constraints imposed on it. The basic workflow of Mondrian is as follow:

Algorithm 1 Mondrian Algorithm
(1) Partition the raw dataset into k-groups using kd-tree. k-groups means that each group contains at least k records.

Partition(region, k)
1. Choose the best dimension that results in even k-anonymous partition 2. If possible, partition the region according to that dimension into R1 and R2 3. Return Partition(R1, k) U Partition(R2, k) 4. If not possible, Return

(2) Generalization each k-group, such that each group has the same quasi-identifier.

Differential privacy[4] aims to protect the private information of any single individual by limiting the privacy risk raised by the data of the individual is used by certain analysis, compared with the result when the data is not used. In other words, differential privacy guarantees that the information which adversaries get from the outputs of any analysis with or without the presence of any single individual is approximately the same, so that it will help hide the presence or absence of any single individual from the adversaries who access to the outputs.

Differential privacy is a mathematically rigorous definition of privacy tailored to analysis of large datasets and equipped with a formal measure of privacy loss. Moreover, differentially private algorithms take as input a parameter, typically called ε, that caps the permitted privacy loss in any execution of the algorithm and offers a concrete privacy/utility tradeoff. One of the strengths of differential privacy is the ability to reason about cumulative privacy loss over multiple analyses, given the values of ε used in each individual analysis.

Algorithm 2 Laplace mechanism
Inputs: Data set B over a universe D; Set Q of linear queries; Number of iterations $t \in N$; Privacy parameter $\varepsilon > 0$; Number of records n
Let A_0 denote n times the uniform distribution over D For iteration i = 1, ..., T:

1. Exponential Mechanism: Select a query $q_i \in Q$ using the Exponential Mechanism parameterized with epsilon value $\varepsilon/2T$ and the score function $$s_i(B, q) = \lvert q(A_{i-1}) - q(B) \rvert$$ 2. Laplace Mechanism: Let measurement $m_i = q_i(B) + Lap(2T/\varepsilon)$ 3. Multiplicative Weights: Let A_i be n times the distribution whose entries satisfy $$A_i(x) = A_{i-1}(x) \times exp(q_i(x) \times (m_i - q_i(A_{i-1}))/2n$$
Output: $A = avg_{i<T} A_i$

Advantage/disadvantage of those research

As social network data is much more complicated than relational data, privacy preserving in social networks is much more challenging and needs many serious efforts. Privacy attacks in the social network data are totally different than most of the attacks in other fields, thus effective and efficient anonymization methods with respect to different attacks should be researched while most of the papers don't consider the social networks' reality.

Your solution to solve this problem

In this paper, we will perform experiments and examine the performance of both algorithms on a reality social network dataset. We will use a reality facebook database with 99000+ records as our dataset. After applying k-anonymity and ε-differential privacy, a series of background knowledge attacks will be used. And we will use the following metrics to compare different privacy preserving methods.

Hidden Failure	The ratio between the sensitive patterns that were not hidden with the privacy-preserving method, and the sensitive patterns found in the original data
Misses Cost	The number of patterns that were incorrectly hidden

Misclassification Error	The percentage of data points that ‘‘are not well classified in the distorted database’’

Where your solution different from others

Instead of handling abstract data, our research will focus on social media and social networks. We will use the real world data so that most of the privacy protection and background knowledge attacks will fit reality. We can also do some optimization for these algorithms when necessary, to make them more suitable for social network PPDM.

Why your solution is better

While analyzing with realistic data, we can find out the weak point of each method, and improve our evaluation metrics. And we will give out more definitive benchmark scores for each of the algorithms.

Hypothesis

Previous research[14] showed that the privacy of k-anonymity is threatened by some possible attacks, e.g. Homogeneity Attack and Background Knowledge Attack. Since k-anonymity does not induce randomization, the information in the dataset could be deduced by attackers. Also k-anonymity does not perform as well when applying to high dimensional dataset. [15] However, ε-differential privacy randomizes the dataset to obscure private attributes, which might cause the loss of utility. This paper assumes that ε-differential privacy is a better method of protecting users' sensitive information. We will perform experiments and examine the performance of both algorithms. By analyzing the experimental results we would be able to determine which algorithm is more efficient and perform better among the two for general privacy preservation while mining social media data.

Methodology

How to generate/collect input data

For input data, we will use a virtual Facebook dataset with 99000+ records. This data will be used for calculating the performance matrices of k-anonymity and differential privacy algorithms.

How to solve the problem

For k-anonymity method, Mondrian is a Top-down greedy data anonymization implementation of GCCG algorithm for relational dataset, proposed by Kristen LeFevre. The resource of Mondrian by Qiyuan Gong and Liu Kun is available on github.[7] For ε-differential privacy method, IBM Differential Privacy

Library[8] and differential-privacy0.1.0[9] are used to implement laplace mechanism differential privacy applications. The output privacy preserved data produced by two algorithms is passed to a Privacy Analyzer which generates the performance matrices as the result of taking the original dataset and privacy preserved dataset as inputs. The comprehensive score by evaluating the performance metrics indicates the better algorithm for preserving privacy among k-anonymity and ε-differential privacy while using data mining.

How to generate output

We will pass the dataset D_o to the privacy algorithm module as input and generate the privacy obscured data output D_p. Then D_p becomes the input of Privacy Analyzer, the performance metrics are calculated thus we can rate the algorithms according to the matrices. Python 3 is used to implement the functional modules in Privacy Analyzer.

How to test against hypotheses

We examine each attribute in the performance matrices, then the comprehensive score of each algorithm is computed. Consequently we can compare the performance of investigated algorithms and therefore we are able to suggest an efficient technique of privacy preserving for data mining.

After finishing the privacy-preserving algorithms, the following steps will be taken:

(1) Compare the output data with the original data, find all of the the sensitive patterns left in output as $\#spattern_{left}$, nonsensitive patterns left as $\#npattern_{left}$, new generated patterns as $\#pattern_{new}$

(2) Go through the original data, find all of the sensitive patterns as $\#spattern_{origin}$ and nonsensitive patterns left as $\#npattern_{origin}$

(3) Do a clustering for original data and output data, find the count of abnormal data points as $\#data\_point$

Then we can get all of the scores for the performance of each algorithm.

Hidden Failure	$\#spattern_{left}/\#spattern_{origin}$
Misses Cost	$\#npattern_{origin} - \#npattern_{left}$
Artifactual Patterns	$\#pattern_{new}$
Misclassification Error	$\#data\_point$

Implementation

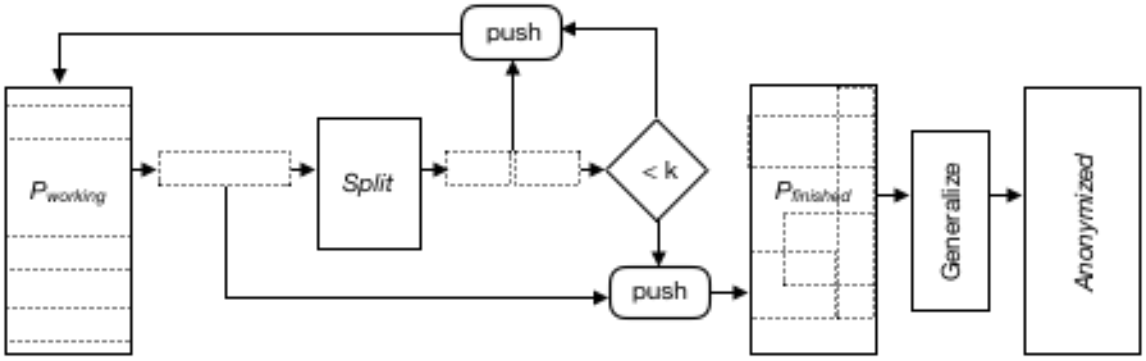
K-anonymity

To make a multidimensional dataset k-anonymous and find the optimal k number is considered NP-hard. In this report we utilize a top-down greedy algorithm called Mondrian[18] which is good enough for this task. The Mondrian partitions the dataset into smaller groups using the hierarchical clustering technique. The Mondrian algorithm flow diagram is shown below.

Mondrian Algorithm:

1. Initialize the finished set of partitions $P_{finished} = \{\}$.
2. Initialize the working set to be the partitions of the entire dataset $P_{working} = \{\{1, 2, ..., N\}\}$.
3. While $P_{working}$ not empty:
    - Pop one partition from the set.
    - Calculate the relative spans of all columns in the partition.
    - Sort the columns by the span in descending order
    - For each column:
        - Try to split the partition along the column at the median of column values.
        - If two partitions are k-anonymous then add them to the $P_{working}$ and break.
    - If no valid split then add the partition to the $P_{finished}$.
4. For each partition in $P_{finished}$:
    - For each sensitive column:
        - Set the numerical value to the mean of the partition.

5. Return the $P_{finished}$.



Mondrian algorithm flow diagram

Code:

split function splits the input partition into two partitions along the column that values below or above the mean value are in different partitions[19].

```
def split(df, partition, column):
    dfp = df[column][partition]
    if column in categorical:
        values = dfp.unique()
        lv = set(values[:len(values)//2])
        rv = set(values[len(values)//2:])
        return dfp.index[dfp.isin(lv)], dfp.index[dfp.isin(rv)]
    else:
        median = dfp.median()
        dfl = dfp.index[dfp < median]
        dfr = dfp.index[dfp >= median]
        return (dfl, dfr)
```

partition_dataset function splits each sorted partition and validates the partitions according to the input validate function provided, is_k_anonynous.

```
def is_k_anonymous(df, partition, sensitive_column, k=3):
    if len(partition) < k:
        return False
    return True

def partition_dataset(df, feature_columns, sensitive_column, scale, is_valid):
    finished_partitions = []
    partitions = [df.index]
    while partitions:
        partition = partitions.pop(0)
        spans = get_spans(df[feature_columns], partition, scale)
        for column, span in sorted(spans.items(), key=lambda x:-x[1]):
            lp, rp = split(df, partition, column)
            if not is_valid(df, lp, sensitive_column) or not is_valid(df, rp, sensitive_column):
                continue
            partitions.extend((lp, rp))
            break
        else:
            finished_partitions.append(partition)
    return finished_partitions
```

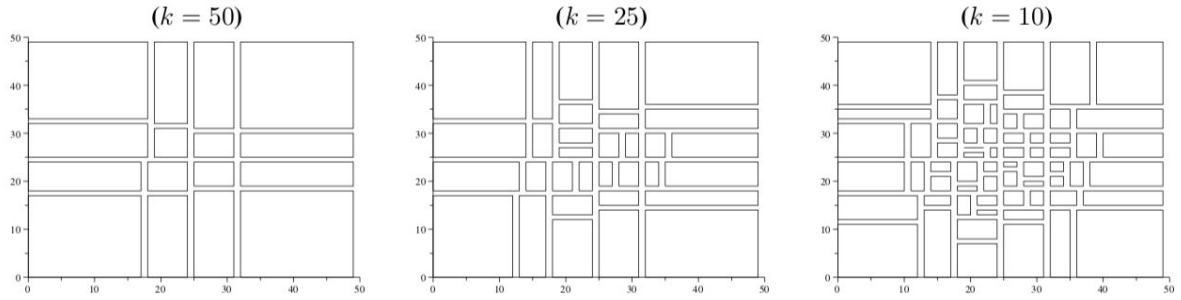
agg_numerical_column function returns the mean value of the column.

```
def agg_categorical_column(series):
    return [','.join(set(series))]

def agg_numerical_column(series):
    return [series.mean()]
```

build_anonimized_dataset function generalizes the group value, such that the value becomes the output of the aggregation function.

```
def build_anonymized_dataset(df, partitions, feature_columns, sensitive_column, max_partitions=None):
    aggregations = {}
    for column in feature_columns:
        if column in categorical:
            aggregations[column] = agg_categorical_column
        else:
            aggregations[column] = agg_numerical_column
    rows = []
    for i, partition in enumerate(partitions):
        if i % 100 == 1:
            print("Finished {} partitions...".format(i))
        if max_partitions is not None and i > max_partitions:
            break
        grouped_columns = df.loc[partition].agg(aggregations, squeeze=False)
        sensitive_counts = df.loc[partition].groupby(sensitive_column).agg({sensitive_column : 'count'})
        values = grouped_columns.iloc[0].to_dict()
        for sensitive_value, count in sensitive_counts[sensitive_column].items():
            if count == 0:
                continue
            values.update({
                sensitive_column : sensitive_value,
                'count' : count,

            })
            rows.append(values.copy())
    return pd.DataFrame(rows)
```

The figure below shows how different numbers of k will influence multidimensional partitioning. In this paper we chose k=3.



Greedy strict multidimensional partitioning[18]

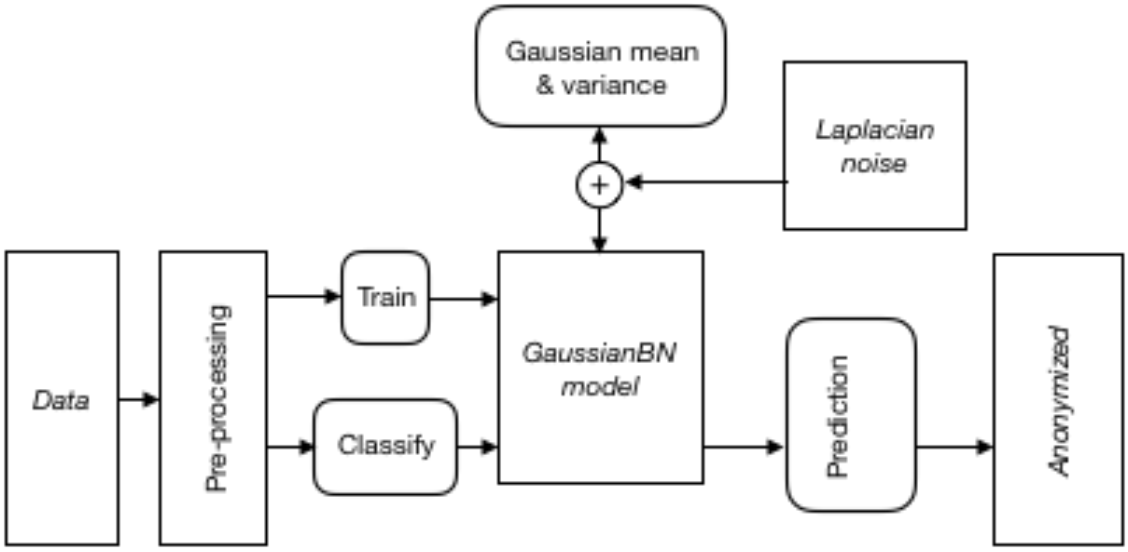
ε-differential privacy

IBM Differential Privacy Library is a general purpose, open source Python library for developing applications for differential privacy. The differential privacy is performed by machine learning model Gaussian naïve Bayes classifier which will be trained and add Laplacian noise with respect to the ε while computing gaussian mean and variance. In this paper default ε =1 is used.

ε-differential privacy algorithm:

1. Pre-process the non-numeric data to assign it into numeric classes in order to train the model.
2. For each sensitive column:
    - Assign the classification target values to be the same as original values, performing supervised learning and expect the prediction results to be the same.
3. Initialize the GaussianNB classifier model with ε and sensitivity.
4. For each record in the dataset:
    - Fit the record to the GaussianNB classifier.
    - Update the gaussian mean and variance with Laplacian noise added to the value.
5. For each record in the dataset:
    - For each sensitive column:
        - Predict the value of sensitive attributes in the record with classification.
        - Overwrite the original data.
6. Return the anonymized dataset.

The ε-differential privacy flow diagram is shown below:



ε-differential privacy algorithm flow diagram

Code:

Laplace.randomise function add the laplacian noise to the input value according to the given ε and sensitivity.[8]

```
136     def randomise(self, value):
137         """Randomise `value` with the mechanism.
138         """
139         self.check_inputs(value)
140         scale = self._sensitivity / (self._epsilon - np.log(1 - self._delta))
141         unif_rv = random() - 0.5
142         return value - scale * np.sign(unif_rv) * np.log(1 - 2 * np.abs(unif_rv))
```

class GaussianNB(sk_nb.GaussianNB) inherits the :class:'sklearn.naive_bayes.GaussianNB' class from Scikit Learn.

Function _update_mean_variance is called by the sklearn.naive_bayes.GaussianNB._partial_fit. The Laplacian noise is added to the learned mean and variance while training the model.

```
    def _update_mean_variance(self, n_past, mu, var, X, sample_weight=None):
        """Compute online update of Gaussian mean and variance.
        """
        if X.shape[0] == 0:
            return mu, var

        # Compute (potentially weighted) mean and variance of new datapoints
        if sample_weight is not None:
            warn_unused_args("sample_weight")

        n_new = X.shape[0]
        new_var = np.var(X, axis=0)
        new_mu = np.mean(X, axis=0)

        # Apply differential privacy to the new means and variances
        new_mu, new_var = self._randomise(new_mu, new_var, self.new_n_samples)

        if n_past == 0:
            return new_mu, new_var

        n_total = float(n_past + n_new)

        # Combine mean of old and new data, taking into consideration
        # (weighted) number of observations
        total_mu = (n_new * new_mu + n_past * mu) / n_total

        # Combine variance of old and new data, taking into consideration
        # (weighted) number of observations. This is achieved by combining
        # the sum-of-squared-differences (ssd)
        old_ssd = n_past * var
        new_ssd = n_new * new_var
        total_ssd = old_ssd + new_ssd + (n_past / float(n_new * n_total)) * (n_new * mu - n_new * new_mu) :
        total_var = total_ssd / n_total

        return total_mu, total_var
```

Data analysis and discussion

Output Generation

Hidden Failure

Hidden Failure (HF) is used to measure the balance between privacy and knowledge discovery. The hidden failure may be defined as the ratio between the sensitive patterns that were hidden with the privacy-preserving method, and the sensitive patterns found in the original data. If HF = 0, all sensitive patterns are successfully hidden, however, it is possible that more non-sensitive information will be lost in the way.

(1) Hidden Failure in K-anonymity Result

The suggested parameter of K-anonymity is K=3. We will consider the records whose k value is less than 3 as the failed records.

(2) Hidden Failure in ε-differential privacy Result

For the ε-differential privacy, we will compare the generated result with the original data. The unchanged data will be considered to be leaked data.

Misses Cost

The MC measures the number of patterns that were incorrectly hidden. That is non-sensitive patterns that were lost in the process of privacy preservation.

Let $f_x(x)$ be the original density function and $\widehat{f_x(x)}$ the reconstructed density function. Then, the information loss is defined as:

$$I = \frac{1}{2}E[\int_{\Omega X} |f_x(x) - \widehat{f_x(x)}|dx]$$

We will use kernel density estimation (KDE), which is a non-parametric way to estimate the probability density function $f_x(x)$ and $\widehat{f_x(x)}$.

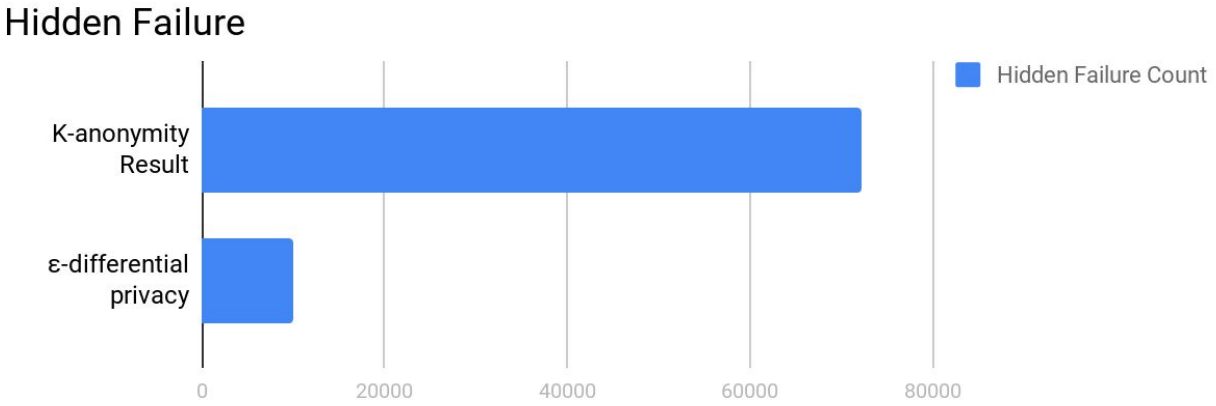
Misclassification Error

Misclassification Error measures the percentage of data points that ‘‘are not well classified in the distorted database’’. We use the euclidean distance and Density-based spatial clustering of applications with noise (DBSCAN) to cluster the data. DBSCAN groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). Data points outside of 3-sigma will be thought as abnormal points.

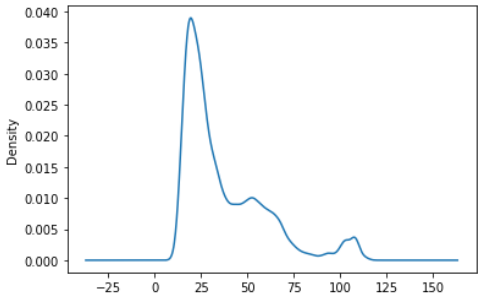
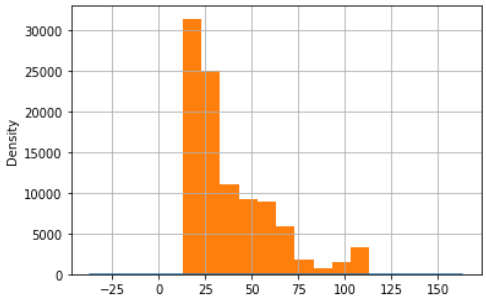
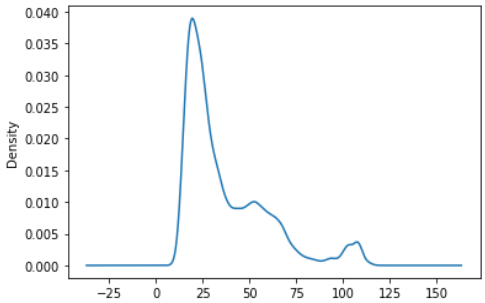
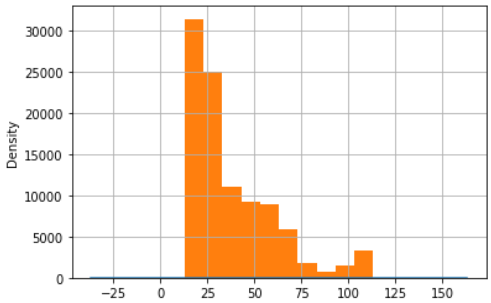
Output Analysis

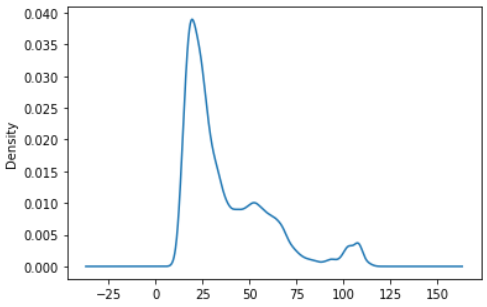
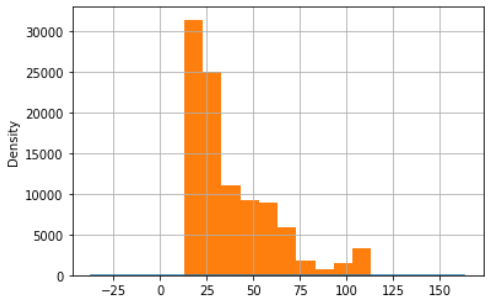
Hidden Failure

	Hidden Failure Count	Hidden Failure Rate
K-anonymity Result	72228	72.96%
ε-differential privacy	10006	10.11%



The hidden failure analysis shows that both of these 2 algorithms fails to hide all of the sensitive information. The ε-differential privacy has a better performance on privacy-preserving. For K-anonymity, it fails to hide most of the information. We will analyze this phenomenon later.

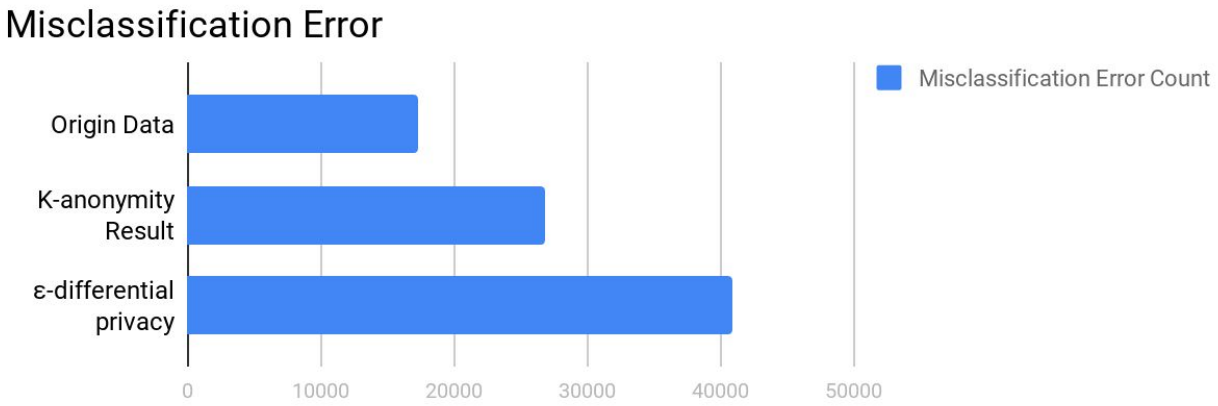
	PDF	Histogram
Original Data		
K-anonymity		

ε-differential privacy		

Density functions of these 2 algorithms show no difference with the original density functions, which indicates that K-anonymity and ε-differential privacy have no misses cost.

Misclassification Error

	Misclassification Error Count	Misclassification Error Rate
Original Data	17312	17.49%
K-anonymity Result	26775	27.04%
ε-differential privacy	40805	41.22%



The misclassification error analysis shows that both of these 2 algorithms will increase the misclassification error rate. K-anonymity has a smaller influence on misclassification error than

ε-differential privacy as it makes smaller changes to the original data. This is in line with theory as k-anonymity makes fewer changes to sensitive data.

Compare Output Against Hypothesis

In hypothesis, we give out the following assumptions:

(1) k-anonymity does not induce randomization, the information in the dataset could be deduced by attackers. ε-differential privacy should have a good privacy-preserving performance over k-anonymity.
(2) k-anonymity does not perform as well when applying to high dimensional dataset.

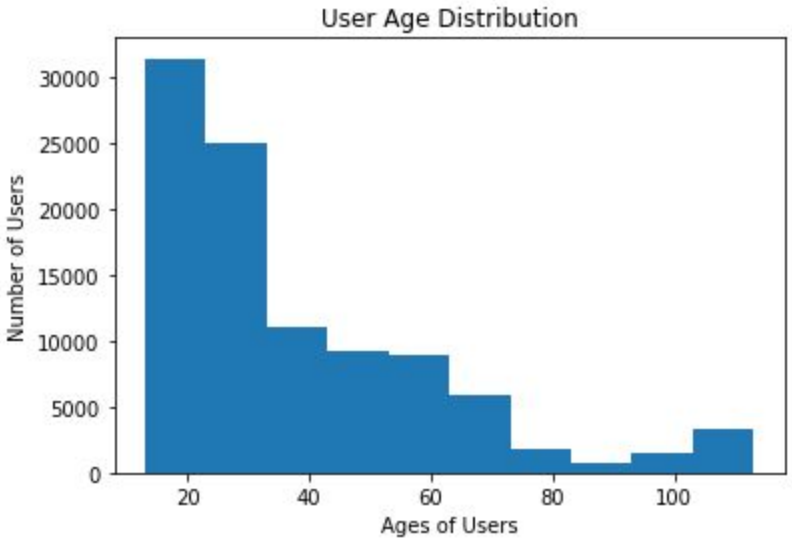
These two assumptions are consistent with the experiment, as ε-differential privacy has less hidden failure than k-anonymity. While the hidden failure rate of k-anonymity is 72.96%, the hidden failure rate of ε-differential privacy is just 10.11%. Due to the complexity of social data, neither of these 2 algorithms can fully protect sensitive data. The hidden failure rate of k-anonymity is extremely high, which indicates that k-anonymity is not suitable for social network privacy-preserving. We will analyze it in detail in the following part.

(3) ε-differential privacy randomizes the dataset to obscure private attributes, which might cause the loss of utility

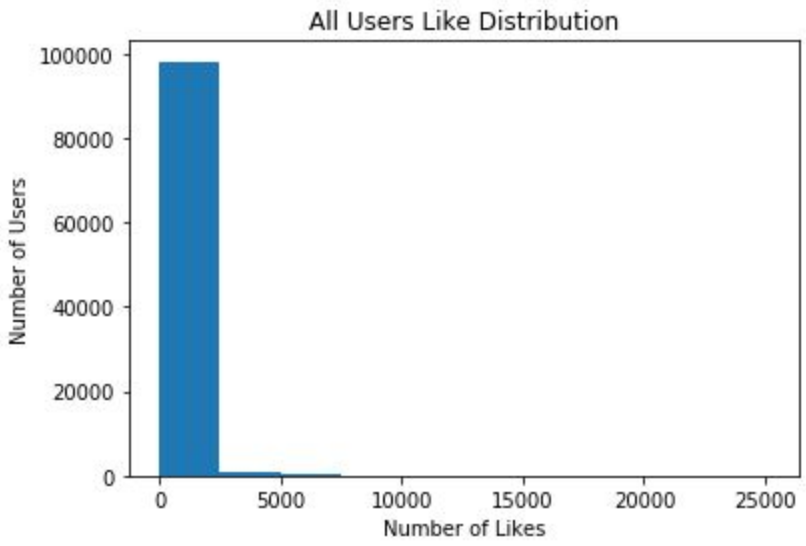
This hypothesis is also consistent with the experiment, as there are less Misclassification Error on k-anonymity. This is because k-anonymity makes fewer changes to sensitive data. When doing data protection, we need to do a trade off between data utility and data privacy.

Abnormal Case Explanation

The abnormality in our results around the k-anonymity implementation comes from the dataset of users itself. As k-anonymity functions by ensuring that any row in a dataset appears identical to at least k-1 other rows, it accomplishes this by collecting similar users into buckets representing a larger group through a process called generalization. The weakness of k-anonymity comes when users in the dataset are so disparate that it becomes impossible to place them into buckets without making them so large as to be functionally useless for the purposes of conducting any meaningful analysis off of, which results in significant hidden failure.

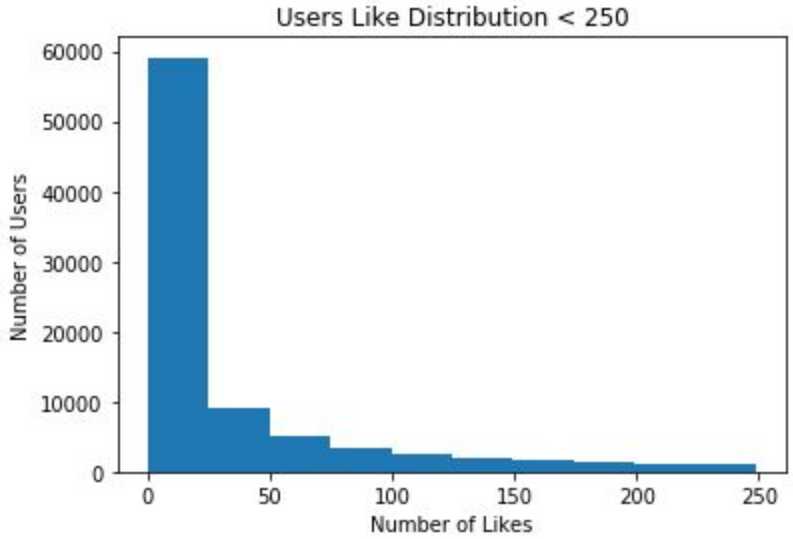


Histogram of Ages in Dataset



Histogram of Likes Given in Dataset

From the figures above, it seen that the relative distribution of ages among the facebook users was as expected, leaning heavily on younger users, but not so much that it would on its own pose a problem for the anonymity of the users. However, for the “Likes” attribute which measured how many likes a given user gave demonstrated a significant difference. The attribute clustered its results heavily towards the lower end, with 87.15% of users having given under 250 likes, but with some users giving well in excess of 20000 likes.



Histogram of All Users That Gave Less Than 250 Likes

While it may be initially tempting to blame the dataset for its distribution of likes being far too low, due to the fact that it is a pseudo-dataset. However, many sources, such as the digital consumer intelligence company Brandwatch, place the average monthly likes a Facebook user gives to be ten likes per month [17]. Which reflects well on how in the pseudo-dataset, 53.17% of users have given less than fifteen likes. The way in which this ends up damaging the Mondrian implementation of k-anonymity is that all users that have given fifteen or more likes are spread out so thinly that it becomes difficult to maintain meaningful buckets that would ensure that there are k-1 other users from which they are indistinguishable from.

As age was selected to be the sensitive feature that would need to be protected, it would be unsurprising that it would be easy to identify a user based on their like count. This does not, however, fully account for the 72.96% hidden failure rate generated by the algorithm. For that, it must be considered that there are more features than just likes given available as viewable data. When the results of those features are provided as well, it becomes easier to identify a specific user based on their non-sensitive data.

Discussion

High levels of variance in demographics means that it is better to anonymize social media data by altering the data itself rather than trying to group them together to prevent individual identification. In this experiment, a high-dimensional social data is used, which is an extreme case for most privacy-preserving methods. Grouping methods like k-anonymity show a bad performance over privacy-preserving. ε-differential privacy will do some modification to sensitive data, so it is more suitable for protecting social data. Altering the data will reduce the data utility. As the experiment points out, misclassification

error rate is increased in both algorithms. K-anonymity and its derivative algorithms should be chosen if data utility is crucial.

Conclusion and Recommendations

Summary and Conclusions

Initially setting out to compare two different frameworks of privacy, k-anonymization and ε-differential privacy, in terms of their ability to provide security to social media data, this paper has come to conclusive results. For highly dimensional data, which would be generally expected for any dataset acquired from a site that gathers as much information as social media sites do, it was found that the Mondrian implementation of k-anonymization was significantly less effective in maintaining the privacy of the users. While the Laplacian implementation of ε-differential privacy did result in a notable drop in the utility of the dataset, the difference in the level of privacy that it provided compared to k-anonymization was high enough to counterbalance the loss in utility. A dataset that does not maintain the anonymity of a majority of its members cannot be seen as superior.

Ultimately, the failure of k-anonymity comes from its approach to providing privacy to members of a dataset. Unlike ε-differential privacy which alters the contents of the dataset itself, k-anonymization attempts to ensure that there are sufficient similar data points so that an individual cannot be identified among them. For social media datasets which can provide a high level of features, this presents a problem, as it becomes difficult to generalize non-sensitive columns to ensure that k-anonymity is achieved. When the features exhibit a high level of spread for their data, the problem compounds upon itself. Since these are aspects that would be expected to be common to social media datasets, it cannot be said that k-anonymity is a privacy framework well suited to them. Thus, in accordance with initial

expectations, it was determined that ε-differential privacy was the most effective way to ensure that social media user's privacy would be maintained while data mining sites such as Facebook.

Recommendations for Future Studies

The most obvious paths for future work on the subject of this paper would be to expand the scope to include other privacy frameworks, and to expand the number of implementations of those frameworks in order to have a clearer understanding of their strengths and weaknesses. For example, l-diversity is an expansion on k-anonymity which would be interesting to investigate the differences between itself and k-anonymity.

The main goal of any future exploration, though, must be searching for frameworks and algorithms that effectively preserve the privacy of outliers. Being the most vulnerable members within a dataset to identification under the k-anonymity framework, they raise the question of whether it is possible to effectively anonymize outliers without altering the data as ε-differential privacy does. If not, the next point that must be answered becomes how much privacy loss is acceptable compared to the loss in utility of the ε-differential privacy method.

Bibliography

[1] Agrawal, Rakesh, and Ramakrishnan Srikant. "Privacy-preserving data mining." *In Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 439-450. 2000.

[2] Sweeney, Latanya. "k-anonymity: A model for protecting privacy." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10, no. 05* (2002): 557-570.

[3] Ni, Sang, Mengbo Xie, and Quan Qian. "Clustering Based K-anonymity Algorithm for Privacy Preservation." *IJ Network Security 19*, no. 6 (2017): 1062-1071.

[4] Dwork, Cynthia. "Differential privacy: A survey of results." *In International conference on theory and applications of models of computation*, pp. 1-19. Springer, Berlin, Heidelberg, 2008.

[5] "Trustpilotreviews." PyPI. Accessed February 24, 2020. https://pypi.org/project/trustpilotreviews/.

[6] "Amazon Customer Reviews Dataset." Amazon Customer Reviews Dataset - Registry of Open Data on AWS. Accessed February 24, 2020. https://registry.opendata.aws/amazon-reviews/.

[7] Qiyuangong. "Qiyuangong/Mondrian." GitHub. Accessed February 24, 2020. https://github.com/qiyuangong/Mondrian.

[8] Ibm. "IBM/Differential-Privacy-Library." GitHub. Accessed February 24, 2020. https://github.com/IBM/differential-privacy-library.

[9] "Differential-Privacy." PyPI. Accessed February 24, 2020. https://pypi.org/project/differential-privacy/.

[10] Nuclearstar. "Nuclearstar/K-Anonymity." GitHub, October 20, 2019. https://github.com/Nuclearstar/K-Anonymity.

[11] Nuclearstar. "Nuclearstar/K-Anonymity." GitHub, October 20, 2019. https://github.com/Nuclearstar/K-Anonymity.

[12] "(Tutorial) Simplifying Sentiment Analysis in Python." *DataCamp Community*. Accessed February 24, 2020. https://www.datacamp.com/community/tutorials/simplifying-sentiment-analysis-python.

[13] "Part of Speech Tagging with Stop Words Using NLTK in Python." GeeksforGeeks, February 2, 2018. https://www.geeksforgeeks.org/part-speech-tagging-stop-words-using-nltk-python/.

[14] Hale, J., and S. Shenoi. "Catalytic Inference Analysis: Detecting Inference Threats Due to Knowledge Discovery." Proceedings. *1997 IEEE Symposium on Security and Privacy (Cat. No.97CB36097)*, n.d. https://doi.org/10.1109/secpri.1997.601333.

[15] Aggarwal, Charu C. (2005). "On k-Anonymity and the Curse of Dimensionality". *VLDB '05 – Proceedings of the 31st International Conference on Very large Data Bases*. Trondheim, Norway. CiteSeerX 10.1.1.60.3155. ISBN 1-59593-154-6.

[16] Dwork, Cynthia, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. "Our data, ourselves: Privacy via distributed noise generation." *In Advances in Cryptology-EUROCRYPT* 2006, pp. 486–503. Springer Berlin Heidelberg, 2006.

[17] Smith, Kit. "53 Incredible Facebook Statistics and Facts." Brandwatch. Brandwatch, June 1, 2019. https://www.brandwatch.com/blog/facebook-statistics/.

[18] Lefevre, K., D.j. Dewitt, and R. Ramakrishnan. "Mondrian Multidimensional K-Anonymity." *22nd International Conference on Data Engineering (ICDE06)*, 2006. https://doi.org/10.1109/icde.2006.101.

[19] Nuclearstar. "Nuclearstar/K-Anonymity." GitHub, October 20, 2019. https://github.com/Nuclearstar/K-Anonymity.

Appendices

Program Source Code, input/output files, and readme documentation can be found on the following Github repo:

https://github.com/zhehedream/COEN281