Final Report

Generating Differentially Private Datasets Using Deep Learning
(in collaboration with: Office for National Statistics)

Module: 1819-CM3203 - 40 credits

Author: Wei Loon Teh

Supervisor: George Theodorakopoulos

Moderator: Luis Espinosa-Anke

Acknowledgements

I would like to extend my sincere gratitude and thanks to my supervisor Professor George Theradoukopoulos and to Dr Ioannis Kaloskampis from the Office of National Statistics. Their expertise and guidance was invaluable in the forming of the project, especially the methodology

I would also like to thank my family for all their love and support. And to my friends, for the happy diversions and helpful discussions.

Abstract

The release of sensitive private data for the purposes of analytics requires the protection and guarantees of privacy. This project implements two different mechanisms to achieve differential privacy and evaluates and compares them together. It is found that the Generative Adversarial Networks are capable of generating data that closely mimics distribution of table data while providing aspects of data quality and privacy, but the Gaussian Noise mechanism is more controllable and accurate in terms of data quality and privacy

Introduction

Government organisations, businesses, academia, members of the public and other decision-making bodies require access to a wide variety of administrative and survey data to make informed and accurate decisions. However, the collecting bodies are often unable to share sensitive data with third-parties without risking breaking the confidentiality and consent checks required to obtain this data.

Even aggregation methods or other functions that distort original data could still leak information through reconstruction attacks or model inversion attacks, such as described below in Fedrikson *et al.* (2015)

Therefore, researchers have proposed many methods for generating synthetic data to replace the raw data for the purposes of processing and analysis. A good synthetic dataset has two properties: it is representative of the original data and it provides strong guarantees about privacy.

Aside from synthetic data generation there exists methods to provide a more robust form of quantitative privacy called Differential Privacy (Dwork et al 2006). A differentially private mechanism has similar constraints of the tradeoff between utility of the data and privacy provided.

This project explores and compares methods of generating synthetic data using Deep Learning and Neural networks with traditional Differential Privacy methods while evaluating privacy and utility of the data.

Background & Related Work

Data sets have been released throughout the ages with data anonymization techniques and data aggregation used to protect privacy of the users within the dataset.

These methods include not only technological means of data protection but choices in what kind of data and how to publish. Examples include excluding data with small number of correspondents (cell suppression), creating and upper ceiling or limit (top-coding) and more. These techniques together limit statistical disclosure of the data. (Adam, N. R., & Worthmann, J. C. 1989)

Yet data released in this manner is still vulnerable to attacks such as tracing and reconstruction attacks. Reconstruction attacks seek to uncover underlying microdata by comparing released data and known attributes across multiple queries to determine secret information hidden within the dataset. A more subtle attack is the tracing attack. This attack determines whether or not a specific individual is a member or not a member of a given dataset, a more modest goal (Dwork et al, 2017)

These types of attacks can be mitigated by Differential Privacy, (Dwork et al 2006). Differential Privacy is a type of quantitative privacy goal where a procedure M take sensitive data x and releases output M(x). Output M(x) when compared with a different output M(y) where y and x differs only one one single individual, M(x) and M(y) are indistinguishable. This doesn't prevent no information about a person being released, but mitigates the problem.

There are established mechanisms of differential privacy, such as adding noise to the data. But here we consider exploring a different method of doing so, synthetic data.

Synthetic data is the idea of generating data to meet specific needs or conditions that may not be found in original data. It is often representative of the authentic dataset and dates back to 1993, proposed by Rubin to release data samples without disclosing microdata.

With regards to synthetic data, the project was proposed in conjunction with Dr. Ioannis Kaloskampis, attached to the Office of National Statistics Data Science Campus. He has done preliminary work with the office of Data Science regarding data generation with GANs.

There are various methods of generating synthetic data, (Variational Auto-Encoders, SMOTE, etc) but the one that is of interest for this project is Generative Adversarial Networks. GANs for short, (Goodfellow et al 2014)

This method is to simultaneously train two models which Generator G and Discriminator D. Generator G tries to create data that imitates the real data, while discriminator D tries to tell between generated and real data.

The idea here is to evaluate an compare GAN data generation, which releases no 'real' data, offers any inherent privacy when evaluated as a differential privacy mechanism.

Design & Implementation

Overview

The project will be split into two stages: implementing and evaluating two different mechanisms for achieving differential privacy, synthetic data generation using GANs (Goodfellow et al 2014) and gaussian noise addition

The two methods will be experimentally tested vs the definition of differential privacy described below

> Definition : (Differential Privacy, Dwork et al. 2006b). A mechanism M satisfies ε - differential privacy if, for any datasets x and y differing only on the data of a single individual and any potential outcome q,
>
> $$P[M(x) = q] \leq e^{\varepsilon} \cdot P[M(y) = q]$$

To give a brief overview of the overall project, the steps are below.

1. Synthetic Data Creation
    - 1.1. Develop and evaluate GAN suitability to produce synthetic data
    - 1.2. Tune GAN to provide closest representation to real data
    - 1.3. Create GAN training system to save and load the weights of trained layers.
    - 1.4. Train WGAN on full data, save the weights of the dataset
2. Gaussian Noise Addition
    - 2.1. Test adding Gaussian noise to histogram plots of the data
    - 2.2. Create system to change std variation to provide different noise
3. Compare and evaluate the mechanisms
    - 3.1. Evaluate quality of mechanism by comparing M(data) with original data
    - 3.2. Evaluate privacy of mechanism by comparing M(data) vs M(data minus i-th row)

Further details on implementation are below, and more details on evaluation methodology in the evaluation section

Dataset

For the purposes of our project, we are working with the US Adult Census dataset (Dua, D. and Graff, C. 2019), limiting it to 6 continuous variables and the first 5000 records.

Data	Description
age	age
fnlwgt	Final Weight, a sampling weight of the dataset
education-num	Numerical representation of education
capital-gain	Income from sources apart from wages
capital-loss	Losses from sources apart from wages
hours-per-week	Hours worked per Week

Example shown:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
0	39	77516	13	2174	0	40
1	50	83311	13	0	0	13
2	38	215646	9	0	0	40
3	53	234721	7	0	0	40
4	28	338409	13	0	0	40
5	37	284582	14	0	0	40
6	49	160187	5	0	0	16
7	52	209642	9	0	0	45
8	31	45781	14	14084	0	50
9	42	159449	13	5178	0	40
10	37	280464	10	0	0	80
11	30	141297	13	0	0	40
12	23	122272	13	0	0	30
13	32	205019	12	0	0	50

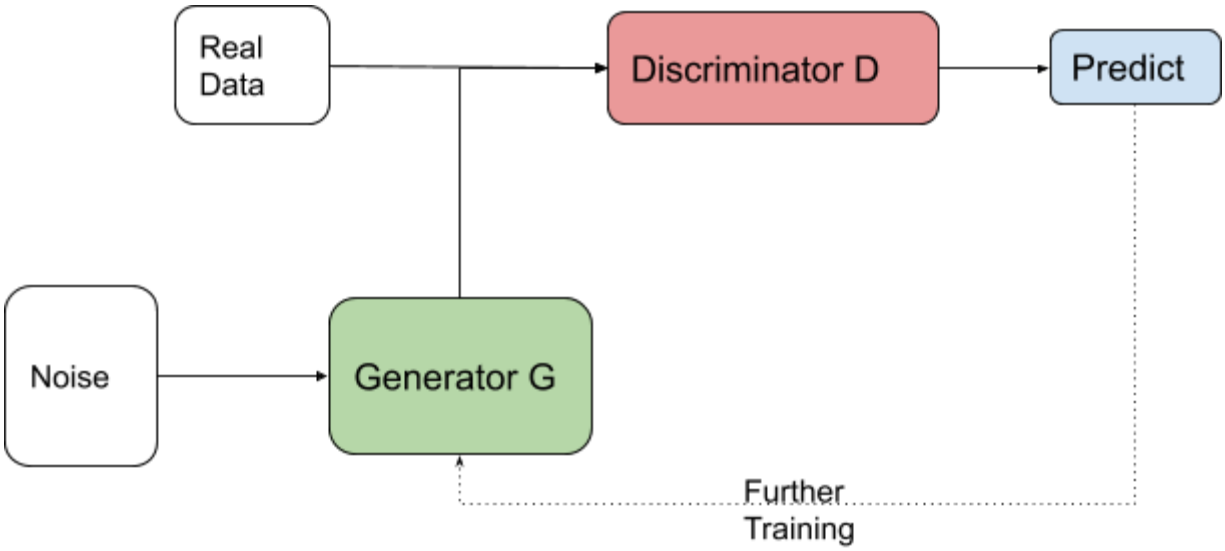
All variables are treated as normal data features and network is trained on the dataset.
Data was pre-processed and normalized before training

Synthetic Data Creation

Generative Adversarial Networks is a class of machine learning algorithm that takes a game-theoretic approach. The Generator and Discriminator learns to generate data that approximates the distribution of real data through an adversarial mini-max game.

Two competing networks play against one another. The Generator takes noise as input and creates samples of approximate real data. The Discriminator network is fed both real and generated data and tries to determine which one is the real data. These two networks learn off of each other, each performing better at their task to achieve the goal of creating synthetic, highly realistic data.

Figure 1.1 : Generative Adversarial Network



Most of the achievements in the field of generative adversarial networks have not been with table data, such as this project but rather images, such as CycleGAN and StyleGAN which does domain and style transfer, (real scenery to paintings, zebras to horses) or image combination such as GANbreeder.

This portion of the project was supervised and advised by Dr Ioannis Kaloskampis, attached to the Data Science Campus at the Office of National Statistics. His work and advice was indispensable to the project.

GAN Implementation

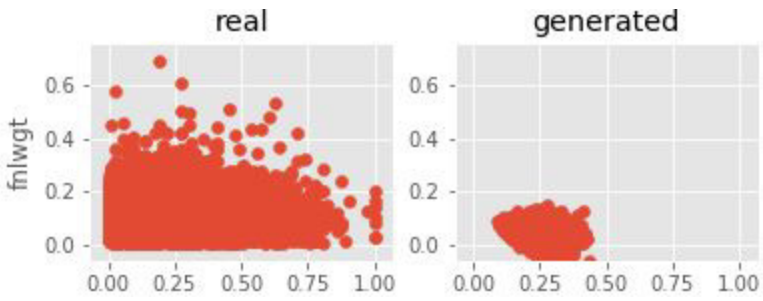
GANs were implemented in Python, using the Keras and Tensorflow packages for building neural networks. Matplotlib and seaborn were used for plots and sklearn package for data preprocessing and normalization.

GAN training and weight saving-loading system of code adapted from publicly available github code of Cody Nash, which was used for credit card fraud data.

Keras and Tensorflow was used to build the GANs and WGAN due to suitability for deep neural networks, compared to the more NLP focused packages available in PyTorch. Keras High-level API was sufficient for the task.

Initial 'Vanilla' GAN implementation was very difficult. Multiple problems with GANs exist such as mode collapse, only generating one type of correct data, leading to a lack of diversity in distribution

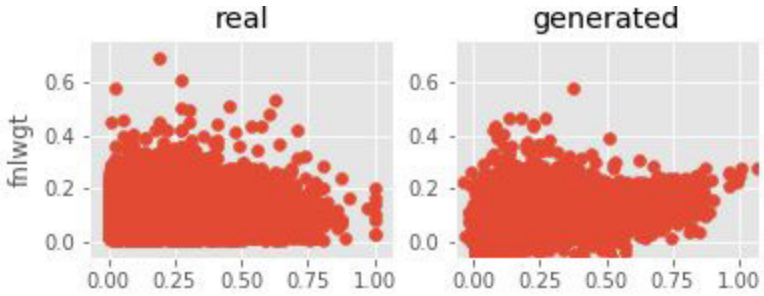
Figure 1.2: Mode Collapse in standard GAN, data not distributed properly



We explored developing two types of GANs with different objective functions: WGANs (Arjovsky et al 2017) using Wassertein Earth Mover distance WGAN and traditional GAN using Jensen Shannon Divergence.

Earth Mover distance is an objective function which assesses distance between probabilities WGANs were developed by Martin Arjovsky and colleagues to improve the stability of learning, and reduce problems of mode collapse. Dr Kaloskampis's previous work had also shown that WGANs perform better for the dataset

Figure 1.3 : fnlweight distribution real vs generated plotted at step 5000 using WGAN



Other problems included problem with training time. Training time using full dataset 32,251 records proved over 20 mins long, with no batch size training. By reducing the dataset size to 5000, and implementing a batch size of 128 to train the network it took about 6 minutes to train to 5000 steps which gave good results on distribution charts like above.

GANs were hypersensitive to Hyperparameters, which took trial, time and experimentation with the parameters and guidance from Dr. Ioannis to solve. Tested multiple different batch sizes, neurons per layer and steps.

Final Hyperparameters which showed promising results, evaluated through distribution histograms and correlation matrices are as such :

Figure 1.4

GAN Hyperparameters	
Layers	3 Dense Layers
Neurons per Layer	128
Batch Size	128
Steps (epochs)	5000
Learning Rate	5e-4
Critic Pre-Train Steps	100

Below is an example of a few histogram plots and correlation matrices which we used to explore the differences and compare real and generated data.

Figure 1.5 Correlation Matrix with WGAN data at step 5000

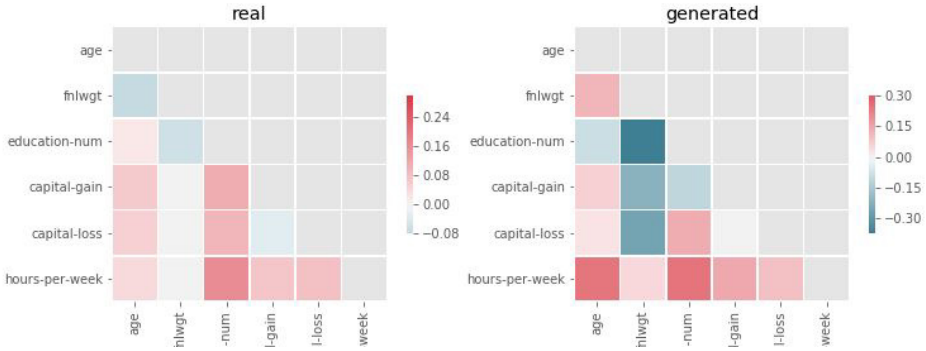
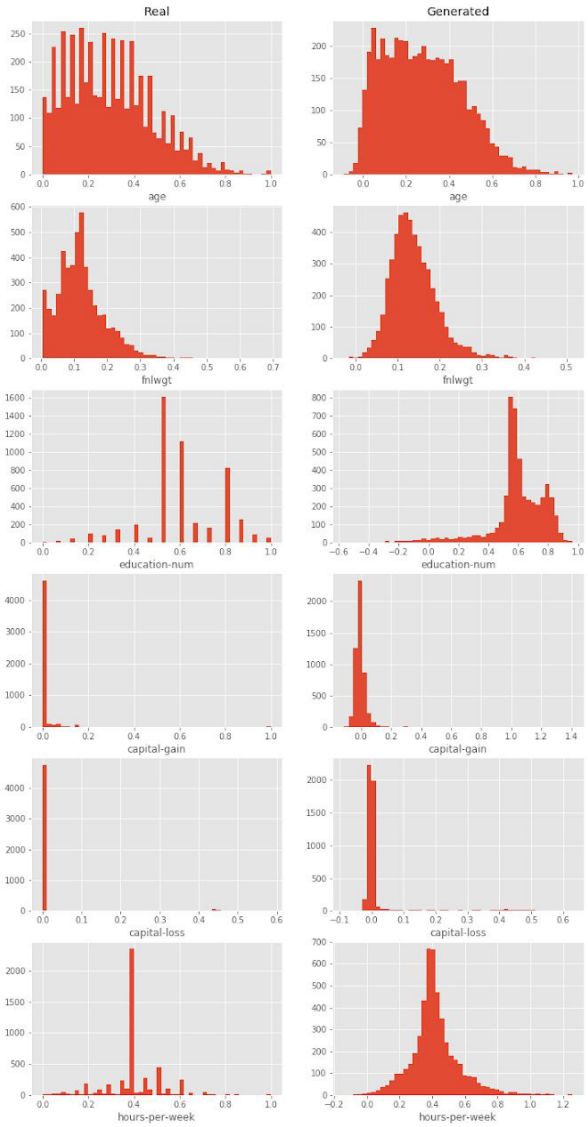


Figure 1.6 : Histogram comparison of Real vs Generated Data at step 5000 using WGAN



Gaussian Noise Addition

Output perturbation is one of the existing families of differential privacy mechanisms.
In output perturbation, after computing the output of a query, f from the user,(in our case, a histogram of the distribution of values) random noise variable Z is added to the output, which is calculated by function N (Dwork and Roth 2014)

$$M(x) = f(x) + Z \text{where } Z \sim N$$

There exist multiple different styles of output perturbation but the one of focus for our project is Gaussian noise output perturbation.

$$Z \sim N(0, \sigma^2)$$

The Gaussian Mechanism calculates a zero mean isotropic Gaussian perturbation based on σ (std deviation)

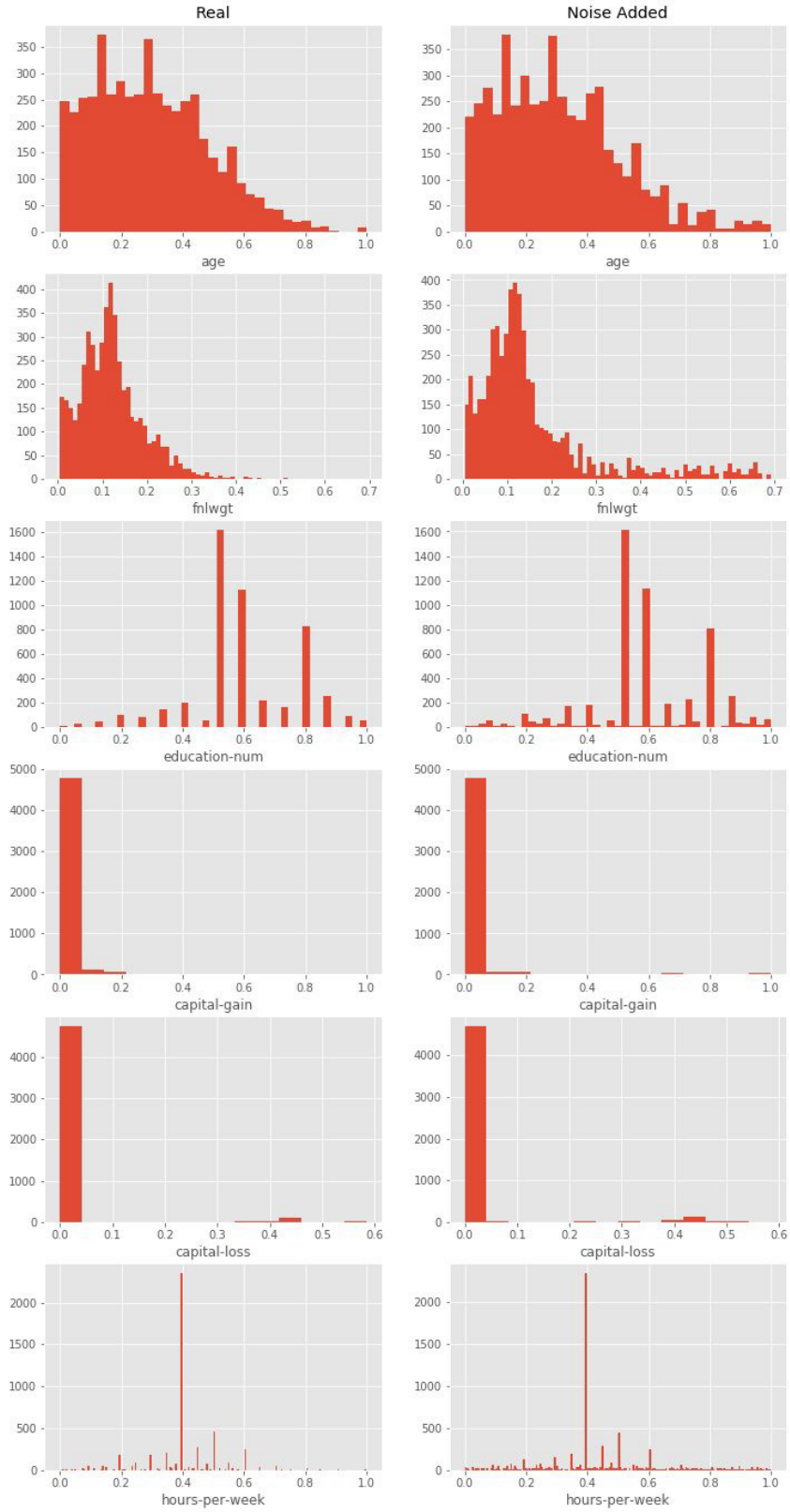
The higher the value of σ, the more noise is added to the data and the more privacy you achieve. The lower the value of σ, the more representative the data is.

The trade-off of adding higher or lower amounts of noise depends on the priority of the users of the dataset. Is it more important to achieve better privacy protection, which lowers the probability of an individual being detected within the dataset, or to preserve the representativeness of the data that is being protected by differential privacy.

Implementation was done using numpy.random packages that can generate gaussian distributed data in a given shape. To account for accurate data in the histogram, after adding random noise, data is rounded up or down to the nearest integer, given that histogram counts cannot have decimal points.

Example data is shown on the next page.

Figure 1.7 : Histogram Plots of Real data vs Noise added data with a $\sigma = 20$



Measuring and Comparing Histograms

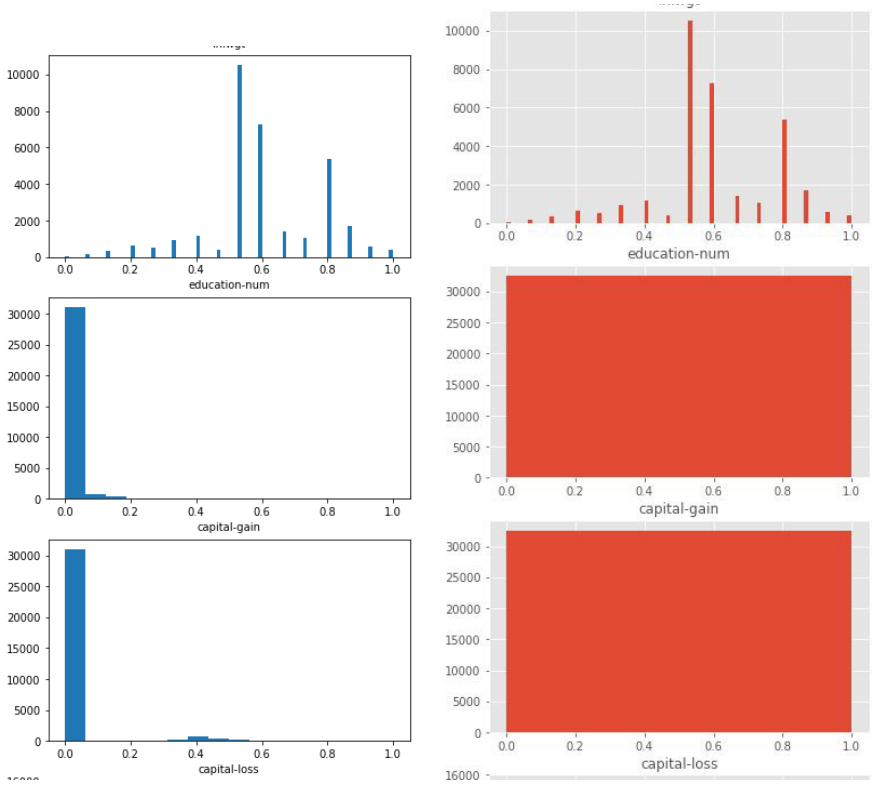
To compare the output of two histograms, the output can be calculated by calculating the sum of absolute differences between each bin, so long as the bin edges remain the same for both histograms.

$$\sum_{n=bins} | hist(i) - hist(j) |$$

By determining and setting the bin edges of the histograms, we use the sum of the absolute difference between each bin as our indicator as to how similar the two distributions are.

Bin edges are determined by using a combination of the of the Freedman Diaconis Estimator (Freedman et al 1981), for large datasets and the Sturges (Scott D.W. 2009) estimator for smaller datasets.

Figure 1.8 Comparison of histograms using both (left) vs Freedman Diaconis alone (right)



Evaluation & Conclusions

Data Quality

For the purposes of the project, we are considering distribution of the data as the main query performed and will evaluate quality based on distribution.

Comparing two histogram distributions is challenging, we opted to try two different approaches, Chi-Squared tests and Absolute Difference.

After experimenting with chi-squared tests for frequency using the Python stats package, the data frequencies of the dataset were found to be unsuitable for the test due to the limitation of chi squared test requiring at least 5 counts in all frequency categories.

To evaluate data quality of the mechanisms, the solution is to compare histogram of full raw data hist(x) and histogram of data after hist(M(x))

GAN mechanism

$$\sum_{n=bins} | hist(x) - hist(GAN(x)) |$$

Noise mechanism

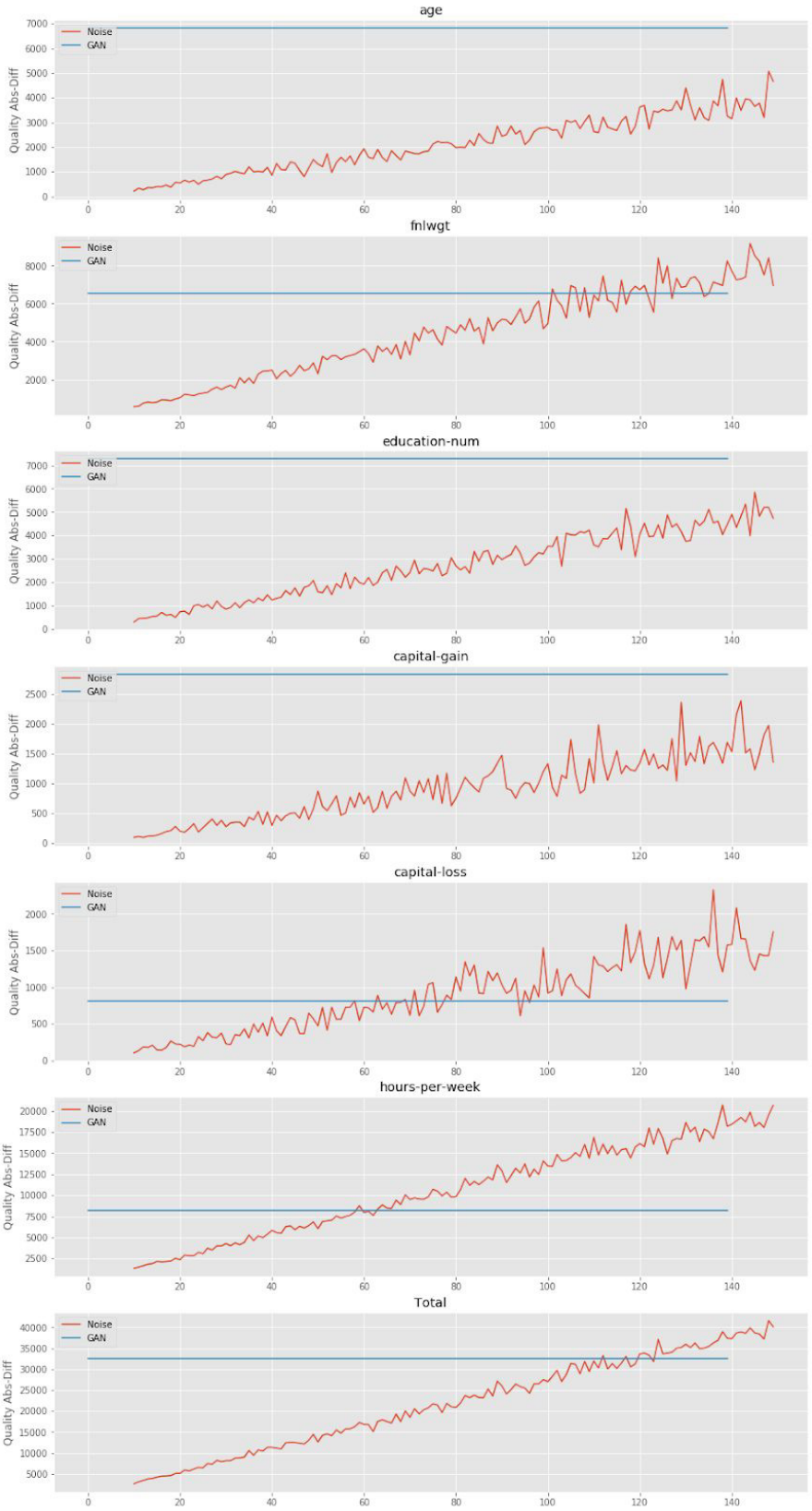
$$\sum_{n=bins} | hist(x) - hist(Noise(x)) |$$

This gives the absolute difference between distributions before and after each mechanism. The smaller the absolute difference, the higher the data quality, as it is more representative of the real data.

To take into account randomness of both mechanisms, as data from noise and GAN is generated from random input, a mean average output after 30 times is used to compare with real data.

Results are below.

Figure 2.1 : Absolute Difference between hist against σ noise. GAN parameters remain static.


age
Noise
GAN
Quality Abs-Diff
fnlwgt
Noise
GAN
Quality Abs-Diff
education-num
Noise
GAN
Quality Abs-Diff
capital-gain
Noise
GAN
Quality Abs-Diff
capital-loss
Noise
GAN
Quality Abs-Diff
hours-per-week
Noise
GAN
Quality Abs-Diff
Total
Noise
GAN
Quality Abs-Diff

As you can see, the GAN achieves high difference in distribution compared to the low noise noise sigma variable.

This is because although the trained data retains similar distributions and correlations, there are significant differences to real data still. Also, the method of evaluation using absolute differences is not an exact measure of the goodness of fit between two distributions.

Noise perturbation is still the more accessible method of preserving data quality, as tuning GANs is a trial and error process which can drastically alter output based on the hyperparameters

Testing for Data Privacy

Each mechanism will be tested by removing a single line of data, running the mechanism on both the full data M(x) and the 'data minus i-th row' M(y) and compared against each other. This will be repeated for the top 40 rows of the dataset to get an average.

GAN mechanism

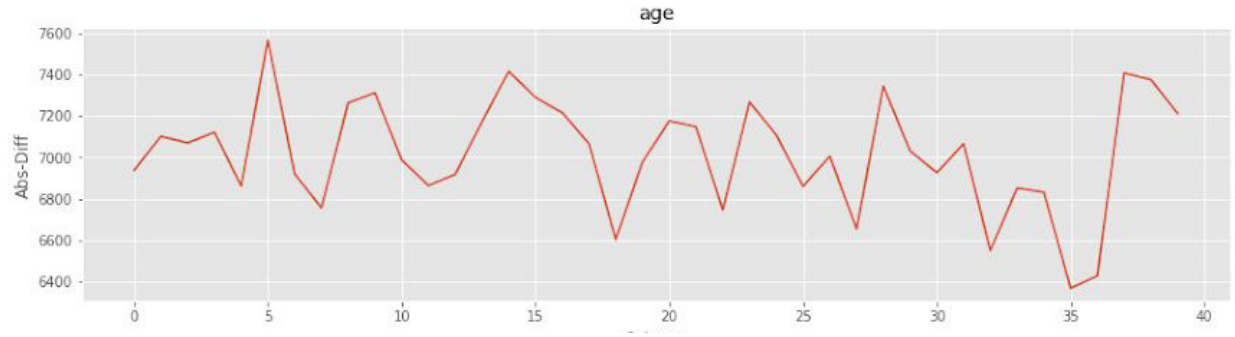
$$Avg \sum_{n=bins} | hist(GAN(x)) - hist(GAN(y)) |$$

Noise mechanism

$$Avg \sum_{n=bins} | hist(Noise(x)) - hist(Noise(y)) |$$

Where y = x with i-th data row removed, 0< i < 40

Figure 2.2 : i-th Row impact on absolute difference



This plot shows the removal of which rows have a higher impact on the overall mechanism output. The lower the absolute difference, the lower the likelihood of the data being discovered by an attacker

However, the definition of differential privacy states that it is the likelihood of detection of a single individual that determines how privacy protected the dataset is. Therefore by determining

a threshold T where if the absolute difference is over T, the detector/attacker will be able to discover the information hidden, we can split the absolute difference and obtain a probability. This could be an arbitrary threshold but a better way for exploring that will be explored in future work.

Figure 2.3 Histogram of Absolute Difference & Row Count for GAN mechanism

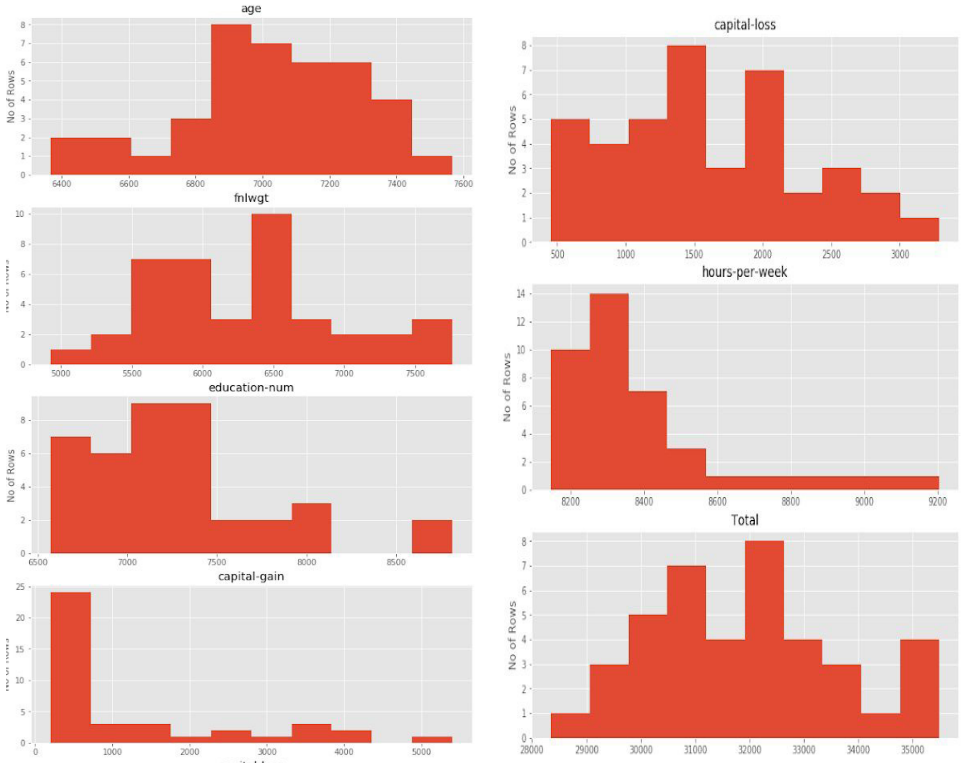
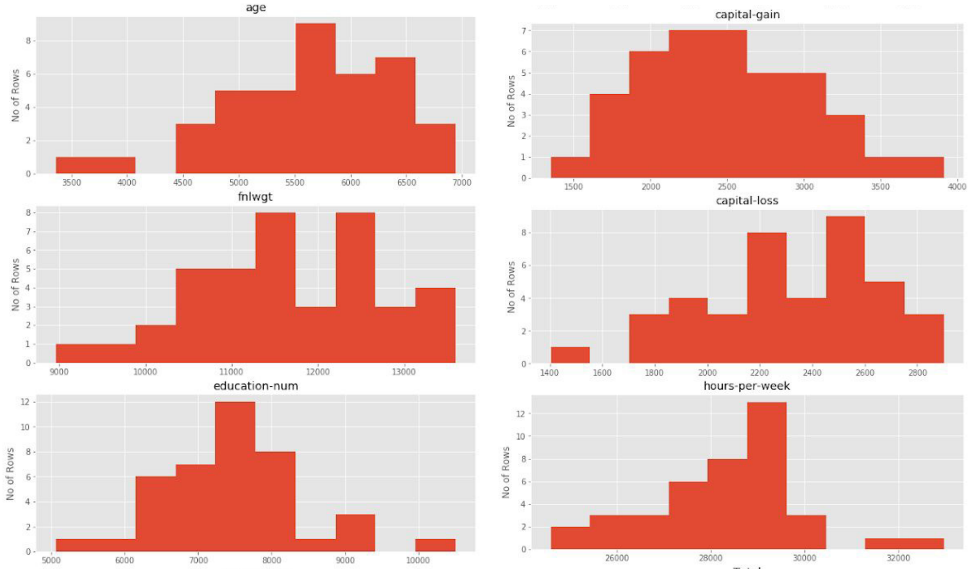


Figure 2.4 Histogram of Absolute Difference & Row Count for Noise mechanism



5. Future Work

For Future work in the project, we will first look at future work available on the mechanisms themselves.

More experimentation on the hyperparameters of the GAN to further improve the output of the synthetic data would be needed to compare against differential privacy. Changing the number of layers, which was not experimented on heavily during the testing of the project, or even trying out other types of GANs such as Conditional GANs. Adding support for categorical or labeled data would be an improvement, as currently the GAN only works on categorical data

More work could be done in the area of the output perturbation noise mechanisms. Laplace distribution noise is an alternative that could be explored

As for the evaluation process, further looking into methods of comparison would be better. Log-Likelihood tests or a kernel density estimate sampling could improve the comparison of the frequencies to give a better idea of goodness of fit.

Implementing an attacker/discriminator to build a more robust system for experimental testing of Differential Privacy, as it would be able to empirically assess of how often and how probable it is to obtain the protected data from the real data.

Finally, experimenting with a hybrid combination would be interesting to compare to both pure GAN and pure noise. Adding noise to the data pre-gan training could show interesting results.

Reflections on Learning

Reflecting on the journey of the entire project, I thoroughly believe that I've learned a lot. There are skills that I've gained throughout this project that I will carry with me into my work and apply it liberally.

Firstly I now know exponentially more about neural networks, GAN's, statistical analytics and programming in python, I've learned the basics about Differential Privacy and its mechanisms as well as how to troubleshoot more thoroughly, carefully and deliberately. The guidance of Professor George Theodorakopoulos and Dr Ioannis was key into getting the project off the ground, as I had no prior experience with neural network programming and differential privacy.

Not to undermine the 'hard skills' that I have learned throughout the project but there are a few other key skills I identified that would carry me into my career. Number one would be the how to perform research. I've never undertaken a research type project where the goal of the project is not only to complete the project in the standard manner, but also to test and evaluate all different methods available. A research is not only guided by the prompt or the supervisor, it is more independent. How would I go about solving this problem, or testing this theory in the context of the project. It challenges your own assumptions and concepts that you understand about the subject. Many of the skills I have now at the end of the project, would greatly improve the process of any research type project that I undertake in the future.

There were a few assumptions that I had made about the project that in hindsight could've saved me much time and effort. Reading the internal documentations more clearly on the packages as well as understanding that a thorough understanding of the methodology would help speed things along.

These are the skills that I will carry on into my next projects. I will continue to ask more pertinent questions of myself, when evaluating a problem. Is this methodology truly the right way? Are there any key areas that need more research with the decisions about the data, the relevant approach to reduce trial and error and wasting time?

Thank you once more for the learning experience and I will build upon the skills and approaches that I have learned during this project and grow through the projects I undertake in the future.

5. References

1. Adam, N. R., & Worthmann, J. C. (1989). Security-control methods for statistical databases: a comparative study. ACM Computing Surveys, 21(4), 515–556. https://doi.org/10.1145/76894.76895
2. Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. Retrieved from http://arxiv.org/abs/1701.07875
3. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. UCI Adult Learning Dataset https://archive.ics.uci.edu/ml/datasets/adult
4. Dwork, C., Roth, A., Dwork, C., & Roth, A. (2014). The Algorithmic Foundations of Differential Privacy. Foundations and Trends R in Theoretical Computer Science, 9, 211–407. https://doi.org/10.1561/0400000042
5. Dwork, C. (2006). Differential privacy. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4052 LNCS, 1–12. https://doi.org/10.1007/11787006_1
6. Dwork, C., Smith, A., Steinke, T., & Ullman, J. (2017). Exposed! A Survey of Attacks on Private Data. Annual Review of Statistics and Its Application (2017). Retrieved from https://privacytools.seas.harvard.edu/publications/exposed-survey-attacks-private-data
7. Fredrikson, M., Jha, S., & Tech, C. (n.d.). Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures Thomas Ristenpart. https://doi.org/10.1145/2810103.2813677
8. Freedman, D., & Diaconis, P. (1981). On the histogram as a density estimator:L 2 theory. Zeitschrift F�r Wahrscheinlichkeitstheorie Und Verwandte Gebiete, 57(4), 453–476. https://doi.org/10.1007/BF01025868
9. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative Adversarial Networks. Retrieved from http://arxiv.org/abs/1406.2661
10. Kaloskampis, Ioannis (2019) Synthetic data for public good https://datasciencecampus.ons.gov.uk/synthetic-data-for-public-good-and-art/
11. Nash, Cody [Source code] GANs for Fraud Data https://github.com/codyznash/GANs_for_Credit_Card_Data
12. Scott, D. W. (2009). Sturges' rule. Wiley Interdisciplinary Reviews: Computational Statistics, 1(3), 303–306. https://doi.org/10.1002/wics.35