Project Report : CS 7643
Differential Privacy

Krishna Kumar
Georgia Institute of Technology
Atlanta, Georgia
kkumar80@gatech.edu

Sumit Machwe
Georgia Institute of Technology
Atlanta, Georgia
smachwe3@gatech.edu

Abstract

Differential Privacy ensures to protect the private data such as social security numbers, age, gender, or any sensitive information (PII - Personally Identifiable Information) which can identify the individual persons in the dataset. This has wide applicability in domains such as finance, healthcare, demographic data, etc. [2]
Our project demonstrates the use of Differential Privacy on a Bank Credit Card Fraud detection dataset provided on Kaggle[1]. This will improve the fraud detection identification by encouraging banks, financial institutions to share the data in an differential private way, so that federated learning can be done for fraud detection and solve the class imbalance (less percentage of fraud for individual bank). Given above dataset, our project will generate a synthetic dataset that will have distributional similarity as real dataset, and Machine Learning utility which could be sub optimal to the real dataset performance by some threshold. We will demostrate distributional similarity using KDE Plots and feature correlation plots. ML utility will be demonstrated by comparing performance of real vs. synthetic dataset using AUC and F1 score.

1. Introduction/Background/Motivation

Across industry there is always a need for sharing the data which can help in solving complex problems, such as medical drug discovery, financial fraud prevention, network security while providing end user services etc. All these cases contain PII information such as age, gender, location, ipaddress etc that discourages sharing of data and collaboration.
The problem we tried to solve was to generate a synthetic dataset from a given fraud detection dataset of a bank. This synthetic dataset will have similar data distribution and machine learning utility as real dataset without compromising PII.

Data Anonymization (3-way or 5-way) is the standard practice today to protect PII. Today's data disclosure practices aim to address this issue by minimizing the number of attributes that are particularly vulnerable to re-identification. However it is prone to reverse attack where alternative datasources could be linked with anonymized data to identify the person. For example:

- 87% of Americans (based on 2010 US Census data) can be uniquely identified with only three pieces of data: Gender, Birthday and Zip Code. This is the reason why 2020 Census data is being released in a differential private dataset.
- University of Texas at Austin demonstrated reverse attack on anonimized dataset provided by Netflix in 2006. UT Austin linked the anonimized dataset to the publicly available IMDB dataset and were able to uncover many customer names.

There is wide applicability across industries to protect PIIs. Institutions and organizations will collaborate more by sharing the data for solving industrial problems while not compromising customer sensitive information. Individuals will be encouraged to participate in surveys without the concern of their information being compromised. Government agencies and regulators will be able to protect citizen data, for example, implement GDPR while sharing their data.

We will use the **Bank Account Fraud Dataset Suite (NeurIPS 2022)**[1] dataset provided on Kaggle.

This suite of datasets is based on recommendation here:

- Realistic, based on a present-day real-world dataset for fraud detection.
- Biased, each dataset has distinct controlled types of bias. Imbalanced, this setting presents a extremely low prevalence of positive class

- Dynamic, with temporal data and observed distribution shifts.

- Privacy preserving, to protect the identity of potential applicants we have applied differential privacy techniques (noise addition), feature encoding and trained a generative model (CTGAN).

- This dataset meets the section 3.1 motivations of creating the training data for the model, such as purpose, creator and funding of dataset.

This dataset provides multiple variants. However, for our experimentation, we will only use the following variants:

- Variant-V: Has **better separability in train** for one of the groups.

2. Approach

- **EDA**: Analyze and visualize the dataset. Cleanup null values, convert datatypes, identified categorical variables and visualize the data distribution. We realized that the original dataset was 1 million records with more than 40 columns. Given the compute resource limitations we decided to reduce the data size from 1 million to 300,000 records. We used the *stratify* parameter of the `train_test_split` of sklearn library on `FRAUD_BOOL` label. We also reduced the number of columns (features) based on correlation plot, retaining any feature that has ± 0.2 correlation with `FRAUD_BOOL`. This also helped to avoid Multicollinearity.

- **Baseline ML Utility**: We trained `Random Forest`, `Naive Bayes` and `Logistic Regression` models on the real (reduced) dataset. Evaluated and plotted the Precision, Recall, F1 score and AUC/ROC. This provided us the ML utility baseline performance. This should act as an upperbound on the synthetic dataset generated by our our implementation of Differential Privacy.

- **Main Algorithm for Differential Privacy:** Differential Privacy is a randomized algorithm (K), which gives
  ϵ-differential privacy if for all data sets D and D' differing on at most one row, and any $S \subseteq Range(K)$, [4]

$$Pr[K(D) \in S] \leq exp(\epsilon) \times \Pr[K(D')] \in S'$$

  Following two quantities must be considered in DP algorithms:

  - Epsilon (ϵ): A metric of privacy loss at a differentially change in data (adding or removing 1 entry). The smaller the value, the better privacy protection.
  - Accuracy: The closeness of the output of DP algorithms to the pure output measured by PMSE ratio [5]

- In order to identify the optimal DP model, we trained all three models (DP-CTGAN, PATE-CTGAN and QUAIL) on $\epsilon = 3.0$. For QUAIL we used DP-Naive Bayes as its classifier and used DP-CTGAN and PATE-CTGAN as its synthesizers.

- **Algorithm Implementation**: As a first step, we created base models using default hyper-parameters and $\epsilon = 3.0$. We evaluated the model performance and ML utility (Precision, Recall, ROC-AUC, F1 score) using below tests.

  - `TRTR`: *Train Real Test Real*
  - `TSTR`: *Train Synthetic Test Real*
  - `TSTS`: *Train Synthetic Test Synthetic*

- We compared base model performance on `TRTR` vs. `TSTR`.
  `TSTR` performance is expected to be sub-optimal to `TRTR` but not overfit. However, `TSTS` performance is expected to be the best, but overfit. Hence we decided to use `TSTR` performance as the baseline model performance.

- Based on the base model performance comparisons `DP-CTGAN` synthesizer has better performance than `PATE-CTGAN`. Our test observed that QUAIL with `DP-CTGAN` and `PATE-CTGAN` synthesizers was creating a single class for `FRAUD-BOOL`. Hence we decided to drop QUAIL from further experimentation for this dataset.

- We then selected `DP-CTGAN` and `PATE-CTGAN` for further experimentation (change $\epsilon = 10.0$ and keep other parameters intact) and repeated above steps for comparing the model performance.

- Model performance improved in the new $\epsilon = 10.0$. We again observed performance of `DP-CTGAN` to be better than that of `PATE-CTGAN`.

- Hence we decided to do various hyper-parameter tuning on `DP-CTGAN` in pursuit of finding the best model (in terms of ML Utility and Distributional Similarity). Experiments and Results section has further details on hyper-parameter tuning.

- Above experiments would be successful because we are trying to recursively narrow down on synthesizers with various test setup to find an optimal model which produces a synthetic dataset which is close to the real dataset only by data distribution and ML utility. This is a standard approach for differential privacy.
- Synthetic dataset would have a distributional similarity. We will demonstrate this by using KDE Plot and histograms. We will use the PMSE ratio as a measure.[5]
- We will run ML models (Logistic Regression, Random Forest, Decision Tree, and Naive Bayes) and compare and contrast the performance of:
  - $TRTR$: *Train Real Test Real*
  - $TSTR$: *Train Synthetic Test Real*
  - $TSTS$: *Train Synthetic Test Synthetic*
- Our approach requires hyperparameter tuning to find optimal parameters.

Problem Anticipation:

- We anticipated the runtime environment creation of Differential Privacy would be challenging due to strict library version requirements. It took more than 3 days to setup the correct set of libraries, conda environment creation.
- We also anticipated the creation of runtime environment in Cloud (GCP) would have a learning curve specially given the library versions. Local setup worked, however we could not port the setup to GCP or Google Colab. Created an Ed Discussion post, but could not solicit any good solution. Hence we decided to work on local environment for training GANs.
- Training GANs require heavy resource: GPUs, Memory and computation time. We found training on GANs is extremely time consuming. Most of the experiments took 12 to 14 hours to finish. Many experiments failed due to kernel crashes and required restarts. Local setup had no ability to parallize experiments, which slowed us down. We ultimately ended up running the expeiments on on 4 M1 Macbook Pro (32 GB CPU mem/ 8GB GPU mem) laptops.
- We expected `TSTS` test performance to overfit. Our experiments confirmed our anticipation. Hence we chose to use `TSTR` as our metrics for comparison.
- We expected to run numerous experiments to find the best model with optimal hyper-parameters. Although we ran 10 experiments, we really could not find better model performance than the baseline model. We could have expanded our hyper-parameter search if we had more time and resources in terms of hardware (GPUs and parallelization).

Important: Mention any code repositories (with citations) or other sources that you used, and specifically what changes you made to them for your project.

3. Experiments and Results

- We applied Differential Privacy algorithms on the reduced dataset (Please refer to EDA in Approach section).
- Trained `Random Forest`, `Naive Bayes`, and `Logistic Regression` models on the real (reduced) dataset and evaluated its performance using these metrics: `Precision`, `Recall`, `F1 score`, and `AUC/ROC`). As discussed in Apporach section, this is the `TRTR model` (Train Real, Test Real with 70:30 split).
- As Discussed in the approach section, For Synthetic data created using differential privacy, We calculate `Precision`, `Recall`, `F1 score`, and `AUC/ROC`).
- To measure the success of our approach, we compared ML metrics of TSTR. We also published results for `TSTS` (Train Synthetic, Test Synthetic - with 70:30 split)
- For distributional similarity between real and synthetic datasets, we measure it through KDE plots and pMSE ratios.
- **Baseline Solution**: Train the baseline models with $\epsilon = 3.0$ and $\epsilon = 5.0$. Please find below ML Utility and PMSE Ratios.

S.No	Model Key	Model Type	Model Split	Precision	Recall	F1 score	ROC-AUC	PMSE
1	Real Data	TRTR	train	0.04	0.75	0.07	0.8354	
2	Real Data	TRTR	test	0.03	0.73	0.07	0.8291	
3	Synthetic PATECTGAN	TSTR	train	0.36	0.5	0.42	0.5311	47004.33
4	Synthetic PATECTGAN	TSTR	test	0.01	0.41	0.01	0.3281	47004.33
5	Synthetic PATECTGAN	TSTS	train	0.36	0.5	0.42	0.5311	47004.33
6	Synthetic PATECTGAN	TSTS	test	0.36	0.49	0.42	0.5232	47004.33
7	Synthetic DPCTGAN	TSTR	train	0.14	0.98	0.25	0.9913	5917.88
8	Synthetic DPCTGAN	TSTR	test	0.04	0.23	0.07	0.7602	5917.88
9	Synthetic DPCTGAN	TSTS	train	0.14	0.98	0.25	0.9913	5917.88
10	Synthetic DPCTGAN	TSTS	test	0.14	0.98	0.25	0.9901	5917.88

Figure 1. Baseline ML Utlity Performance metrics. Real Dataset with 300K rows and 11 features. Synthetic Dataset with $\epsilon = 3.0$. Best model: DPCTGAN

- **Hyperparameter Tuning** We conducted 12 experiments. Please find below ML Utility, PMSE Ratios metrics for experiments having comparable results with real dataset.

S.No	Model Key	Model Type	Model Split	Precison	Recall	F1 Score	ROC-AUC	PMSE
1	Real Data	TRTR	train	0.04	0.75	0.07	0.8354	
2	Real Data	TRTR	test	0.03	0.73	0.07	0.8291	
3	Synthetic PATECTGAN	TSTR	train	0.08	0.58	0.15	0.5442	4117.04
4	Synthetic PATECTGAN	TSTR	test	0.01	0.42	0.02	0.5078	4117.04
5	Synthetic PATECTGAN	TSTS	train	0.08	0.58	0.15	0.5442	4117.04
6	Synthetic PATECTGAN	TSTS	test	0.08	0.55	0.14	0.5172	4117.04
7	Synthetic DPCTGAN	TSTR	train	0	1	0.01	0.9862	8387.17
8	Synthetic DPCTGAN	TSTR	test	0.08	0.26	0.12	0.7947	8387.17
9	Synthetic DPCTGAN	TSTS	train	0	1	0.01	0.9862	8387.17
10	Synthetic DPCTGAN	TSTS	test	0	0.79	0	0.9705	8387.17

Figure 2. Baseline ML Utlity Performance metrics. Real Dataset with 300K rows and 11 features. Synthetic Dataset with $\epsilon = 5.0$. **Champion model: DPCTGAN**

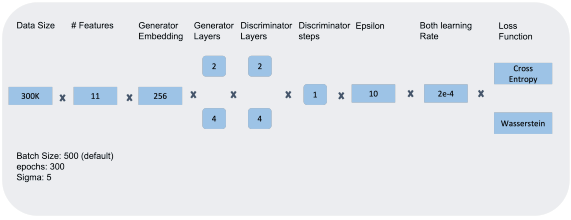

Figure 3. Experiment Designs. Base experiments were conducted as above. In addition, we conducted specific experiments to achieve better performance. Details described in next figure.

.

	Model Key	Hyper Params	Model Type	Model Split	Precision	Recall	F1 score	ROC-AUC	PMSE
1	Real Data		TRTR	train	0.04	0.75	0.07	0.8354	
2	Real Data		TRTR	test	0.03	0.73	0.07	0.8291	
3	Synthetic DPCTGAN	eps=10; loss=CE; gdepth=2; ddepth=2; epochs=300	TSTR	test	0.01	0.43	0.01	0.3409	99702.94
4	Synthetic DPCTGAN	eps=10; loss=CE; gdepth=4; ddepth=2; epochs=300	TSTR	test	0.01	0.81	0.02	0.4918	99672.45
5	Synthetic DPCTGAN	eps=10; loss=WA; gdepth=2; ddepth=2; epochs=300	TSTR	test	0.04	0.05	0.04	0.7169	88513.8
6	Synthetic DPCTGAN	eps=10; loss=WA; gdepth=4; ddepth=2; epochs=300	TSTR	test	0	0.16	0.01	0.2959	99918.93
7	Synthetic DPCTGAN	eps=10; loss=WA; gdepth=2; ddepth=2; epochs=500	TSTR	test	0	0	0	0.6676	56630.7
8	Synthetic DPCTGAN	eps=10; loss=WA; gdepth=2; ddepth=2; epochs=500	TSTR	test	0	0	0	0.5082	35468.47
9	Synthetic DPCTGAN	eps=05; loss=CE; gdepth=1; ddepth=1; epochs=350	TSTR	test	0.03	0.61	0.06	0.784	27195.78
10	Synthetic DPCTGAN	eps=05; loss=WA; gdepth=1; ddepth=1; epochs=300	TSTR	test	0	0	0	0.6314	10597.89

Figure 4. Hyperparamter Tuning ML Utility Performance metrics.

- Based on all above experiments, baseline model with synthesizer DPCTGAN, $epsilon = 5.0$ is the champion model for the given dataset. Please find its ML Utility and KDE plots.

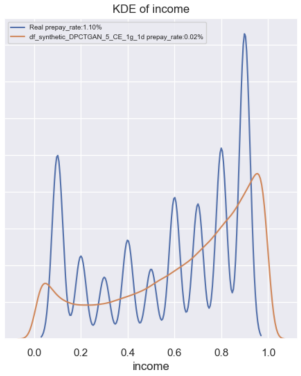

Figure 5. Real vs. Synthetic data KDE Plot: Income

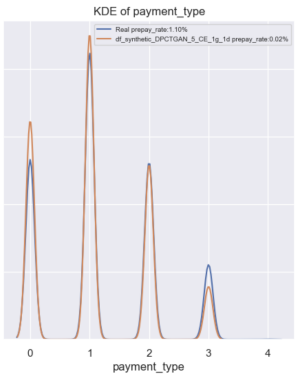

Figure 6. Real vs. Synthetic data KDE Plot: Payment Type

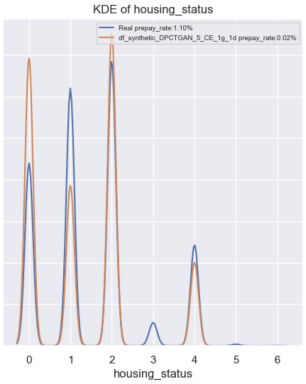

Figure 7. Real vs. Synthetic data KDE Plot: Housing Status

- **Discussion on results**:
  - In terms of ML Utility, the baseline model with EPS 5.0 is the champion model.
  - Observing the KDE Plots, we notice that generated synthetic categorical features (e.g. Payment Type, Housing Status KDE Plots) show a close similarity to real data, whereas some continuous features exhibit mode collapse (e.g. Income KDE Plot shown above)
  - Mode collapse can be rationalized by the fact that the real dataset is unbalanced. The mode collapse causes the synthesization of the label `Fraud_Bool` to be 0.02% compared to the real dataset of 1.10%.
  - We measured distributional similarity using PMSE Ratio. As expected, the champion model has the lowest PMSE Ratio.
- **discussion on success / failure**:
  - We were able to generate synthetic data which has similar ML utility as real dataset. Synthetic

dataset AUC/ROC degreaded by 4.15%, which is reasonable.

- However, we could not achieve a reasonable distributional similarity for continuous features (as evident by KDE plots)
- PMSE Ratio figures can also be improved with additional data and experimental setups.
- Training GANs is an exceptionally resource and time intensive exercise. Each expeiment took on average 12 hours to run. There is a scope to improve model performance with better resource and training/experimentation time.

4. Other Sections

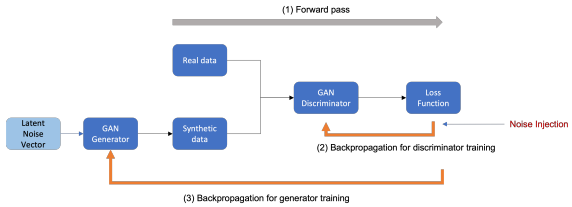

Figure 8. GAN based models Architecture

.

- Data Sourcing, EDA, Pre & Post-Processing implementation of ML Utility and distributional plots (KDE Plots) are not based on deep-learning. However training the model using GANs (Generative Advarsarial Neural Networks) are based on deep learning and involves following hyper-parameter tuning:
  - embedding layers
  - learning rates
  - epochs
  - Generator depth
  - Discriminator depths
  - Sigma and Epsilon
- We used the open-source opendp/smartnoise-sdk and the following Neural Network Synthesizers to train on the real data:
  - `DP-CTGAN`: GAN-based synthesizer that uses conditional masks to learn tabular data.
  - `PATE-CTGAN`: Conditional tabular GAN using Private Aggregation of Teacher Ensembles.
  - `QUAIL`: Quailified Architecture to Improve Labeling. Divide epsilon in a known classification task between a differentially private synthesizer and classifier. Train DP classifier on real, fit DP synthesizer to features (excluding the target label), and use synthetic data from the DP synthesizer with the DP classifier to create artificial labels.
- Pre-processing involved reducing the features and dataspace (11 features, 1 label and 300,000 records). This was input to the GAN network as an embedding.
- Output from learned GAN network was the synthetic data with same feature and record count as input data.
- **Loss Functions:**
  - `Cross-Entropy`: is a metric used to measure how well a classification model in machine learning performs.
  - `Wasserstein`: is a distance metric used in GANs to measure the difference between the generated and real data distributions, allowing for more stable training and improved quality of generated samples.
- For `TSTS` our model overfit. And hence, we decided to use `TSTR` as our baseline metrics for ML Utility comparison. Please refer to Approach section for further details.
- Future potential: Tuning GANs is resource intensive. Any GAN based approach that optimizes or improves the training time will improve differential privacy framework and its applicability to the real world.
- Currently Differential Privacy algorithms are not generalizable across datasets. Further research is required to develop generic algorithm(s) for data synthesization using differential privacy.
- Project Repository: https://github.gatech.edu/cs7643-spring-2023/differential-privacy

5. Work Division

1.

Student Name	Contributed Aspects	Details
Krishna Kumar	Data Sourcing and selection of target data	Identify and Source Dataset from Kaggle.
Sumit Machwe	Exploratory Data Analysis (EDA)	Generate Correlation plots, identify categorical and continuous variables and reduce dimentionality for experimentation.
Sumit Machwe	DiffPriv Runtime Environment Setup	Setup the complex runtime with various dependent libraries and their specific versions.
Krishna Kumar	Baseline Model: Implementation	Implement DPCTGAN, PATECTGAN and QUAIL with default parameters and chosen epsilon using smartnoise library.
Sumit Machwe	ML Utility implementation	Implement calculations for Precion, Recall, F1-Score and AUC/ROC and compiling the results in dataframe.
Krishna Kumar	KDE Plot and PMSE Ratio implementation	Implement KDE Plot and PMSE Ratio to calculate distributional similarity.
Krishna/Sumit	Hyper parameter Tuning Experimentation Setup	Design experimentation framework for hyper-parameter tuning.
Sumit Machwe	Hyperparameter Tuning: Batch-1	Run 6 experiments on DPCTGAN based on defined experimental setup.
Krishna Kumar	Hyperparameter Tuning: Batch-2	Run 6 experiments on DPCTGAN based on defined experimental setup.
Krishna/Sumit	Results, Analysis and Report	Assimilate all experiment results, plots etc and develop the final report.

Table 1. Contributions of team members.

References

[1] Kaggle Dataset: https://www.kaggle.com/datasets/sgpjesus/bank-account-fraud-dataset-neurips-2022

[2] Create privacy-preserving synthetic data for machine learning with SmartNoise https://cloudblogs.microsoft.com/opensource/2021/02/18/create-privacy-preserving-synthetic-data-for-machine-learning-with-smartnoise/

[3] 10 Breakthrough Technologies 2020 https://www.technologyreview.com/10-breakthrough-technologies/2020/differential-privacy

[4] Understanding Differential Privacy https://medium.com/towards-data-science/understanding-differential-privacy-85ce191e198a

[5] General and specific utility measures for synthetic data https://arxiv.org/pdf/1604.06651

[6] Quail Paper https://arxiv.org/abs/2011.05537

[7] DP-GAN Paper https://arxiv.org/pdf/1808.00087.pdf

[8] PATE-GAN Paper https://openreview.net/forum?id=S1zk9iRqF7

[9] CT-GAN Paper https://link.springer.com/chapter/10.1007/978-3-031-09342-5_17