IUDX Differential Privacy Pipeline

Differential Privacy at a Glance

Differential Privacy describes a promise made by a data holder or curator to a data subject: “You will not be affected, adversely or otherwise, by allowing your data to be used in any study or analysis, no matter what other studies, data sets, or information sources, are available.”

Mathematically, this is represented as:

$$\forall\ D \text{ and } D'\ \forall\ x:\ \mathbb{P}[M(D) = x] \leq \exp(\varepsilon) \cdot \mathbb{P}[M(D') = x]$$

where D and D' are any two neighbouring datasets. The differential privacy guarantee offered by this statement is that for any two neighbouring datasets D and D’ that differ only by the data of a single user, any function or process M is considered ε-differential private iff for every possible output *x*, the probability of this output being observed never differs by more than exp(ε) between the scenario with and without that user’s data.

Differential privacy requires that any query made to a dataset, regardless of the intention or motivation behind the query, should produce nearly identical results whether or not any particular individual's data is included in the dataset.

To achieve this, differential privacy involves adding a small amount of random noise to the dataset before releasing it, which makes it harder for an attacker to distinguish between the presence or absence of any individual's data in the dataset. This ensures that any analysis performed on the dataset will not be able to uncover sensitive information about any individual in the dataset, while still preserving the overall accuracy of the analysis. The amount of noise added is carefully calibrated to balance the level of privacy protection provided against the level of accuracy preserved.

An Example of DP in Use

Suppose there exists a hospital that wants to share information about the number of patients it treats for a particular disease with an external research team, but it wants to protect the privacy of its patients. To achieve this, the hospital would add noise to the data through the differential privacy mechanism to the number of patients currently in treatment and historically treated for the disease. This would make it difficult for an attacker to identify any individual patient.

The research team would receive only the noisy data, but would still be able to draw meaningful and usable statistical inferences from it, because the noise is random and small (depending on the choice of ε). Thus through the use of differential privacy, the hospital was able to share important data with the research team while minimising the risk of leaving its patients vulnerable to reidentification.

Implementing Differential Privacy

While implementing Differential Privacy in practice, it is important to keep some of its characteristics in mind. Some of these important considerations are:

- Choosing the right privacy budget: The privacy budget determines the level of privacy protection that is provided. If the budget is too small, the level of privacy protection may not be sufficient, while if the budget is too large, the amount of noise added to the data may be too high, resulting in inaccurate results.

- Determining the appropriate noise level: The noise level determines the amount of random noise that is added to the data to protect privacy. If the noise level is too low, the data may still be vulnerable to privacy attacks, while if it is too high, the accuracy of the data may be compromised.

- Understanding the data: It is important to understand the data being analysed and the potential privacy risks associated with it. Differential privacy may not be appropriate for all types of data, and certain types of data may require additional privacy protections.

- Balancing privacy and accuracy: Differential privacy involves a trade-off between privacy and accuracy. It is important to find the right balance between the two, depending on the specific use case and the sensitivity of the data being analysed.

DP-Pipeline Use Case

This Differential Privacy pipeline currently offers the option to choose a privacy loss budget ε and compute the amount of noise to be added by using the formula:

$$\varepsilon = \frac{S}{b}$$

where,
ε is the privacy loss budget,
S is the Sensitivity,
b is the noise

The tool is currently limited to a few temporal and geospatial datasets and work is being done to expand the functionality to all types of datasets.

Sensitivity

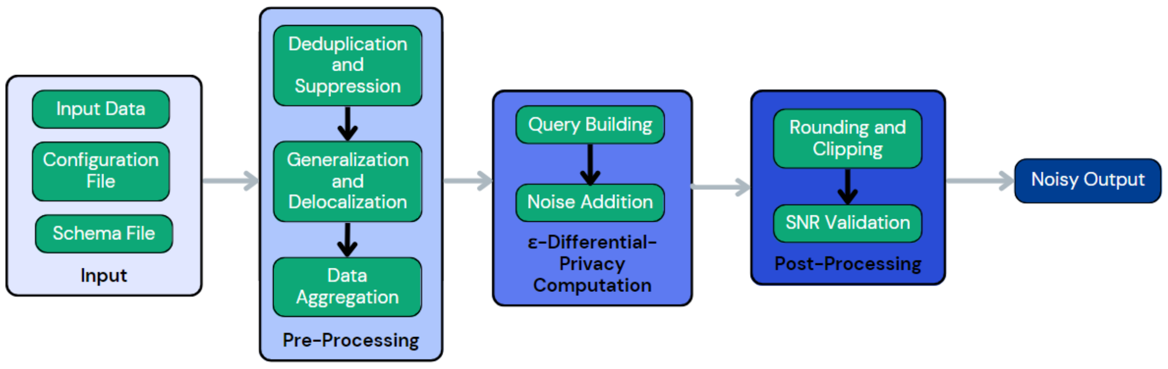
An important choice to be made when applying DP is choosing between local and global sensitivity.

Local sensitivity is best used in situations where the data is highly sensitive or there are strict privacy requirements. In these cases, each party applies noise to their data before sharing it with the central aggregator, which preserves differential privacy. Local sensitivity provides stronger privacy guarantees but can be more complex and computationally expensive to implement.

Global sensitivity, on the other hand, is typically used when the data is less sensitive, or there are fewer privacy requirements. It works by adding noise to the aggregated data to ensure differential privacy is preserved. This approach is often used when the data can be aggregated centrally, making it a more cost-effective and easier-to-implement solution.

Pipeline

This pipeline has the following structure:



The individual modules have the following functionality:

1. Pre-Processing
    a. Deduplication
    This function removes any duplicates found in the dataset based on the choice of columns from the config file. If there exists two or more data packets with identical data in both columns then one of those packets is treated as a duplicate.

    b. Suppression
    This function removes unnecessary columns from the data as determined and selected by the user in the suppressCols parameter in the configuration file.

    c. Generalisation and Delocalisation

This function generalises specific information in the dataset that could be used to reveal identifying information of the users whose data is present in the dataset. This could apply to location, time, etc.

d. Data Aggregation
   This function aggregates the data by statistical releases such as count, mean, sum, min, max.

2. ε-Differential Privacy Computation
   a. Query Building
      This set of functions serve to implement the type of query that the user wants to observe in the data. Currently a limited number of queries and datasets are supported.

   b. Noise Addition
      This function computes the global sensitivity of the dataset and the amount of noise to be added based on the privacy loss budget assigned to each query. This noise is added to the aggregated query output.

3. Post-Processing
   a. Rounding and Clipping
      This function rounds the query outputs to ensure integer values are returned and clips to the lower and upper bounds defined by the global sensitivity. These are not required steps in the posprocessing pipeline.

   b. Signal to Noise Ratio Validation
      This function computes the ratio of signal to noise using the formula
      $SNR = \frac{\mu}{\sigma}$
      where,
      μ is the mean of the true value or the signal,
      σ is the standard deviation of the noise parameter b.
      If the SNR is too high, it is recommended that the parameter privacy budget be adjusted to avoid potential data leakage from addition of a very small amount of noise.

The output of the pipeline is an ε-differential private output for the specific query requested and the chosen value of ε.

User Guide

Setting up the Workspace

Clone the differential-privacy repository using the following command:

```
git clone https://github.com/datakaveri/differential-privacy.git
```

This sets up a directory with the structure seen below:

```
differential-privacy
├── config/
│   │   ├── DPConfig.json
│   │   ├── DPSchema.json
├── data/
├── pipelineOutput/
├── scripts/
│   │   ├── .gitkeep
│   │   ├── diffPrivPipeline.py
│   │   └── modules.py
├── requirements.txt
└── README.md
```

Workspace Components

config/

This folder stores the *config* and *schema* files. In the config file, you can set up the pipeline parameters, such as ε, locality factor, etc. These are further described in the *setting parameters* section of this document.

data/

This folder holds the data that is to be processed with the differential privacy pipeline.

scripts/

This folder contains the modules of the pipeline as well as the execution script. Running the execution script runs the data through the pipeline.

pipelineOutput/
The noisy query output is stored in this folder.

requirements.txt

This file contains the package dependencies for this tool. Installation instructions are provided below.

Required libraries and packages

Ensure that *pip* is installed on your machine. Navigate to the scripts folder, and run the following command to install the package and library dependencies:

```
pip install -r requirements.txt
```

Setting Parameters

The config file *DPConfig.json* requires the user to set the following parameters:

- duplicateDetection
  - Input type: array of 2 strings (column names)
  - This parameter takes two inputs, which are column names in the dataset. If there exists two or more data packets with identical data in both columns then one of those packets is treated as a duplicate and the entire row is removed for that data packet.
- suppressCols
  - Input type: array of strings (column names)
  - This parameter takes an array of strings as input, with the strings being column names in the dataset. These columns are treated as not required for the choice of query and are dropped from the output data.
- groupbyCol
  - Input type: string
  - This parameter takes a column name as string, and is used to aggregate the chosen column for query creation
- trueValue
  - Input type: string
  - This parameter takes a column name as string, and this column is treated as the column used for query creation and for application of a noise distribution to achieve ε-Differential Privacy.
- localityFactor
  - Input type: number
  - This parameter is added to 1 and is used to define a “locality factor” that determines how close to true the values of certain parameters are.
  - Applies to: globalMaxValue, globalMinValue, K
- globalMaxValue
  - Input type: integer
  - This parameter sets the maximum possible value for the trueValue parameter. This is used to compute the global sensitivity for certain queries and for post-processing (clipping).
- globalMinValue
  - Input type: integer
  - This parameter sets the minimum possible value for the trueValue parameter. This is used to compute the global sensitivity for certain queries and for post-processing (clipping).
- privacyLossBudgetEpsQuery
  - Input type: array of integers
  - This parameter determines the privacy loss budget for the queries chosen. The length of the array is equal to the number of queries.
- h3Resolution
  - Input type: array of integers
  - This parameter determines the resolution of the hexagons used to visualize spatial data using Kepler.gl.
- startTime
  - Input type: integer

    - This parameter is used to select a starting time slot to create a window of data to be processed further.
- endTime
    - Input type: integer
    - This parameter is used to select an ending time slot to create a window of data to be processed further.
- minEventOccurences
    - Input type: integer
    - This parameter is used to set a threshold to filter out aggregate query results that fall below that threshold.
- trueValueThreshold
    - Input type: integer
    - This parameter is used to filter out rows where the trueValue column values are below a certain threshold.
- mapeThreshold
    - Input type: integer
    - This parameter is used to determine the threshold for the mean absolute percentage error between the true value and the noisy output value.
- snrThreshold
    - Input type: integer
    - This parameter is used to determine the threshold for the signal (mean of the true value) to noise ratio (standard deviation of the noisy value)

Running the Tool

To run the tool, ensure that the parameters are all defined in the config file in the format defined by the schema. Once that is done, run the following command:

```
python3 diffPrivPipeline.py
```