Documentación del Transformador de Privacidad Diferencial

Luisa María De La Hortúa[*], David Moreno[†], Allan Ramírez[‡], and David Romero[§]

Master's student MIIA, Industrial Engineering Department, Universidad de Los Andes, Bogotá, Colombia

Contents

[*]Email: lm.delahortua@uniandes.edu.co
[†]Email: ds.morenom1@uniandes.edu.co
[‡]Email: as.ramirez2@uniandes.edu.co
[§]Email: ds.romerog1@uniandes.edu.co

1 Introducción

La privacidad diferencial es un marco matemático que permite el análisis de datos sensibles garantizando la privacidad individual. Este documento presenta un transformador de privacidad diferencial que facilita la aplicación de diferentes mecanismos de privacidad a conjuntos de datos, tanto para variables cuantitativas como categóricas. El objetivo es proporcionar una herramienta flexible y fácil de usar para investigadores y profesionales que deseen proteger la privacidad en sus análisis de datos.

2 Descripción General del Paquete

El paquete proporciona una clase principal, `DifferentialPrivacyTransformer`, que actúa como interfaz para aplicar mecanismos de privacidad diferencial a un conjunto de datos. Además, incluye funciones para graficar y analizar el impacto de la privatización en los datos, así como implementaciones de varios mecanismos de privacidad diferencial para datos cuantitativos y categóricos.

3 Estructura del Paquete

El paquete está organizado de la siguiente manera:

- **dp_transformer.py**: Contiene la clase `DifferentialPrivacyTransformer`.
- **dp_plotting.py**: Incluye funciones para la visualización y análisis de datos privatizados.
- **dp_mechanisms**: Directorio que alberga los mecanismos de privacidad diferencial.
  - **quantitative.py**: Implementaciones de mecanismos para datos cuantitativos.
  - **categorical.py**: Implementaciones de mecanismos para datos categóricos.

4 Clase `DifferentialPrivacyTransformer`

La clase `DifferentialPrivacyTransformer` es la interfaz principal para aplicar mecanismos de privacidad diferencial a un conjunto de datos de pandas. Permite transformar variables cuantitativas y categóricas utilizando diferentes mecanismos y ajustar el nivel de privacidad mediante el parámetro ϵ.

4.1 Inicialización

```
DifferentialPrivacyTransformer(
    df,
    epsilon_quantitative=1.0,
    epsilon_categorical=1.0,
    quantitative_vars=None,
    categorical_vars=None
)
```

Parámetros:

- `df` (*pd.DataFrame*): DataFrame que contiene los datos a transformar.
- `epsilon_quantitative` (*float*): Presupuesto de privacidad para variables cuantitativas (valor por defecto: `1.0`).
- `epsilon_categorical` (*float*): Presupuesto de privacidad para variables categóricas (valor por defecto: `1.0`).
- `quantitative_vars` (*list* de *str*): Lista de nombres de variables cuantitativas. Si es `None`, se detectan automáticamente.

- `categorical_vars` (*list* de *str*): Lista de nombres de variables categóricas. Si es `None`, se detectan automáticamente.

4.2 Métodos Principales

4.2.1 `fit_quantitative`

Aplica mecanismos de privacidad diferencial a las variables cuantitativas seleccionadas.

```
fit_quantitative(
    quantitative_vars=None,
    method='duchi',
    epsilon=None
)
```

Parámetros:

- `quantitative_vars` (*list* de *str*): Variables cuantitativas a transformar. Si es `None`, se utilizan las variables definidas en la instancia.
- `method` (*str*): Método de privatización a utilizar. Opciones disponibles:
  - `'duchi'`
  - `'laplace'`
  - `'piecewise'`
  - `'multidimensional_duchi'`
  - `'multidimensional'`
- `epsilon` (*float*): Presupuesto de privacidad. Si es `None`, se utiliza el valor definido en la instancia.

4.2.2 `fit_categorical`

Aplica mecanismos de privacidad diferencial a las variables categóricas seleccionadas.

```
fit_categorical(
    categorical_vars=None,
    method='direct_encoding',
    epsilon=None
)
```

Parámetros:

- `categorical_vars` (*list* de *str*): Variables categóricas a transformar. Si es `None`, se utilizan las variables definidas en la instancia.
- `method` (*str*): Método de privatización a utilizar. Opciones disponibles:
  - `'direct_encoding'`
  - `'oue'` (Optimized Unary Encoding)
  - `'rappor'`
- `epsilon` (*float*): Presupuesto de privacidad. Si es `None`, se utiliza el valor definido en la instancia.

4.2.3 `calculate_utility_metrics`

Calcula métricas de utilidad para evaluar el impacto de la privatización en las variables cuantitativas.

```
calculate_utility_metrics(
    variables=None
)
```

Parámetros:

- `variables` (*list* de *str*): Variables cuantitativas para calcular las métricas. Si es `None`, se utilizan las variables definidas en la instancia.

Devuelve:

- `pd.DataFrame`: DataFrame que contiene las métricas de utilidad calculadas.

Métricas Calculadas:

- **Error Cuadrático Medio (MSE)**: Mide la diferencia promedio al cuadrado entre los datos originales y privatizados. Se calcula como:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \tilde{x}_i)^2$$

  donde x_i es el valor original y $\tilde{x}_i$ es el valor privatizado.

- **Error Absoluto Medio (MAE)**: Mide la diferencia promedio absoluta entre los datos originales y privatizados. Se calcula como:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - \tilde{x}_i|$$

- **Correlación**: Calcula el coeficiente de correlación de Pearson entre los datos originales y privatizados, indicando la relación lineal entre ambos conjuntos de datos. Se calcula como:

$$\rho_{x,\tilde{x}} = \frac{\text{Cov}(x, \tilde{x})}{\sigma_x \sigma_{\tilde{x}}}$$

  donde $\text{Cov}(x, \tilde{x})$ es la covarianza entre x y $\tilde{x}$, y σ_x, $\sigma_{\tilde{x}}$ son las desviaciones estándar.

Interpretación:

- Un **MSE** o **MAE** menor indica que los datos privatizados están más cerca de los datos originales en términos de valores numéricos, lo que implica una mejor utilidad.
- Una **Correlación** más alta (cercana a 1 o -1) sugiere una relación lineal más fuerte entre los datos originales y privatizados, indicando que la privatización preserva las tendencias y patrones del conjunto de datos.
- Si la desviación estándar de los datos originales o privatizados es cero, la correlación es indefinida y se representa como `NaN` (Not a Number).

4.2.4 Métodos de Graficación

`plot_mean_vs_privatized` Genera gráficos que comparan la media de las variables originales con las medias de las variables privatizadas para diferentes valores de ϵ.

```
plot_mean_vs_privatized(
    variables=None,
    method='duchi',
    epsilons=None,
    **kwargs
)
```

plot_histograms Genera histogramas comparativos de las variables originales y privatizadas.

```
plot_histograms(
    variables=None,
    method='duchi',
    epsilon=1.0,
    bins=30,
    **kwargs
)
```

5 Funciones de Graficación

5.1 plot_mean_vs_privatized

Genera gráficos que muestran cómo varía la media de las variables privatizadas con respecto a diferentes valores de ϵ.

5.2 plot_histograms

Genera histogramas que comparan la distribución de las variables originales con las privatizadas.

6 Mecanismos de Privacidad Diferencial

6.1 Para Datos Cuantitativos

Los mecanismos para datos cuantitativos se implementan en `dp_mechanisms/quantitative.py`.

- **Mecanismo de Duchi et al.** (`'duchi'`): Adecuado para datos en el rango $[-1, 1]$, utiliza respuesta aleatorizada para garantizar la privacidad.
- **Mecanismo de Laplace** (`'laplace'`): Agrega ruido Laplaciano proporcional a la sensibilidad del dato.
- **Mecanismo por Tramos** (`'piecewise'`): Divide el dominio en tramos y aplica ruido de acuerdo con la posición del dato.
- **Mecanismo Multidimensional de Duchi** (`'multidimensional_duchi'`): Extensión del mecanismo de Duchi para datos multidimensionales.
- **Mecanismo Multidimensional Personalizado** (`'multidimensional'`): Aplica mecanismos unidimensionales de forma selectiva en dimensiones aleatorias.

6.2 Para Datos Categóricos

Los mecanismos para datos categóricos se implementan en `dp_mechanisms/categorical.py`.

- **Codificación Directa** (`'direct_encoding'`): Asigna probabilidades a cada categoría y aplica respuesta aleatorizada.
- **Codificación Unaria Optimizada (OUE)** (`'oue'`): Representa categorías mediante codificación unaria y aplica perturbación optimizada.
- **RAPPOR** (`'rappor'`): Utiliza codificación unaria y múltiples rondas de perturbación para mejorar la privacidad.

7 Ejemplos de Uso

7.1 Privatización y Evaluación de Variables Cuantitativas

```
import pandas as pd
import seaborn as sns
from dp_transformer import DifferentialPrivacyTransformer

# Cargar el conjunto de datos Titanic
df = sns.load_dataset('titanic')

# Eliminar filas con valores nulos
df = df.dropna()

# Definir variables cuantitativas
quantitative_vars = ['age', 'fare']

# Crear instancia del transformador
dp_transformer = DifferentialPrivacyTransformer(
    df,
    epsilon_quantitative=1.0,
    quantitative_vars=quantitative_vars
)

# Aplicar privatización usando el mecanismo de Duchi
df_priv = dp_transformer.fit_quantitative(method='duchi')

# Calcular métricas de utilidad
metrics = dp_transformer.calculate_utility_metrics()
print(metrics)

# Generar gráfico de la media original vs privatizada
dp_transformer.plot_mean_vs_privatized(
    variables=quantitative_vars,
    method='duchi',
    epsilons=[0.1, 0.5, 1.0]
)
```

7.2 Privatización de Variables Categóricas

```
import pandas as pd
import seaborn as sns
from dp_transformer import DifferentialPrivacyTransformer

# Cargar el conjunto de datos Titanic
df = sns.load_dataset('titanic')

# Eliminar filas con valores nulos
df = df.dropna()

# Definir variables categóricas
categorical_vars = ['sex', 'class', 'embarked']
```

```
# Crear instancia del transformador
dp_transformer = DifferentialPrivacyTransformer(
    df,
    epsilon_categorical=1.0,
    categorical_vars=categorical_vars
)

# Aplicar privatización usando codificación directa
df_priv = dp_transformer.fit_categorical(method='direct_encoding')

# Mostrar las primeras filas del DataFrame transformado
print(df_priv[categorical_vars].head())
```