pseudPy Documentation

version 1.0

2024, Oleksandra Popovych

June 10, 2024

Contents

Welcome to pseudPy’s documentation!

Modules

class `Pseudonymization.`**`Aggregation`** (`column`, `method`, `df`=None, `input_file`=None, `output`=None)

Bases: **`object`**

Class for data aggregation.

Parameters:

- **column** (*str*) – Column to aggregate.
- **method** (*list*) – Aggregation method and the range - [‘number’, int] for numerical data or [‘dates-to-years’, int] for dates.
- **df** (*Pandas Dataframe*) – Input Dataframe. Required if input path is not provided.
- **input_file** (*str*) – Path to input file. Required if dataframe is not provided.
- **output** (*str*) – Path to output folder.

`group` **()**

Main function to initiate aggregation of csv.

Return type: Output to file, if the output path is passed, or to a Pandas Dataframe.

Example

Aggregate salary data to the groups of range 10000.

```
>>> import pseudPy.Pseudonymization as pseudPy
>>> agg = pseudPy.Aggregation(
>>>     column='salary',
>>>     method=['number', 10000],
>>>     input_file='some_user_data.csv',
>>>     output='/output/dir')
>>>
>>> agg.group()
```

class `Pseudonymization.`**`KAnonymity`** (`df`, `k`, `depths`=None, `mask_others`=True, `output`=None)

Bases: **`object`**

Class for data k-anonymization.

Parameters:

- **df** (*Pandas Dataframe*) – Input data.
- **k** (*int*) – The level of anonymity. Each row in the data must be matched to at least k other rows.
- **depths** (*dict*) – The depths of rounding the values in form {column_name: depth}. Ex. value=55222, depths = {value: 1}, result=55220.
- **mask_others** (*bool*) – Enable or disable tokenizing all string values with token ‘ * ‘. To achieve k-anonymity, string data must be masked in most cases, unless it has also been k-anonymized externally beforehand.
- **output** (*str*) – Path to output folder.

`is_k_anonymized` **()**

Check if the data is k-anonymous.

Return type: True or False.

Example

Check if the data is k-anonymous.

```
>>> import pseudPy.Pseudonymization as pseudPy
>>> grouped = pl.read_csv('/path/to/input.csv')
>>> is_k_anonym = pseudPy.KAnonymity(
>>>             df=grouped,
>>>             depths={'salary': 1},
>>>             k=2)
>>>
>>> print(is_k_anonym.is_k_anonymized())
```

k_anonymity ()

k-anonymize the data by providing the dataframe, k, depths of anonymization.

Return type: k-anonymized Dataframe or output to file, if the output path is passed.

Example

k-anonymize salary data with k of 2.

```
>>> import pseudPy.Pseudonymization as pseudPy
>>> df = pl.read_csv('/path/to/input.csv')
>>> k_anonymity = pseudPy.KAnonymity(
>>>             df=df,
>>>             depths={'salary': 1},
>>>             k=2,
>>>             output='/output/dir')
>>>
>>> grouped = k_anonymity.k_anonymity())
```

class Pseudonymization.**Pseudonymization** (map_method='counter', map_columns=None, input_file=None, output=None, df=None, mapping=True, encrypt_map=False, text=None, all_ne=False, seed=None, pos_type=None, patterns=None)

Bases: **object**

Main class for data pseudonymization.

Parameters:

- **map_method** (*str*) – Pseudonymization method. Select one of the following: *'counter', 'random1', 'random4', 'hash', 'hash-salt', 'merkle-tree', 'encrypt', 'decrypt', 'faker'*. Or specify the faker method: *'faker-name', 'faker-loc','faker-email', 'faker-phone', 'faker-org'*.
- **map_columns** (*str or list*) – Column(s) to be pseudonymized for structured data.
- **input_file** (*str*) – Path to input file. Use if the parameter df is not specified.
- **output** (*str*) – Path to output folder. Use if the output to file is required.
- **df** (*Polars DataFrame*) – Input CSV, read as Polars DataFrame. Use if data is structured and the input_file is not specified.
- **mapping** (*bool*) – Enable or disable mapping output. Required for reversibility of pseudonyms.
- **encrypt_map** (*bool*) – Enable or disable encryption of the mapping table. Output includes additionally secret key file for decryption.
- **text** (*str*) – Input text. Use if data is unstructured and the input_file is not specified.
- **all_ne** (*bool*) – Enable or disable pseudonymization of all named entities such as names, locations, and organizations. Use if data is unstructured and no other entities have to be pseudonymized.
- **seed** (*int*) – Seed random pseudonymization methods like random1, random4, or faker. Return always expected result.
- **pos_type** (*str or list*) – Type(s) of entities in data to be pseudonymized such as names, locations, organizations, emails, and phone numbers. Use if data is unstructured.
- **patterns** (*str or spaCy Matcher*) – If data is structured, use *"column,operation,value"*, else *patterns = [[{"LOWER": "abc"}, {"LOWER": "corporation"}]…]*.

nlp_pseudonym ()

Main function for pseudonymization of free text.

Returns: *A pseudonymized String or, if output parameter is passed, writes the pseudonymized String and mappings to files. If the encryption is involved, the secret keys are written to the .txt files by default.*

Example

Pseudonymization of unstructured data using 'merkle-tree' method and filtering.

```
>>> import pseudPy.Pseudonymization as pseudPy
>>> pseudo = pseudPy.Pseudonymization(
>>>        map_method = 'merkle-tree',
>>>        input_file = '/path/to/input.txt',
>>>        output='/output/dir',
>>>        patterns = [[{"LOWER": "Emily"}, {"LOWER": "White"}],
>>>        [{"LOWER": "Bob"}]])
>>>
>>> pseudo.pseudonym()
```

pseudonym ()

Main function for pseudonymization of csv data.

Returns: *Pseudonymized Dataframe or a String. If output parameter is passed, writes pseudonymized and mapping files. If the encryption is involved, the secret keys are written to the .txt files by default.*

Example

Pseudonymization of structured data using 'faker-name' method and filtering.

```
>>> import pseudPy.Pseudonymization as pseudPy
>>> pseudo = pseudPy.Pseudonymization(
>>>        map_method = 'faker-name',
>>>        map_columns = 'names',
>>>        input_file= '/path/to/data.csv',
>>>        output='/output/dir',
>>>        patterns=['salary', '>', 100000])
>>>
>>> pseudo.pseudonym()
```

`revert_nlp_pseudonym` **(**`revert_df`**,** pseudonyms=None**)**

Revert free text to original.

Parameters:

- **revert_df** (*Polars DataFrame*) – The mapping table.
- **pseudonyms** (*list*) – Filter for exact pseudonyms to revert. Optional.

Return type: A String or, if output parameter is passed, output as a file.

Example

Revert pseudonymized values to the original. For unstructured data.

```
>>> import pseudPy.Pseudonymization as pseudPy
>>> with open('/path/to/input.txt', "r") as file:
>>>      text = file.read()
>>> output = '/output/dir'
>>> pseudo = pseudPy.Pseudonymization(
>>>        map_columns = 'Names',   # select from: 'Names', 'Locations', 'Organizations',
>>>        text=text,
>>>        output=output)
>>> df_revert = pl.read_csv(f'{output}/mapping_output_Names.csv')
>>>
>>> pseudo.revert_nlp_pseudonym(df_revert)
```

`revert_pseudonym` **(**`revert_df`=None**,** pseudonyms=None**)**

Revert structured data to original in form of Dataframe.

Parameters:

- **revert_df** (*Polars DataFrame*) – The mapping table in form of Polars Dataframe.
- **pseudonyms** (*list*) – Filter for exact pseudonyms to revert. Optional.

Return type: A reverted Dataframe and if output parameter is passed, a file with reverted pseudonyms.

Example

Revert pseudonymized values to the original. For structured data.

```
>>> import pseudPy.Pseudonymization as pseudPy
>>> df = pl.read_csv('/path/to/data.csv')
>>> output='/output/dir'
>>> pseudo = pseudPy.Pseudonymization(
>>>        map_columns = 'column1',
>>>        df=df,
>>>        output=output)
>>> df_revert = pl.read_csv(f'{output}/mapping_output_column1.csv')
>>>
>>> pseudo.revert_pseudonym(df_revert)
```

Indices and tables

- **genindex**
- **modindex**
- **search**

Index

Python Module Index