pycanon

Release 1.0.1.post1

Spanish National Research Council (CSIC)

May 18, 2023

CONTENTS

pyCANON is a Python library which allows the user to know the anonymity level of a dataset based on a set of quasi-identifiers (QI), and a set of sensitive attributes. To do so, it provides a set of functions to compute the anonymity level of a dataset by means of the following techniques:

- k-anonymity.
- (α,k)-anonymity.
- ℓ-diversity.
- Entropy ℓ-diversity.
- Recursive (c, ℓ)-diversity.
- t-closeness.
- Basic β-likeness.
- Enhanced β-likeness.
- δ-disclosure privacy.

CHAPTER

ONE

USER DOCUMENTATION

1.1 Getting started

Using pyCANON is quite straighforward.

1.1.1 Install

We recommend using `pip` for installing pyCANON inside a `virtualenv`:

```
virtualenv .venv
source .venv/bin/activate
pip install pycanon
```

Installing with support for PDF reports: If you want to generate PDF reports with ReportLab you need to issut the following install command:

```
pip install pycanon[PDF]
```

If you want to use the latest development version, you can use:

```
virtualenv .venv
source .venv/bin/activate
git clone https://gitlab.ifca.es/privacy-security/pycanon/
pip install pycanon
```

1.1.2 Example usage

You can use pyCANON through the command line or via its Python API.

1.1.2.1 Command line

Example with the adult dataset:

```
$ pycanon k-anonymity --qi age --qi education --qi occupation --qi relationship --qi sex␣
↪--qi native-country adult.csv

$ pycanon report --sa salary-class --qi age --qi education --qi occupation --qi␣
↪relationship --qi sex --qi native-country adult.csv
```

1.1.2.2 Python API

Example with the adult dataset:

```
from pycanon import anonymity, report

FILE_NAME = "adult.csv"
QI = ["age", "education", "occupation", "relationship", "sex", "native-country"]
SA = ["salary-class"]
DATA = pd.read_csv(FILE_NAME)

# Calculate k for k-anonymity:
k = anonymity.k_anonymity(DATA, QI)

# Print the anonymity report:
report.print_report(DATA, QI, SA)
```

1.2 pycanon

1.2.1 pycanon package

1.2.1.1 Subpackages

pycanon.anonymity package

Subpackages

pycanon.anonymity.utils package

Submodules

pycanon.anonymity.utils.aux_anonymity module

Module with different functions which calculate properties about anonymity.

k-anonymity, (alpha,k)-anonymity, l-diversity, entropy l-diversity, (c,l)-diversity, basic beta-likeness, enhanced beta-likeness, t-closeness and delta-disclosure privacy.

pycanon.anonymity.utils.aux_anonymity.**aux_calculate_beta**(*data: DataFrame*, *quasi_ident: Union[list, ndarray]*, *sens_att_value: str*) → Tuple[ndarray, list]

Beta calculation for basic and enhanced beta-likeness.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att_value** (*string*) – sensitive attribute under study.

Returns
proportion of each value of the sensitive attribute in the entire database and distance from the proportion in each equivalence class.

Return type
np.array and list.

pycanon.anonymity.utils.aux_anonymity.**aux_calculate_delta_disclosure**(*data: DataFrame*, *quasi_ident: Union[list, ndarray]*, *sens_att_value: str*) → float

Delta calculation for delta-disclosure privacy.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att_value** (*string*) – sensitive attribute under study.

Returns
delta for the introduced SA.

Return type
float.

pycanon.anonymity.utils.aux_anonymity.**aux_t_closeness_num**(*data: DataFrame*, *quasi_ident: Union[list, ndarray]*, *sens_att_value: str*) → float

Obtain t for t-closeness.

Function used for numerical attributes: the definition of the EMD is used.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att_value** (*string*) – sensitive attribute under study.

Returns
t for the introduced SA (numerical).

Return type
float.

pycanon.anonymity.utils.aux_anonymity.**aux_t_closeness_str**(*data: DataFrame*, *quasi_ident: Union[list, ndarray]*, *sens_att_value: list*) → float

Obtain t for for t-closeness.

Function used for categorical attributes: the metric "Equal Distance" is used.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.

- **sens_att_value** (*string*) – sensitive attribute under study.

Returns
t for the introduced SA (categorical).

Return type
float.

pycanon.anonymity.utils.aux_anonymity.**get_equiv_class**(*data: DataFrame*, *quasi_ident: Union[list, ndarray]*) → list

Find the equivalence classes present in the dataset.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*is a list of strings*) – list with the name of the columns of the dataframe that are the quasi-identifiers.

Returns
equivalence classes.

Return type
list.

pycanon.anonymity.utils.aux_functions module

Module with different auxiliary functions.

pycanon.anonymity.utils.aux_functions.**check_qi**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*) → None

Check if the entered quasi-identifiers are valid.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.

pycanon.anonymity.utils.aux_functions.**check_sa**(*data: DataFrame*, *sens_att: Union[List, ndarray]*) → None

Check if the entered sensitive attributes are valid.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **sens_att** (*is a list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.

pycanon.anonymity.utils.aux_functions.**convert**(*ec_set: set*) → list

Convert a set with an equivalence class to a list.

Parameters
ec_set (*set*) – set which will be convert into a list.

Returns
equivalence class into a list.

Return type
list.

pycanon.anonymity.utils.aux_functions.**intersect**(*tmp: list*) → list

Intersect two sets: the first and the second of the given list.

Parameters
tmp (*list of numpy arrays*) – list of sets sorted in decreasing order of cardinality

Returns
list obtained when intersecting the first and the second sets of the given list.

Return type
list.

pycanon.anonymity.utils.aux_functions.**read_file**(*file_name: Union[str, Path]*, *sep: str = ','*) → DataFrame

Read the given file. Returns a pandas dataframe.

Parameters

- **file_name** (*string or pathlib.Path*) – file with the data under study.
- **sep** (*string*) – delimiter to use for a csv file.

Returns
dataframe with the data.

Return type
pandas dataframe.

Module contents

Package containing auxiliary functions related with privacy models.

Module contents

Module with different functions which calculate properties about anonymity.

k-anonymity, (alpha,k)-anonymity, l-diversity, entropy l-diversity, (c,l)-diversity, basic beta-likeness, enhanced beta-likeness, t-closeness and delta-disclosure privacy.

pycanon.anonymity.**alpha_k_anonymity**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*, *sens_att: Union[List, ndarray]*, *gen=True*) → Tuple[float, int]

Calculate alpha and k for (alpha,k)-anonymity.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – boolean, default to True. If true, it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA

Returns
alpha and k values for (alpha,k)-anonymity.

Return type
alpha is a float, k is an int.

pycanon.anonymity.**basic_beta_likeness**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*, *sens_att: Union[List, ndarray]*, *gen=True*) → float

Calculate beta for basic beta-likeness.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – boolean, default to True. If true, it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA

Returns
beta value for basic beta-likeness.

Return type
float.

pycanon.anonymity.**delta_disclosure**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*, *sens_att: Union[List, ndarray]*, *gen=True*) → float

Calculate delta for delta-disclousure privacy.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – boolean, default to True. If true, it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA

Returns
delta value for delta-discloure privacy.

Return type
float.

pycanon.anonymity.**enhanced_beta_likeness**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*, *sens_att: Union[List, ndarray]*, *gen=True*) → float

Calculate beta for enhanced beta-likeness.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.

- **gen** (*boolean*) – boolean, default to True. If true, it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA

Returns
beta value for enhanced beta-likeness.

Return type
float.

pycanon.anonymity.**entropy_l_diversity**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*, *sens_att: Union[List, ndarray]*, *gen=True*) → float

Calculate l for entropy l-diversity.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – boolean, default to True. If true, it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA

Returns
l value for entropy l-diversity.

Return type
float.

pycanon.anonymity.**k_anonymity**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*) → int

Calculate k for k-anonymity.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.

Returns
k value for k-anonymity.

Return type
int.

pycanon.anonymity.**l_diversity**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*, *sens_att: Union[List, ndarray]*, *gen=True*) → int

Calculate l for l-diversity.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – boolean, default to True. If true, it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA

Returns
l value for l-diversity.

Return type
int.

pycanon.anonymity.**recursive_c_l_diversity**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*, *sens_att: Union[List, ndarray]*, *imp=False*, *gen=True*) → Tuple[float, int]

Calculate c and l for recursive (c,l)-diversity.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – boolean, default to True. If true, it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA

Returns
c and l values for recursive (c,l)-diversity.

Return type
c is a float, l is an int.

pycanon.anonymity.**t_closeness**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*, *sens_att: Union[List, ndarray]*, *gen=True*) → float

Calculate t for t-closeness.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – boolean, default to True. If true, it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA

Returns
t value for basic t-closeness.

Return type
float.

pycanon.report package

Submodules

pycanon.report.base module

Get report values for all privacy models.

pycanon.report.base.**get_report_values**(*data: DataFrame*, *quasi_ident: list*, *sens_att: list*, *gen=True*) → Tuple[int, Tuple[float, int], int, float, Tuple[Any, int], float, float, float, float]

Generate a report with the parameters obtained for each anonymity check.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*is a list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – default to true. If true it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA.

pycanon.report.json module

Get report values as JSON for all privacy models.

pycanon.report.json.**get_json_report**(*data: DataFrame*, *quasi_ident: list*, *sens_att: list*, *gen=True*) → str

Generate a report (JSON) with the parameters obtained for each anonymity check.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*is a list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – default to true. If true it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA.

pycanon.report.pdf module

pycanon.report.pdf_utility_report module

Module contents

Generate reports with all privacy model's scores.

pycanon.report.**get_json_report**(*data: DataFrame*, *quasi_ident: list*, *sens_att: list*, *gen=True*) → str

Generate a report (JSON) with the parameters obtained for each anonymity check.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*is a list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – default to true. If true it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA.

pycanon.report.**get_report_values**(*data: DataFrame*, *quasi_ident: list*, *sens_att: list*, *gen=True*) → Tuple[int, Tuple[float, int], int, float, Tuple[Any, int], float, float, float, float]

Generate a report with the parameters obtained for each anonymity check.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*is a list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – default to true. If true it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA.

pycanon.report.**print_report**(*data: DataFrame*, *quasi_ident: list*, *sens_att: list*, *gen=True*) → None

Generate a report with the parameters obtained for each anonymity check.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data under study.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*is a list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.
- **gen** (*boolean*) – default to true. If true it is generalized for the case of multiple SA, if False, the set of QI is updated for each SA.

pycanon.utility package

Module contents

Module with different functions for calculating the utility.

pycanon.utility.**average_ecsize**(*data_raw: DataFrame*, *data_anon: DataFrame*, *quasi_ident: Union[List, ndarray]*, *sup=True*) → float

Calculate the metric average equivalence class size.

Parameters

- **data_raw** (*pandas dataframe*) – dataframe with the data raw under study.
- **data_anon** (*pandas dataframe*) – dataframe with the data anonymized.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sup** (*boolean*) – boolean, default to True. If true, suppression has been applied to the original dataset (some records may have been deleted).

pycanon.utility.**classification_metric**(*data_raw: DataFrame*, *data_anon: DataFrame*, *quasi_ident: Union[List, ndarray]*, *sens_att: Union[List, ndarray]*) → float

Calculate the classification metric.

Parameters

- **data_raw** (*pandas dataframe*) – dataframe with the data raw under study.
- **data_anon** (*pandas dataframe*) – dataframe with the data anonymized.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.
- **sens_att** (*list of strings*) – list with the name of the columns of the dataframe that are the sensitive attributes.

pycanon.utility.**discernability_metric**(*data_raw: DataFrame*, *data_anon: DataFrame*, *quasi_ident: Union[List, ndarray]*) → float

Calculate the discernability metric.

Parameters

- **data_raw** (*pandas dataframe*) – dataframe with the data raw under study.
- **data_anon** – dataframe with the data anonymized. Assuming that all the

equivalence classes have more than k records, and given each suppressed record a penalty of the size of the input dataset. :type data_anon: pandas dataframe

Parameters

quasi_ident (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.

pycanon.utility.**sizes_ec**(*data: DataFrame*, *quasi_ident: Union[List, ndarray]*) → dict

Calculate statistics associated to the equivalence classes.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data anonymized.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.

pycanon.utility.**stats_quasi_ident**(*data: DataFrame*, *quasi_ident: str*) → dict

Calculate statistics associated to a given quasi-identifier.

Parameters

- **data** (*pandas dataframe*) – dataframe with the data anonymized.
- **quasi_ident** (*list of strings*) – list with the name of the columns of the dataframe that are quasi-identifiers.

1.2.1.2 Submodules

1.2.1.3 pycanon.cli module

Module providing command line tools for pyCANON.

pycanon.cli.**alpha_k_anonymity**(*filename: ~pathlib.Path = <typer.models.ArgumentInfo object>*, *qi: ~typing.List[str] = <typer.models.OptionInfo object>*, *sa: ~typing.List[str] = <typer.models.OptionInfo object>*, *gen: bool = <typer.models.OptionInfo object>*)

Calculate (alpha,k)-anonymity.

pycanon.cli.**basic_beta_likeness**(*filename: ~pathlib.Path = <typer.models.ArgumentInfo object>*, *qi: ~typing.List[str] = <typer.models.OptionInfo object>*, *sa: ~typing.List[str] = <typer.models.OptionInfo object>*, *gen: bool = <typer.models.OptionInfo object>*)

Calculate basic beta-likeness.

pycanon.cli.**delta_disclosure**(*filename: ~pathlib.Path = <typer.models.ArgumentInfo object>*, *qi: ~typing.List[str] = <typer.models.OptionInfo object>*, *sa: ~typing.List[str] = <typer.models.OptionInfo object>*, *gen: bool = <typer.models.OptionInfo object>*)

Calculate delta-disclosure.

pycanon.cli.**enhanced_beta_likeness**(*filename: ~pathlib.Path = <typer.models.ArgumentInfo object>*, *qi: ~typing.List[str] = <typer.models.OptionInfo object>*, *sa: ~typing.List[str] = <typer.models.OptionInfo object>*, *gen: bool = <typer.models.OptionInfo object>*)

Calculate enhanced beta-likeness.

pycanon.cli.**entropy_l_diversity**(*filename: ~pathlib.Path = <typer.models.ArgumentInfo object>*, *qi: ~typing.List[str] = <typer.models.OptionInfo object>*, *sa: ~typing.List[str] = <typer.models.OptionInfo object>*, *gen: bool = <typer.models.OptionInfo object>*)

Calculate entropy l-diversity.

pycanon.cli.**k_anonymity**(*filename: ~pathlib.Path = <typer.models.ArgumentInfo object>*, *qi: ~typing.List[str] = <typer.models.OptionInfo object>*)

Calculate k-anonymity.

pycanon.cli.**l_diversity**(*filename: ~pathlib.Path = <typer.models.ArgumentInfo object>*, *qi: ~typing.List[str] = <typer.models.OptionInfo object>*, *sa: ~typing.List[str] = <typer.models.OptionInfo object>*, *gen: bool = <typer.models.OptionInfo object>*)

Calculate l-diversity.

pycanon.cli.**main**(*version: ~typing.Optional[bool] = <typer.models.OptionInfo object>*)

Check the level of anonymity of a dataset.

pycanon.cli.**recursive_c_l_diversity**(*filename: ~pathlib.Path = <typer.models.ArgumentInfo object>*, *qi: ~typing.List[str] = <typer.models.OptionInfo object>*, *sa: ~typing.List[str] = <typer.models.OptionInfo object>*, *gen: bool = <typer.models.OptionInfo object>*)

Calculate recursive (c,l)-diversity.

pycanon.cli.**report**(*filename: ~pathlib.Path = <typer.models.ArgumentInfo object>*, *qi: ~typing.List[str] = <typer.models.OptionInfo object>*, *sa: ~typing.List[str] = <typer.models.OptionInfo object>*, *gen: bool = <typer.models.OptionInfo object>*)

Generate a complete privacy report.

pycanon.cli.**t_closeness**(*filename: ~pathlib.Path = <typer.models.ArgumentInfo object>*, *qi: ~typing.List[str] = <typer.models.OptionInfo object>*, *sa: ~typing.List[str] = <typer.models.OptionInfo object>*, *gen: bool = <typer.models.OptionInfo object>*)

Calculate t-closeness.

pycanon.cli.**version_callback**(*version: bool*)

Return version info.

1.2.1.4 Module contents

pyCANON is a library to check the values of the most common data privacy models.

1.3 Notes and Warnings

Note: In the case of multiple sensitve attributes, two approaches can be considered:

- Calculate the different properties for each sensitive attribute individually, and take the maximum or minimum as appropriate (for example, the minumum value of ℓ for ℓ-diversity and the maximum value for α in the case of (α,k)-anonymity).
- Calculate the different properties for each sensitive attribute individually but modifying the set of quasi-identifiers by adding to this set, in addition to the initial quasi-identifiers, all the sensitive attributes except the one under analysis. Then, as in the previous approach, the minimum or maximum of each parameter as appropriate is taken. It is important to note that since the set of quasi-identifiers is updated each time the calculations are made for each SA, the computational cost is much higher in this case.

Specifically, in order to address this challenge, a parameter gen is introduced in all functions except k-anonymity (not applicable). If *gen=True* (default value), the process of the first approach is followed: generalizing. Otherwise, the second approach is followed, updating the quasi-identifiers.

Warning: It's important to take into account that the values of t and δ for t-closeness and δ-disclosure privacy respectively must be strictly greater than the ones obtained using *pyCANON* (see the definition of that techniques).

1.4 Utility metrics

The details of the three utility metrics implemented are presented below:

- *Average equivalence class size* (C_{avg}): Two versions are implemented depending on whether or not complete rows have been deleted from the original dataset (in this case, *sup=True* in `get_utility_report_values`). Be *DB* the original database and *DB'* the anonymized one. Be *|EC|* the number of equivalence classes in the anonymized dataset and *k* the verified value for *k-anonymity*:

$$C_{avg} = \begin{cases} \frac{|DB'|}{k\ |EC|} & \text{if } sup = True \\ \frac{|DB|}{k\ |EC|} & \text{if } sup = False \end{cases}$$

- *Classification metric (CM)*: This metric is calculted for only one sensitive attribute. Be *N* the number of rows of the raw datatset. Be $penalty(r_i) = 1$ if the row r_i has been suppresed or if its associated sensitive attribute takes a value other than the majority in the equivalence class to which it belongs, and 0 otherwise. The classification metric is defined as folows:

$$CM = \frac{1}{N} \sum_{i=1}^{N} penalty(r_i),$$

- *Discernability metric* (DM^*): A version of the classical discernability metric has been considered in order to penalize the deletion of records. Thus, it has been implemented according to the following equation, with *DB* the raw database, *DB'* the anonymized one, N_{ec} the number of equivalence classes and EC_i the *i-th* equivalence class:

$$DM^* = \sum_{i=0}^{N_{ec}} |EC_i|^2 + |DB|^2 - |DB||DB'|$$

Note: for the calculation of these three metrics, it is necessary to enter the original dataset prior to anonymization.

CHAPTER

TWO

LICENSE

pyCANON is licensed under Apache License Version 2.0 (http://www.apache.org/licenses/)

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

R

S

T

V