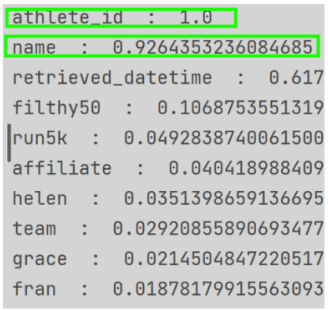
Hugo Chamillard

Assignment #1 : Pseudonymisation Techniques and Considerations

Dataset : Fitness Data

1. Pseudonymisation

a. Identify Personally Identifiable Information



Document 1

The first challenge of Pseudonymisation is to identify which attributes qualify as a Personally Identifiable Information (PII). A PII attribute is an attribute that can immediately identify a person regardless of any other attributes. This means that each element of a PII Attribute is unique or nearly unique with respect to the other elements contained in this attribute. Then to identify all of these attributes, a simple function *attributes_identification(data_frame)* can be written that calculates the percentage of uniqueness of each attribute *(Document 1)*. As we can see, the two highest attributes are *athlete_id* and *name*. Obviously this was expected since these are the two attributes that are used to identify the individuals.

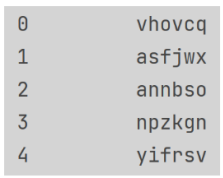
b. Pseudonymisation function

Once we have figured out which attributes should be pseudonymised to prevent the individuals to be immediately recognized, we write the function *pseudo_(data_frame)* using the *anonymizedf* library and the *fake_names* function. Now each name in the attribute *name* gets its own pseudonym and the individuals cannot be identified by their names anymore *(Document 2)*. Regarding the athlete_id attribute, depending on its use it should be either kept, pseudonymised or deleted. If it is linked to the name in another Database that is public, the attribute should be pseudonymised or deleted. Otherwise we could keep it as it is in the Database. Since I don't have any other Database, I just kept it. However, using the function *fake.ids* from *anonymizedf* would easily pseudomyze it.

Document 2

2. Randomisation

a. Simple randomization

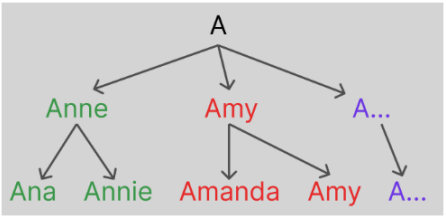


Document 3

An alternative to Pseudonymization is Randomization. Randomization can be really easy by simply replacing any Data with a randomized number or *String* used by a finite combination of random numbers or string using the *random* library. To show an example of simple randomization, we randomized the *name* attribute *(Document 3)* using the *randomizer(path, col_to_randomize)*.

b. Randomization with meaningful replacements

The problem with simple randomization is that it destroys absolutely all information contained in the attributes. Another way of Randomization is to randomize values with meaningful replacements. The idea was to go from “Anne” to “Amy”. However, I’m not sure I did what was asked. What I tried to do was to create a look-up table working as a Tree where each node was a different first letter. To do this, I created one dictionary that associates each first letter in the attribute *name* using the function *name_generator(data_frame)* which is based on the use of the *fake.first_name()* function and its randomness. Then you just have to replace the names in the attribute and keep a look-up table. If one were to try to get the names back, it could easily be done by iterating the look-up table, picking the first letter, and the first in the table would be the first iteration of the node (which is the new name in the table). If I only did it once as a proof of concept, one could apply the same function for each node and split every name depending on their second letter. The look-up table would then look like *Document 4*. You could reuse the function as many times as you want, depending on the degree of uniqueness you’re trying to achieve. I did meet difficulties with names that were starting with characters that weren't generated by the *fake.first_name()* function. For them, I deleted all the characters except the first one.



Document 4

3. Aggregation

a. Identify quasi-identifiers

The second part of anonymizing the dataset is to anonymize the riskier quasi-identifiers. For this, we must first identify them. A quasi-identifier is an attribute that can’t by itself be linked to one individual but if you were able to group many QI together you could. To that extent, I estimated them using the known QI such as *age*, *weight*, and *height* and the attributes that had a ratio of uniqueness *(Document 1)* superior to an arbitrary value (in my function 0.05%). To calculate these QI, I used the *identify_quasi_identifiers(data_frame)* function which is based on the *attributes_identification(data_frame)*. Then to aggregate the QI, I only picked the ones which were in the *float64* format. I eventually treated the attributes : *age*, *weight*, *height*, *filthy50*, *run5k*, *helen*, *grace* and so on.

b. Create intervals

Once we have identified the quasi-identifiers which are numbers, we can create intervals to avoid having the exact number. For instance if the age is 27, we replace it by an interval (25, 30]. To create these intervals, we use the function *aggregation_df(data_frame, column)* which splits each attribute into intervals containing each N% value (here I chose arbitrary N=10) using *panda.qcut()* and *numpy.arrange()* functions. The intervals are calculated using the unique values and not the population. This choice was made to avoid having a single interval with only one value. Eventually, I replaced every value in the attributes such as *age* (values shown in *Document 5)*. Even though I am aware that there were better methods to choose the intervals, I have so much inexperience in Big Data that I wasn’t able to create them using any other algorithm and it seemed like a fair idea because it respected the standard deviation and distribution of the values and lowered by many factors of 10 the uniqueness ratio of the attributes.

```
0          (23.0, 26.0]
1          (39.0, 43.0]
2          (39.0, 43.0]
3          (31.0, 34.0]
4          (36.0, 39.0]
```

Document 5

4. Perturbation

Once the PII attributes and the QI are aggregated, depending on the number of attributes, there might still be a lot of unique rows like on this Fitness data. To make sure that the individuals cannot get recognized, it is possible to add noise and perturbate the Data. To do this, I used the *add_noise(data_frame, attribute, noise_factor, noise_fraction)* function that uses the *numpy.random.normal()* function to randomly distribute noise in respect to the standard deviation. I set the noise_factor at 0.1 and the fraction of value perturbed at 0.5 and I applied it to the *age* attribute. Then I compared the standard deviation of the original attribute and the new attribute to make sure that I didn't tempered the data *(Document 6)*. We can see that the Standard Deviation is still a bit tempered (0.31%) which it shouldn't be but I wasn't able to find a function that wouldn't change the STD at all.

```
Std changed percentage  :  0.31 %
```
Document 6

5. Data Analysis

a. Computation

To be able to do define the Information Loss, I used the following formula that I adapted :

$$IL = \frac{1}{pn}\sum_{j=1}^{p}\sum_{i=1}^{n}\frac{x_{ij}-y_{ij}}{\sqrt{2S_j}}$$

The distance between the original Data Set and the intervalles was actually the distance to the mean value of these intervalles. Regarding the distance between two Strings, I arbitrarily set a rule : d = 0 if the Strings are equals, d = 1 if the Strings are not equal and d = 0.6 if the Strings are still a bit meaningful. I also created a function for their Uniqueness Score, computing how much variation was between the original Uniqueness and the new one. I also chose to only evaluate one column per transformation so that it could be easier to compare. Eventually using the *analysis()* function, I obtained these results :

```
Pseudonymisation         => Information Loss Indicator :  0.84  / Uniqueness Score :  -29.292
Simple Randomization     => Information Loss Indicator :  0.84  / Uniqueness Score :  7.302
Meaningful Randomization => Information Loss Indicator :  0.58  / Uniqueness Score :  -92.63
Aggregation              => Information Loss Indicator :  -0.38  / Uniqueness Score :  -0.013
Perturbation             => Information Loss Indicator :  0.03  / Uniqueness Score :  0.001
```

First of all, the 0.84 value is a bit strange because literally all the values changed. However I think it comes from the difficulties I had computing a "Standard Deviation" for the original strings. I think it's still representative. We also find the IL of Perturbation as 0.3 which is coherent to the Standard Deviation variation that I already observed. Lastly the IL Indicator concerning the aggregation is negative which is explained by the absence of absolute value in the formula but I struggle to be sure what it means.

b. Discussions, Pros and Cons

Pseudonymisation & Simple Randomization : As expected from these transformations, there is a great information loss since it's the actual objective : being unable to recognize the individuals. However we can see that Pseudonymisation loses a lot of Uniqueness because it can only be as wide as the Database you use to generate Pseudonymes. It can be a problem because it adds patterns that shouldn't be existing whereas Simple Randomization is much more effective.
Meaningful Randomization : Meaningful Randomization with its medium IL Indicator allows it to preserve some information and pattern while still being destructive to the original Dataset. However, we can see that Meaningful Randomization can be used to highly reduce the Uniqueness of the Data.

It can be really useful if you fear that people might get recognized using a combination of QI. Using it on different attributes allows it to make Outliers much more difficult to be seen.

Aggregation : As said earlier I'm not really sure how to interpret a negative IL Indicator but I think it's safe to say that while replacing all the Data, not that much information is lost which proves that it can be really useful. However, as studied during this Assignment, it depends a lot on the parameters chosen.

Perturbation : Perturbation is I think one of the most effective transformation techniques because really little Information is lost and yet still, 50% of the Data was transformed. Although, the Uniqueness remains the same so using other QI could still allow you to identify Outliers.