Assignment 5: Anonymization

This project is due on **December 3, 2021** at **5 pm**. You may work with a partner on this assignment and submit one project per team. You may NOT work with the same partner you have worked with on a previous assignment. Submit your solutions electronically via Moodle.

Background

The release of publicly available datasets is essential for public health, public policymaking, transparent governance, and collaborative research & development. However, public datasets often contain sensitive attributes that could cause harm if they are deanonymized to reveal the identity of the individuals they describe.

Public datasets are often prone to deanonymization by linking multiple datasets and background knowledge on attributes known as **quasi-identifiers** (QI). Naïve anonymization techniques, such as removing names or addresses, do not account for the breadth of possible quasi-identifiers and remain vulnerable to deanonymization. Fortunately, computer scientists have developed anonymization algorithms that provide stronger theoretical guarantees.

In this assignment, you will gain experience with the k-anonymity, l-diversity, and t-closeness algorithms on a public dataset.

Objectives

- Implement k-anonymity and l-diversity on a public dataset.
- Calculate the t-closeness of the resulting k-anonymous and l-diverse datasets.

Provided Files

- `Readme.pdf`: This file.
- `CensusData.csv`: Curated subset of the 1994 U.S. Census data maintained for machine learning practice by UC Irvine.
- `Questions.txt`: File for you to complete with written responses to open-ended questions.

Dataset Overview

The `CensusData.csv` file consists of a header row followed by 32561 data rows. Each row contains 6 columns with the following attributes:

- **Age:** The age of the respondent as an integer
- **Occupation:** The respondent's occupation category and sub-category as a hyphen-separated string (e.g. "Farming-fishing")
- **Race:** The race of the respondent as a string
- **Sex:** The sex of the respondent as a string
- **HoursPerWeek:** The approximate hours per week worked by the respondent as an integer
- **EducationNum:** The respondents' number of years of education as an integer (e.g. 12 = high-school diploma)

Unknown values not entered by Census respondents are denoted with a question mark. For more details about the full dataset, see `https://archive.ics.uci.edu/ml/datasets/Adult`.

Part 1. *k*-Anonymity (25%)

Your first task is to write a Python script `kAnonymize.py` that can be run as follows:

```
python3 kAnonymize.py ./path/to/CensusData.csv k
```

Your script should *k*-anonymize the census dataset for quasi-identifiers QI = `{Age, Occupation, Race, Sex, HoursPerWeek}` and command-line argument *k*. The resulting dataset should be written to a new file `kCensusData.csv` in the same directory as the script.

Your *k*-anonymization algorithm should group the rows of the dataset into anonymity sets and reduce the precision of QI attributes to ensure that every set has at least *k* members. You may assume that the value of *k* is between 2 and the number of rows in the dataset (inclusive). You may choose how you reduce the precision of attributes. For example, you may round numerical attributes (e.g. `Age`) to the nearest 5 or 10. You may remove portions of categorical attributes (e.g. `Exec-managerial` $\rightarrow$ `Exec`) or (as a last resort) replace them entirely with `*`.

Your algorithm does NOT need to find the optimal *k*-anonymization of the dataset resulting in the least loss of precision. However, it should strive to minimize the amount of imprecision where possible to produce a useful dataset for downstream analysis.

Your script should also track the number of individual attributes that are subject to a precision reduction and print this count at the end of the script. For example, if your algorithm reduces the precision of the `Age` attribute in 3 rows and the `HoursPerWeek` attribute in 2 rows, it should print `"Reduced precision of 5 attributes"`.

Part 2. l-Diversity (25%)

Your next task is to write a new script named lDiversify.py that can be run as follows:

```
python3  lDiversify.py  ./path/to/kCensusData.csv  L
```

Your lDiversify.py script should read the kCensusData.csv file created by your kAnonymize.py and modify the dataset to increase its l-diversity for sensitive attribute `EducationNum` to command-line argument L. You may assume that L is between 2 and the number of unique values of `EducationNum` in the dataset.

The script must maintain *or increase* the dataset's existing k-anonymity for QI = `{"Age", "Occupation", "Race", "Sex", "HoursPerWeek"}`. The resulting dataset should be written to a new file lCensusData.csv. If the input k-anonymized dataset is already l-diverse for command-line argument L, the script should simply write the dataset unchanged to lCensusData.csv

Otherwise, your l-diversification algorithm should swap rows between anonymity sets or merge anonymity sets such that there are at least L different values of sensitive attribute `EducationNum` in each set. Your algorithm may need to further reduce the precision of some QI attributes to achieve this goal. Your algorithm does NOT need to find the optimal l-diversification of the dataset resulting in the least loss of precision. However, it should strive to minimize the amount of imprecision where possible to produce a useful dataset for downstream analysis.

lDiversify.py should track the number of attributes that are subject to a precision reduction and print this count at the end of the script.

Your lDiversify.py may import functions from or otherwise depend on your kAnonymity.py and assume that these scripts are located in the same directory.

Part 3. t-Closeness (25%)

Your third task is to measure the t-closeness of your k-anonymized and l-diversified datasets. You should write a Python script named tCloseness.py that can be run as follows:

```
python3   tCloseness.py   ./path/to/[k or l]CensusData.csv
```

This script should calculate and print the t-closeness of the dataset in the argument CSV file. More specifically, tCloseness.py should use the **earth mover's distance metric** to compute the **maximum** distance between the distribution of sensitive attribute `EducationNum` within any anonymity set and the distribution of `EducationNum` in the entire original CensusData.csv dataset. The earth mover's distance (also known as the Wasserstein distance) can be calculated using the `scipy.stats.wasserstein_distance()` function. You will need to `pip3 install scipy` in order to import this function. The function documentation is here:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wasserstein_distance.html

Your tCloseness.py may import functions from or otherwise depend on your kAnonymity.py and/or lDiversify.py and assume that these scripts are located in the same directory.

Implementation Tips

It will be helpful to create a toy dataset with the first 100 rows of `CensusData.csv` to use for testing and debugging.

The Python `Pandas` library (`https://pandas.pydata.org/docs/`) is very helpful for reading and manipulating datasets. For example, the `Pandas` `read_csv()` function will read `CensusData.csv` directly into an easy-to-manipulate `DataFrame` object. The `.groupby()` method will then group all rows that have the same values in the argument columns list.

You are not required to use `Pandas` for this assignment. Vanilla Python with dicts, lists, & sets will work just as well. However, if you are looking to minimize the amount of code you have to write, are already familiar with `Pandas`, or would like to learn a widely used Python data science library, it is recommended. If you do decide to use `Pandas`, you will need to `pip3 install pandas` first.

Deliverables

Upload the following files to Moodle:

- `kAnonymize.py` (25%)
- `lDiversify.py` (25%)
- `tCloseness.py` (25%)
- `Questions.txt` with your responses to open-ended questions (25%)

Extra Credit "Bug Bounty"

If you find a bug anywhere in this assignment, please inform Prof. Apthorpe. The first student (or partners) to find any particular bug will be given a small amount of extra credit. This is an incentive to start the assignment early and will help make the course better for students in future years.