Exercise 1 - Phase 2: Security, Privacy and Explainability in Machine Learning

Student: Marko Georgiev 12442103 **Mentor**: Prof. Rudolf Mayer

Overview

This report serves as an overview of what I did during Phase 2 (P2) of Exercise 1 (E1) of the SPEML course at TU Wien. Unlike P1, where we are given a fingerprinted dataset and we must attack it, here we are given two datasets, one from P1 and another from Bob. We are also allowed to organize between each other, something which I took full advantage of during this phase. In contrast to P1, where I had to figure out how to measure data utility, here I can focus strictly on removing the fingerprint. My approach combines all datasets I could get my hands on and introduces noise. I also measure data utility after the attack to ensure the resulting dataset preserves utility.

Datasets Analysis

Before anything I need to analyze all the datasets, I've compiled for E1. I have a total of 5 datasets, two of my own and three of fellow students taking the course. Having access to multiple copies of the data is extremely important for my approach in terms of removing the report.

Importance of Counting Unique Datasets

As pointed out to us by the exercise description, there is a chance that some of the Bob copies given out to different students might be the same. This makes it necessary to check whether every copy of the dataset that we have is completely unique. By analyzing the mechanism by which fingerprints are introduced to the datasets and by having multiple copies of the exact same fingerprinted dataset, I might introduce bias into the fingerprint removal process. For example, since I have 10 copies at the time of writing this report, if three copies of the data are the same, they will always have the majority in terms of changed values in the 'cleaned' dataset. I will explain in more detail why this bias occurs in the 'Attack Execution Details' section.

Dataset Difference Analysis

Before I worked on the attack, I needed to know how the datasets were different from each other. My goal is to identify positions in the dataset where values differ and mark these positions as the locations where fingerprint modification occurred. To assess this, I looked at every cell across all unique datasets and counted how many unique values occur. Results are saved in a file '`diff_map.csv`' and this difference map is then analyzed into a '`summary.csv`' file that tells us how many cells in total will be modified. Lastly, I have a file titled '`uniqueness.csv`' which tells me the number of times a dataset had unique values in a cell where multiple datasets disagree (the number of unique values was > 1 because of that dataset). The complete notebook that does these computations is in '`dataset-diff-analysis.ipynb`'.

Attack Execution and Results

Now, it is finally time to discuss the attack. Since I have what I consider to be enough dataset versions, my attack only focuses on combining them all into a single one.

Dataset Utils

An extremely important script in my approach is the script titled '`dataset_uniqueness_utils.py`'. This script contains two functions: '`get_unique_datasets`' a function that reads all files in the datasets-p2 directory and returns all unique datasets, their filenames, and how many there are, and '`save_cleaned_output`', which saves the cleaned version of the datasets and all modification logs into files. Results saving will be discussed in section 'How are Attack Results Saved'

Attack Execution Details

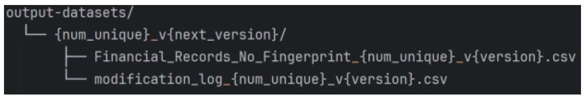
Finally, in terms of the attack, I will break it down like a pipeline. First, I iterate across all cells from all unique datasets, to collect the values. For example, for column `employment_since`, I collect ['3', '3', '3', '3']. Next, since I have both categorical and numeric values, I try to cast all unique values into a float. If the result is `NaN`, then it's treated as a categorical value, and if not as numeric. The code also handles mixed-type data like e.g. if `employee_since` is ['<1 year', '<1 year', '46', '46']. With this information, I can finally start the fingerprint removal. The cleaning follows simple rules. For fully numeric values, if the standard deviation is low, I clean the cell by averaging all the values. If the standard deviation is high, I clean it by selecting the most common value after rounding all numbers to four decimal places. For fully categorical values, I clean the cell by choosing the most frequently occurring value. For mixed-type values, where some values are numeric and others are categorical, I perform a coin flip. There is a 50% chance that I will use the meaning of all the numeric values, and a 50% chance that I will randomly select one of the non-numeric (categorical) values. The full attack is saved in the '`P2-Collusion-Attack.ipynb`' notebook, including a version of the attack as well as explanation why a new version was created.

Introduction of Noise

For categorical cells with unanimous values across all datasets, there is a 0.5% chance of replacing the value with a random alternative from that column. For mixed-type cells, a 0.5% chance allows the algorithm to choose a categorical value instead of the numeric consensus. See V3 of the '`P2-Collusion-Attack.ipynb`' file.

How are Attack Results Saved

The cleaned dataset and its associated modification log are saved in a versioncontrolled structure under the **`output-datasets/`** directory. Each attack execution creates a uniquely named subdirectory following the pattern shown in Figure 1, where `num_unique` is the number of unique datasets used in the cleaning process, **`version`** is automatically incremented based on previously created subdirectories.



```
output-datasets/
  └── {num_unique}_v{next_version}/
        ├── Financial_Records_No_Fingerprint_{num_unique}_v{version}.csv
        └── modification_log_{num_unique}_v{version}.csv
```

Figure 1

With this approach I can fully trace what happens and how the resulting dataset is created (by inspecting modification logs).

Post-Attack Data Utility

After the attack, I measured post-attack data utility by comparing my cleaned dataset to all fingerprinted versions using cell change rate, KS tests, mean/std differences, Jaccard similarity, and correlation matrix distance. The results show minimal distortion, confirming my method preserves statistical structure and categorical integrity while effectively removing the fingerprint. Refer to the '**`utilitiy-comparisson.ipynb`**' file for the full output.

Exercise Questions

In this section I will answer the questions given to us in the exercise description.

1. Having multiple datasets, how obvious do you think the fingerprint was before you started attacking it? (0-10)

1/10

Considering that we have multiple dataset copies, before attacking it or doing any kind of analysis, the fingerprint is completely abstract. We can only make assumptions as E1 gives us a detailed explanation on how fingerprints are embedded, but that's it.

2. How eager were you to include Bob's copy to design your attack? (0-10)

10/10

Ever since completing P1, and struggling with including noise in a smart way, and possibly ruining the dataset, I couldn't wait to create an attack that includes different dataset copies.

3. How eager were you to include more copies to design your attack? (0-10)

10/10

Again, from the moment I read that we can interact with students, I was extremely excited to create a forum post. And I was even more excited when I finally got replies.

4. How confident are you that you broke the fingerprint, and that no collaborator can be detected? (0-10)

9/10

Considering my probability (coin-flip) approach detailed in 'Attack Execution Details', I am very confident that the fingerprint is removed without trace.

5. How confident are you that at least one collaborator cannot be detected? (0-10)

8/10

Again, considering my probability based (coin-flip) approach, I am certain that no collaborator can be detected. However, I do think that with more dataset copies my confidence will increase. The introduction of noise further supports my confidence.

6. How difficult did you find the task of disrupting the fingerprint? (0-10)

4.5/10

Considering my access to multiple dataset copies, of the datasets 7 are unique. So, the only difficult part was deciding what to do with mixed values and writing the code.

7. How many collaborators (including Bob) did you work with? (Answer truthfully, there is no wrong answer here)

I worked with Bob and ~**7 students**, as of writing this report.

8. Why do you believe your attack was effective (or not)?

I believe my attack **was successful**, due to the randomized handling of values, introducing unpredictability.

9. If you had more time, what would you do differently?

Given the added time (date extension to May). I previously didn't have time to introduce noise. Perhaps, with more skill and time to study fingerprinting, I would try to **extract the fingerprint pattern**.

10. Did you notice any patterns in the data or in the dataset differences that gave you hints about the fingerprinting method?

The difference map is the only reference point I have for the data. My whole attack is based on using the heat map implicitly. Where there is difference there is an opportunity for cleaning. So, while I can't explicitly state where the fingerprint is, **my attack can and does**.

11. What was your biggest challenge in balancing fingerprint disruption and data utility?

For my approach since I am only changing about 2.6% to 3% of the data and using only given datasets with no noise. The cleaning process relied solely on values already present in the fingerprinted datasets, using consensus-based strategies such as averaging or majority voting (See section 'Attack Execution Details'. This limited the risk of degrading data utility, as no synthetic values were introduced (apart from the 0.5% noise). Therefore, the challenge was not in preserving utility but ensuring that modifications improved robustness against fingerprinting without degrading interpretability or downstream task performance. Given that all modifications were derived from real, observed values, I can confidently argue that the cleaned dataset **preserves at least the utility of the worst fingerprinted dataset** used in its creation, if not more. Refer to section 'Post-Attack Data Utility' and the mentioned file, for more details.

12. How does having multiple datasets help or complicate your attack?

My attack relies on having multiple dataset copies, so my answer is not only it strongly **helps** my attack, but my attack depends on it.