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

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

Notice: All supporting documentation (e.g., notebooks, code, datasets, sources, etc.), referenced are included in the .zip file submitted to TUWEL. Additionally, in the contents of that .zip file you can find many other documents, all of which simply didn't make it into the final version, and I sadly could not include descriptions why this report is due to the 2-page limit.

Overview

This report serves as an overview of my activities during Phase 1 (P1) of Exercise 1 (E1) from the SPEML (194.055) course at TU Wien. The exercise puts us in a scenario where we are given a fingerprinted dataset titled '`Financial_Records.csv`'. This is a copy of the original data which has a fingerprint embedded, allowing the original owner to trace it back to us. As per the P1 description, we have gone rouge, and we now want to redistribute this dataset while ensuring the original owner cannot confirm this data came from us. In Section 1 (Benchmarking), this report contains the approach to assessing data utility and fidelity, both for the fingerprinted and attacked dataset. While Section 2 (Fingerprint Attack), contains the full attack methodology. I also answer the Questions highlighted at the P1 description on the SBA research website. In the spirit of transparency, I state, during the executing of P1 I used LLMs, and their uses will be explicitly stated where relevant.

Benchmarking

After taking the time to fully comprehend the fingerprinting process covered in SBA research website, it was clear that I needed to set some sort of benchmark on the original dataset. Meaning, I had to have some way to compare the original and attacked dataset, to ensure the usability of the data, after my attack.

Data Utility

Initially I wanted to use data utility in the context of model performance, but after many attempts I could not get a model to reliably produce accuracy >51% (mainly due to the data being limited inherently). Unfortunately, describing my attempts is beyond this report, but it was an interesting experience. Following the discussion on the TUWEL forum should give a clear idea why this approach was not potent.

Data Fidelity

So, instead I decided to use a fidelity centric approach for benchmarking the data.

Choosing Fidelity Metrics

First, I used a python script at [1]'`dataset-analysis.py`' which gave me a rough idea of the data. The script python code reads each column a .csv and tells me its range, average, standard deviation, and type (numeric, categorical, or mixed). This information is extremely useful as it allows me to choose specific fidelity metrics depending on the column characteristics. After researching online, I decided on these metrics:

- Mean Squared Error (MSE) for numeric columns
- Categorical agreement rate and Jensen-Shannon Divergence (JSD) for categorical columns

In our case the `checking_account` is another mixed attribute, for which we will use MSE for the numeric values and apply Exact Match Rate and JSD for distributional comparison.

Preparing Data for Fidelity Checks

When assessing fidelity metrics, data still needs to be prepared. In my context, this was straight forward, as the different fidelity metrics clearly

[1] code generated with DeepSeek LLM.

give their requirements for input data. Firstly, all numeric values should be converted to integer. This must be done for all numeric columns (`age`, `liable_people`, `existing_credits`, `duration`, `installment_rate`, `credit_amount`, `monthly_rent_or_mortgage`, `residence_since`). For all categorical values (`sex`, `marital_status`, `job`, `credit_hist`, `purpose`, `debtors`, `property`, `installment_other`, `foreign`, `housing`, `tel`, `online_banking`,) we ensure that there are no trailing values, all are lowercase, and there are no missing values. Lastly, for the mixed attribute `employment_since` containing two string options '`unemployed`' and '`1<year`'. For this attribute we will define a custom mapping '`unemployed`' to -1, and '`1<year`' to 0, the rest of the values are converted to integers. The full preparation process can be found in the [2]'`comparison-preparation-data.ipynb`' file.

Fingerprint Attack

With that we finally get to the attack. My attack on the dataset, doesn't attempt to remove the fingerprint, which would revert the data to the original state.

Attack Plan

The first noticeable problem is that PID is an attribute not mentioned in the dataset description. This leads me to believe that it is no a mission critical attribute, and therefore likely a record identifier that helps with the fingerprint's PRNG synchronization. Another assumption I am making is that fingerprint embedding follows the process described in the forum. Therefore, my attack plan will target the PID in a way that makes it not easily recoverable. After this I will systematically perturb both categorical and numerical attributes across the dataset to introduce fingerprint bit collisions. I will introduce controlled modifications across the dataset to disrupt the fingerprint detection process, which will alter value distributions, introducing ambiguity in frequently marked attributes, and degrading the reliability of fingerprint bit reconstruction without significantly affecting overall data utility.

[2] Comments and code sections generated by OpenAI LLM ChatGPT 4o

Attack Execution

This was the fun part of the whole exercise. The first thing I did was create a script with the file name '`dataset-shuffler.py`'. The file takes a .csv file and shuffles the rows inside based purely on a key (passcode). I need this as I must shuffle the entries in the dataset, so that the attack on the `PID` makes sense. If I don't shuffle the records, the owner can simply recompute the `PID` and find places where there has been tampering (or at least make it more difficult). Next, I implemented a full-scale fingerprint removal attack implemented in a file called '`dataset-attack.py`'. The next phase of the attack was chosen with the goal of exploiting the fingerprint generation structure described in the instructions of the SBA research website. I chose bit collision engineering. For this part of the attack, I aimed to confuse fingerprint detection by tweaking the statistical patterns in the data. I randomly swapped categorical values like `marital_status`, `job`, and `property` with other common ones. For numeric fields such as `credit_amount` and `duration`, I applied small random noise (around ±15%) and rounded the results to keep them as integers. I also flipped binary values like `default` with a 20% chance. This disrupted local distributions without breaking the dataset structure, to much.

Results

After performing the attack, I analyzed all the results Table 1 shows the results for the numerical values, Table 2 for the mixed columns and Table 3 the results for the categorical results. These results were generated by running '`comparison-preparation-data.ipynb`', which if you recall also contains the preparation code.

Now, in terms of the outcome of my attack. I measured both numeric distortion and categorical distribution changes. For numeric features, I used Mean Squared Error (MSE) between the original and attacked values. Most fields like installment_rate, residence_since, and liable_people showed very low MSE (<0.3), confirming that the utility of the data was preserved. Higher MSE values in fields like credit_amount (421,119) as well as the monthly_rent_or_mortgage (3,227) were expected, as these were intentionally perturbed more heavily to shift quantile boundaries.

For categorical attributes, I calculated agreement rate (unchanged values) and Jensen-Shannon Divergence (JSD) to capture distributional drift. Most fields had agreement rates between 75%–85% and relatively low JSDs, indicating moderate but controlled changes. For example, job and credit_hist had lower agreement (~77% and ~76%) and higher JSDs (~0.10 and ~0.11), showing that bit collision engineering successfully reshaped their local distributions. Overall, the results confirm that the fingerprint was attacked effectively: numerical signals were disrupted without breaking data realism, and categorical distributions were reshaped just enough to mislead the density-based fingerprint decoder. The data remains highly usable, yet much harder to trace back to the original fingerprint.

Execution of the whole attack in one notebook (including the comparison) can be done by running the '`full-notebook.ipynb`'.

Column	Type	MSE
age	numeric	35.217943
credit_amount	numeric	421119.294171
duration	numeric	17.212443
monthly_rent_or_mortgage	numeric	3227.077114
installment_rate	numeric	0.304414
residence_since	numeric	0.291686
existing_credits	numeric	0.047786
liable_people	numeric	0.017514
default	numeric	0.200343
employment_since	numeric	1.154229

Table 1

Column	Type	MSE	Agreement Rate	JSD
checking_account	mixed	14324.373822	1.0	0.0

Table 2

Column	Type	Agreement Rate	JSD
sex	categorical	0.848371	0.065110
marital_status	categorical	0.801757	0.091862
job	categorical	0.776743	0.099901
credit_hist	categorical	0.759071	0.111292
purpose	categorical	0.754714	0.074388
debtors	categorical	0.798557	0.145328
property	categorical	0.775986	0.022937
installment_other	categorical	0.800271	0.110993
housing	categorical	0.802714	0.101846
tel	categorical	0.851400	0.006557
online_banking	categorical	0.848786	0.077840
foreign	categorical	0.849757	0.136580

Table 3

Final Questions

How obvious do you think the fingerprint was before you started attacking it? (0-10)

I interpret this question as, how easy it was to find the embedding of the fingerprint inside the data, before any attack. From the my perspective it's impossible (1/10), but since no change has been made yet, from the perspective of the data owner, it would be extremely easy (**10/10**).

How confident are you that you broke the fingerprint? (0-10)

Considering my methodological and thought out approach I am very confident that I broke the fingerprint or at least made it much more difficult to find it (**8.5/10**).

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

Disrupting the fingerprint was very fun, but also very difficult. I had to do a lot of research, and unfortunately relied on many assumptions about the fingerprint generation, so I would say it was difficult, but worth it (**8/10**).

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

The attack focused on restoring natural-looking patterns and distributions, especially in areas suspected to be manipulated for fingerprinting.

Since all the changes were targeted, and contextual, I believe my attack was effective.

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

This is an easy one, I would definitely implement more than one fully working attack strategy.

Did you notice any patterns in the data that gave you hints about the fingerprinting method?

Unfortunately, no I didn't. Nothing in particular stood out when plotting the original fingerprinted data.

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

My biggest issue was choosing metrics by which we can determine data utility. While, the forum helped this remained the main issue. Consequently, preserving the semantic and statistical integrity of the data while making changes was the hardest part. It's hard to preserve semantics when you 're not sure what they are.