Supplementary Materials for: Privacy-Preserving Feature Selection with Secure Multiparty Computation

Xiling Li, Rafael Dowsley, Martine De Cock

1 Introduction

In this document, we describe supplementary materials for the manuscript "Privacy-Preserving Feature Selection with Secure Multiparty Computation" [2]. We implemented the proposed protocols in the open source Secure Multiparty Computation (MPC) framework MP-SPDZ.[1] To empirically verify the runtime improvements that can be obtained with our MS-GINI criterion compared to traditional GI, we also created an implementation of an MPC protocol for computing GI proposed by [1] that is based on oblivious sorting. To this end, we adapted code made available to us by Mark Abspoel. Three related files are submitted with this document:

- `pp_ms_gini_fs.py`: implementation of our protocols
- `pp_ss_gini_fs.py`: implementation of sorting-based protocols
- `icml_submission.mpc`: script to evaluate the protocols on the data sets described in Section 2

Further details about the experiments are described in Section 4.

2 Data sets

We used three data sets that we obtained online. We refer to the paper [2] for a description of the data sets and the urls where they can be downloaded. For the experiments, we converted the data into a proper format for MP-SPDZ in the directory `mpspdz_data` submitted with this document:

- Cognitive load data set: `Input-P0-0-COG-DATA` and `Input-P1-0-COG-LABEL-CLASS`
- LSVT data set: `Input-P0-0-LSVT-DATA` and `Input-P1-0-LSVT-LABEL-CLASS`
- Speed dating data set: `Input-P0-0-SD-DATA` and `Input-P1-0-SD-LABEL-CLASS`

`Input-P0-0-*-DATA` is data matrix D described in Section 4.1 of the manuscript [2].
`Input-P1-0-*-LABEL-CLASS` is label-class matrix L described in Section 4.2 of the manuscript [2].

3 Setup of machines

We went through the following steps to set up the environment for the experiments on MS Azure:

- Create multiple virtual machines on Azure (we used Azure F32s V2 machine in this work, e.g. 3 machines for 3-party computation, namely P0, P1 and P2).
- Set up inbound and outbound rules for each machine (e.g. port 5000) to make sure machines can communicate with each other.

[1] `https://github.com/data61/MP-SPDZ`

- Clone the MP-SPDZ framework from `https://github.com/data61/MP-SPDZ///releases` on each machine.
- Compile the framework as described in `https://github.com/data61/MP-SPDZ` on each machine.
- Move
  - `pp_ms_gini_fs.py` to `*/mp-spdz/Compiler/`,
  - `pp_ss_gini_fs.py` to `*/mp-spdz/Compiler/`,
  - `icml_submission.mpc` to `*/mp-spdz/Programs/Source/`, and
  - `mpspdz_data` to `*/mp-spdz/`

  on each machine.
- Create a directory named `Player-Data` on each machine by `mkdir mp-spdz/Player-Data`.
- Set up an ssl connection by `Scripts/setup-ssl.sh 3` on P0.
- P0 distributes `.key` and `.pem` files to `*/mp-spdz/Player-Data` of other machines. For example, P1 gets `P0.pem`, `P1.pem`, `P1.key` and `P2.pem`.
- Execute `c_rehash Player-Data` on each machine to allow ssl to recognize `.key` and `.pem` files during communication.
- Create a file called `HOSTS` on each machine by including the IP addresses of all computing machines.

4 Experiments

Experiments can be performed for each desired data set in the following manner:

- Edit `*/mp-spdz/Programs/Source/icml_submission.mpc` for use of the desired data set and desired approach.
- Copy the desired data in `*/mp-spdz/mpspdz_data`. For example, to perform experiments for the "cognitive load" data, we need to execute
  - `cp mpspdz_data/Input-P0-0-COG-DATA Player-Data/Input-P0-0` on P0
  - `cp mpspdz_data/Input-P1-0-COG-LABEL-CLASS Player-Data/Input-P1-0` on P1
- Compile `*/mp-spdz/Programs/Source/icml_submission.mpc` on each machine by

  ```
  ./compile.py -R 64 icml_submission
  ```

  Note that we use $k = 64$ in this work.
- On each machine, execute
  - for passive 3PC:

    ```
    ./replicated-ring-party.x party_id -R 64 icml_submission -pn 5000 -h P0_ip
    ```
  - for active 3PC: same as above, with `sy-rep-ring-party.x`
  - for active 4PC: same as above, with `rep4-ring-party.x`
- The accuracy results in Table 1 in the manuscript [2] can be obtained as follows:
  - Matrix D', selected by Protocol 1, can be used to train a logistic regression model with

    ```
    sklearn.linear_model.LogisticRegression
    ```

    of `scikit-learn`.
  - Feature selection in-the-clear based on GI, PCC, or MI, and training logistic regression models over the selected features, can be done directly using `Pandas`, `Numpy` and `scikit-learn`.

References

[1] Mark Abspoel, Daniel Escudero, and Nikolaj Volgushev. Secure training of decision trees with continuous attributes. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, pages 167–187, 2021.

[2] Xiling Li, Rafael Dowsley, and Martine De Cock. Privacy-preserving feature selection with secure multiparty computation. In *38th International Conference on Machine Learning*, volume 139 of *PMLR*, 2021.