Federated Learning for Credit Risk Modelling

Hamid Aliakbarlou, Rongde Liang, Jesse Thibodeau

December 20 2021

Abstract

We motivate the use of federated learning as a solution to data privacy issues in the financial sector. In particular, we adapt a classic federated learning framework to a logistic regression model with the goal of predicting loan defaults in a privacy-preserving manner. We compare our FL model's accuracy to that obtained through centralized methods. As expected, the accuracy of the FL model obtained through distributed computation was slightly lower than that of our benchmark model, but we argue that the resulting gain in privacy far outweighs the marginal reduction in accuracy.

Contents

1 Introduction

Nowadays, smartphones and tablets are widely used and serve as people's primary computing devices [1, 2]. Equipped with powerful sensors such as gyroscopes, accelerometers and GPS, and generally carried frequently, they can access huge amounts of sensitive data. Models learned on such data could show promise of significantly improving the user experience, but its sensitive quality means that storing it in a centralized server poses considerable risks. In this project, we investigate a privacy-preserving learning technique which allows models to be trained on local data without the need for central storage, and apply this technique to a credit risk modelling problem. This privacy-preserving approach is called Federated Learning (FL) [3].

1.1 Federated Learning Background

Federated Machine Learning was initially introduced by Google Inc. in 2016 [3, 4, 5]. In a federated machine learning setting, various nodes e.g., business substructures like hospital servers, or end-user systems like mobile phones, contribute to solving a machine learning problem. For instance, a classifier could be obtained by a Deep Neural Network algorithm, or a simpler algorithm, with fewer variables like SVM or a logistic regression model [6]. Federated learning (FL) models are distinguished in the way that the training data never leave the related local device where they are accessed. Each device (commonly called a *client*) retains a version of the same FL model, which gets updated by each new observation. Every training iteration, a parameter update is performed on individual clients in a distributed manner i.e. in a way that does not involve data being shared between them. Only these learned parameters get shared with a central component (a *server*), where they are aggregated algorithmically. When the updated model is learned, it is redeployed to all available clients for further training. This is an iterative process that, in theory, will eventually converge just as a centralized approach would.

A graphical presentation of this process is shown in Figure 1 [7]. In (A) the initial weights are pushed to the clients, which subsequently train the model on local data. As shown in (B), local updates are then performed by the clients. Subsequently, new models are sent to the central component and get aggregated to obtain the model in (c). Then, this updated shared model is redeployed to all devices and the whole process starts again.

There are three main types of FL, which are "Horizontal Federated Learning", "Vertical Federated Learning", and "Federated Transfer Learning" [8].

- Horizontal FL involves the use of datasets sharing the same feature space across devices, as shown in Figure 2, left.

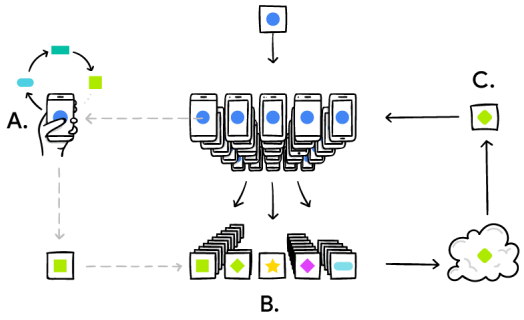

Figure 1: FL graphical conceptual architecture

- Vertical FL involves learning from datasets with different feature spaces, but similar sample spaces as shown in Figure 2, middle.
- Federated transfer learning is an approach designed to deal with datasets that share neither a sample space or a feature space, as shown in Figure 2, right.

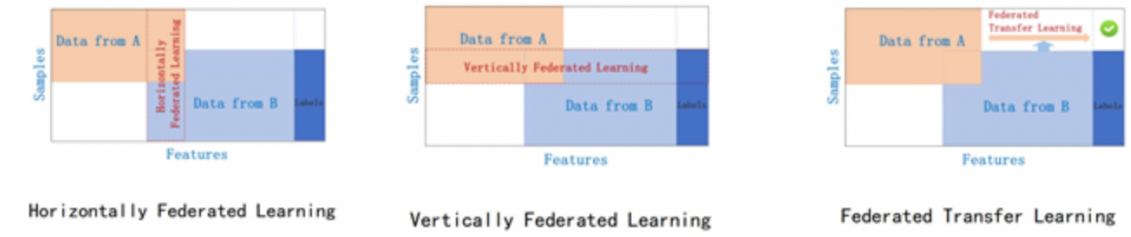

Figure 2: Different types of FL data architectures

1.2 Importance of Privacy Preservation

Federated learning is evolving rapidly and has become a hot and interesting research topic in the field of ML [9, 10]. There are three facts that FL techniques capitalize on: (a) the never-ending growth of data, (b) successful progressive applications to ML technologies, (c) the increasing importance of data privacy regulations worldwide.

Data privacy regulations play an important role in the emergence of FL innovations. As these regulations are established, data breaches continue to threaten privacy. For example, in 2019, a huge number of records associated with Facebook users was exposed on Amazon's cloud service [11]. This data breach, among others, sparked serious legal and social challenges. In response to such problems, multiple regulatory measures were set for preserving users' data privacy like the Singapore Personal Data Protection Act introduced in

Singapore [12], the General Data Protection Regulation in the European Union [13], and the California Consumer Privacy Act ruling in the US [14]. These measures are meant to allow users to delete or withdraw their personal data. In addition, any kind of autonomous activity in collecting, transferring or using private/sensitive data is prohibited. The role of such regulations is significant in promoting the development of FL, specifically Privacy-Preserving FL (PPFL) [15].
FL methods therefore provide three main benefits for data modelling: 1. privacy, 2. computational power, and 3. real-time learning.

1. FL has distinguished privacy features that are enabled by the models ability to be trained in parallel without the sharing or storage of local data.

2. Transferring computations down to multiple devices remarkably reduces the process power necessary in a central place. This stems from the fact that in FL, the central entity's role is just to aggregate the updates obtained by all participating local devices.

3. By performing the model training process locally, updates occur instantly, time-to-prediction and consequently user experience is therefore enhanced.

1.3 Problem Statement and Objective

Data breaches are unfortunate, yet commonly occur in the financial sector [16]. They are costly, result in lower client trust, and could be very threatening to the sector's potential to adequately manage a growing influx of transactional (and other) data, which is sensitive in nature. Our goal is therefore to introduce privacy-preserving methods as a solution to privacy concerns in financial data. To achieve this, we will explore an application of federated learning to a credit risk model for loan approval, and compare its accuracy to that of a traditional, centralized modelling approach.

2 Theoretical Groundwork

The problem we face is twofold: first, we must build a credit model for predicting default, and finally, we must federate the model in order to preserve data privacy. To accomplish the former, we use logistic regression for binary classification, and for the latter, we implement a well-known parameter aggregation technique. Let us now provide the theoretical background for both tasks.

2.1 Logistic Regression

The model we will be fitting to our data is a logistic regression. We chose it for its simplicity and ease of implementation. The logistic model uses the sigmoid function,

$$\sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} \qquad \text{Sigmoid function}$$

to take a design matrix (covariates) as input and outputs a value between 0 and 1, which can be interpreted as a binary classification prediction. The outputted value itself is a probability that the data instance belongs to class 0 or 1, or in this case, whether or not a loan applicant defaults on their debt. We use a decision boundary situated at 0.5. That is, we predict DEFAULT for applicants with $\sigma > 0.5$ and therefore they do not receive a loan, and those with $\sigma < 0.5$ are granted their loan. To prevent overfitting our model to the training data, we use L2 regularization, which penalizes learned weights for being too large according to the their squared values, scaled by hyperparameter λ. The loss function we aim to minimize is therefore the following:

$$\mathcal{L}_2 = (\bar{y} - y)^2 + \lambda w^2 \qquad \text{L2 loss function}$$

As a result, a proper selection of λ is crucial for our model to generalize well to out-of-sample data.

2.2 Shared Model Specification

For determining the shared model that was to be the result of our federated learning implementation, we use a classic algorithm introduced by McMahan, et. al. called federated averaging, or `FedAvg` [3]. The algorithm aims to determine the parameters obtained from the distributed training by aggregating the locally-computed weights from the individual clients through averaging. Algorithm 1 details the iterative process by which model parameters are learned and shared in a federated environment. The algorithm is directly adapted from the McMahan publication [3], but captures a few contextual changes we made. For its intuitiveness and simplicity, `FederatedAveraging` was our aggregation algorithm of choice when fitting our shared model. Naturally, we would want to compute a weighted average of the locally-computed parameters in order to obtain a shared model derived from each client's data, without said data being shared or stored in a centralized environment. That is precisely what `FedAvg` accomplishes. Specifically, the server (Avengers) component begins by taking in randomly initialized weights. Then, for every training round, the model is trained on a subset of participating clients (banks), the size of which are determined by hyperparameter C. Learned weights are updated using gradient descent. They are then aggregated by averaging between every

Algorithm 1 FederatedAveraging. The K banks are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Avengers Server Executes:
initialize w_0
for each round $t = 1, 2, ...$ **do**
  $m \leftarrow max(C * K, 1)$
  $S_t \leftarrow$ (random set of m banks)
  for each bank $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{BankUpdate}(k, w_t)$
  end for
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for

BankUpdate(k, w)**:** ▷ Run on bank k
$B \leftarrow$ (split P_k into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in B$ **do**
    $w \leftarrow w - \eta \nabla l(w; b)$
  end for
  Return w to server
end for

updated client weight to obtain shared global weights. This new shared model therefore represents the average of locally-computed models, updated over a predetermined number of local epochs and training rounds.

3 Data

An individual's financial data is very sensitive in nature, and can therefore be difficult to come by in public domains. In another course, Statistical Learning with Prof. Jean-François Plante, we were provided a simulated data set of features typically taken into consideration by financial models when determining whether or not an applicant is eligible for a loan. That is, present-day models will take into account an applicant's financial profile, made up of features such as loan amount, age, income (net and gross), value of savings, transactional records, assets, liabilities and more, to determine their likelihood of default. In total, we are given 27 descriptive features and one target variable, which was an indicator dummy telling us whether a default had occurred. The dataset was simulated by a professional with over 20 years of experience in financial modelling, and is meant to closely approximate what is observed in the real world.

Despite this reassurance, we investigated their distribution. Evidence of simulation is present. In particular, Figure 1 shows that the feature MNT_DEMANDE, which represents the requested loan amount, is uniformly distributed. That is, every requested loan amount is equally likely to show up in the data, which may not be very realistic. However, we did not anticipate that this would cause any issues with respect to

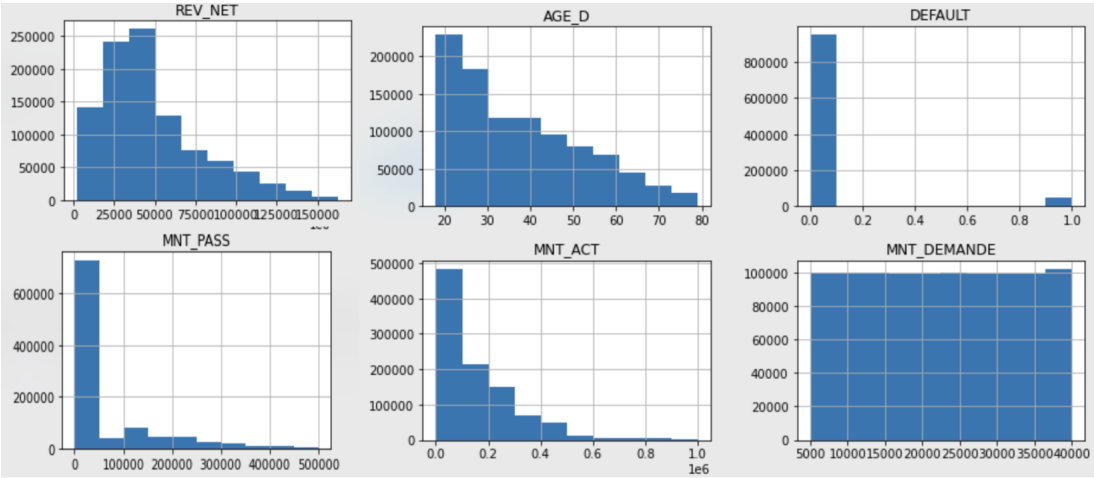

Figure 3: Distributions of select features using `matplotlib`

our implementation, as the purpose of the project was not to obtain accurate results, but to successfully implement an FL framework to a credit model.

4 Code Environment

We continue with a discussion about our coding environment, particularly, how we split our data and created numerous Python scripts that would eventually run in parallel.

4.1 Data File Structure

There are two dataset in a "credit" file folder: `CreditGame_TRAIN_all.csv` is the training dataset; `CreditGame_TEST_all.csv` is the testing dataset. In our FL setup, the server component utilizes the whole training dataset and testing dataset while each client uses a random sample of one quarter of both datasets. Therefore, the data is partitioned evenly between clients. Our sampling method does this automatically when we run the code.

4.2 Code File Structure

There are 7 python files in our project, described in Table 1. The coding component comprises three parts: the client scripts, the server script and the utils code. The utils code must be put together with client code or server code since it contains the command parameters and utility functions. The server code and client code can be run in parallel on different machines.

No.	File Name	Function
1	Centralize_LR	Centralized LR method for all bank data
2	Server_Avengers	Server part of FL method
3	Client_BMO	Client part of Fl method
4	Client_Scotia	Client part
5	Client_CIBC	Client Part
6	Client_TD	Client Part
7	Utils	Command parameters and utility functions

Table 1: Python scripts and their uses

The evaluation function computes the result of the shared model in the server script.

Tables 2 and 3 summarize our federated learning and logistic model specifications.

Name	Value	Description
round	20	The number of FL rounds
strategy	FedAvg	Implementation of the abstract base class
minimum_available_clients	4	Minimum participating clients

Table 2: `flwr` model specifications

Name	Value	Description
cv	5	Number of folds used in CV
random_state	0	Random state instance for solver
penalty	l2	Norm of the penalty
max_iter	10	Max iterations of optimization algorithm
solver	lbfgs	Algorithm used in optimization problem
Cs	[0.01, 0.1, 1, 10]	Grid of values representing $1/\lambda$

Table 3: `LogisticRegressionCV` model specifications

5 Results and Discussion

5.1 Shared Model Comparison

We run two experiments for comparison. First, Table 4 compares the classification accuracy of the centralized method to that of the federated learning method. This represents the results obtained when testing the shared model parameters on the aggregated bank test data. We find that the accuracy in both centralized and FL approaches is very comparable, but it is slightly lower in the Avengers model, as expected. By design, FL methods won't obtain better accuracy than centralized methods, since they rely on the aggregation of locally computed parameters, rather than simply obtaining these parameters directly by accessing raw data. However, we deem this marginal reduction in accuracy to be a small price to pay for the resulting gain

in privacy our implementation has provided. The goal was to approximate the results of the centralized approach, and we accomplished this task nicely.

Size	Thanos (Centralized Model)	Avengers (Shared Model)	Improvement
120000	69.32%	69.23%	-0.09%

Table 4: Global model accuracy comparison

We might now discuss the accuracy values themselves. In the centralized approach, we achieved an accuracy of 69.32%, and for the FL approach, we achieved an accuracy of 69.23%. In general, this level of accuracy shouldn't strike one as particularly high or low, given that a logistic regression model was used. However, in the context of our problem, which is to predict whether or not a loan applicant will default, it is important to consider the costs associated to the true positives and true negatives that contribute to our accuracy values. A financial institution in this setting would only be profitable if it receives the principal amount by the end of the lending period, plus interest payments. That is, it is only profitable by the amount of interest paid in the case of a true negative, where we accurately predict that a loan will be repaid. In contrast, a financial institution would be unprofitable if its false negative rate were high, that is, we predict that the loan will be repaid when in reality it won't. We note however, that in the case of a true negative, the bank is only profitable by the amount of interest, while in the case of a false negative, the bank loses the principal amount, which is considerably larger.

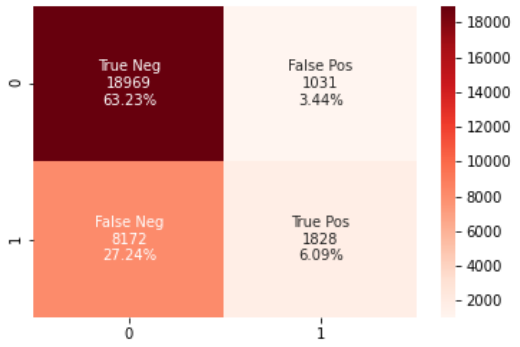

Figure 4: Centralized model confusion matrix

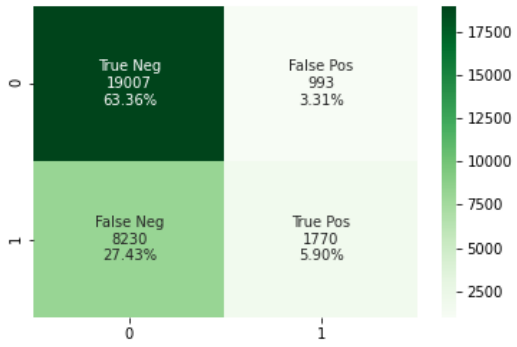

Figure 5: Shared model confusion matrix

We therefore aim to maximize the number of true negatives, and minimize the number of false negatives, as the negative predictions will be the ones who are granted loans. In other words, they generate all of the

financial risk. Our accuracy measure was concerned with maximizing true positives and true negatives, but it did not consider the greater cost of a false negative. Therefore, in practice, we would recommend that financial institutions experiment with different classification thresholds in order to minimize the number of false negatives while still maintaining a profitable amount of true negatives, thereby optimizing profitability. From Figures 4 and 5, we note that the centralized model had better detection of "Default", but in fact had worse detection of "Not Default", so it is actually less likely to give profitable loans than the shared model obtained through federated learning. However, in contrast, we observe that the centralized model is also better at avoiding giving unprofitable loans. It is therefore more conservative and is less likely to give a loan than the shared model. Given the information we have, it is difficult to directly determine the profitability of each model, as this would depend on the size of the loans and resulting costs of errors.

5.2 Client Model Comparison

Second, we compare the performance of the centralized and shared model approaches based on the individual banks' test sets. In other words, this compares the performance of the individual banks using the shared model obtained from FL rather than their own centralized model. In theory, we would expect to have a reduction in accuracy across all clients, unless each client had so little data that any model trained on it would likely not generalize well due to overfitting. However, in this case, we note that the expected reduction in performance only held true for half of the participating clients. As can be seen in Table 5, after the federated learning, Scotia and CIBC witnessed a decline in performance on their own test set, while BMO and TD improved theirs. Given the very marginal differences in performance, we suspect that they are simply due to uncontrollable factors, but it may also be due to an imbalance in the way the data was partitioned, not in size, but in presence of the minority "Default" class. For instance, if BMO's training set happened to contain a relatively larger amount of loan defaults, then it may be biased towards the "Default" class, since it sees it more frequently. So, when introduced to parameters learned from clients with lower default rates, it may actually have learned a model that generalizes better to its test set, which may not have contained disproportionately high default rates.

Name	Size	Centralized Accuracy	FL Accuracy	Improvement
BMO	30000	69%	69.14%	0.20%
Scotia	30000	69.42%	69.09%	-0.47%
CIBC	30000	70.09%	69.77%	-0.45%
TD	30000	68.77%	68.93%	0.23%

Table 5: Client model accuracy comparison

A potential solution to this problem would have been to standardise the proportion of the minority class that is present in all training samples, that is, we could have used oversampling for all clients to increase the representation of defaults in order for the local models to be able to better predict the likelihood of default.

5.3 A Marvel Analogy

As we observed, the results of our experiment were mostly in line with what the intuition and theory behind federated learning would dictate. That is, if properly executed, the model learned through distributed computation should somewhat approximate the centralized model. From this, we see that it is indeed possible for users to directly benefit from machine learning models without compromising their privacy. As we become more educated on the ways data is accessed from our everyday usage, it is paramount for policy-makers to implement regulations that preserve our individuality and enforce our right to maintain privacy. Throughout this report, we made occasional reference to Marvel's Avengers (shared model), treating them as the sum of individual heroes (small banks) coming together to face the larger entity (central bank), which we refer to as Thanos. In the end, the Avengers challenged Thanos and almost equalled his accuracy.

Figure 6: Uniting forces to take down the behemoth

Besides an attempt at humor, the Marvel analogy also serves to show how, though not always, there may be practices at large corporate entities, including financial institutions, that are questionable with respect to their use of sensitive data. Beyond data breaches, there are legitimate moral debates to be had about how our private data is used to, for example, target advertisements. Are these models complementing our intrinsic behaviour, or are they introducing new behaviours to which we are not predisposed? To what degree is the use of our private data indirectly compromising our individuality? As the field of machine learning continues to advance, the opportunities for data use, both benevolent and malicious, will undoubtedly grow. It is therefore imperative that we reduce the opportunities for malicious practice by careful implementation of privacy-preserving methods.

6 Conclusion

We successfully introduced federated learning to a credit risk model. The challenge was to obtain a model through decentralized computation that is comparable to a benchmark model obtained through centralized means. We used `FedAvg`, and the Flower framework for federated learning to learn the shared weights in a logistic regression model. In the end, the centralized model was 69.32% accurate, while the FL model was 69.23% accurate. We tested both models on the test data aggregated from each of the client banks. This reduction in accuracy was expected given the imprecision introduced by parameter aggregation. We also computed the changes in client accuracy when making predictions with the shared model on their own test set. We expected all clients to suffer a reduction in accuracy relative to the centralized method, but this was only true for half. This is likely explained by the imbalance of the "Default" class in the training sets.

While this project was an introductory experience with federated learning, we note that there are significant improvements to be made to our model and implementation. First, we would consider adapting more complex models to our classification problem, such as SVMs or multi-layered perceptrons. We would also consider the challenge of accounting for different data partitions, that is to allow for heterogeneity in the amounts of data held by each client. In addition, the context of the problem encourages some consideration for the costs of errors. In other words, we would like to see how false negatives and true negatives affect profitability. The problem of predicting default among individual applicants may also introduce some socio-demographic bias. For instance, it would be useful to obtain applicant socio-demographic features in order to evaluate whether our models are biased towards any particular social group. Finally, to further enhance our privacy-preserving techniques, it would be relevant to explore Differential Privacy [17] as a way of obscuring an individual's contribution to the shared model.

A Appendix

The following appendix is a walk-through of our coding implementation. The tools we use in our implementation are Python library scikit-learn [18], and FL framework Flower [19].

A.1 Client Code

First, we load the client training dataset and testing dataset. The load function is in utils.py which evenly divides the training and testing datasets into quarters.

```
(X_train_1, y_train_1) = utils.load_credit_train_1_Default()
(y_test, X_test, y_test_p_1) = utils.load_credit_test_1_Default()
```

Second, we configure the logistic regression model with cross-validation. We set 5 folds, 100 local epochs (set in utils.py), and regularization parameters λ between 0.01 and 10.

```
model = LogisticRegressionCV(cv=5, random_state=0,
                             penalty="l2"
                             max_iter=utils.load_parameter_C()[0],
                             Cs=[0.01,0.1,1,10]
                             )
```

Third, we define the Flower client. The client function should follow three methods: "get_parameters" returns the current local model parameters, "fit" sets the train model, "evaluate" configures the evaluation method.

```
class MnistClient(fl.client.NumPyClient):
    def get_parameters(self):
        return utils.get_model_parameters(model)

    def fit(self, parameters, config):
        utils.set_model_params(model, parameters)
            model.fit(X_train_1, y_train_1)
            print(f"Training finished for round {config['rnd']}")
        return utils.get_model_parameters(model), len(X_train_1), {}

    def evaluate(self, parameters, config):
        utils.set_model_params(model, paramaters)
        loss = log_loss(y_test, model.predict_proba(X_test))
        accuracy = model.score(X_test, y_test)
        return loss, len(X_test), {"accuracy":, accuracy}
```

Finally, we run the client with the main function `fl.client.start_numpy_client`.

```
(X_test, y_test, y_test_p, test_all) = utils.load_credit_test_Default()
```

A.2 Server Code

First, we load the server training and testing data. The load function is in utils.py.

```
fl.client.start_numpy_client("0.0.0.0:8080", client=MnistClient())
```

Second, we configure the federated learning method. Here, we use `FedAvg` and set a minimum of 4 participating clients.

```
strategy = fl.server.strategy.FedAvg(
    min_available_clients=4,
    eval_fn=get_eval_fn(model),
    on_fit_config_fn=fit_round,
)
```

Third, we define the Flower server. The server function follows two methods: "fit_round" sends the round number to the client, "get_eval_fn" does testing and prints out the results.

```
def fit_round(rnd: int) -> Dict:
    """Send round number to client"""
    return("rnd": rnd)

def get_eval_fn(model: LogisticRegression):
    """Return an evaluation function for server-side evaluation."""
    (X_test, y_test, y_test_p, test_all) = utils.load_credit_test_Default()

    # The "evaluate" function will be called after every round
    def evaluate(parameters: fl.common.Weights):

        #update model with the latest parameters
        utils.set_model_params(model, parameters)
        loss = log_loss(y_test, model.predict_proba(X_test))
        accuracy = model.score(X_test, y_test)

        predict = model.predict(X_test)
        print('accuracy: '+str(accuracy))
        return loss, {"accuracy": 'accuracy: '+str(accuracy)}

    return evaluate
```

Finally, we run the server with main function `fl.server.start_server`. We run 20 rounds of federated learning (set in utils.py).

```
fl.server.start_server(
    "0.0.0.0:8080",
    strategy=strategy,
    config={"num_rounds": utils.load_parameter_C()[2]}
)
```

A.3 Utility Code

There are three main functions in our utility code: a first to load our data, a second to load model parameters and a third to perform evaluation.

The loading function loads and divides our data into a design matrix and target variables.

```
def load_credit_test_Default() -> Dataset:
    col_n_y = ['DEFAULT']
    y_test = pd.DataFrame(test, columns=col_n_y)
```

```
    col_n_X = [""" 27 features named here """]
    x_test = pd.DataFrame(test, columns=col_n_x)

    col_n_y_k = ['PROFIT_LOSS']
    y_test_p = pd.DataFrame(test, columns=col_n_y_k)

    test_all = pd.DataFrame(test)

    return (x_test, y_test, y_test_p, test_all)
```

The parameter-loading function can load the common parameters since they are shared by all clients.

```
def load_parameter_C():
    max_iter=10 # client iterations
    round=20 # server rounds

    return(max_iter, round)
```

Finally, we define our evaluation function.

```
def accuracy_evaluation(predict_all, test_y):
    accuray = np.mean(predict_all == rave(test_y))

    return accuracy
```

[1] Jacob Poushter et al. Smartphone ownership and internet usage continues to climb in emerging economies. *Pew research center*, 22(1):1–44, 2016.

[2] Monica Anderson. Technology device ownership: 2015. 2015.

[3] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[4] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[5] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[6] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[7] Federated Learning. Collaborative machine learning without centralized training data, 2017.

[8] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), jan 2019. ISSN 2157-6904. doi: 10.1145/3298981. URL `https://doi.org/10.1145/3298981`.

[9] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.

[10] Kang Loon Ng, Zichen Chen, Zelei Liu, Han Yu, Yang Liu, and Qiang Yang. A multi-player game for studying federated learning incentive schemes. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 5279–5281, 2021.

[11] Jason Silverstein. Hundreds of millions of facebook user records were exposed on amazon cloud server. *CBS News*, 2019. URL `https://www.cbsnews.com/news/millions-facebook-user-records-exposed-amazon-cloud-server/`.

[12] Benjamin Wong. Data privacy law in singapore: The personal data protection act 2012. 2017.

[13] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676, 2017.

[14] Lydia de la Torre. A guide to the california consumer privacy act of 2018. *Available at SSRN 3275571*, 2018.

[15] Xuefei Yin, Yanming Zhu, and Jiankun Hu. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[16] Timeline of cyber incidents involving financial institutions. *Carnegie Endowment for International Peace*, Updated 2021. URL `https://carnegieendowment.org/specialprojects/protectingfinancialstability/timeline`.

[17] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2006.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[19] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. Flower: A friendly federated learning research framework. *CoRR*, abs/2007.14390, 2020. URL `https://arxiv.org/abs/2007.14390`.