A Horizontal Federated Random Forest for Heart Disease Detection from Decentralized Local Data

Shafin Mahmud Jalal
Dept. of Computer Science and Engineering
Brac University
Mohakhali, Dhaka – 1212, Bangladesh
shafin.mahmud.jalal@bracu.ac.bd

Md. Ashfaqul Haque
Dept. of Computer Science and Engineering
Brac University
Mohakhali, Dhaka – 1212, Bangladesh
md.ashfaqul.haque@g.bracu.ac.bd

Md. Rezuwan Hasan
Dept. of Computer Science and Engineering
Brac University
Mohakhali, Dhaka – 1212, Bangladesh
md.rezuwan.hasan@bracu.ac.bd

Md. Golam Rabiul Alam
Dept. of Computer Science and Engineering
Brac University
Mohakhali, Dhaka – 1212, Bangladesh
rabiul.alam@bracu.ac.bd

Abstract— **In the modern world, reliable data is a thriving need in every sector. As the data increases, maintaining data privacy is also becoming a big concern. The healthcare sector is no different than that. Privacy in the healthcare sector is a topmost concern when sharing with other institutes. As data from a single healthcare institute is not always enough to get properly predicted outputs in machine learning approaches. There comes the idea of sharing data among multiple hospitals for having a more specified model with keeping the data details private. So, we have designed a model combining a federated central model and clients for the application of Federated Learning on heart disease patients' data. Here, we have implemented an approach for sharing only the model parameters among the clients and central in horizontal federated learning infused with random forest. At the evaluation of our model, we have come up with improved accuracy of 7.1, 2, and 6 percent respectively for the federated central and both clients.**

Keywords— Horizontal Federated Learning, Random Forest, Data Privacy, Heart Disease.

I. Introduction

With the advancement of Artificial Intelligence, our day-to-day lives have become much easier than ever before. Conventionally, We implement AI by collecting data from the real-world data centers, creating a model, training and preparing them with all the data, and then implementing it. The key reason behind the outstanding performance of these models is the huge amount of data that is used to train and prepare these systems. But the problem is that sometimes these data are not very organized and there's also a matter of confidentiality that is always a considerable criterion. As with the advancement of technologies, cyber threat has also increased as data is a valuable asset for any organization, let that be a business organization or a government organization. An organization can meet its demise if the right data falls into the hands of the wrong entity. These chunks of stored data are scattered across discrete domains creating data islands that cannot be shared. For this reason, organizations do not straightforwardly share data and have a high-security protocol for sharing confidential information. There are a lot of enacted legal codes regarding the protection and regulations of data. Not only organizations, but even individual people are also reluctant to share their data depending on the susceptibility of the information. [12] The European Union passed the General Data Protection Regulation (GDPR) in 2018, providing individuals more control over their data and mandating how businesses should use it transparently. Personal data should only be used with the individual's consent, and the organization's goals and intentions should be disclosed in advance. It's also important to remember that data is always changing and information is updated over time. This means along with the production of new data, stored data are constantly becoming obsolete. All these make data collection even more difficult as these AI models should always be up to date for the best performance. Maintaining data privacy while constantly updating these discretely located data on a server is a big issue and should be taken into account. To confront it, an approach named "Federated learning" [13,14,15] takes place. All these issues could be solved if there was a way to decentrally use these data with constant updates and the main idea of federated learning is decentralized learning where the user data is never sent to a central server as they do for conventional machine learning models. Federated learning is a collaborative and decentralized approach to machine learning that allows for smarter models, lower latency, and less power consumption all while maintaining data privacy and updates.

II. Literature Review

Federated learning has been applied with different approaches. One noble method is called federated forest where the CART tree has been applied with the bagging for both the classification and regression problems [3]. Federated learning has been applied to Meta-learning to implement a proposed recommender system [11], Solve multi-task problems [11], and a loss-based AdaBoost method was developed which has been in [8]. Peter et al. [1] introduced the idea of federated learning with many clients orchestrated under a central server with de-centralized data. [18] proposed a lossless federated random forest algorithm where all clients are jointly connected with each built tree and the split information about the features is stored in each client's nodes. The local models were outperformed by the method which had similar accuracy as the centralized model. A federated optimization scheme has been developed for support vector machines to predict future hospitalizations from EMR data shown in [9]. To predict mortality during ICU, stay, A federated autonomous deep learning (FADL) method was developed by Liu et al. [10]. Huang et al. [5] demonstrated that patient clustering can improve the efficiency of federated learning to predict mortality and stay time in the ICU.

There are three types of federated learning methods up to this day, Those are,

- Horizontal federated learning
- Vertical federated learning
- Federated transfer learning

Although the idea of federated learning starts with a model on a central server initially, in the next step, it is distributed to remote devices/domains called "clients" and trains the models in those devices with the data stored in those domains.

In horizontal federated learning, all the clients share the same feature space. Only the training gradients will be passed after a model is deployed to the clients and is locally trained and masked with encryption to preserve the privacy of each client [17]. After the training gradients are returned to the central server, The server then calculates the means of all the training gradients and updates the master model. After that, the newly updated model is again encrypted and deployed to the clients and the clients update their local models again with the latest model parameters[16]. In contrast with this in the vertical federated Learning same feature values are needed in all sites which is not always possible in healthcare sectors.

Let a set of hospitals located in the same city be considered, for example, These hospitals share the same feature sets but barely any common users. Horizontal federated learning can be implemented to pick up the training from the locally deployed models in the client domains by the central server model. Different machine learning algorithms can be integrated with horizontal federated learnings based on the implementation without changing the main framework.

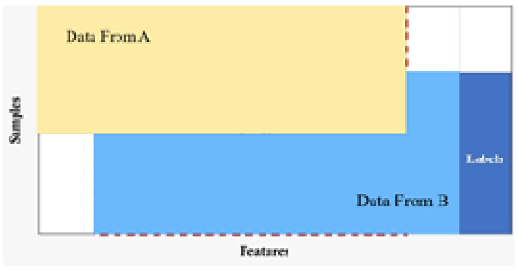


Fig. 1. Horizontal Federated Learning.

In this paper, we used the horizontal federated learning method with random forest to increase or bring stability to the performance.

III. Dataset

A. Dataset Description

The dataset we have taken here is on cardiovascular diseases which is a labeled one. The dataset here has been taken from The Nature article listing the heart disease dataset [2]. Here, label 1 defines there is heart disease and 0 means normal. The dataset has a total of 1190 samples containing info of patients from five hospitals. Here are the five hospitals which are Cleveland, Hungarian, Switzerland, Long Beach, and Statlog containing respectively 303, 294, 123, 200, and 270 samples respectively.

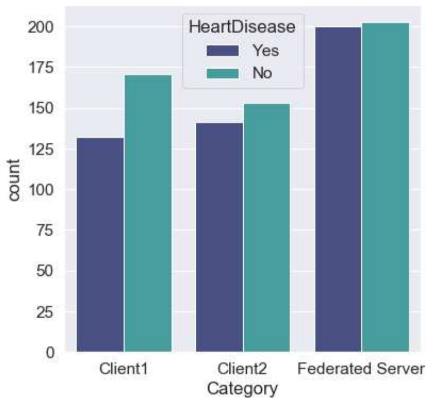


Fig. 2. Heart disease distribution comparison on the federated server and both clients.

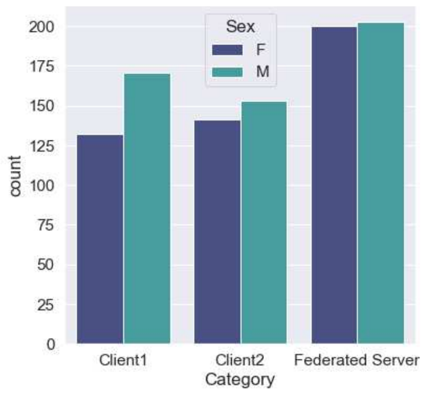
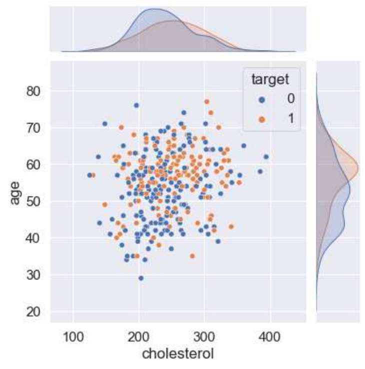
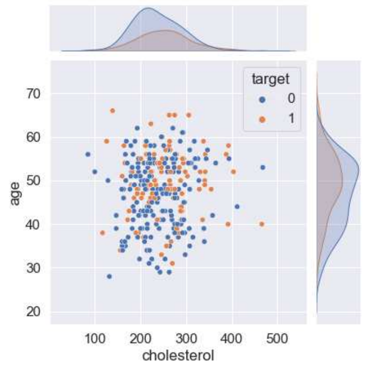


Fig. 3. Gender distribution comparison on the federated server and both clients.

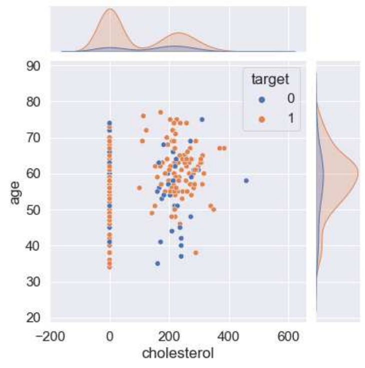
Here in our work, we have defined Cleveland and Hungarian as client 1 and client 2 respectively, and combined the rest three as a federated central server. In fig. 2 the target, if there is heart disease (1) or not (0), in all our sites, shows that the data is in a balanced state. Then in fig. 3, the male percentage in client1 is a little high and the other two sites are in a balanced state.



(a)

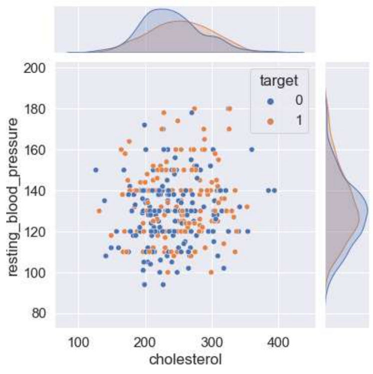


(b)

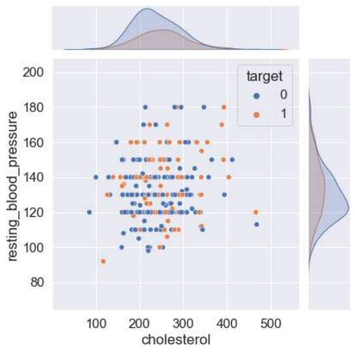


(c)

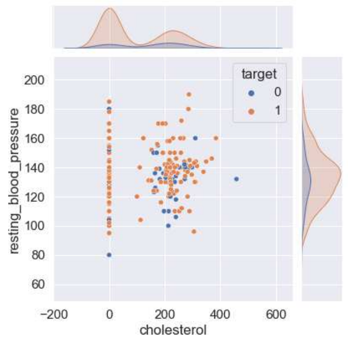
Fig. 4. Age versus Cholestorel in terms of Heart Disease (a) Federated Central, (b) Client1, (c) Client2.



(a)



(b)



(c)

Fig. 5. Blood Pressure Versus Cholstorel Plot in terms of Heart Disease (a) Federated Central, (b) Client1, (c) Client2.

B. *Outlier Removal*

Here fig. 4 and fig. 5 show respectively age distribution versus cholesterol and resting blood pressure versus cholesterol in terms of the target class. These plots show that there are some outliers in all the server datasets. So, we have removed these outliers before starting work by calculating the z-score.

IV. Methodology

A. *Proposed Methodology*

The implementation of federated learning with machine learning techniques has been applied here for maintaining the privacy of the patients' data sharing among different hospitals due to an approach towards a better healthcare system which has been brought to light with some preprocessed steps. Here, the data we have used in both the client and the federated server we have seen from the fig. 6 that there are some outliers, so we have removed those outliers and Algorithm-1 depicts the workflow of fig. 6 methodology. Now, As the client and federated servers have similarities in parameter columns, we have approached the horizontal federated learning infused with the Random Forest classifier. The approach we have taken here is such a way that we have started our learning from the central federated servers and traversed through the client and ended by returning to the federated server while sharing only the model parameters in the process keeping the data private to each local site. Starting from the federated central server we have gone through hyperparameters tuning and then taken the best model parameters from the existing data. In our case, we have utilized the data of the central server and tuned the model on that, and then send the parameters to the client and observed the performance. Then again tune the model on the clients and send the parameters to the federated central server. We have created two helper functions to select the parameters that come from the clients. These helper functions have been depicted in Algorithm-2. These functions help to choose non-repetitive parameters from the clients. After the model parameters from the clients have been selected, then they are used in the central server model for further optimization of the federated central server. Then this optimal model has been used to evaluate the performance in all sites and compared its performance to the previous ones. As for being an online data sharing among the clients and central, at each step there is some new data to be entered and a rise in the data on all sites has happened. In this step, we have expanded the training data by accumulating all the new and previous data on both the federated central server and the clients and concluded our analysis by repeating the procedure of tuning the models by transferring parameters. Here on each step, we have split 75 percent of the used portion of the dataset as training and the other 25 percent as the testing.

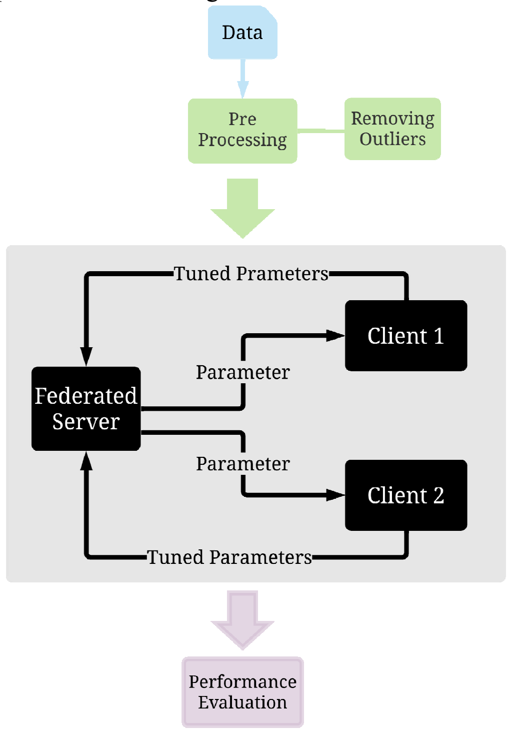


Fig. 6. Proposed Methodology.

B. *Random Forest*

Random forest is an algorithm consisting of a bunch of decision trees and at least three decision trees are required to call the group a forest. This exploits a technique named “Ensemble learning” that provides complex problem solutions by combining multiple classifiers. Random forest methods can be used for both regression and classification tasks. Random forests perform better than individual decision trees but less than gradient boosted trees.

The random forest creates multiple decision trees called “instances” and the idea is to collect the output from each of the decision trees and then the mode is reckoned from those results derived from each of the decision trees and then the final output of the random forest itself named as “Final class”.

Algorithm 1 Federated Random Forest with hyperparameter tuning

```
    Federated To client (Initial)
        1.  Data Preprocessing
        2.  Implementing Random Forest
        3.  saving the initial model parameters
    Client-side
        4.  importing/loading federated global parameters
        5.  applying Random forest with only global parameters
        6.  improving the model further by hyperparameter
            tuning
        7.  saving the model parameters
    Client to federated
        8.  importing/loading clients' parameters
        9.  applying helper functions to receive non-repetitive
            parameters
        10. applying Random forest with only the client's non-
            repetitive parameters
        11. improving the model further by hyperparameter
            tuning
        12. saving the model parameters(global)
13. Repeat 4-12 until the target requirement is achieved
```

Algorithm 2 Helper Functions

```
Helper Function 1:
    Defining a function for eliminating repetitive parameters
    Function parameter: parameter-list
        a.  Create an empty-list
        b.  for every element in parameter-list:
                i.  only accept the non-repetitive parameter
        c.  end for.
Helper Function 2:
    Defining a function to create a dictionary from non-repetitive
    parameters.
    Function Parameters: client-models
        a.  Create an empty-dictionary
        b.  Create an empty list for storing all clients'
            parameters
        c.  for each model on a range of model-list:
                i.   Load client-models
                ii.  Append the loaded models' parameters on
                     the empty list
                iii. Extracting each parameter from the list
                     and creating individual lists for each
                     parameter
                iv.  Creating a key for each parameter,
                     applying the no-repeat function on each
                     parameters list, stored it as the value of
                     each key with each parameters list
        d.  end for.
```

C. *Hyperparameter Tuning*

Hyperparameter tuning is a fundamental prerequisite for the application of any machine learning algorithm, Especially for any optimization algorithm. Hyperparameter tuning is a process of selecting a set of parameters or model arguments that are manually set before fitting the model or beginning the learning process.

Selecting the hyperparameter value to get the maximum output from a model or the best version of a model is called hyperparameter tuning or hyperparameter optimization.

If we have a lot of experience figuring out what the ideal hyperparameter combination is or if we have a lot of domain knowledge about the model type, manually configuring the hyperparameter settings can be highly useful. Otherwise, it may end up being a time-consuming effort with poor outcomes.

Using some automated methods to set the hyper tuning settings would be the optimal approach in this case. Two simple automatic hyperparameter tuning algorithms are grid search and random search [4]. In addition to the above two, there are other hyperparameter tuning techniques. Other optimization methodologies which are worth mentioning are Bayesian optimization, gradient-based optimization, and evolutionary optimization. In this paper, the grid search approach was used. Grid search is used to select a range of values for each hyperparameter that has to be tweaked. The continuous-range hyperparameters must be discretized. The grid search then goes through all of the optimal hyperparameter combinations at the same time, trains and evaluates them all, and then selects the hyperparameter combination that results in the best optimal output. The grid search looks over all of the hyperparameter space for the optimal hyperparameter values within the given ranges. One of the advantages of grid search is this.

V. Experimental Evaluation

As we have portrayed a system with data up-gradation, we have presented the conducted experiments in two phases where Random forest is the base machine Learning model. Here in 1st phase, we have gone through three spheres of Procedures.

A. *Phase 1*

(a) Federated Central to Clients:

In this first step, we have tuned the central data of the federated central server and determined the best hyperparameters from it. Then send the data to the clients. On the clients, we have applied the received parameters and trained the model based on that to evaluate performance. The evaluation metrics we have put here are Accuracy (Acc), Precision (Prec), Recall (Rec), and f1-score (f1) which are in table 1.

TABLE I. Evaluation of Initial Data (Federated Central to Clients)

	Sample size	**No. of Estimators**	**Acc**	**Prec**	**Rec**	**F1**
Federated Central	463	11	81.29	87.75	68.25	76.78
Client 1	230	11	86.21	73.07	95	82.61
Client 2	232	11	86.21	85.71	100	92.31

(b) Clients to Federated Central:

In this step, we have turned the model on local data and sent the model parameters from each client to the central, and evaluated model performance based on the client parameters. The Accuracy (Acc), Precision (Prec), Recall (Rec), and f1-score (f1) are in table 2.

TABLE II. Evaluation Of Initial Data (Clients to Federated Central)

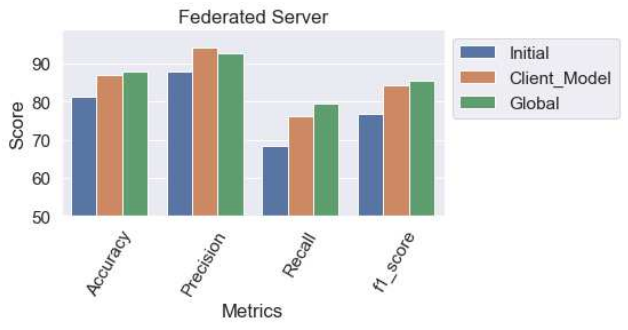
	Sample size	**No. of Estimators**	**Acc**	**Prec**	**Rec**	**F1**
Client 1	230	85	87.93	74.07	100	85.11
Client 2	232	35	89.65	90.38	97.92	94
Federated Central	463	35	87.05	94.12	76.19	84.21

(c) Federated Central to Clients:

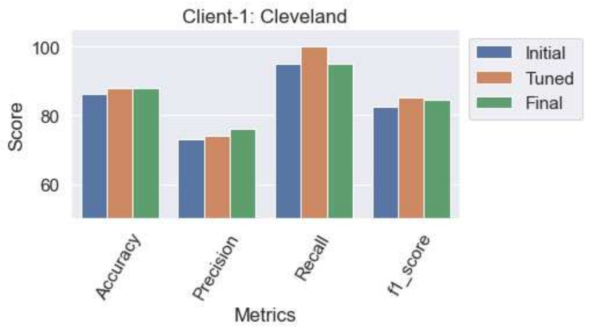
Now using the best parameters of the clients' model, we have again turned the model of the central server and shared the generalized model of the central server with the clients, and evaluated performance.

TABLE III. Evaluation Of Initial Data (Federated Central to Clients)

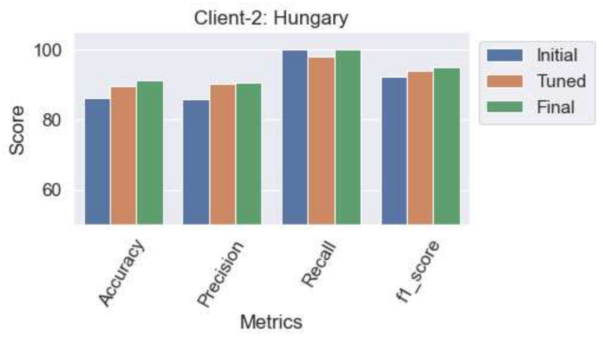
	Sample size	**No. of Estimators**	**Acc**	**Prec**	**Rec**	**F1**
Federated Central	463	30	87.77	92.59	79.36	85.47
Client 1	230	30	87.93	76	95	84.44
Client 2	232	30	91.38	90.57	100	95.05



(a)



(b)



(c)

Fig. 7. Performance Evaluations of (a) Central Server, (b) Client1, and (c) Client2.

In Fig. 7 we have plotted the performance comparison in each of the servers and how the model has been upgraded with each step of sharing the model. Here we can observe that in the central server from the 1st step to the last one there is a significant rise in all the evaluation metrics. In the client1 there is a rise in the accuracy and f1-score which depicts stability. And in client2 there is also a significant rise in accuracy and f1-score. So, this evaluation plotting here shows the stability of our model. So, we have ended up with improved accuracy of 7.1% in the federated central server, 2% in client 1, and 6% in client 2. In addition to that, we have achieved an f1-score improvement of 9.6% in the federated central server, 2.2% in client 1, and 2.9% in client 2.

B. *Phase 2 (Federated Central to Clients with Added Samples:)*

Now the model has been again tuned and trained on the federated central data as the data has increased. Newly updated parameters from the center have been sent to the clients and model performance has been evaluated after training the model utilizing the new parameters on the updated data of the clients. The Accuracy (Acc), Precision (Prec), Recall (Rec), and f1-score (f1) are in table 4.

TABLE IV. EVALUATION OF UPDATED DATA (FEDERATED CENTRAL TO CLIENTS)

	Sample size	**No. of Estimators**	**Acc**	**Prec**	**Rec**	**F1**
Federated Central	579	35	92.41	95.08	87.88	91.34
Client 1	288	35	97.22	96	96	96
Client 2	291	35	90.41	89.55	100	94.49

VI. CONCLUSIONS

We have approached federated learning in a simple way while maintaining data privacy using Random Forest as it's a robust technique for minimizing the overfitting issue. Here, with the work of sharing model in the health care sector, there is a possibility of new data flow. So, we have to keep that in mind while evaluating our proposed process. Here we have done our implementation only on heart disease patients' data but we can apply our model in other areas of the healthcare sector where patients' data from one institute would be beneficial to others and other sectors where data sharing is beneficiary. In our approach, we have got a balanced and improved performance in terms of accuracy and f1-score in every step while sharing models both in federated central servers and clients. At the last presentation of the evaluation metrics for a central server we have got stable performance in terms of accuracy and f1-score.

REFERENCES

[1] Peter Kairouz et al., 'Advances and Open Problems in Federated Learning', 2021.

[2] Manu Siddhartha, "Heart Disease Dataset (Comprehensive)", *IEEE Dataport*, November 5, 2020.

[3] Yang Liu, Yingting Liu, Zhijie Liu, Junbo Zhang, Chuishi Meng, and Yu Zheng, 'Federated Forest', *ArXiv*, 2019, abs/1905.10053.

[4] Philipp Probst, Marvin N. Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. WIREs Data Mining and Knowledge Discovery, 9(3), 2019.

[5] Li Huang, Andrew Shea, Huining Qian, Aditya Masurkar, Hao Deng, and Dianbo Liu. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. Journal of Biomedical Informatics, 99:103291, 2019.

[6] Hankz Hankui Zhuo, Wenfeng Feng, Qian Xu, Qiang Yang, and Yufeng Lin. 2019. Federated Reinforcement Learning.

[7] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He, 'Federated Meta-Learning for Recommendation', 2018.

[8] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. 2018. LoAdaBoost: Loss-Based AdaBoost Federated Machine Learning on medical Data.

[9] Theodora Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. International Journal of Medical Informatics, 112, 2018.

[10] Dianbo Liu, Timothy Miller, Raheel Sayeed, and Kenneth Mandl. Fadl:federatedautonomous deep learning for distributed electronic health record. arXiv:1811.11400, 2018.

[11] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. 2017. Federated Multi-Task Learning.

[12] General Data Protection Regulation. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46. Official Journal of the European Union (OJ) 59, 1-88 (2016), 294.

[13] J. Konecny, H. B. McMahan, D. Ramage, and P. Richtárik, 'Federated Optimization: Distributed Machine Learning for On-Device Intelligence', *CoRR*, abs/1610.02527, 2016.

[14] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, 'Federated Learning: Strategies for Improving Communication Efficiency', *CoRR*, 1610.02527, 2016.

[15] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüeray Arcas. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv:cs.LG/1602.05629.

[16] Henning Best and Christof Wolf. The SAGE Handbook of Regression Analysis and Causal Inference. SAGE Publications Ltd, 2014.

[17] Christopher Bishop. Pattern Recognition and Machine Learning. Springer, 2007.

[18] Cynthia Dwork. Differential privacy. Automata, Languages and Programmin, 4052:1– 12, 2006.