International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET207644

Early Detection of Type-2 Diabetes Using Federated Learning

M. Lincy, Dr. A. Meena Kowshalya
Department of Computer Science and Engineering, Government College of Technology, Coimbatore, Tamil Nadu, India

ABSTRACT

Article Info
Volume 7 Issue 6
Page Number: 257-267
Publication Issue :
November-December-2020

Article History
Accepted : 15 Dec 2020
Published : 30 Dec 2020

Data privacy and security are incredibly important in the healthcare industry. Federated learning is a new way of training a machine learning algorithm using distributed data which is not hosted in a centralized server. Numerous centralized machine learning models exists in literature but none offers privacy to users' data. This paper proposes a federated learning approach for early detection of Type-2 Diabetes among patients. A simple federated architecture is exploited for early detection of Type-2 diabetes. We compare the proposed federated learning model against our centralised approach. Experimental results prove that the federated learning model ensures significant privacy over centralised learning model whereas compromising accuracy for a subtle extend.

Keywords Federated learning, decentralized model, Differential Privacy, Feature Selection, Type 2 diabetes

I. INTRODUCTION

Most of the machine learning algorithms follow a centralized approach that exploits a server for training and prediction. This infrastructure facilitates easy maintenance but poses few disadvantages such as leakage of data if the server alone is compromised, lack of privacy and difficulty in handling very large datasets. To overcome the above-mentioned drawbacks, a decentralised learning can be opted. Decentralised learning involves two components, namely client and server. The server communicates and coordinates with multiple clients in sharing and training the machine learning model. This technique avoids the restriction on requirement capacity of the server and also reduces the exposure of user data to the server. One such decentralised machine learning technique which is becoming very popular nowadays is Federated Learning. It facilitates to learn a prediction model distributed across many clients from the server. The federated learning process is shown in Figure 1.

Federated Learning Process

The federated learning approach is primarily opted in situations where there is a need to train models on larger datasets on their own or in applications where privacy of data is of greater concern than the accuracy of the model. The applications so far where federated learning has been used are Google Gboard, Healthcare, Mobile computing and in Credit Card fraud detection.

Data privacy and security are the central factors focused in the healthcare industry. It is the widely focused domain with respect to federated learning approach, because healthcare data comprises of sensitive and valuable patient details like patient name, disease and diagnosis and other health related factors.

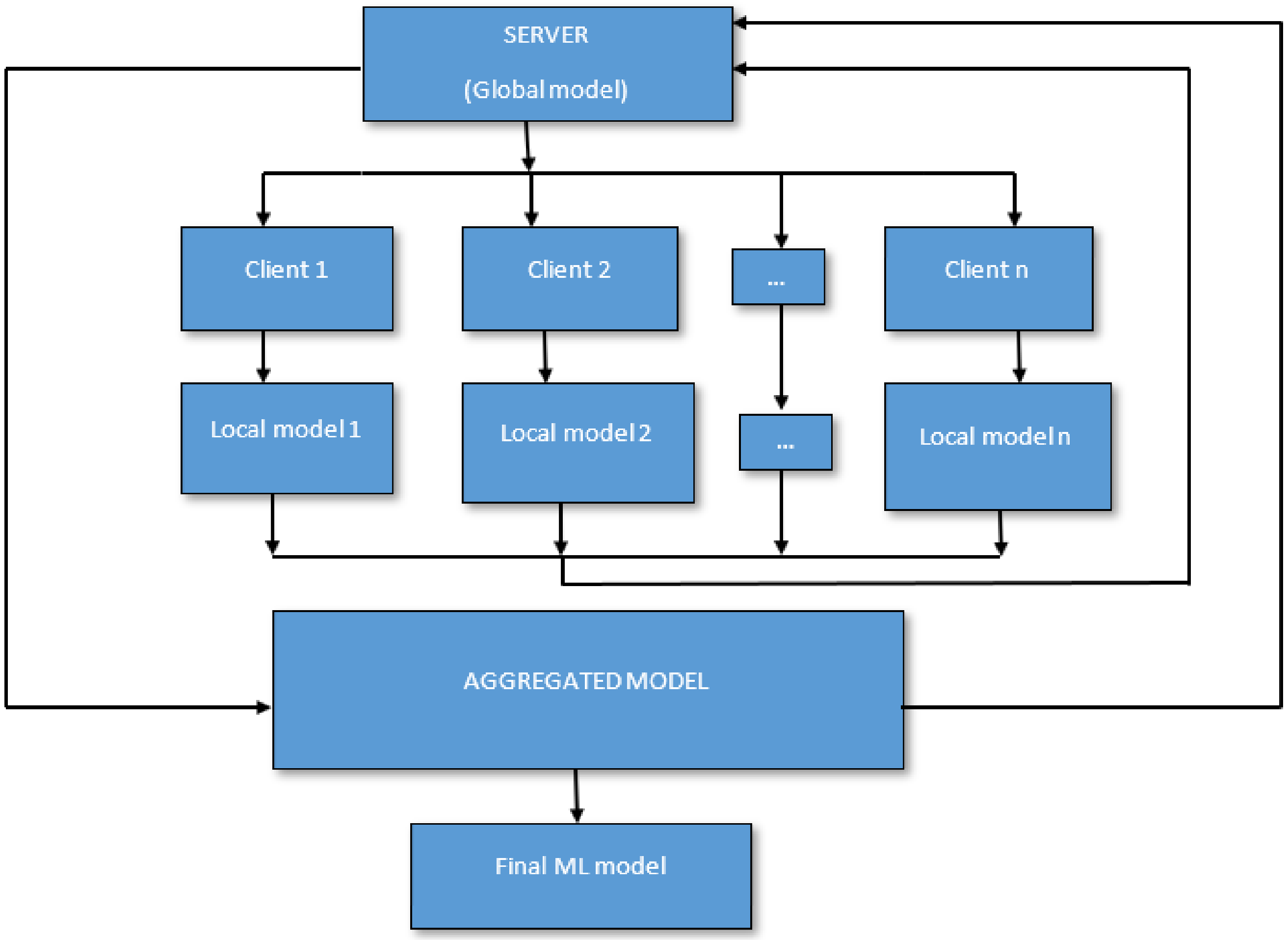


Figure 1 Federated learning process

Federated Learning offers the following advantages

- Data Security and Privacy - model training happens only on federated clients and hence the sensitive information when stored in central server is less prone to hacking.
- Real Time Prediction - prediction happens on the client device and hence the latency due to transmitting input back to a central server and then shipping results back to the device is minimized.
- Offline prediction - Since the ML models are present on the client device, the predictions work even when there is no internet connection available.

The server contains the machine learning model. It handles the process of distributing the model to multiple clients and training it on local datasets that reside in the clients. At every iteration, each client sends its trained model to the server. The server constructs an aggregated model from the locally trained models. This aggregated model is distributed again to the client devices. It can be noted that the server has no direct access to the user's data which ensures privacy. This process is repeated in every iteration and each time the constructed model is improvised with user's features. A final federated machine learning model is obtained after required number of iterations. In a centralised model, privacy is not preserved whereas in a federated model, differential privacy is assured because the data remains private as it is confined within the client devices. This paper proposes a federated learning framework for early prediction of Type-2 Diabetes.

The major contributions of this paper are:

1. Implementation of federated learning approach for early detection of Type-2 Diabetes.
2. Comparing centralized feature selection and prediction model with decentralised model of federated feature selection and prediction to highlight the privacy advantages of federated learning approach.

The rest of the paper is organised as follows. Section 2 discusses the proposed work and section 3 discusses the experimental results followed by conclusion.

II. FEDERATED LEARNING BASED EARLY PREDICTION OF TYPE-2 DIABETES

Federated Learning is a decentralised learning technique where the machine learning process is distributed over to the local devices. It helps the local devices to collaboratively learn a shared model by utilizing the local training data. This ensures that the local data remains private and restricted from any kind of access. Federated Learning allows for faster deployment and testing of smarter models, lower latency and less power consumption and also ensures privacy.

We for the first time exploit Federated Learning approach for the early detection of Type-2 Diabetes which allows client devices to train a shared global model without sharing raw training data. The initial set of features undergo feature selection process and the selected set of features form the training data involved in the federated learning process. The Pima Indian Diabetes datasets with 768 instances and 2000 instances are used for experimental purpose. The feature selection methods used are Mean based Feature Selection (MbFS), Correlation based Feature Selection (CbFS) and Recursive Elimination based Feature Selection (REbFS). The final prediction is made using Federated MultiLayer Perceptron method.

In Mean based Feature Selection (MbFS) method, XGBoost classifier is used which is a gradient boosted decision tree. The feature selection process involves two parameters, namely the Feature Importance value and Threshold value. The threshold value obtained is 0.124 which results in an optimal feature set. In Correlation based Feature Selection (CbFS) method, the features are expanded as subsets with the size of the subsets ranging from one feature up to n features. Each feature subset is evaluated and ranked based on a performance metric (roc_auc score) and the subset with the highest value of the metric is returned as the best correlated subset of features. The corresponding roc_auc score obtained is 0.97. Recursive Elimination based Feature Selection (REbFS) method is a backward feature removal technique which removes features, one after the other and builds a model on the remaining features. During each feature elimination process, a corresponding accuracy is computed. This is an iterative process until the best set of features that highly contribute to the prediction of target variable is obtained. The accuracy is calculated using R-squared and the score obtained is 0.38.

The selected set of features form the training data that is to be fed into the neural network model. The neural network model used is Multilayer Perceptron (MLP) [10]. The prediction process is carried out in a federated manner. The training data and the MLP model are distributed to the client workers where local training of the model is performed. The client workers are created in a virtual environment. After every iteration the local model is sent to the server where it is updated and again distributed to all the workers. A final model is obtained at the server by performing an aggregation function which is the federated average function. This function calculates the federated average of a list of models passed to it. The prediction is performed by evaluating the test data.

2.1 Federeated Learning Algorithm

By running the feature selection algorithms, we obtain the selected subset of features. These features are fed into the federated MLP model. The baseline steps are illustrated in the following algorithm.

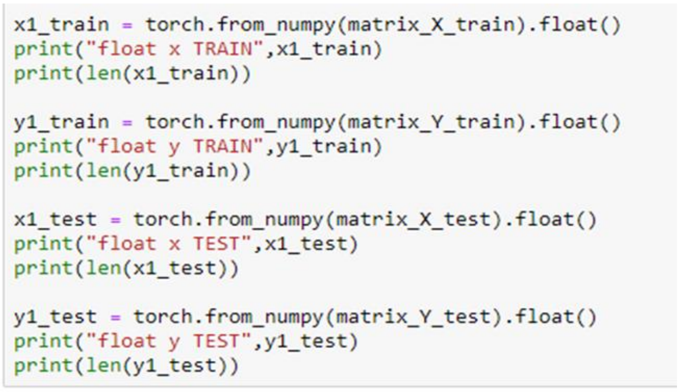
```
Begin

Convert the set of features as tensors

Initialise the MLP model parameters

Create virtual client workers

Connect the server to the client workers

Build and load the subset of features along with
target values, as

federated datasets

Send the MLP model to the workers

for each worker

Train the model at the workers using its local dataset

Send all the trained models to the server after every
epoch

Prediction is made and corresponding loss is
calculated by (1) and

a new updated model is build

Distribute the updated model to all the workers in the
next epoch

Repeat the process until required number of epochs

Evaluate the test data using the final federated MLP
model where

accuracy is calculated by (2)

End
```

The feature values are converted to tensor using PyTorch as shown in figure 2. The parameters of MultiLayer Perceptron model namely number of epochs, learning rate, batch size, number of input, output and hidden layers, number of input, output and hidden neurons should be initialised. The virtual workers are created and connected to the server using PySyft functionality sy.TorchHook(torch). The subset of features with target values are built and loaded as federated datasets using PyTorch functionality TensorDataset() and DataLoader() as shown in figure 3 and 4. Mean Squared Error loss is calculated during training as

MSE loss = (input-target) **2.mean() (1)

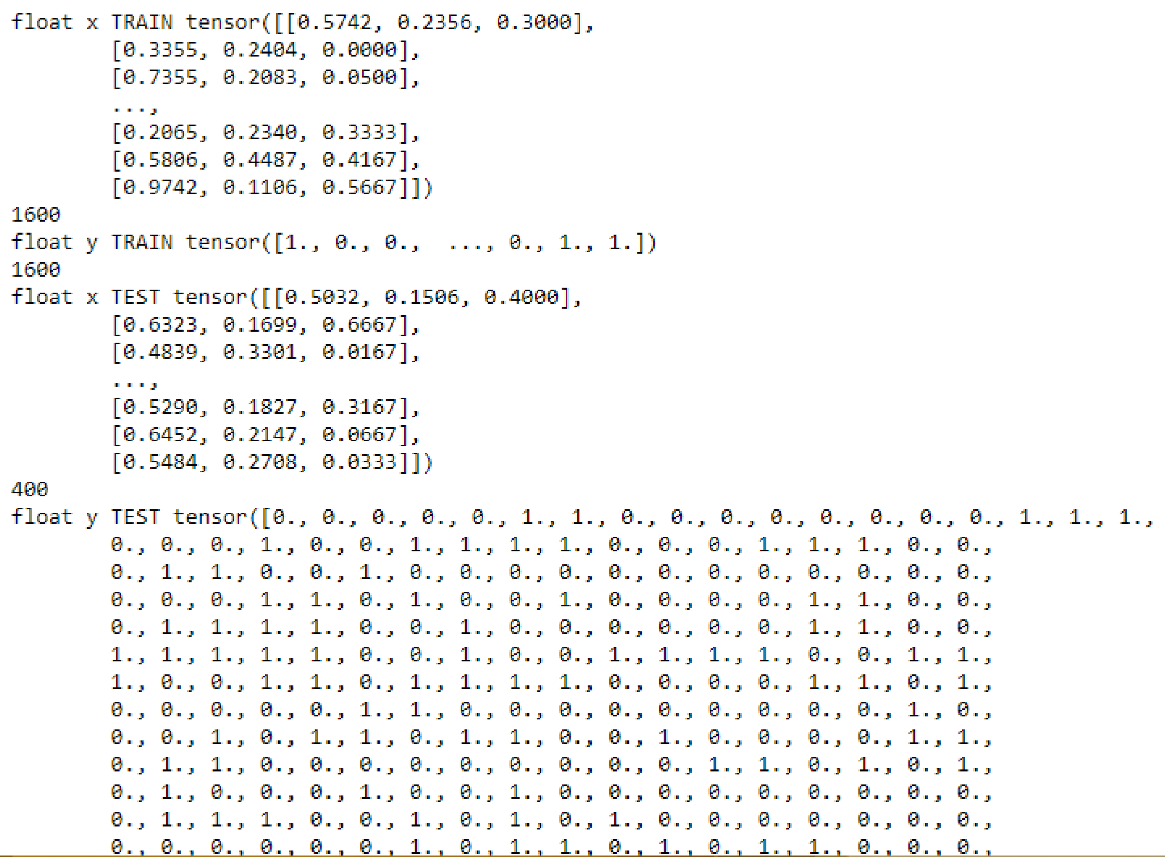
The accuracy is calculated as

Accuracy = ((predicted test data) / (actual test data)) (2)



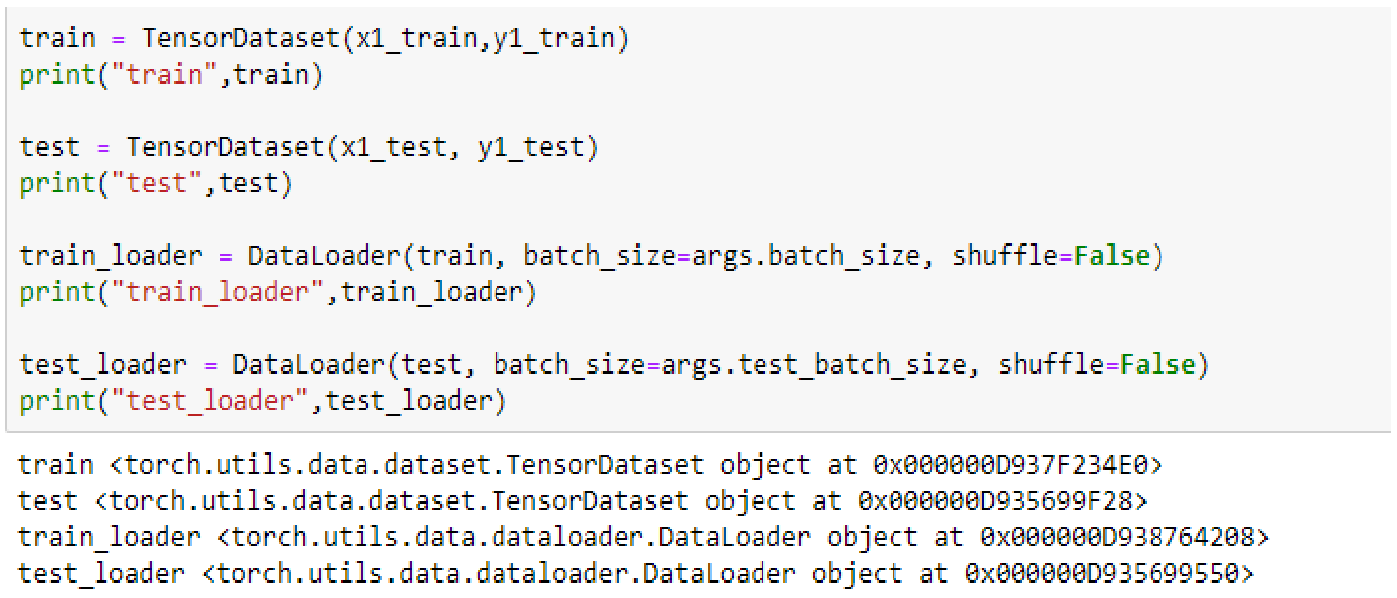
```
x1_train = torch.from_numpy(matrix_X_train).float()
print("float x TRAIN",x1_train)
print(len(x1_train))

y1_train = torch.from_numpy(matrix_Y_train).float()
print("float y TRAIN",y1_train)
print(len(y1_train))

x1_test = torch.from_numpy(matrix_X_test).float()
print("float x TEST",x1_test)
print(len(x1_test))

y1_test = torch.from_numpy(matrix_Y_test).float()
print("float y TEST",y1_test)
print(len(y1_test))
```

Figure 2 Converting to tensors using PyTorch



```
float x TRAIN tensor([[0.5742, 0.2356, 0.3000],
        [0.3355, 0.2404, 0.0000],
        [0.7355, 0.2083, 0.0500],
        ...,
        [0.2065, 0.2340, 0.3333],
        [0.5806, 0.4487, 0.4167],
        [0.9742, 0.1106, 0.5667]])
1600
float y TRAIN tensor([1., 0., 0.,  ..., 0., 1., 1.])
1600
float x TEST tensor([[0.5032, 0.1506, 0.4000],
        [0.6323, 0.1699, 0.6667],
        [0.4839, 0.3301, 0.0167],
        ...,
        [0.5290, 0.1827, 0.3167],
        [0.6452, 0.2147, 0.0667],
        [0.5484, 0.2708, 0.0333]])
400
float y TEST tensor([0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1.,
        0., 0., 0., 1., 0., 0., 1., 1., 1., 1., 0., 0., 0., 1., 1., 1., 0., 0.,
        0., 1., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0.,
        0., 1., 1., 1., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0.,
        1., 1., 1., 1., 1., 0., 0., 1., 0., 0., 1., 1., 1., 1., 0., 0., 1., 1.,
        1., 0., 0., 1., 1., 0., 1., 1., 1., 1., 0., 0., 0., 0., 1., 1., 0., 1.,
        0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
        0., 0., 1., 0., 1., 1., 0., 1., 1., 0., 0., 1., 0., 0., 0., 0., 1., 1.,
        0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 1., 0., 1.,
        0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 1., 1., 1., 0., 0., 1., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 1., 0., 1., 1., 0., 1., 0., 1., 1., 0., 0., 0.,
```

Figure 3 Train and Test data as Tensors

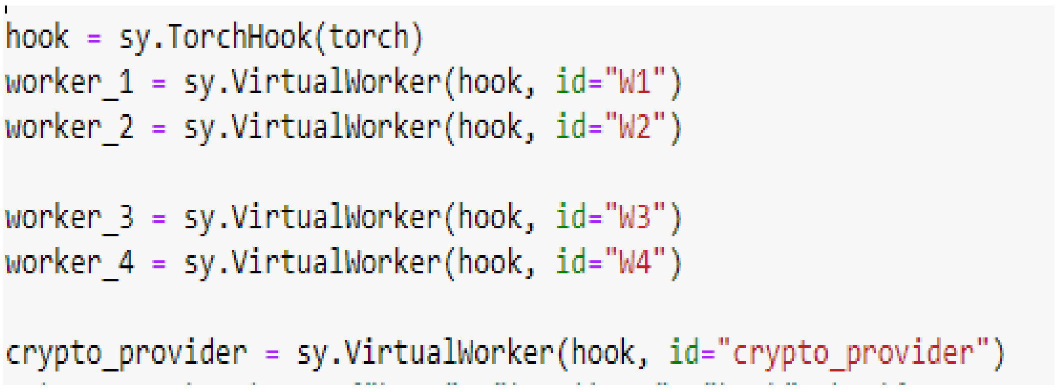


```
train = TensorDataset(x1_train,y1_train)
print("train",train)

test = TensorDataset(x1_test, y1_test)
print("test",test)

train_loader = DataLoader(train, batch_size=args.batch_size, shuffle=False)
print("train_loader",train_loader)

test_loader = DataLoader(test, batch_size=args.test_batch_size, shuffle=False)
print("test_loader",test_loader)
```

```
train <torch.utils.data.dataset.TensorDataset object at 0x000000D937F234E0>
test <torch.utils.data.dataset.TensorDataset object at 0x000000D935699F28>
train_loader <torch.utils.data.dataloader.DataLoader object at 0x000000D938764208>
test_loader <torch.utils.data.dataloader.DataLoader object at 0x000000D935699550>
```

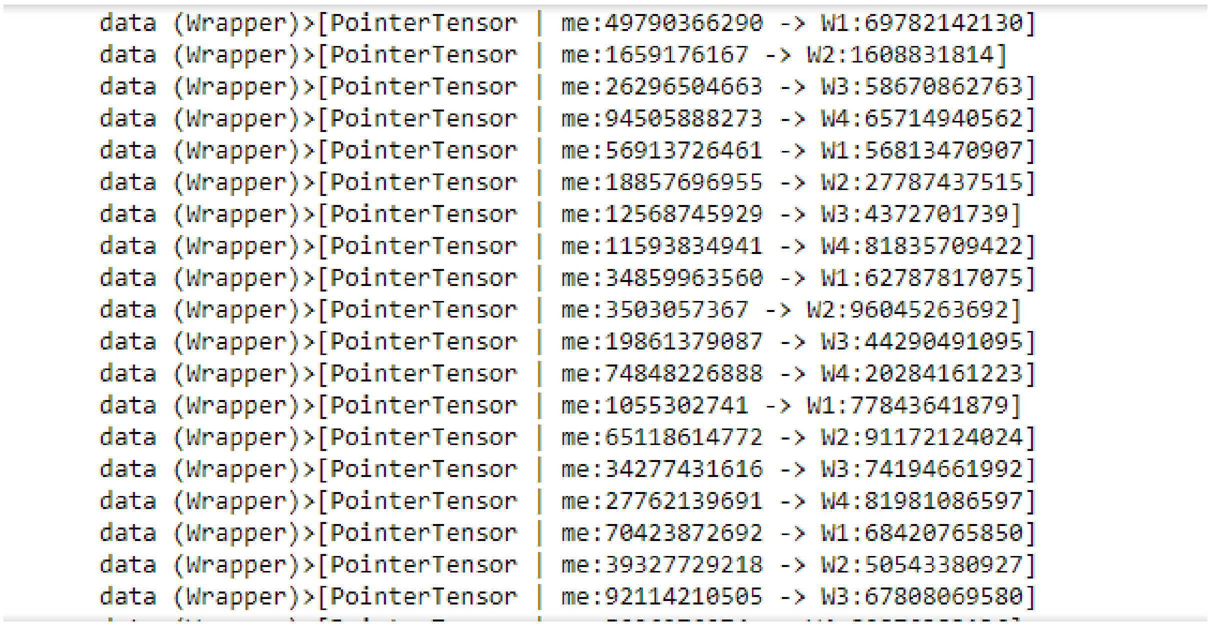
Figure 4 Building and Loading of train and test data

Figures 5, 6 and 7 illustrates the process of creating workers, sharing and assignment of load to workers.



```
hook = sy.TorchHook(torch)
worker_1 = sy.VirtualWorker(hook, id="W1")
worker_2 = sy.VirtualWorker(hook, id="W2")

worker_3 = sy.VirtualWorker(hook, id="W3")
worker_4 = sy.VirtualWorker(hook, id="W4")

crypto_provider = sy.VirtualWorker(hook, id="crypto_provider")
```

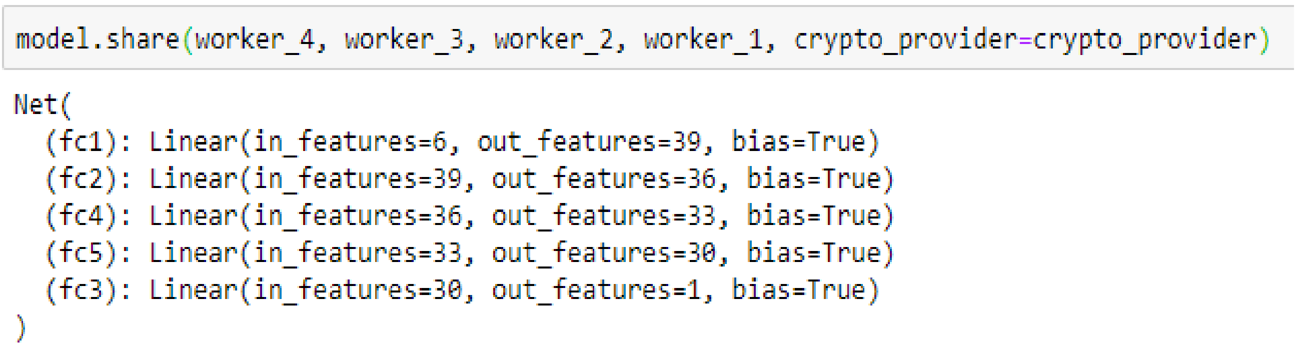
Figure 5 Creating virtual workers using PySyft



```
data (Wrapper)>[PointerTensor | me:49790366290 -> W1:69782142130]
data (Wrapper)>[PointerTensor | me:1659176167 -> W2:1608831814]
data (Wrapper)>[PointerTensor | me:26296504663 -> W3:58670862763]
data (Wrapper)>[PointerTensor | me:94505888273 -> W4:65714940562]
data (Wrapper)>[PointerTensor | me:56913726461 -> W1:56813470907]
data (Wrapper)>[PointerTensor | me:18857696955 -> W2:27787437515]
data (Wrapper)>[PointerTensor | me:12568745929 -> W3:4372701739]
data (Wrapper)>[PointerTensor | me:11593834941 -> W4:81835709422]
data (Wrapper)>[PointerTensor | me:34859963560 -> W1:62787817075]
data (Wrapper)>[PointerTensor | me:3503057367 -> W2:96045263692]
data (Wrapper)>[PointerTensor | me:19861379087 -> W3:44290491095]
data (Wrapper)>[PointerTensor | me:74848226888 -> W4:20284161223]
data (Wrapper)>[PointerTensor | me:1055302741 -> W1:77843641879]
data (Wrapper)>[PointerTensor | me:65118614772 -> W2:91172124024]
data (Wrapper)>[PointerTensor | me:34277431616 -> W3:74194661992]
data (Wrapper)>[PointerTensor | me:27762139691 -> W4:81981086597]
data (Wrapper)>[PointerTensor | me:70423872692 -> W1:68420765850]
data (Wrapper)>[PointerTensor | me:39327729218 -> W2:50543380927]
data (Wrapper)>[PointerTensor | me:92114210505 -> W3:67808069580]
```

Figure 6 Sending data batch wise to workers

Differential privacy is applied to model outputs to guarantee that the model remains inaccessible to any client worker that tries to intercept its data. This involves compromising model accuracy. MultiParty Computation (MPC) is a method where multiple parties can run a computation in which the parties provide a set of inputs and this input remains private to other parties except the provider party. Here crypto_provider is used that provides crypto primitives needed during the secret sharing of the model and data. The model.share() functionality used at the server helps to distribute the model to the workers without revealing the weights and other sensitive information the model holds.



```
model.share(worker_4, worker_3, worker_2, worker_1, crypto_provider=crypto_provider)
```

```
Net(
  (fc1): Linear(in_features=6, out_features=39, bias=True)
  (fc2): Linear(in_features=39, out_features=36, bias=True)
  (fc4): Linear(in_features=36, out_features=33, bias=True)
  (fc5): Linear(in_features=33, out_features=30, bias=True)
  (fc3): Linear(in_features=30, out_features=1, bias=True)
)
```

Figure 7 secret sharing of MLP model to workers

III. EXPERIMENTAL RESULTS

Python libraries namely sklearn, numpy, Tensorflow, PySyft and PyTorch are used for experimental purpose. Sklearn consists of a variety of tools for statistical modelling of data. Numpy is a fundamental package for scientific computing in python. Tensorflow helps to build and train machine learning models. PySyft is a library for secure, encrypted and privacy preserving deep learning. It decouples private data from model training. PyTorch is an open source machine learning library based on the Torch library. It provides a deep learning research platform with maximum flexibility and speed.

For experimentation, the Pima Indian Diabetes (PID) dataset with 768 instances [20] and with 2000 instances [21] are used. The dataset consists of 8 feature variables namely Pregnancies, Glucose, Blood Pressure, Skin thickness, Insulin, Body Mass Index, Diabetes Pedigree Function, Age and 1 target variable Outcome.

The Mean based Feature Selection algorithm (MbFS) resulted in 3 feature subsets namely Glucose, BMI, Age. The Correlation based Feature Selection (CbFS) yielded 6 feature subsets namely Pregnancies, Glucose, BloodPressure, BMI, DiabetesPedigreeFunction, and Age. The Recursive Elimination based Feature Selection (REbFS) yielded 4 feature subsets namely Pregnancies, Glucose, BMI, DiabetesPedigreeeFunction. A final classification is performed using Federated Multilayer Perceptron classifier (Fed MLP) assuring data privacy and the corresponding accuracy results are tabulated in Table 2. The detailed algorithmic explanation and implementation of these algorithms can be found in our previous paper [10].

Table 1 MLP classifier accuracy (Centralized model)

ALGORITHM	**ACCURACY USING MLP CLASSIFIER**	
	Dataset 1	**Dataset 2**
Mean based Feature Selection(MbFS)	80%	79%
Correlation based Feature Selection(CbFS)	80%	80%
Recursive Elimination based Feature Selection(REbFS)	64%	75%

Figure 8 Feature Selection algorithms vs MLP classifier accuracy

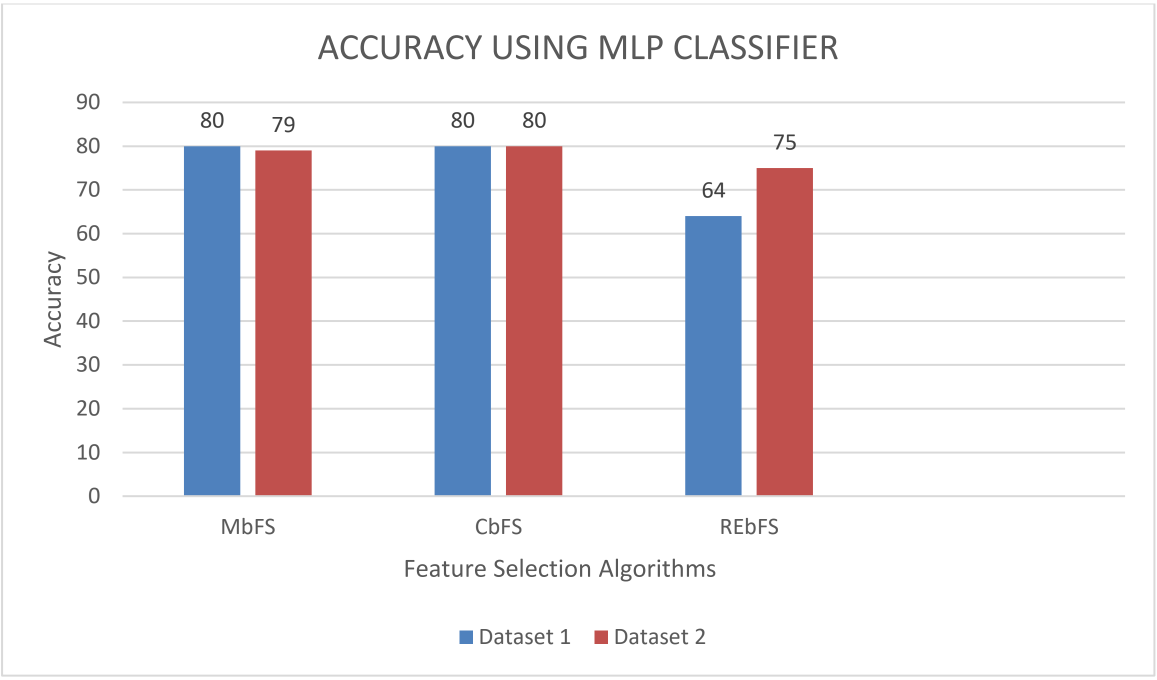


Table 1 shows the accuracy results of feature selection algorithms implemented using centralized approach (our previous work [10]). Figure 8 shows the accuracy values of the feature selection algorithms using centralized approach, for Dataset 1(with 768 instances) and Dataset 2(with 2000 instances).

Table 2 Feature Selection algorithms vs Federated MLP classifier accuracy (Decentralized model)

ALGORITHM	**ACCURACY USING Fed MLP CLASSIFIER**	
	Dataset 1	**Dataset 2**
Mean based Feature Selection(MbFS)	66%	66%
Correlation based Feature Selection(CbFS)	66%	66%
Recursive Elimination based Feature Selection(REbFS)	66%	70%

Figure 9 Feature Selection algorithms vs Federated MLP classifier accuracy

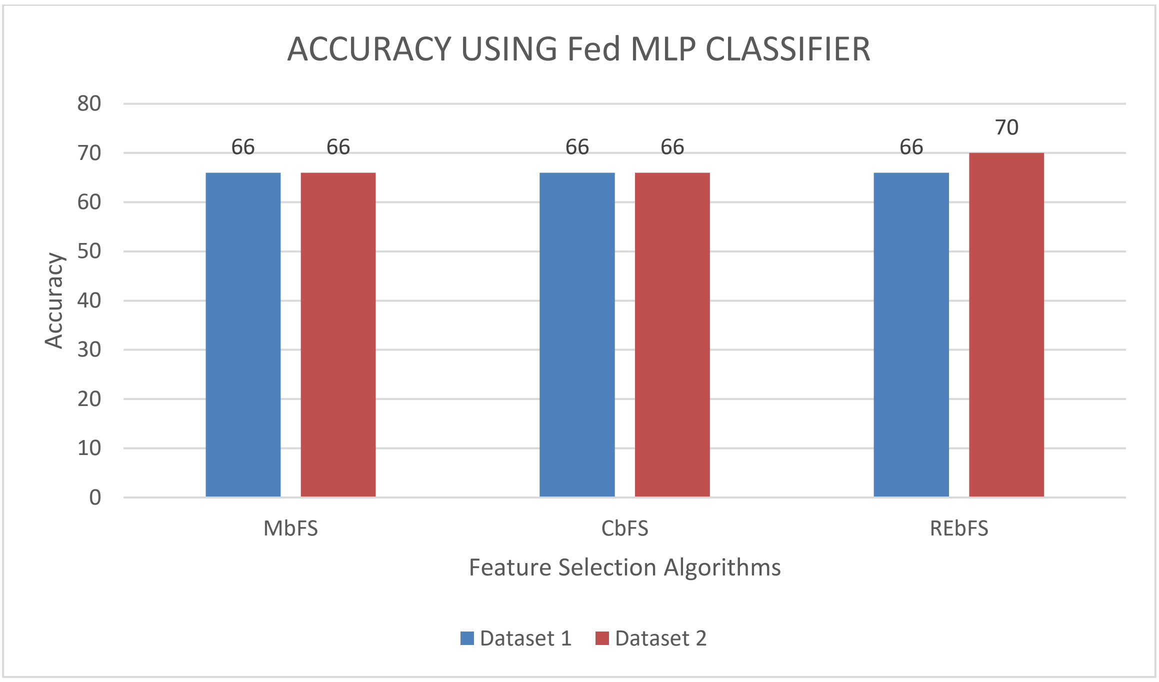


Figure 9 shows the accuracy values of the feature selection algorithms using federated approach, for Dataset 1(with 768 instances) and Dataset 2(with 2000 instances). The Mean based Feature Selection (MbFS) yields an accuracy of 66% with an MSE loss of 0.2227 for Dataset 1 and an accuracy of 66% with an MSE loss of 0.2313 for Dataset 2. The Recursive Elimination based Feature Selection (REbFS) yields an accuracy of 70% with an MSE loss of 0.2105 for Dataset 1 and an accuracy of 66% with an MSE loss of 0.2390 for Dataset 2. The Correlation based Feature Selection (CbFS) yields an accuracy of 66% with an MSE loss of 0.2238 for Dataset 1 and an accuracy of 66% with MSE loss of 0.2265 for Dataset 2.

The centralized machine learning model results in high accuracy than the decentralized model (FL model). The reason for decreased accuracy in FL model is due to increased privacy. To ensure privacy of any sensitive data, accuracy gets compromised. Hence, it can be inferred that any application that requires privacy must opt for the FL model and those applications that contain less sensitive data and in need of high accuracy can opt for a centralised ML model.

IV. CONCLUSION

Privacy of data is essential especially in current scenario where data breaching is becoming very common. This paper proposed a federated learning approach for early prediction of Type-2 Diabetes that involves feature selection algorithms with a federated MLP method. Also, comparison of centralized machine learning model with a decentralized federated learning model is made to highlight the importance of a decentralized model in terms of data privacy. Experimental results using the standard Pima Indian Diabetes dataset proves that the decentralised federated model ensures privacy though it results in decreased accuracy. This is because the data is stored in separated client devices which can be accessed only by the client devices. Hence privacy is ensured and because it involves huge time in collecting the separately stored data at the server, accuracy decreases. Hence, privacy-oriented applications can opt for a federated model and those that seek for accuracy oriented can opt for centralized model.

V. REFERENCES

[1]. Sajratul Yakin Rubaiat, Md Monibor Rahman, Md.Kamrul Hasan, 2018, "Important Feature Selection & Accuracy Comparisons of Different Machine Learning Models for Early Diabetes Detection", International Conference on Innovation in Engineering and Technology (ICIET).

[2]. H. Wu, S. Yang, Z. Huang, J. He, and X. Wang, 2018, "Type 2 diabetes model based on data mining", Informatics in Medicine Unlocked, vol. 10, pp. 100–107.

[3]. Aliza Ahmad, Aida Mustapha, Eliza Dianna Zahadi, Norhayati Masah, Nur Yasmin Yahaya, 2011, " Comparison between Neural Networks against Decision Tree in Improving Prediction Accuracy for Diabetes Mellitus", Digital Information Processing and Communications, Springer.

[4]. Dilip Kumar Choubey, Sanchita Paul & Santosh Kumar, Shankar Kumar, 2017 , "Classification of Pima indian diabetes dataset using naive bayes with genetic algorithm as an attribute selection", Communication and Computing Systems – Prasad et al. (Eds) Taylor & Francis Group, London, ISBN 978-1-138-02952-1.

[5]. Kamer Kayaer, Tulay Yildirim, 2003, "Medical diagnosis on pima indian diabetes using general regression neural networks", Proceedings of the International Conference on Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP), pp. 181–184.

[6]. Manjeevan Seera, Chee Peng Lim, 2014, "A hybrid intelligent system for medical data classification", Expert Systems with Applications, vol. 41, no. 5, pp. 2239–2249.

[7]. YoichiHayashi, ShonosukeYukita, 2016, "Rule extraction using recursive-rule extraction algorithm with j48graft combined with sampling selection techniques for the diagnosis of type 2 diabetes mellitus in the pima indian dataset," Informatics in Medicine Unlocked, vol. 2, pp. 92–104.

[8]. Humar Kahramanli, Novruz Allahverdi, 2008, "Design of a hybrid system for the diabetes and heart diseases", Expert systems with applications, vol. 35, no. 1-2, pp. 82–89.

[9]. R. Priyadarshini, N. Dash, and R. Mishra, 2014, "A novel approach to predict diabetes mellitus using modified extreme learning machine", Electronics and Communication Systems (ICECS), International Conference on. IEEE, pp. 1–5.

[10]. M.Lincy, Dr.A.MeenaKowshalya, "Leveraging Feature Selection Algorithms for Early Detection of Type-2 Diabetes", Journal of Computer Technology & Applications, May 2020, ISSN: 2229-6964, Volume 11.

[11]. Evita Bakopoulou, Bálint Tillman, and Athina Markopoulou, "A Federated Learning Approach for Mobile Packet Classification", arXiv: 1907.13113v1, 2019.

[12]. Li Huang, 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, Volume 99, November 2019.

[13]. Xin Yao, Chaofeng Huang, Lifeng Sun, "Two-Stream Federated Learning: Reduce the Communication Costs", IEEE Visual Communications and Image Processing (VCIP), April 2019.

[14]. Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang,, Hao Deng, and Dianbo Liu, "LoAdaBoost:Loss-Based AdaBoost Federated Machine Learning on medical data", arXiv:1811.12629v3, Aug 2019.

[15]. Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, Chunyan Miao, "Federated Learning in Mobile Edge Networks: A Comprehensive Survey", Computer Science, Engineering, ArXiv 2019.

[16]. Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, Vikas Chandra, "Federated

Learning with Non-IID Data", Machine Learning, arXiv:1806.00582, June 2018.

[17]. Theodora S. Brisimia, Ruidi Chena, Theofanie Melac, Alex Olshevskya, Ioannis Ch. Paschalidisa, Wei Shia,d, "Federated learning of predictive models from federated Electronic Health Records", International Journal of Medical Informatics, Volume 112, April 2018.

[18]. Nguyen H. Tran, Wei Bao, Albert Zomaya, Minh N.H. Nguyen, Choong Seon Hong, "Federated Learning over Wireless Networks: Optimization Model Design and Analysis", IEEE Conference on Computer Communications, June 2019.

[19]. Dianbo Liu, Dmitriy Dligach, Timothy Miller,"Two-Stage Federated Phenotyping and Patient Representation Learning", arXiv:1908.05596 , August 2019.

[20]. Dataset 1(768 instances) https://www.kaggle.com/uciml/pima-indians-diabetes-database/download

[21]. Dataset 2(2000 instances) https://www.kaggle.com/johndasilva/diabetes/download

[22]. A. Meena Kowshalya, R. Madhumathi, N. Gopika, 2019, "Correlation Based Feature Selection Algorithms for Varying Datasets of Different Dimensionality", Wireless Personal Communications, October, Volume 108, Issue 3, pp 1977-1993.

[23]. Gopika, N, A. Meena Kowshalya "Correlation based feature selection algorithm for machine learning." In 2018 3rd International Conference on Communication and Electronics Systems (ICCES), pp. 692-695. IEEE, 2018.

Cite this article as :

M. Lincy, Dr. A. Meena Kowshalya, "Early Detection of Type-2 Diabetes Using Federated Learning", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 7 Issue 6, pp. 257-267, November-December 2020. Available at doi : https://doi.org/10.32628/IJSRSET207644
Journal URL : http://ijsrset.com/IJSRSET207644