COMP5434 BIG DATA COMPUTING

Group Project

Group 3 on Monday class

22049616G	ZHANG Shimin	张时敏	22049616g@connect.polyu.hk
22057918G	HUANG Qiuyi	黄秋逸	22057918g@connect.polyu.hk
22028269G	TANG Ka Shun	鄧嘉淳	22028269g@connect.polyu.hk
22068387G	QIU Ruipeng	邱锐鹏	22068387g@connect.polyu.hk
22045145G	QUAN Hongli	全虹历	22045145g@connect.polyu.hk

The Hong Kong Polytechnic University

2023

Contents

1 Introduction **1**

2 Exploratory data analysis **2**

2.1 Data preprocessing 2

2.2 Data analysis 3

3 Task 1 **6**

4 Task 2 **8**

4.1 Decision Tree 8

4.2 KNN model 9

4.3 Support Vector Machine 10

4.4 Artificial Neural Network 10

5 Task 3 **11**

5.1 Split the data 12

5.2 FL algorithm and implementation 12

5.3 discussion of the result 13

6 Conclusion and future work **14**

References **16**

1 Introduction

The real estate market in the United States is a significant contributor to the economy, with house prices influenced by various factors such as location, size, noise level, and air conditions. A US-based real estate sales company provides sales data to help investors make informed decisions. The data reflects the latest market developments and provides comprehensive insights into housing trends, emerging markets, and key drivers of price changes.

By analyzing this data, investors can develop predictive models that aid in making data-driven decisions on house transactions. The data-driven approach enables investors to better understand market trends, predict price fluctuations, and mitigate risks associated with making poor investments.

The project aims to analyze and predict the total cost of houses in Hong Kong based on various factors such as residence and building costs, using MapReduce and Machine/Deep learning models. The project consists of three tasks.

Task 1 requires employees of a company to calculate the total cost of each house based on the data in Train_Data using MapReduce. The total cost is calculated based on the unit price of residence and building space, and the exchange rate.

Task 2 involves designing a Machine/Deep learning model to predict the total cost of each house. The columns for unit price of residence and building space are removed, and the model is trained using the training data. The total cost is then divided into four classes, and the price range for each sample in Test_Data is predicted.

Task 3 requires the use of Federated Learning (FL) algorithm[1] to train the model as the data cannot be shared among four different real estate companies. The dataset is split into four parts, and the FL model is compared to the model obtained in Task 2. Finally, the price range for the total cost of each sample in

Test_Data is predicted.

Overall, the project aims to provide a comprehensive analysis of the real estate market and develop a predictive model that can help investors make informed decisions. The logical connection between each task is that the results of each task are used as input for the subsequent task. The calculation results from Task 1 are used to train the predictive model in Task 2 and the FL model in Task 3. The predictive model in Task 2 is then used to determine the price range of the total cost, which is tested in Task 3. Finally, the FL model in Task 3 is compared to the model obtained in Task 2 to evaluate its performance.

2 Exploratory data analysis

2.1 Data preprocessing

Before processing the data, we need to clean the data set. We'll check for any missing values in our dataset using the 'isna()' method in Python. Fortunately, no such values were found, and data cleaning is not necessary in this regard.

Apart from that, we also check the data validation. Through the analysis of continuous variables and categorical variables in Table.1, their values are within the reasonable range. All of the data appears to be reasonable and no inconsistencies are immediately noticeable.

We classify the total cost into four categories, which is where we're going to go from here. The sample number of each group is shown in Table.1 Meanwhile, we use one-hot coding for the classification variable "city" and drop useless columns: date, building year, decoration year, district, city, zip code, region, exchange rate and total cost. To make the meaning of the variable more intuitive. We converted "building year" to "building time", which is how long the house has been built. So we've done the data processing.

	number of rooms	security level of the community	residence space	building space	noise level	waterfront	view
count	4000.0	4000.0	4000.0	4000.0	4000.0	4000.0	4000.0
mean	3.4	2.2	2138.4	11604.2	1.5	0.0	0.2
std	0.9	0.8	902.8	18085.1	0.5	0.1	0.7
min	0.0	0.0	430.0	747.0	1.0	0.0	0.0
25%	3.0	1.8	1500.0	5100.8	1.0	0.0	0.0
50%	3.0	2.3	1990.0	7696.5	1.5	0.0	0.0
75%	4.0	2.5	2600.0	10666.8	2.0	0.0	0.0
max	9.0	8.0	13540.0	307752.0	3.5	1.0	4.0

	air quality level	aboveground space	basement space	building year	decoration year	exchange rate
count	4000.0	4000.0	4000.0	4000.0	4000.0	4000.0
mean	3.4	1825.4	313.0	1973.0	788.3	7.0
std	0.7	827.2	455.7	28.5	975.3	0.6
min	1.0	430.0	0.0	1900.0	0.0	6.0
25%	3.0	1200.0	0.0	1954.0	0.0	6.5
50%	3.0	1600.0	0.0	1977.0	0.0	7.0
75%	4.0	2310.0	620.0	1998.0	1999.0	7.5
max	5.0	9410.0	4130.0	2014.0	2014.0	8.0

Table 1: Description of training data

0 <= total cost < 300000	830
300000 <= total cost < 500000	1490
500000 <= total cost < 700000	944
700000 <= total cost	736

Table 2: Sample size of four total costs classes

2.2 Data analysis

First and foremost, we check the correlation between features in Figure 1.

Taking 0.5 as the threshold of absolute value of correlation coefficient, we can find a lot of interesting information:

- There is a significant positive correlation between the number of rooms, security level of the community, residence space and aboveground space. This may be because these factors are positively correlated with house prices. In other words, for the property with higher price, there are generally more rooms, better security environment, larger residence space and aboveground space.

- There is a strong positive correlation between noise level and security level of the community, which indicates that safer areas tend to be noisy, while

quiet areas may not have good security.

- There is a strong negative correlation between building time and security level of the community, noise level, which indicates that buildings built for a long time tend to be noisy and have poor security.
- Since the correlation coefficient between "residence space" and "above ground space" is 0.9, which indicates these two variables may contain quite similar information. Therefore, to reduce variable redundancy, we just choose one of them when building models.

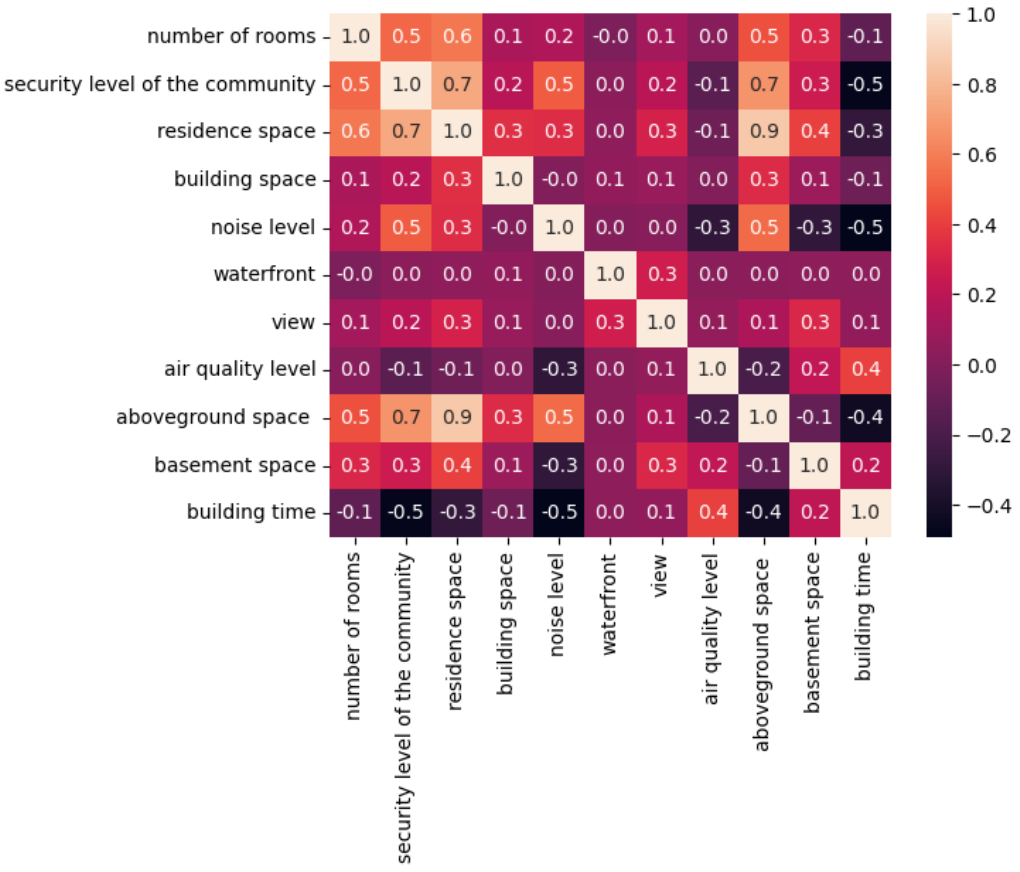

Figure 1: Heatmap of correlation

After that, we drew the scatter plot matrix. We hope to find variables with

better classification ability. In order to facilitate observation, 1000 samples were randomly selected for drawing. There are a number of interesting observations that we can make, but we will only cover some of them for space:

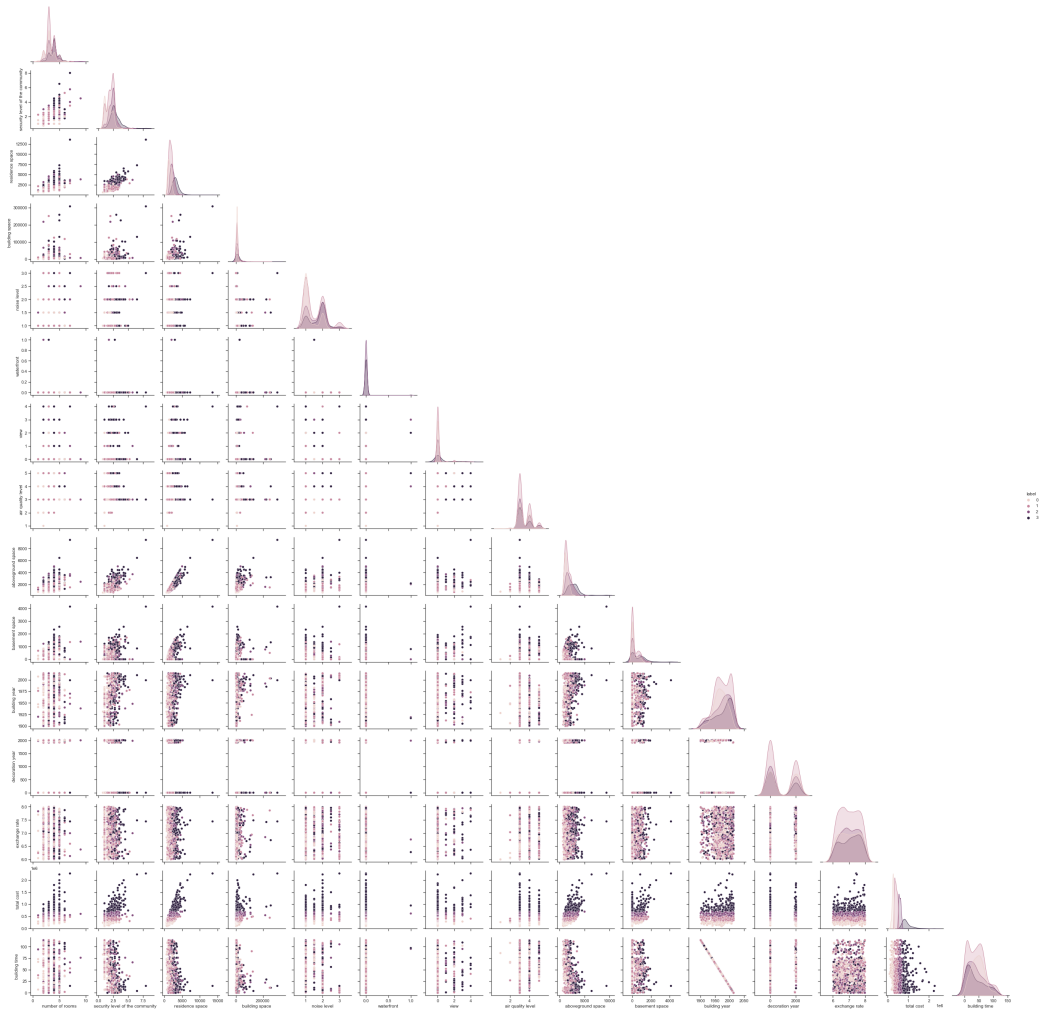

Figure 2: Scatter plot matrix

- The older the building time, the higher the total prices, especially for those in the lowest housing class.
- There is an obvious linear relationship between the number of rooms, security level of the community, residence space and aboveground space, but

the variance gradually increases.

- The attribute residence space and above ground space are highly correlated.
- The attribute residence space, above ground space, building space are severe right skewed.
- The attribute waterfront is a very sparse column (3987 samples-“0”,13 samples-“1”).

Therefore, we fix the skewness of residence space, basement space and building space by taking the logarithm. And drop variable waterfront and above ground space.

3 Task 1

In order to calculate the total cost through MapReduce, we first uploaded the housing price dataset to hdfs[2].

Step 1: Create the hdfs: dfs -mkdir -p /project/housecost project directory first. And upload the file to the project directory: hdfs dfs -put /data/file/house.csv /project/housecost.

Step 2: Write an entity class for calculating housing costs.

Step 3: Write the mapper class： Keys are unit price of residential area, unit price of residential area, building space and exchange rate.

Step 4: Write the reducer class.

Step 5: Write the driver class.

Step 6: Package the project and submit it to hadoop for running.

Figure 3: Submit project to hadoop

Step 7: Output Reduce result: Because there are two reducer, the result will have two files. Aggregate the two files based on the index, then we get the final output.

Figure 4: The last column is the final output

4 Task 2

This is a classification problem. We first build a classification model based on classical machine learning algorithms. Here we use Decision Tree[3], KNN[4] and SVM[5]. And we also build a model with ANN[6], which is in a seperate python file.

4.1 Decision Tree

For the decision tree, the Cost Complexity Pruning (CCP)[7] method was used to prevent overfitting. The CCP method prunes and simplifies the tree after it has been constructed by minimizing a loss function which considers both the cost and the complexity of the tree, and aims to find a balance between the complexity and the accuracy of the tree.

In Figure 5, we plot the accuracy of the decision tree with respect to the ccp_alpha, which is a parameter indicates the complexity of the decision tree.

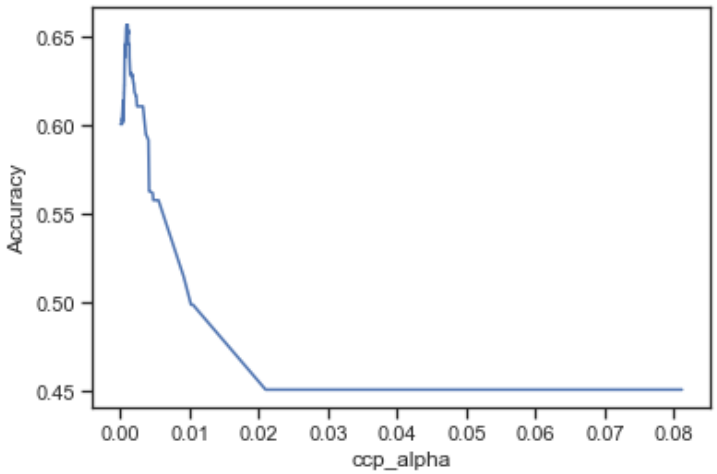

Figure 5: Cost complexity pruning plot

Then we set the best ccp_alpha with highest accuracy. The testing accuracy of the best decision tree model is 65.2%. Table.3 is its confusion matrix on testing

data.

		Predicted				
		Class0	Class1	Class2	Class3	Total
Actual	Class0	149	70	2	1	222
	Class1	51	246	58	13	368
	Class2	1	68	144	38	251
	Class3	0	8	33	118	159
	Total	201	392	237	170	1000

Table 3: Confusion matrix of decision tree model on testing data

4.2 KNN model

For KNN model, we use cross validation method to find the best k. According to Figure 6, the optimal k=9. The testing accuracy of the best KNN model is 66.9%. Table.4 is its confusion matrix on testing data.

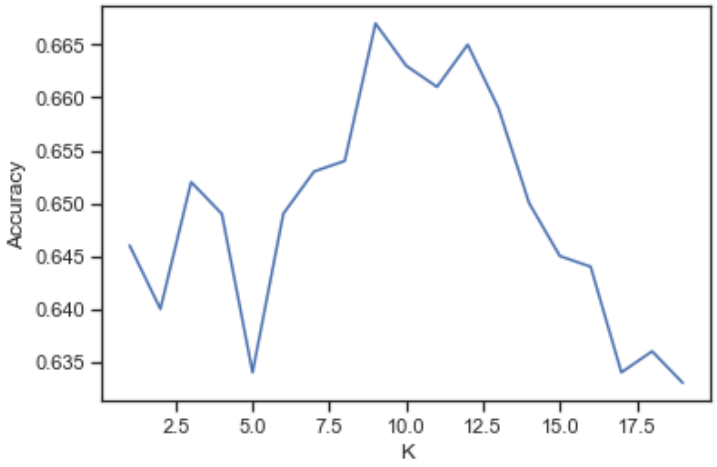

Figure 6: Cross validation of KNN model

		Predicted				
		Class0	Class1	Class2	Class3	Total
Actual	Class0	149	70	3	0	222
	Class1	36	293	35	4	368
	Class2	5	93	112	41	251
	Class3	0	17	29	113	159
	Total	190	473	179	158	1000

Table 4: Confusion matrix of KNN model on testing data

4.3 Support Vector Machine

Finally, we use support vector machines for classification. We don't need to determine any parameters, and the classification results are better than the previous methods. The testing accuracy of the best KNN model is 69.1%. Table.5 is its confusion matrix on testing data.

		Predicted				
		Class0	Class1	Class2	Class3	Total
Actual	Class0	148	73	1	0	222
	Class1	25	303	36	4	368
	Class2	3	93	111	44	251
	Class3	0	4	26	129	159
	Total	176	473	174	177	1000

Table 5: Confusion matrix of SVM on testing data

4.4 Artificial Neural Network

In order to perform the given task, we constructed a fully connected neural network with five layers. The input dimension of our data was 50, and we included three hidden layers with an architecture of 128-256-128. To prevent overfitting, we added a Dropout layer[8] after each neuron layer, with a probability of 0.5 for

a neuron being frozen. We chose ReLU as our activation function and used four neurons in the output layer to denote the scores of the four classes. The classification result of the forward calculation was determined by the category with the highest score.

To optimize the network, we selected Adam with a learning rate of 0.0001 as the optimizer, and Cross Entropy Loss as the loss function for the multi-classification task. To ensure the comparability of results, we fixed the random seed using the seed_everything method for both Numpy and Torch. We chose seed-1234 for our training, and found that the model fit in approximately 200 epochs. We used a mini-batch approach to train the model and calculated the average loss of all data batches at each epoch. The parameters were saved according to the test accuracy of the current epoch.

The maximum test accuracy achieved by our ANN model was 80.47. Table.6 is its confusion matrix on testing data.

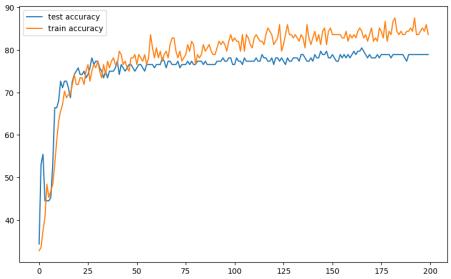

Figure 7: Training and testing accuracy

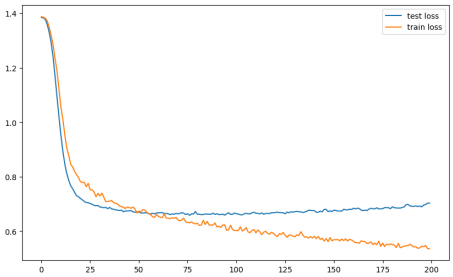

Figure 8: Training and testing loss

5 Task 3

In this task, we were given a real estate dataset and were required to split it into four parts to apply the Federated Learning algorithm to train the model. Then compare the results with those obtained in Task 2.

		Predicted				
		Class0	Class1	Class2	Class3	Total
Actual	Class0	138	41	3	0	182
	Class1	30	213	47	6	296
	Class2	0	45	97	54	196
	Class3	0	0	16	110	126
	Total	168	299	163	170	800

Table 6: Confusion matrix of ANN on testing data

5.1 Split the data

To split the data into four parts, we use the sklearn library's train_test_split function with the stratify parameter. The stratify parameter ensures that the data is split in a way that maintains the proportion of classes in each split. We use this function four times in a row, each time extracting a subset that is equivalent to $\frac{1}{5}$ of the total amount of the original set. The first call with the test size $\frac{1}{5}$ is used to separate the training data and the testing data, and the following three calls set the test size to $\frac{1}{4}$, $\frac{1}{3}$, and $\frac{1}{2}$ respectively to separate the training sets for use by each client network.

5.2 FL algorithm and implementation

Our FL algorithm works as follows:

1. Each party involved in the collaboration trains a local model on their own data.

2. The local models are then aggregated to form a global model.

3. The global model is sent back to each party, and the process is repeated iteratively until convergence.

We implemented this algorithm in two steps. In the first step, we defined a function called get_client_grad to obtain the gradients of the loss function with respect to the parameters of the client network. Then the gradients will be used to update the global model. In the second step, we create a training loop that iterates over a

fixed number of epochs. During each epoch, the loop trains the model using the Federated Learning approach, and evaluates the model's performance on the test data. After the loop over the training data is completed, the loop sets the model to evaluation mode using net.eval(), and loops over the test data in batches. For each batch, it calculates the model's predictions and loss on the test data, and stores the accuracy and loss values.

5.3 discussion of the result

For the results, We obtained an accuracy of 75.00 for the FL model, which is slightly lower than the accuracy of 80.47 achieved by the ANN model. One possible reason is that the data is distributed across different devices or servers, which may result in a smaller training dataset for each device or server. This can lead to a less representative sample of the overall data, which can negatively affect the accuracy of the model. Besides that, the loss curve and accuracy curve of the

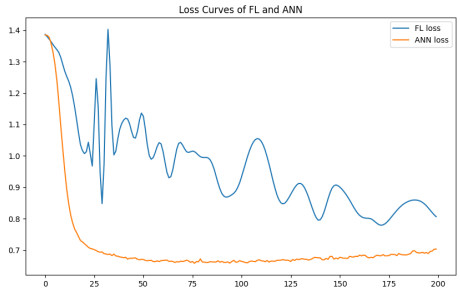

Figure 9: Loss curve comparison

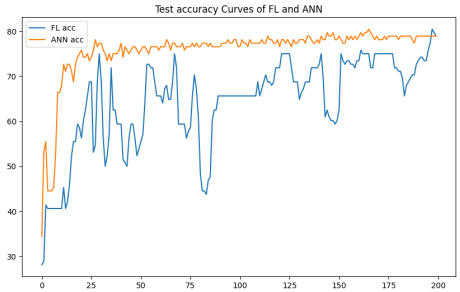

Figure 10: Accuracy curve comparison

FL model exhibit more fluctuations than the ANN. The cause may be The learning progress of multiple sub-models is different. It's also possible that the neural network architecture used in the federated learning method was not as optimal as the one used in the non-federated method, which can also affect the accuracy of the model. Lastly, the FL algorithm may require more time to converge to a stable accuracy compared to the ANN, as it involves more rounds of optimization

and communication between the parties. All in all, we can conclude that the FL algorithm performed slightly worse than the ANN. However, the difference in accuracy is not significant, and we can still use the FL algorithm to train the model if there are privacy concerns.

		Predicted				
		Class0	Class1	Class2	Class3	Total
Actual	Class0	142	37	2	1	182
	Class1	27	220	44	5	296
	Class2	1	43	109	43	196
	Class3	0	0	16	110	126
	Total	170	300	171	159	800

Table 7: Confusion matrix of federal learning on testing data

6 Conclusion and future work

In this study, we have investigated various machine learning models and a federated learning approach to address a multi-class classification problem using an anonymized real estate dataset. We began by training classical machine learning algorithms, including Decision Trees, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM), and compared their performance. Next, we implemented an Artificial Neural Network (ANN) with a fully connected architecture, which outperformed the classical algorithms with an accuracy of 80.47%. Finally, we explored the Federated Learning (FL) approach, which achieved an accuracy of 75.78%, providing a viable option for training the model when privacy concerns are considered.

The result of these experiments demonstrate that amount classical learning algorithms, SVM achieved the highest accuracy. While deep learning algorithm

achieved a maximum accuracy of 80.47%. The performance of FL model was slightly worse compared to the ANN. But it still perform better than all of the classical models that we had tried. Besides, it could potentially be improved with additional optimization.

In future work, several aspects can be explored to enhance the model's performance and investigate other approaches to address privacy concerns:

1. Hyperparameter optimization: Further tuning of hyperparameters for both the ANN and FL models could improve performance. This could include the number of layers, neurons, learning rate, dropout rate, and other parameters that affect model behavior.

2. Replacing artificial neurons by spiking neurons: Spiking neurons in SNNs, more biologically plausible than ANNs, generate and accumulate membrane voltages. Upon reaching a threshold, they spike and reset, helping SNNs achieve or exceed DNN performance in various AI tasks.

3. Ensemble methods: Combining multiple models or algorithms to improve the overall performance of the classification task can be considered. Ensemble methods like stacking, bagging, or boosting could help achieve better results.

4. Data augmentation and feature engineering: Enhancing the dataset by generating additional synthetic data or creating new features based on domain knowledge could help improve the model's performance.

5. Differential privacy: Implementing mechanisms for Differential Privacy, which adds noise to the data or the model's output, could provide better privacy protection while maintaining reasonable performance. This approach could be combined with the Federated Learning method to further strengthen privacy.

6. Scalability and efficiency: Investigating the trade-offs between accuracy, privacy, and computation efficiency in a real-world, large-scale setting is essential. Techniques for reducing communication overhead and improving the speed of

model updates in the federated learning setting should be explored.

In conclusion, this study has showcased the potential of various machine learning algorithms and the federated learning approach to address a multi-class classification problem while considering privacy concerns. These results pave the way for further exploration of advanced techniques and privacy-preserving methods in machine learning, which are increasingly important in today's data-driven world.

References

[1] https://research.ibm.com/blog/what-is-federated-learning

[2] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[3] https://en.wikipedia.org/wiki/Decision_tree

[4] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[5] https://en.wikipedia.org/wiki/Support_vector_machine

[6] https://en.wikipedia.org/wiki/Artificial_neural_network

[7] https://www.analyticsvidhya.com/blog/2020/10/cost-complexity-pruning-decision-trees/

[8] https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9