FedLR and FedFor

Abbas Zal

***Abstract*—Federated learning is critical for decentralized model training while preserving data privacy. In this paper, I propose two frameworks—Federated Logistic Regression (FedLR) and Federated Forest (FedFor)—applied to the Spambase and HuGaDB datasets. My approach integrates robust data preprocessing, local model training, and federated aggregation through averaging or majority voting. Performance is evaluated using global accuracy, Shapley values, and Nash equilibrium analysis. Experiments show that FedFor effectively captures non-linear patterns and often outperforms FedLR, Despite some clients having low-quality data, full client participation still emerges as a stable Nash equilibrium in most cases. These insights advance the state-of-the-art in federated learning and offer valuable benefits for future research and privacy-preserving commercial applications.**

***Index Terms*—Federated Learning, Spambase, HuGaDB, Logestic Regression, Dessicion Tree, FedAVG, FederatedForest,Nash Equilibrium**

I. INTRODUCTION

Federated learning (FL) has emerged as a powerful scheme to facilitate the collaborative learning of models amongst a set of agents holding their own private data. Imagine a hospital is trying to evaluate how effective a certain medical procedure . They analyze data from past patients and build a linear regression model with parameters $\hat{\theta}$. However, since they only have data from their own patients, the model isn't very accurate—it has a high error rate.

Fortunately, other hospitals have also performed the same procedure and collected their own data. But due to privacy concerns, technical differences, and other limitations, they cannot simply share their raw patient data. Instead, they decide on a different approach: each hospital trains its own model separately, and then they combine their models by taking a weighted average of the parameters. If there are M hospitals, and hospital i has n_i patient samples, the final combined model is [1]:

$$\hat{\theta}^f = \frac{1}{\sum_{i=1}^{M} n_i} \sum_{i=1}^{M} \hat{\theta}_i \cdot n_i$$

This method is an example of *federated learning*, a powerful approach to machine learning that is gaining popularity [2]. However, federated learning isn't always perfect. In the hospital example, there could be challenges such as differences in data quality, variations in how hospitals collect patient information, or even biases in individual models. These factors can impact the accuracy of the final combined model(**Global Model Accuracy**). This report investigates approaches to addressing these challenges, structuring data to achieve both high global model accuracy and client satisfaction, and managing client formations in combinations. I address federated learning challenges in heterogeneous environments without and with low-quality clients. My work introduces two frameworks—**FedLR** and **FedFor**—that merge local model training (logistic regression or decision trees) with global aggregation (FedAvg or majority voting). By leveraging Shapley values, i assess client contributions and identify Nash equilibria to ensure fairness and stability. Experiments on Spambase and HuGaDB demonstrate the practical effectiveness of my approach.

- **Framework Innovation:** Proposing FedLR and FedFor for federated learning.
- **Performance under Degradation:** Evaluating methods with standard and low-quality clients scenarios.
- **Fairness Analysis:** Using Shapley values and Nash equilibria to measure client impact.

The paper is organized as follows. Section II reviews related work; Sections II and III describe the system and data models and details the proposed methods; Section IV presents experimental results; and Section V concludes with future directions.

II. PREPROCESSING AND PIPELINE

Two datasets were used in this work:
The **spambase** dataset [3]:

was manually partitioned among 10 clients using an equal data-splitting approach. Given $X \in \mathbb{R}^{n\times d}$ and labels $y \in \mathbb{R}^n$, i divided the dataset into n_{clients} subsets. Each client received approximately $\frac{n}{n_{\text{clients}}}$ samples, **where n is total number of samples**. If shuffling was enabled, the data was first randomly permuted to avoid bias. After partitioning, the dataset was split into training and testing subsets, where 80% of the data was used for training and 20% for testing. To improve the performance and

I would like to thank my professor,Thomas Marchioro for his invaluable guidance and support.

stability of the learning model, standardization was applied.

The **HuGaDB** [4]:
which originally consisted of data collected from 18 clients. However, for this study, only data from 10 clients was used to simulate a federated learning environment. The dataset includes multiple sensor-based activity recognition data points, which were preprocessed before use. The global training and testing datasets were constructed by combining data from the selected 10 clients. Any missing values were removed to ensure data consistency. The input features consisted of sensor readings, while the target variable represented different activity labels. Since the activity labels were categorical, label encoding was applied. The same scaling transformation (As spambase) was applied to both training and testing sets to maintain consistency.

III. MODEL ARCHITECTURES AND LEARNING FRAMEWORKS

In my study, the base training models for client data training were **Logistic Regression and Decision Trees**. Logistic Regression is a linear classification algorithm that estimates the probability of class membership using the sigmoid function and optimizes parameters through maximum likelihood estimation. It is effective for both binary and multi-class classification. In contrast, a Decision Tree is a non-parametric model that hierarchically partitions data by selecting optimal feature splits based on criteria such as Gini impurity or entropy. It constructs an interpretable tree structure where internal nodes represent decision rules, and leaf nodes correspond to class predictions, making it well-suited for capturing complex, non-linear relationships.

A. Federated Logestic Regresssion - FedLR -

The First learning framework utilizes a **Federated Logistic Regression model**, which enables decentralized training across multiple clients while preserving data privacy. The FedLR framework follows a structured approach that includes local model training, federated aggregation, and performance evaluation. Each client trains a logistic regression classifier on its local dataset. Given a dataset $\mathcal{D}_i = \{(x_j, y_j)\}_{j=1}^{N_i}$ for client i, where $x_j \in \mathbb{R}^d$ is the feature vector and $y_j \in \mathcal{Y}$ represents class labels. Once client models are trained, they are aggregated using **Federated Averaging (FedAvg)** (aggregates multiple logistic regression models by averaging their coefficients and intercepts.). To evaluate the aggregated model, i compute the global accuracy using a test dataset. Additionally, i compute per-client accuracies on Global test set. (see Fig 1 - a)

B. Federated Forest - FedFor -

The Second learning framework employed a **Federated Forest (FedFor)** approach. The framework consists of local decision tree training, federated aggregation, and performance evaluation. Each client trains a **Decision Tree** classifier on its local dataset. The model is trained by selecting the best feature split at each node based on impurity reduction. Once client models are trained, they are aggregated into a **FedFor**, where individual trees contribute to a collective decision-making process. The predictions from client models are combined using majority voting:

$$y_{\text{final}} = \arg\max_k \sum_{i=1}^{M} \mathbb{I}(y_i = k), \tag{1}$$

where M is the number of participating clients and y_i represents predictions from the i^{th} model. (see Fig 1 - b)

C. FedLR and FedFor With Low Quality Clients(LQC)

To analyze the my **Federated Learning models**, i introduced data corruption techniques that affect the client datasets before applying federated learning strategies.

Two distinct corruption methods are employed for different datasets:

1) Spambase Corruption

I introduced missing values and noise using the following approach:

- Noise Injection: Features of randomly selected samples are perturbed by adding Gaussian noise $\mathcal{N}(0, \sigma^2)$, where σ is a predefined noise standard deviation.
- Missing Values: A subset of the corrupted samples is further modified by replacing selected feature values with NaN.

Given a dataset (X, y), the corruption is defined as:

$$X_{ij} = \begin{cases} X_{ij} + \mathcal{N}(0, \sigma^2), & \text{if Noise Injection applies} \\ \text{NaN}, & \text{if missing value applies} \end{cases} \tag{2}$$

The labels y remain unchanged due to the limited sample size available per client in **Spambase** Dataset.

2) HuGaDB Dataset Corruption

For the **HuGaDB dataset**, both feature corruption and label corruption are applied:

- Feature Noise and Missing Values: Similar to Spambase.

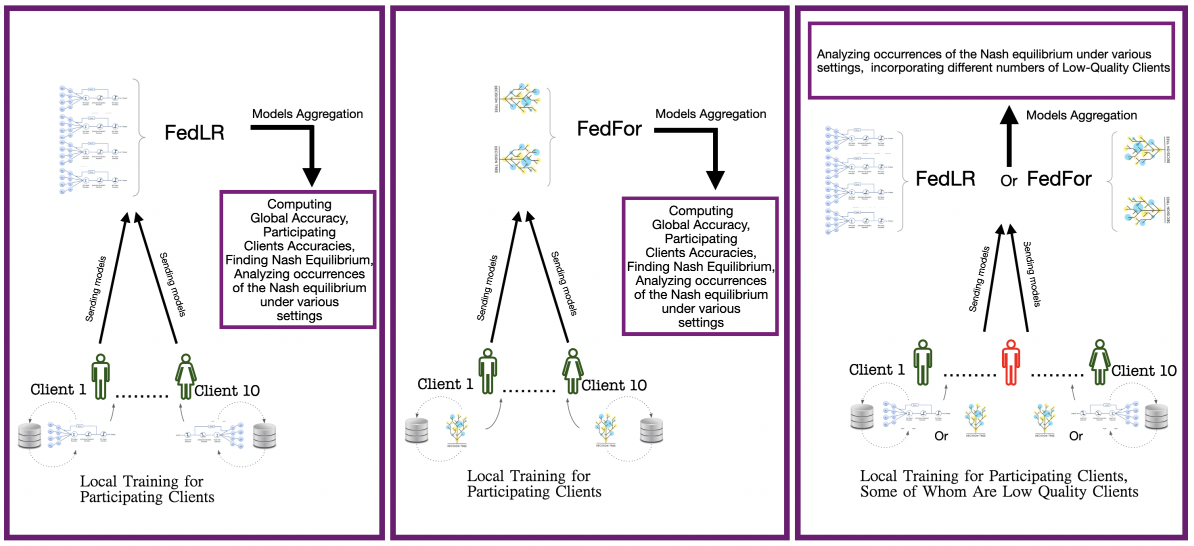
FedLR
Models Aggregation
Computing Global Accuracy, Participating Clients Accuracies, Finding Nash Equilibrium, Analyzing occurrences of the Nash equilibrium under various settings
Sending models
Client 1
Client 10
Local Training for Participating Clients
FedFor
Analyzing occurrences of the Nash equilibrium under various settings, incorporating different numbers of Low-Quality Clients
Or
Local Training for Participating Clients, Some of Whom Are Low Quality Clients
a
b
c

Fig. 1: Overview of the proposed federated learning frameworks under different scenarios. **(a)** Illustration of FedLR (see Section III-A), where each client trains a Logistic Regression model locally and the server aggregates their coefficients using **Federated Averaging (FedAvg)**. **(b)** Illustration of FedFor (see Section III-B), where each client trains a **Decision Tree classifier** locally and the server combines their predictions via majority voting (Equation 1). **(c)** Illustration of FedLR or FedFor in the presence of Low Quality Clients (LQC) situation (see Section III-C1 and Section III-C2), where data are corrupted through noise injection, missing features, and/or label flipping before local model training. The aggregated model is then evaluated in terms of global accuracy, per-client accuracy, and the **occurrences of a Nash Equilibrium under various settings**.

- Label Corruption: Labels of non-NaN samples are flipped to different classes with a fixed probability.(see Fig 1 - c)

IV. Results

Before presenting the results, it is beneficial to review the methodology used for exploring all configurations.

Methodology for Exploring All Configurations

To systematically analyze performance across all client combinations, the following approach was employed:

Iterative Analysis:
The analysis began with the binary representation '0000000001' (indicating that only Client 1 participated) and proceeded through to '1111111111' (where all clients participated), evaluating every valid combination.

Nash Equilibrium Methodology: The Nash equilibrium is determined by assessing whether any individual client can improve its performance—measured by the global accuracy—through a unilateral change in its participation decision. Here, the global accuracy of each configuration is used as a payoff for a client's utility. For every coalition, the method compares the coalition's global accuracy when a client is part of the coalition with the client's accuracy when training independently (represented by a one-hot configuration). In practice, if a client is not participating (denoted by a 0), the approach checks whether the global accuracy would improve if the client joined the coalition (i.e., if the global accuracy of the new coalition exceeds that of the client's standalone training). Conversely, if a client is already participating (denoted by a 1), it evaluates whether its standalone training accuracy is higher than the current coalition's global accuracy. A configuration qualifies as a Nash equilibrium if, for each client, switching its participation (either joining or leaving the coalition) does not lead to a higher global accuracy.

The utility function for a client i is defined as:

$$u_i(S) = \text{Global Accuracy}(S)$$

where S represents a coalition configuration. I did not consider communication cost and other related parameters, assuming that they are equal for all clients.

Nash Equilibrium Check:
In the initial experiment, I did not run the models for different configurations; however, I also wanted to examine the Nash results across various configurations. For experiments conducted both without and with LQC, I employed **FedLR** and **FedFor** as the learning strategies. These models,(as described in Sections III-A and III-B), were tested under multiple configurations. In particular, I varied the maximum number of iterations for the **FedLR** models and the maximum depth for the **FedFor** models. Each model was executed over 10 trials, with different parameter values and a unique random seed in each trial. Notably, for the **HuGaDB** dataset, it was

necessary to use a subsample of each client's dataset as well as a subsample of the global dataset in every trial. This subsampling was crucial because, without it, the results remained unchanged across trials even when the maximum iteration or depth settings were modified. In contrast, this approach was not required for the **Spambase** dataset, since the data was manually split among clients. A different random seed was applied to the splitting process in each trial to ensure variability in the results.

A. *FedLR For Spambase and HuGaDB*

For Federated Logistic Regression (FedLR) experiments on the Spambase and HuGaDB datasets, I focused on (1) the best global model accuracies under different client configurations, (2) the Shapley values quantifying each client's contribution to the global model, and (3) the corresponding Nash Equilibrium (NE).

a) Spambase Dataset

The centralized training accuracy on the Spambase dataset was 0.9197. Table I shows the best federated configurations in terms of global accuracy. These combinations of clients yield global accuracies higher than the centralized baseline.

Combination	Clients	Global Accuracy
1111010101	[1, 3, 5, 7, 8, 9, 10]	0.939197

TABLE I: The best global accuracy for the FedLR Spambase dataset.

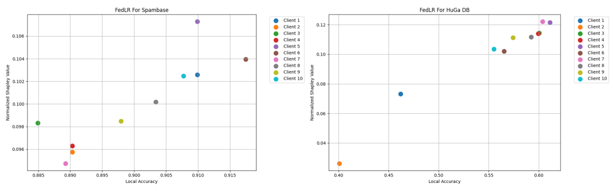

Fig. 2: Shapley values and client contributions for **left** Spambase and **right** HuGaDB in FedLR

Figure 2 presents the Shapley values, which quantify the marginal contribution of each client to the global model performance. Notably, the Nash Equilibrium for this dataset is achieved when `Combination = 1111111111`, i.e., all clients $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$ participate, yielding a global accuracy of 0.932682.

b) HuGaDB Dataset

For the HuGaDB dataset, the centralized training accuracy was 0.6603. Table II shows the best global accuracy for different client subsets. These results show that certain client combinations can perform closely match the centralized approach.

Figure 2 illustrates the Shapley values for each client in HuGaDB. Two distinct Nash Equilibria emerged in this dataset(TableIII).

Combination	Clients	Global Accuracy
0001011101	[1, 3, 4, 5, 7]	0.655780

TABLE II: The best global accuracy for the FedLR HuGaDB dataset.

Combination	Clients	Global Accuracy
0000000011	[1, 2]	0.471286
1111111111	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	0.645829

TABLE III: NE For FedLR on HuGaDB

B. *FedFor For Spambase and HuGaDB*

In this section, I present the results of my FedFor experiments on the Spambase and HuGaDB datasets. I focused on the same results as previous section.

a) Spambase Dataset

The centralized training accuracy on the Spambase dataset was 0.9402. Table IV shows the best federated configurations in terms of global accuracy.

Combination	Clients	Global Accuracy
1110011101	[1, 3, 4, 5, 8, 9, 10]	0.921824

TABLE IV: The best global accuracy for the FedFor Spambase dataset.

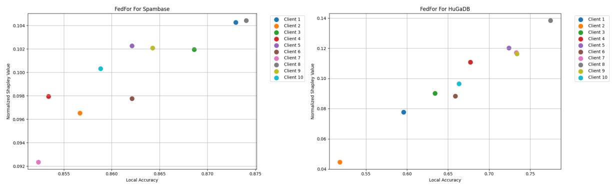

Fig. 3: Shapley values and client contributions for **left** Spambase and **right** HuGaDB in FedFor

Figure 3 presents the Shapley values. Also, the Nash Equilibrium that for this dataset were achieved are in TableV:

b) HuGaDB Dataset

For the HuGaDB dataset, the centralized training accuracy was 0.9542. Table VI shows the best global accuracy for different client subsets.

Figure 3 illustrates the Shapley values for each client in HuGaDB. Six distinct NE emerged in this dataset (Table .VII):

C. *Nash Equilibrium Check Without LQC for Spambase and HuGaDB*

As mentioned earlier, in the initial experiments where I aimed to find the NE , I did not run the models for different configurations. However, I wanted to analyze the Nash equilibrium occurrences across various settings.

To achieve this, I ran **FedLR** and **FedFor** under different configurations for both datasets. Specifically, for the **HuGaDB** dataset , I experimented

Combination	Clients	Global Accuracy
0001000000	[7]	0.852334
1111111111	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	0.910966

TABLE V: NE For FedFor on Spambase

Combination	Clients	Global Accuracy
1111011101	[1, 3, 4, 5, 7, 8, 9, 10]	0.883882

TABLE VI: The best global accuracy for the HuGaDB dataset.

with different subsample ratios:$[0.5, 0.6, 0.7, 0.8]$ and varying maximum iterations:$[10, 100]$ for FedLR Model and I used different subsample ratios:[0.6, 0.7, 0.8] while varying the maximum depths:$[10, 100]$ For FedFor Model. For the **Spambase** dataset, I conducted experiments using the same variations in max depths and max iterations.
All configuration was run for **10 trials** to ensure robustness. The total number of trials for HuGaDB in FedLR was 80, while in FedFor, it was 60. For Spambase, both models had 20 trials .The results are presented in Figures 4 and 5. **The 'All Clients' combination** was the most frequent Nash equilibrium across different configurations for both datasets and models.

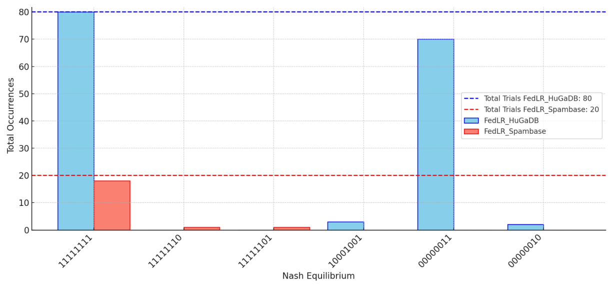

Fig. 4: Nash Equilibrium Occurrences in FedLR for HuGaDB and Spambase across Different Configurations

D. Nash Equilibrium Check With LQC For Spambase and HuGaDB

In this section, I analyze the occurrences of NE under various settings, incorporating different numbers of Low-Quality Clients (LQC). To achieve this, I ran FedLR and FedFor under different configurations for both datasets. Specifically, I considered scenarios where 3, 4, 6, 8, and 9 out of 10 total clients were designated as low-quality and examined the frequency

Combination	Clients	Global Accuracy
0000000001	[1]	0.595840
0000000010	[2]	0.518314
0000000100	[3]	0.634129
0000100000	[6]	0.658733
1000000000	[10]	0.663258
1111111111	[1,2,3,4,5,6,7,8,9,10]	0.880920

TABLE VII: NE for FedFor on HuGaDB

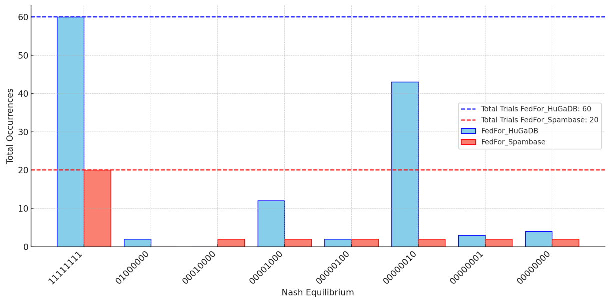

Fig. 5: Nash Equilibrium Occurrences in FedFor for HuGaDB and Spambase across Different Configurations

of the "1111111111" combination appearing across multiple trials under different configurations.

The methods used to generate LQC are described in Sections III-C1 and III-C2. **The corrupting parameters and settings:**

For Spambase, the `corruption_prob` was 0.8 (Applying injection noise or missing values), and `nan_prob` was 0.5.

For HuGaDB, the `corruption_prob` was 0.3 (Applying injection noise or missing values), `nan_prob` was 0.5 , and the `label_corruption_prob` was 0.4. I conducted experiments using the same variations in max depths and max iterations as in the previous section. However, for HuGaDB, I employed subsample ratios:[0.5, 0.6, 0.7, 0.8] for **FedLR** and for **FedFor** employed subsample ratios:[0.5, 0.8]. So, The total number of trials for HuGaDB in FedLR was 80, while in FedFor, it was 40. For Spambase, both models had 20 trials. Also, For Spambase dataset, I explored the impact of different noise standard deviations to assess the effect of low-quality clients on the formation of NE.

The results of these experiments are presented in Figures 6 and 7. As evident, the quality of LQC and the number of LQC impact the formation of Nash equilibrium. As both the poor quality and the number of LQC increase, high-quality clients are more likely to avoid participating in combinations where all clients form a Nash equilibrium.

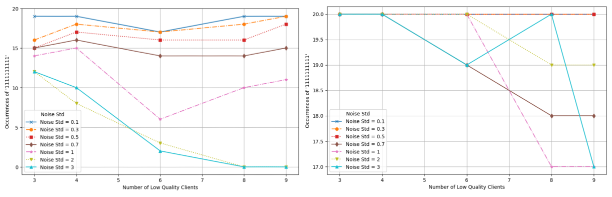

Fig. 6: Occurrences of '1111111111' in Nash vs. varying numbers of low-quality clients across different noise standard deviations. **Left:** FedLR Spambase with LQC. **Right:** FedFor Spambase with LQC.

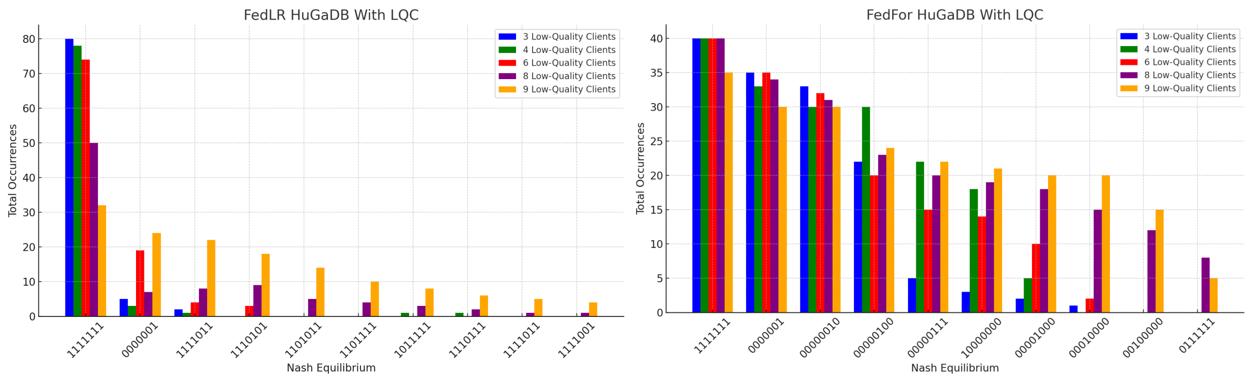

Fig. 7: Occurrences of '1111111111' Nash vs. Different Numbers of Low Quality clients. **Left:** FedLR HuGaDB with LQC. **Right:** FedFor HuGaDB with LQC.

V. Concluding Remarks

This study evaluated two federated learning frameworks, **FedLR** and **FedFor**, using the **Spambase** and **HuGaDB** datasets. My focus was on evaluating global accuracy, individual client contributions using **Shapley values**, and identifying **Nash equilibrium** under both standard and **low-quality client (LQC) scenarios**.

A. *Summary of Findings*

- **Model Performance**
  - **FedFor outperformed FedLR on both datasets**, proving its robustness with high-dimensional data by achieving higher accuracy and better generalization.
- **Shapley Values and Client Contributions**
  - Shapley values effectively quantified each client's data impact and ensured fairness.
  - Clients with poor global test performance consistently received low Shapley values, showing that weak models or low-quality data contribute little.A client's highest local accuracy did not always yield the highest Shapley value due to potential over-specialization.
- **Nash Equilibrium Analysis**
  - The **"All Clients Participate"** combination (`'1111111111'`) often emerged as a Nash equilibrium, indicating that full participation incentivizes even high-quality clients and benefits collaborative learning.
  - In the HuGaDB dataset, alternative NE appeared: excluding the `'1111111111'` combination, FedLR typically led to mixed high- and low-quality clients, while FedFor mostly resulted in NE with low-quality clients.
  - A significant presence of low-quality clients (LQC) and higher noise standard deviation and chosen model reduced good clients' preference for full participation.

B. *Key Lessons and Challenges*

Lessons Learned:

- **Framework Suitability: FedLR** and **FedFor** both performed well on Spambase, yet **FedFor** excelled on HuGaDB by capturing non-linear patterns.
- **Nash Equilibrium Insight:** Understanding NE is essential for predicting stable client participation.

Challenges:

- **Low-Quality Clients:** Noise and missing data from LQC adversely affected performance.
- **Computational Overhead:** Evaluating every client combination is computationally expensive, necessitating efficient approximation techniques.

C. *Future Work*

Future research should extend these experiments to larger, more diverse datasets and validate findings with additional models.

References

[1] Kate Donahue and Jon Kleinberg. Model-sharing games: Analyzing federated learning under voluntary participation, 2020.
[2] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions, 2020.
[3] M. Hopkins, E. Reeber, G. Forman, and J. Suermondt. Spambase [dataset], 1999.
[4] Roman Chereshnev and András Kertész-Farkas. Hugadb: Human gait database for activity recognition from wearable inertial sensor networks, 2018.