**Understanding of Federated Learning:**

**Machine Learning Model and Dataset:**

For this project, we have used a Linear Regression machine learning model to predict housing values based on the California Housing Dataset. The dataset is made up of 20640 total data samples and consists of 8 features including the median income, housing median age, average rooms, average bedrooms, population, average occupancy, and geographical factors like latitude and longitude. There is also one target variable, the median house value. The features included in the dataset are common potential predictors of house prices and thus enable our model to reliably predict the median house prices based on the physical properties and location of houses. Due to the presence of a single target variable, and our objective being a regression problem, the Linear Regression model is suitable choice for predicting the house value based on the given 8 regression factors. This suggests that there would be a linear correlation between the target variable and at least one regression factor. Since our scenario has 5 clients, the dataset is split up across each client, with each client having their own unique testing and training data on which they train their local model.

**Federated Learning Algorithm:**

The Federated Learning algorithm implemented in our project is the FedAvg (Federated Averaging) algorithm. The FedAvg algorithm is a core approach in Federated Learning which allows for collaborative model training across multiple clients while keeping each client’s data private. Our system consists of 1 server and 5 clients where each client has access to its own private dataset which is a unique subset of the California Housing Dataset. In the FedAvg algorithm, each client independently trains a local model on its data, and then sends the model updates, rather than the actual data, to the server. The server then aggregates the updates received from the clients to improve the current global model, and then distributes the new global model back to the clients so they can continue local training and updating. By decentralising data, it maintains data security and privacy, and reduces the storage strain on the server as the data is stored on clients instead which in turn improves scalability by simply adding on clients to the system.

**Implementation:**

Text

**Experimental Results:**

**Simulation Results:**

The simulation results produced by our system demonstrate the significant improvements made to the global model over the course of multiple iterations. Both the testing and training mean squared errors (MSE) showed major reductions from the first iteration to the last iteration, gradually decreasing over the course of the program. In Figure X, we can see the same trend across each client's data where there is a rapid decline in the MSE over the initial iterations followed by a more gradual decline as the MSE nears 0.

By examining the logs for each client, we can see the final iteration produces a substantially lower testing MSE and training MSE. For example, the initial test MSE on client 1’s data was over 309,000 and rapidly decreased to 196.77 after a single iteration. Then by the 10th iteration it was reduced to 4.18, and by the 100th iteration it was only 2.47. Similarly, the other clients demonstrate the same trend but with slightly different values. Client 2’s testing MSE reduced from 367,000 to 2.45, client 3 from 389,000 to 2.78, client 4 from 389,000 to 2.55, and client 5 from 326,000 to 2.41 (see Figure X).

In addition, the pre-update and post-update training MSE results also follows a downwards trend. In Figure X, we can see client 1’s results in its first 4 iterations with the first iteration holding a pre-update training MSE of 402,700 which drastically drops to 223.06 after the update. The training MSE continues to be reduced in each subsequent iteration, with the fourth iteration showing that the pre-update training MSE is now only 55.43 which is significantly less than in the first iteration, and this is reduced even further after the update for a post-update training MSE of 24.85.

Given that our simulation results demonstrate a constant decrease in the testing MSE after each iteration, we can conclude that the server is successfully working to improve the global model. Likewise, the downwards trend in the training MSE results for every client indicates that the local model is also successfully learning and improving alongside the global model. As each client improves its local model, and as the server also improves its global model, the iterative broadcasting of updates allows for all the models to continue improving together as expected by a Federated Learning algorithm.

**Accomplished and Remaining Tasks:**

We have completed the essential criteria such as the integration of the FedAvg algorithm. Its success in learning and refining the existing models continuously is evident as per the simulation results above which outline the MSE trends that we anticipated by a functional FedAvg system. We also implemented two separate optimisation methods: gradient descent and mini-batch gradient descendent. In conjunction, there is an option for enabling subsampling and selecting the number of sub-clients to randomly select in each iteration.

The essential criteria have been satisfied. However, there are some potential tasks that remain unsolved and could improve our understanding of federated learning. This includes implementing other federated learning algorithms such as FedSGD or FedDyn which we could have compared with the results of FedAvg and performed an analysis to determine the pros and cons of each approach.

**Comparative Analysis:**

**Gradient Descent vs Mini-Batch GD:**

Text

**Subsampling:**

Text