\*\*Progress Report:\*\*

\*\*a) What has been done:\*\*

- Completed a literature review on LinUCB, Thompson Sampling, and NeuralUCB algorithms. Evaluated their applications in personalized recommendations and loan recovery strategies.

- Conducted background research on contextual bandits and their use in offline settings, particularly focusing on loan recovery strategies.

- Written custom scripts to simulate loan recovery strategies using LinUCB and Thompson Sampling.

\*\*b) Any questions or issues:\*\*

- Handling delayed rewards remains a challenge as most models assume immediate rewards. We’ve proposed solutions, including ghost rewards and batch updates, but further refinement is needed.

- We would like clarity on whether the main goal is to innovate methodologies, improve predictive accuracy, or explore new datasets.

\*\*c) The plan until the next meeting:\*\*

- Continue exploring NeuralUCB and its application in non-linear reward estimation.

- Implement offline policy evaluation techniques such as Doubly Robust Estimation and Inverse Propensity Scores.

- Test different reward functions based on loan recovery outcomes and analyze model performance.

This format adheres to the request for a short and structured report. Let me know if any adjustments are needed!

My part:

\*\*Code Reading Progress:\*\*

1. \*\*LinUCB Algorithm\*\*:

- We reviewed the implementation of the \*\*LinUCB\*\* algorithm, which selects actions based on upper confidence bounds (UCB).

- The algorithm initializes matrices for storing parameter estimates and context data for each arm.

- It calculates UCB scores for arm selection and updates parameter estimates after each trial using the observed rewards.

- The supporting functions, such as `generate\_reward` and `make\_regret`, were also examined. These functions help simulate rewards with noise and compute cumulative regret over trials.

2. \*\*NeuralUCB Algorithm\*\*:

- NeuralUCB incorporates a neural network to estimate complex, non-linear reward functions, addressing the limitations of linear models like LinUCB.

- The network architecture consists of two hidden layers with ReLU activation functions, and it is trained using the Adam optimizer with mean squared error (MSE) as the loss function.

- This approach allows NeuralUCB to adapt to non-linear contexts, and we have started testing the algorithm in more complex scenarios to observe how it improves reward estimation.

3. \*\*Thompson Sampling with Gaussian Processes (GP)\*\*:

- This algorithm models the reward function using Gaussian Processes (GP) with an RBF kernel, which captures the relationships between contexts and rewards.

- Thompson Sampling is then used to sample from the posterior reward distributions, balancing exploration and exploitation in uncertain environments.

- We examined how GP models are updated with each new batch of data and how Thompson Sampling helps navigate unexplored regions in context space.

4. \*\*Bayesian Neural Networks (BNN) with Thompson Sampling\*\*:

- The BNN approach incorporates uncertainty in reward estimation by using Bayesian layers in the neural network. These layers estimate distributions for the model weights, rather than single point estimates.

- This allows the model to provide more informed decisions in the face of sparse or noisy data, which is particularly useful in scenarios like loan recovery, where uncertainty is prevalent.

- Training BNN with Thompson Sampling allows the model to select actions based on uncertainty in the rewards, and we are exploring how this can be applied to financial data like loan recovery strategies.

5. \*\*Backtesting Framework\*\*:

- We've started backtesting the LinUCB, NeuralUCB, and Thompson Sampling algorithms by comparing cumulative rewards, regret, and accuracy.

- This involves evaluating these models on historical data to determine their effectiveness in selecting optimal actions.

- Visualizations such as cumulative regret plots and action selection frequencies were generated to provide insights into the performance of each model over time.