Final Project Guidelines: Stock Price Prediction Using Fine-Tuned Large Language Models

**Please submit your presentation slides and code to Canvas by 5PM, December 12th.**

**Project Overview** This final project requires you to develop a model for predicting stock closing prices over the next 3 days using fine-tuned large language models (LLMs). Leverage open-source LLMs (e.g., Llama, Mistral, or GPT-like variants) for time-series forecasting by fine-tuning on historical stock data. The focus is on minimizing prediction error (RMSE) while optimizing computational efficiency. Form groups of up to 5 members. Emphasize practical skills, including price prediction and handling unseen test data during your presentation.

**Task Description** Your model must predict the closing prices for the next 3 consecutive days based on input historical stock data.

**Recommended Method**:

1. Input a window of historical data (Day 1 to Day 30) as a text prompt (e.g., "Based on this stock data: [Date1: Open=X, Close=Y, ...], predict the next day's closing price.").
2. Output a single numerical prediction for Day 31.
3. Slide the window: Append the predicted Day 31 (as if it were actual data) to the sequence, remove the original Day 1 to keep a fixed length of 30, and repeat the procedure to predict Day 32. (Repeat step 3 again to predict Day 33)

- **Window Size**: Recommend 30 days for input context to capture trends without excessive computation. Experiment and justify alternatives (e.g., 14-60 days) based on your fine-tuning results.
- **Input data format**: CSV with columns Date (YYYY-MM-DD), Open, High, Low, Close, Volume. Stock identity stays anonymous. Convert data to text prompts for LLM input.
- Train/fine-tune on publicly available datasets (e.g., Yahoo Finance, Kaggle S&P 500, or Stooq.com for free CSVs). Use different PEFT techniques for efficient fine-tuning. Ensure compatibility with our test format (the same format as "sample_test.csv") during presentation.

We provide a sample test CSV file ("sample_test.csv") on Canvas, including approximately 100 days of historical data for an anonymous stock. This matches the structure of the real test data (also about 100 days, but from a different period and stock). These data serve as the input for your models. You can also download other stock data to evaluate your model's performance. During the on-site test, your model must compute the closing stock prices for the three days after the given real test data period and record them in a CSV file.

The data for your own training and testing:

- Yahoo Finance: Search ticker (e.g., AAPL), "Historical Data," filter dates, download CSV.
- Kaggle: Datasets like "Daily Stock Prices" for bulk data.
- Stooq.com: Free historical CSVs for global stocks.

Format final predictions as a list/CSV: [Predicted_Close_Day1, Predicted_Close_Day2, Predicted_Close_Day3]. You can test on our sample code to verify this.

**Development and Resource Monitoring** We provide a standardized testing framework ("framework.py") via the course platform. This framework handles data loading, stock value predictions, output generation, and automatic computation of metrics for evaluation (e.g., total trainable parameters and peak GPU memory usage during inference). You can train your model using the computers in our computer lab, then modify the framework by:

- Replacing the placeholder model class with your fine-tuned LLM implementation (e.g., wrapping a Hugging Face model for numerical output).
- Adapting the loading interface to load your trained model weights (e.g., via torch.load or from_pretrained). Do not alter the rest of the framework, including the sliding window logic, GPU monitoring, or output processes—these ensure fair evaluation. Track GPU usage (via nvidia-smi: time, memory, compute hours) during fine-tuning separately in your logs. Optimize for efficiency—current methods may require multiple passes, so focus on lightweight models. Submit your codes and slides .

**Evaluation Process**

1. **Presentation (15 minutes/group)**: Demo your methodology, sliding window, sample results, and innovations. Scored on clarity, creativity, and technical depth (40%).
2. **On-Site Testing**: Post-presentation, we provide a new CSV test file. Run your modified framework in the container; it will output predictions, parameters, and GPU usage. We compute RMSE on ground-truth 3-day prices (revealed after). Primary metric: Smallest RMSE (40%).
3. **Resource Tiebreaker (20%)**: For similar RMSE (±5%), lower GPU usage (total hours/memory from logs and framework output) and fewer parameters rank higher.

Total score: Presentation + RMSE + Efficiency. Top groups get bonuses.