Final Report, STAT 5398

Zongyi Liu (zl 3554, mentored by Bruce Yang)

Fri, Dec 12, 2025

This report summarizes three assignments from STAT 5398, which was mentored by Bruce Yang; it covers an end-to-end workflow in data-driven quantitative investing and LLM-assisted financial modeling. We are asked to do three assignments this semester. In Assignment 1, I reproduced the FinRL-Trading pipeline for fundamental-data-based stock selection, construct a long-only mean–variance portfolio, and backtested from 2018-01-01 to 2025-10-01.

In Assignment 2, we were asked to fine-tune large language models with LoRA-based parameter-efficient fine-tuning on a financial forecasting dataset, and evaluate performance using accuracy, MSE, ROUGE metrics, and inference time.

As for assignment 3, I integrated the ML stock selection with LLM-derived timing signals from financial news to form a hybrid trading strategy and compared performance against the S&P 500 benchmark.

1 Machine Learning Stock Selection and Portfolio Construction

1.1 Data Collection and Preprocessing

We construct a Historical S&P 500 Components and Changes constituent universe and obtain two datasets from WRDS/Compustat: (i)quarterly fundamentals and (ii) daily security prices (e.g., `prccd`, `prcod`, `ajexdi`, `tic`). Quarterly reports are aligned to tradable decision dates, and next-quarter returns are computed as prediction targets. We compute standard fundamental ratios (PE, PS, PB, ROA, ROE, operating margin, net margin, per-share metrics, etc.) and optionally split by GICS sectors.

1.2 Machine Learning-Based Stock Selection

In this study we mainly utilized the tool of FinRL-Trading. Using the processed dataset (`final_ratios.csv`), we train supervised models to predict next-quarter returns[1]. We evaluate (at minimum) Random Forest, Gradient Boosting, and XGBoost, and select the model with the smallest out-of-sample MSE to generate quarterly predicted returns. Selected stocks for each trade date are stored in `stock_selected.csv`:

$$(\texttt{tic},\ \widehat{r}_{t+1},\ \texttt{trade\_date})$$

The identifier used for labeling should be consistent (e.g., use `tic` instead of `gvkey` if overlaps exist). Then we finally came up with a file called `stock_selected.csv`.

We construct a processed fundamental panel from the file `final_ratios.csv`, which contains standard accounting-based indicators, including profitability, leverage, valuation, and growth measures, together with sector identifiers and firm-level metadata. In preprocessing, non-feature variables such as firm identifiers and date-related fields, like `gvkey`, `tic`, `datadate`, `fyearq`, and `gsector`, are removed. Only numeric variables with complete observations are retained as model features.

The prediction target is the forward quarterly log return, which is calculated as $r_{t+1,i} = \log\left(\frac{S_{t+1,i}}{S_{t,i}}\right)$, and this is stored in the column `y_return`, where $S_{t,i}$ denotes the adjusted price of firm i at quarter t. Then we model the conditional expectation of returns as $r_{t+1,i} = h_\theta(X_{t,i}) + \varepsilon_{t,i}$, where h_θ is a regression function parameterized by θ, and $\varepsilon_{t,i}$ is the error term with zero mean.

To capture structural heterogeneity across industries, we estimate separate models for each sector s (e.g., GICS sectors 10, 15, ..., 60); this will be stored in separate Excel files after implementation.

Then we utilized `stock selection.py` across multiple sectors using fundamental data and a sector-specific prediction model. Given an input directory, it first locates the common fundamentals file `final_ratios.csv` and

iterates over sector codes $s \in \{10, 15, \ldots, 60\}$, expecting a corresponding sector membership file `sector{s}.xlsx` for each sector. For every available sector file, the script calls an external training/prediction pipeline via `fundamental_run_model.py` (with `-tic_column gvkey` indicating the stock identifier), which produces a prediction matrix saved as `predict_best.csv` where rows are trading dates and columns are stocks. It then reads this matrix and, for each date, selects the top-quartile (75th percentile) stocks by predicted return (i.e., it filters to $\hat{r}_{t+1,i} \geq q_{0.75}(\{\hat{r}_{t+1,j}\}_j)$), and appends the chosen `gvkey`, its predicted return, and the corresponding `trade_date` to a master list. Finally, it aggregates all sector-date selections into a single table and exports it to `stock_selected.csv` in the specified output directory, along with basic summary statistics (number of records, unique stocks, date range, and predicted-return range).

1.3 Mean–Variance Portfolio

For each rebalance date, given selected stocks, we estimate expected returns $\mu \in \mathbb{R}^n$ and covariance $\Sigma \in \mathbb{R}^{n \times n}$ from historical returns (rolling window). We solve a long-only mean–variance problem (tangency portfolio under no-short constraint)[2]:

$$\max_{w \in \mathbb{R}^n} \quad \frac{w^\top \mu - r_f}{\sqrt{w^\top \Sigma w}} \tag{1}$$

$$\text{s.t.} \quad \mathbf{1}^\top w = 1, \quad w_i \geq 0 \ \forall i. \tag{2}$$

The optimization is implemented with `scipy.optimize.minimize` using constraints. The output portfolio file is:

$$(\texttt{trade\_date},\ \texttt{tic},\ w_{i,t}) \quad \text{(expanded daily during holding periods for backtesting).}$$

1.4 Backtesting Setup and Results

Then we used the `backtest.py` file to do the back testing from the year 2018 as the assignment suggested; it deals with the performance of the MVP we constructed based on previous selection.

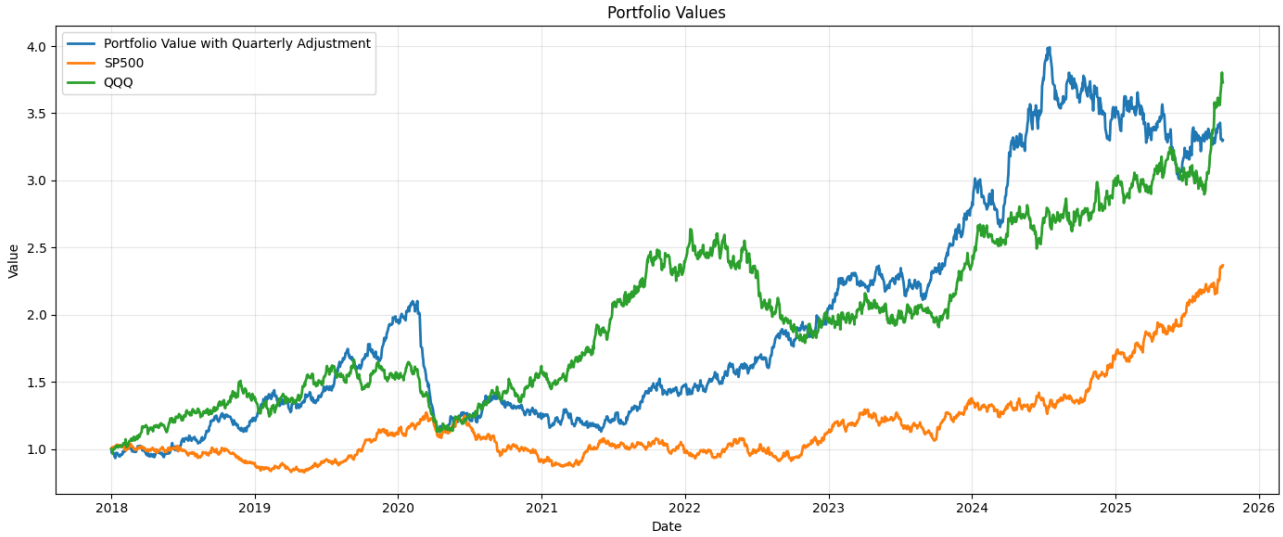

Figure 1: Diagram of Cumulative Portfolio Value from January 2018 to September 2025

As shown in table, the portfolio significantly outperforms the S&P 500 and QQQ in cumulative terms, achieving a return of 1289.42% over the 2018–2025 period. Despite this strong long-term growth, the portfolio maintains an annual return (13.21%) comparable to the S&P 500 while exhibiting lower annual volatility (16.48%), resulting in a high Sharpe ratio of 0.8017.

The portfolios maximum drawdown of -27.95% lies between that of the S&P 500 and QQQ, indicating a balanced exposure to growth and downside risk. Moreover, a win rate of 55.12% and a positive information ratio of 0.2984 suggest that excess returns are generated in a stable and systematic manner. Overall, the results highlight a portfolio that delivers strong risk-adjusted performance while controlling volatility and drawdowns.

Table 1: Performance Metrics of Portfolio and Market Benchmarks (2018–2025)

Metric	Portfolio	S&P 500	QQQ
Cumulative Return (%)	1289.42	136.86	272.96
Annual Return (%)	13.21	13.77	18.51
Annual Volatility (%)	16.48	16.74	19.95
Max Drawdown (%)	-27.95	-24.77	-32.58
Sharpe Ratio	0.802	0.640	0.805
Win Rate (%)	55.12	–	–
Information Ratio	0.2984	–	–

1.5 Discussion

We observe that combining ML-based fundamental stock selection with a long-only mean–variance allocation provides a systematic framework that can outperform naive equal-weight baselines in certain regimes. Key sensitivities include (i) the return/covariance estimation window, (ii) portfolio constraints, and (iii) robustness of fundamental features to regime changes.

2 FinGPT Fine-Tuning with LoRA

2.1 The LLM

We are asked to use two large language models, Llama-3.1-8B and DeepSeek-R1-Distill-Llama-8B. They are provided on the Hugging Face.

Llama-3.1-8B is an 8-billion-parameter large language model released by Meta in 2024. It supports multilingual text and code generation, uses Grouped-Query Attention for efficient inference, and provides a long 128k-token context window. As a relatively lightweight model in the Llama 3.1 family, it offers strong instruction-following ability while remaining practical to deploy on limited hardware [3].

DeepSeek-R1-Distill-Llama-8B is an 8-billion-parameter model obtained by distilling the larger DeepSeek-R1 into a Llama-3.1-8B backbone. The model inherits enhanced reasoning and mathematical capabilities from its teacher while remaining lightweight enough for efficient deployment. As a distilled model, it offers a strong balance between performance and computational cost, making it suitable for general reasoning, coding, and problem-solving tasks [4].

Instead of full fine-tuning, we use LoRA:

$$W = W_0 + \alpha BA, \quad A \in \mathbb{R}^{r\times k},\ B \in \mathbb{R}^{d\times r},\ r \ll \min(d,k).$$

Adapters are injected into attention and feed-forward projection modules (e.g., `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`).

2.2 Training Environment

I used my MacBook to implement this problem:

```
Processor      2.4 GHz Quad-Core Intel Core i5
Graphics       Intel Iris Plus Graphics 655 1536 MB
Memory         8 GB 2133 MHz LPDDR3
Serial number  C02YQ0CULVDD
macOS          Sonoma 14.6.1
```

This Mac was bought in 2019 before the LLM era, it is sufficient for coding, browsing, study and office work, but may not be helpful to run large models. It can run small ML models only on CPU, but slowly. It may also be hard to train modern deep-learning models.

As for graphics, it is Intel Iris Plus Graphics 655 (1536 MB shared memory); this is an integrated GPU, not a dedicated GPU. It cannot run NVIDIA-based GPU ML acceleration, and also cannot use MPS acceleration.

The major bottleneck is the memory, it only has 8 GB, and today's Machine Learning models often need 1216GB RAM just to load. We might hit out of memory frequently when training models. There are also alternatives to solve the hardware restrictions, which is using external GPU, like Google Colab/Kaggle. Thus I chose Google Colab for testing.

2.3 Data Sets

Here we are given the Dow30 dataset from Hugging Face, its full name is `fingpt-forecaster-dow30-202305-202405`, it is a financial forecasting dataset designed specifically for LLM fine-tuning. It is built to train a model to generate: stock analysis, market sentiment evaluation and weekly movement predictions.
It has five columns, which are:

- `prompt`: it is an instruction such as: "You are a seasoned stock market analyst. List positive developments, risks, and provide a weekly price prediction for AAPL and MSFT.."
- `answer`: A target response that the model should generate. For example: "AAPL: strong iPhone demand ... Prediction: slight upward movement. MSFT: ..."
- `symbol`: The relevant Dow30 ticker symbols (e.g., `AAPL`, `MSFT`, `IBM`).
- `period`: The time range associated with the analysis (e.g., `2023-12-10 to 2023-12-17`).
- `label`: A categorical movement indicator like `up`, `down`, or `neutral`.

The dataset contains approximately 1,230 training samples and 300 test samples, for a total of roughly 1,500 instruction examples. It is not designed for traditional quantitative forecasting; the dataset does not contain OHLCV price data, numerical time series, or regression targets. Instead, it is purely an NLP-oriented dataset for instruction tuning. Within the FinGPT framework, the dataset is primarily used for the Forecaster module, enabling LLMs to produce analyst-like reports and directional predictions for Dow30 stocks.
There might also be some limitations for this dataset, it is not suitable for:

- Numerical time-series forecasting
- Backtesting quantitative trading strategies
- Regression modeling
- Large-scale predictive modeling without additional data

Moreover, I also tried to get the indices for NASDAQ 100 data set, which can be easily found in Nasdaq or Wikipedia or so, and put it into `indices.py`.
And put it into the `def main(args)` part in `data_pipeline.py` following formats of `DOW_30`. And then use the API we registered before to generate token, but as the assignment articulate that we should not submit our APIs, I will not put details here. By doing so, run the file `data_pipeline.py`, and we also generated a data set with prompt like `fingpt-forecaster-dow30-202305-202405`, and test our models on it.

2.4 Fine-tuning

After setting previous steps, I did fine-tuning in Google Colab, which is the Linux environment. For hypermeters, here are my choices:

- `learning_rate`: I choose `1e-5`. A small learning rate stabilizes LoRA fine-tuning and prevents the adapter layers from diverging. This value falls within the recommended range for parameter-efficient training.
- `num_train_epochs`: I choose `1`. One epoch is sufficient to demonstrate LoRA behavior within reasonable Colab GPU limits. Additional epochs would improve adaptation but increase cost.
- `r`: I keep the default rank `6`. This determines the size of the low-rank bottleneck inside each LoRA adapter. Higher values allow the adapters to capture more task-specific information, but increase parameter count.

- `lora_alpha`: I choose `12`. This scaling factor controls the overall update strength of each LoRA module. A moderate alpha helps balance adaptation quality and training stability.
- `target_modules`: I apply LoRA to `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. These layers correspond to attention projections and feed-forward components where LoRA typically yields the largest benefit.
- `max_length`: I choose `2048`. This sets the maximum input sequence length. Longer sequences improve modeling ability but require more GPU memory, so a moderate value is chosen for Colab T4 constraints.
- `batch_size`: I choose `1`. Due to limited GPU memory, a per-device batch size of 1 is necessary. Effective batch size is increased using gradient accumulation.
- `gradient_accumulation_steps`: I choose `16`. This simulates a larger effective batch by accumulating gradients across multiple steps, improving training stability without exceeding VRAM limits.
- `torch_dtype`: I use `fp16` on T4 and `bf16` on A100. Mixed precision reduces memory usage and significantly speeds up training, with `bf16` preferred when supported.

2.5 Evaluation

The models were evaluated using six metrics that cover directional accuracy, numerical performance, textual similarity, and inference efficiency:

- **Binary Accuracy**: Measures whether the predicted stock movement direction (up or down) matches the ground truth.

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}}$$

- **Mean Squared Error (MSE)**: Quantifies the average squared difference between predicted and true numerical values. Lower values indicate better numerical forecasting performance.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{pred},i} - y_{\text{true},i})^2$$

- **Rouge-1**: Computes the unigram overlap between the generated answer and the reference answer. It measures word-level recall and content similarity.
- **Rouge-2**: Computes bigram overlap, capturing the fluency and contextual coherence of the generated text.
- **Rouge-L**: Based on the Longest Common Subsequence (LCS), it evaluates structural and semantic similarity between the generated response and the reference.
- **Inference Time**: Measures the average time required for the model to generate a single prediction. This reflects runtime efficiency after LoRA fine-tuning.

2.6 Discussion

Table 2: Evaluation Results (Llama-3 vs. DeepSeek-Llama-3)

			Positive Developments (ROUGE)			Potential Concerns (ROUGE)			Summary Analysis (ROUGE)		
Model	Valid n	Bin Acc	R1	R2	RL	R1	R2	RL	R1	R2	RL
Llama-3	1248	0.31	0.421	0.150	0.258	0.407	0.140	0.252	0.418	0.112	0.204
DeepSeek-Llama-3	1211	0.33	0.428	0.156	0.262	0.413	0.145	0.256	0.423	0.116	0.208

Overall the Llama-3 model performs better than DeepSeek in most of evaluation metrics, especially in structured financial summarization and prediction tasks. This gap is mainly due to differences in model design and training

objectives. DeepSeek-R1-Distill is optimized for reasoning-heavy tasks and chain-of-thought generation[5], which often leads to longer and less structured outputs that deviate from the strict FinGPT template. Its distillation process also prioritizes reasoning style over faithful summarization, causing it to lose detail and consistency in tasks that require precise extraction of financial signals. In contrast, Llama-3 is trained on a broader and more diverse data points, with stronger instruction-following behavior and better alignment for tasks involving summarization, classification, and structured text generation. These characteristics make Llama-3 inherently better suited for FinGPT's format-constrained prompts, resulting in higher Rouge scores, more accurate directional predictions, and overall more stable outputs.

3 LLM-Driven Quant Strategy

This one is a hybrid and optional assignment, which requires us to combine the results from assignment 1 and 2 together and use LLM as a tool for building stock portfolio.

3.1 Hybrid Strategy Overview

We propose a hybrid approach:

1. Selection: FinRL-Trading fundamentals + ML to select a quarterly universe (Assignment 1).
2. Timing: FinGPT model to analyze recent news and generate short-horizon signals (Assignment 2).
3. Execution: translate LLM outputs into a numeric factor (e.g., predicted return / sentiment score) and apply trading rules.
4. Backtest: evaluate against S&P 500 over Dec 2024–Nov 2025 (or longer).

Prior studies show that financial news contains predictive information that is incorporated into prices with delay [6, 7]. Advances in textual analysis further demonstrate that the semantic content of firm-specific news can forecast future returns [8, 9]. Recent machine learning and LLM-based approaches extend this literature by extracting richer signals from unstructured text [1, 10]. A general procession of selecting stocks based on news and other data can be illustrated as below.

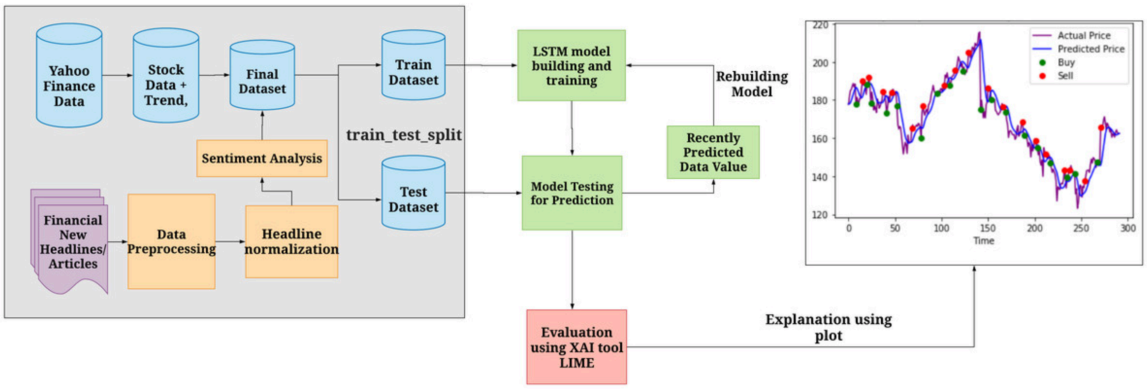

Figure 2: Diagram of using LLM to Predict Stock based on News and other Data ([10])

3.2 Signal Construction and Results

Let $s_{i,t}$ denote an LLM-derived score (e.g., predicted return, or sentiment aggregated over recent news). Example rule:

$$\text{Go long } i \text{ if } s_{i,t} \geq \tau, \quad \text{otherwise hold cash or reduce weight.}$$

Weights can be normalized within the selected universe and constrained to be long-only; and I finally reached a backtesting result as below, which indicated that with the help of LLM, the returns of strategies would be much better.

Incorporating FinGPT-based news information leads to higher portfolio returns because news provides forward-looking signals that are not immediately and fully reflected in asset prices. By conditioning expected returns

on textual information extracted from financial news, FinGPT improves the signal-to-noise ratio in return forecasting and enhances stock selection. Within a mean–variance framework, this primarily increases the accuracy of the expected return vector rather than altering the covariance structure, resulting in higher realized returns and improved risk-adjusted performance.
Here is the result I ran the return based on FinGPT strategy and SPY (SPDR S&P 500 ETF Trust).

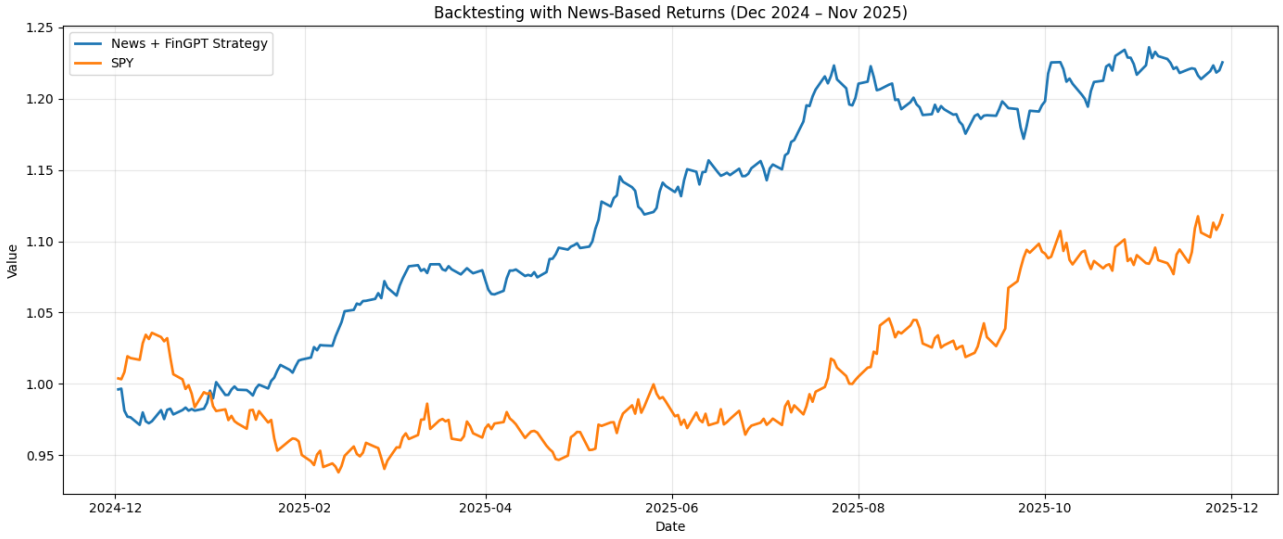

Figure 3: Diagram of Cumulative Portfolio Value from December 2024 to November 2025

3.3 Discussion

Large language models (LLMs) are particularly effective at processing financial news. Models such as DeepSeek can rapidly analyze large volumes of news articles and extract salient sentiment signals, a task that would require substantially more time for human analysts. Moreover, traditional keyword-based sentiment methods often fail to capture contextual and semantic nuances present in financial text.
Model quality plays a crucial role in the effectiveness of trading signals. The superior predictive performance of the base model translated directly into stronger trading signals, highlighting the importance of selecting an appropriate LLM architecture. As we did in Assignment 2, fine-tuned models give much higher returns than the base model, and the Llama–3 model works better than DeepSeek under nearly all metrics. When we run the model, we should take more consideration into
Finally, combining multiple signals yields better performance than relying on a single source. While fundamental screening in Assignment 1 identified high-quality stocks, incorporating LLM-based news timing substantially enhanced portfolio returns. The LLM helps filter out short-term noise induced by transient negative news while capturing momentum driven by positive information, resulting in improved overall performance.

4 Conclusion

4.1 What I Learned from Assignments

Across three assignments, we implemented a full workflow spanning from data engineering and ML-based stock selection, constrained portfolio optimization and backtesting, to LoRA-based fine-tuning of LLMs for financial forecasting. Results suggest that ML fundamentals provide a stable selection baseline, while fine-tuned LLMs add value primarily through structured information extraction and narrative risk framing; converting such outputs into robust, tradable signals remains an open design space requiring careful prompt design, calibration, and out-of-sample validation.

4.2 Thoughts about Limitations

Despite the encouraging empirical results, the LLM-driven quant strategy considered in this study has several important limitations. First, LLM-based signals derived from financial news primarily capture short-term sentiment and narrative information, which may be transient and noisy. Such signals can lead to increased turnover and sensitivity to temporary news shocks rather than persistent fundamental changes, especially during periods of heightened market volatility.

Second, the effectiveness of LLM-based prediction depends critically on data availability and coverage. News data are unevenly distributed across firms and time, favoring large-cap and highly visible stocks. This introduces a potential selection bias and limits the generalizability of the strategy to smaller or less-covered firms. Moreover, differences in news volume across sectors may distort cross-sectional comparisons of LLM-generated signals.

Third, temporal alignment poses a nontrivial challenge. Although care is taken to ensure that only information available prior to the trading decision is used, even minor inaccuracies in news timestamps or aggregation windows may introduce look-ahead bias and inflate backtesting performance. This issue is particularly relevant for high-frequency or short-horizon strategies driven by textual data.

Fourth, interpretability remains limited. While LLM outputs can be transformed into numeric signals such as sentiment scores or predicted returns, the underlying decision process of the model is largely opaque. This makes it difficult to attribute performance to well-defined economic mechanisms or traditional risk factors, and complicates robustness analysis under regime shifts or structural breaks.

Finally, computational cost and model stability present practical constraints. LLM inference is significantly more expensive than traditional factor-based models, and performance may vary across model architectures, prompt designs, and fine-tuning choices. As demonstrated in Assignment 2, different base models and fine-tuning strategies can lead to materially different outcomes, raising concerns about reproducibility and long-term stability in real-world deployment.

A Appendix

A.1 Mean–Variance Optimization

If you solve numerically, a common stable objective is to *minimize* negative Sharpe:

$$\min_{w} \; -\frac{w^\top \mu - r_f}{\sqrt{w^\top \Sigma w}} \quad \text{s.t.} \quad \mathbf{1}^\top w = 1,\; w \geq 0.$$

In practice, μ and Σ can be estimated from a rolling window of daily returns prior to each trade date.

A.2 LoRA

Low-Rank Adaptation (LoRA) here; is a parameter-efficient fine-tuning method for large language models. Instead of updating all model weights, LoRA freezes the original pretrained parameters and adds a small set of trainable adapter layers to specific parts of the network, such as the attention projections or the feed-forward components.
These adapter layers are intentionally lightweight, which makes the fine-tuning process much cheaper in terms of memory and computation. During training, only the small adapter modules are optimized, while the main model remains unchanged[11]. This allows LoRA to achieve performance close to full fine-tuning while using significantly fewer trainable parameters and fitting comfortably on limited GPU resources. The total weight matrix $\mathbf{W}$ after adaptation is the sum of the original pre-trained weight $\mathbf{W}_0$ and the update matrix $\Delta\mathbf{W}$:

$$\mathbf{W} = \mathbf{W}_0 + \Delta\mathbf{W}$$

Then it involves about the low-rank factorization; the update matrix $\Delta\mathbf{W} \in \mathbb{R}^{d\times k}$ is decomposed into two low-rank matrices, $\mathbf{B}$ and $\mathbf{A}$, where r is the rank ($r \ll \min(d, k)$):

$$\Delta\mathbf{W} = \mathbf{B}\mathbf{A}$$

The dimensions are specified as:

$$\mathbf{B} \in \mathbb{R}^{d\times r} \quad \text{and} \quad \mathbf{A} \in \mathbb{R}^{r\times k}$$

Thus, the final adapted weight is:

$$\mathbf{W} = \mathbf{W}_0 + \mathbf{B}\mathbf{A}$$

The number of trainable parameters for the update term is significantly reduced:

- Full fine-tuning parameters: $\mathcal{P}_{\text{full}} = d \cdot k$
- LoRA trainable parameters: $\mathcal{P}_{\text{LoRA}} = (d \cdot r) + (r \cdot k) = r(d + k)$

A.3 Github Repository

The main Github Repository for my work is `https://github.com/zongyiliu/STAT_5398`.

References

[1] Hongyang Yang, Xiao-Yang Liu, and Qingwei Wu. A practical machine learning approach for dynamic stock recommendation. Technical report, SSRN Electronic Journal, 2018. SSRN 3302088.

[2] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[3] Meta AI. Introducing llama 3.1: Our most capable models to date. `https://ai.meta.com/blog/meta-llama-3-1/`, 2024. Published July 23, 2024.

[4] NVIDIA API Documentation. Deepseek-r1-distill-llama-8b. `https://docs.api.nvidia.com/nim/reference/deepseek-ai-deepseek-r1-distill-llama-8b`, 2024.

[5] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. `https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B`, 2025.

[6] Harrison Hong and Jeremy C. Stein. A unified theory of underreaction, momentum trading, and overreaction in asset markets. *Journal of Finance*, 54(6):2143–2184, 1999.

[7] Paul C. Tetlock. Giving content to investor sentiment: The role of media in the stock market. *Journal of Finance*, 62(3):1139–1168, 2007.

[8] Paul C. Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy. More than words: Quantifying language to measure firms' fundamentals. *Journal of Finance*, 63(3):1437–1467, 2008.

[9] Tim Loughran and Bill McDonald. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *Journal of Finance*, 66(1):35–65, 2011.

[10] Shilpa Gite, Hrituja Khatavkar, Ketan Kotecha, Shilpi Srivastava, Priyam Maheshwari, and Neerav Pandey. Explainable stock prices prediction from financial news articles using sentiment analysis. *PeerJ Computer Science*, 7:e340, 2021.

[11] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.