Context-aware Stock Market Movement Prediction: a MAML and Incremental Learning Approach

Stanford CS 330 Fall 2023 - Final Project Report
Project Mentor: Alex Sun

Irina Alexandra Marton
December 12, 2023

Abstract

The stock market's complexity and its sensitivity to a multitude of factors have long posed a challenge for predictive modeling. This study introduces an innovative approach that combines Model-Agnostic Meta-Learning (MAML) with Incremental Learning, implemented in a stacked Long Short-Term Memory (LSTM) model, to enhance the prediction of stock market movements. Our methodology hinges on two key components: a foundational training phase leveraging the MAML algorithm and a subsequent phase of Incremental Learning. The study utilizes a robust dataset from Yahoo! Finance, comprising Adjusted Closing historic prices of NASDAQ companies with significant market capitalization. This dataset spans over 3143 trading days and covers 216 companies across various industries. The models explored include a baseline model StockBot*, an original "context-aware" input inspired by the few-shot classification input which we tested with the ContextLSTM model, and our novel MetaContext model.

StockBot*, the baseline model, an adaptation of Mohanty et al.'s algorithm, provided promising generalization to unseen data but was limited in its decision-making, failing to offer realistic trading strategies. The ContextLSTM model, trained on "context-aware" inputs, marked a considerable improvement in performance, achieving a 96% accuracy rate compared to StockBot*'s 66%. This improvement was attributed to the model's ability to classify data into actionable trading decisions (buy, sell, hold), as opposed to merely predicting directional movements. The MetaContext model, treating each company's stock as a separate classification task, displayed an even closer alignment with ideal trading strategies, especially in short-term predictions. However, the model struggled with long-term accuracy and encountered concept drift during the online learning phase.

ContextLSTM's improved performance underscores the value of reformulating the prediction problem into a classification task and augmenting input data. The MetaContext model, while innovative in its approach to treat each stock as a unique task, highlights the challenges posed by concept drift in dynamic financial environments. This research contributes significantly to the field of financial machine learning by demonstrating the potential of meta-learning in stock market prediction. The findings suggest that while these models can adapt to market conditions and improve prediction accuracy in the short term, they require further refinement to address long-term prediction challenges and concept drift. Future work should explore more sophisticated online learning strategies, potentially incorporating techniques like Incremental LSTM, to enhance the models' adaptability and resilience in the face of continually evolving market dynamics. The successful implementation of such models could revolutionize stock market analysis, offering more reliable and adaptable tools for investors and analysts. This study not only advances our understanding of machine learning applications in finance but also opens up new avenues for research in developing more robust and efficient predictive models for complex, time-sensitive financial data.

1 Introduction

1.1 Motivation

There have been numerous attempts at designing a machine learning model that can accurately predict stock market prices and make profitable investment decisions. However, the high complexity of the stock market and its dependencies on a high amount of factors has made it difficult to develop a model that is able to completely understand the stock market's movements.

1.2 Objective

The objective is to formulate an algorithm capable of adapting to a diverse range of stock market conditions. The approach entails meta-training a stacked LSTM model utilizing MAML and following foundational training with a regimen of incremental learning, systematically applied at predetermined intervals. This methodology is engineered to predict directional movements in stock prices.

2 Related Work

2.1 Long Short-Term Memory (LSTM) Models for stock market predictions

Due to the ability to effectively handle time-series data, LSTM models are one of the preferred methods for making predictions based on stock market data, as presented by Zou et al. [1] in their survey about deep learning techniques used for stock market predictions. In their survey, there is a multitude of papers that implement LSTM models on their own, in combination with other methods, or in hybrid systems for performing different stock market-related tasks such as opening, lowest and highest price prediction, and stock movement prediction. Nelson et al. [2] perform stock movement prediction with a model composed of one LSTM layer that is trained each day on historical price data. By inputting 180 features with a rolling window of 10 months of prior trading data, they achieve an average of 55.9% accuracy when predicting if the price is going to go up or down for a particular stock. Although 55.9% may seem like a high accuracy, a 44.1% risk is considered extremely high in regard to investments. Moreover, simply predicting up or down movement does not translate into a realistic trading strategy when taking into account transaction costs. Therefore, a model should be able to predict up, down, or sideways movement.

Another LSTM based model conceived for stock predictions, but not included in the survey, is Mohanty et al.'s [3] StockBot. They introduced an algorithm that classifies stock data examples into buy, sell, or hold which was able to generalize by being trained on multiple stocks from the same industry. This approach facilitated knowledge transfer, enabling accurate predictions on previously held-out stocks from the same industry. Even though the results were promising, there was an absence of experiments where the model was trained across several industries.

2.2 Labeling Financial Time Series

The most common way of labeling stock data is done by evaluating short-term fluctuations. Such an example is StockBot's decision algorithm that looks at the stock prices forecast, computes the nature of change, and then the curvature of the forecast ending with a -2 (buy), 2 (sell), or 0 (hold) values.

Algorithm 1: StockBot decision making algorithm

1: Obtain the predicted trajectory, **c** (stock forecast for forward_look days)
2: Compute the nature of change: $\delta_i = \text{sign}(c_{i+1} - c_i)$
3: Compute the curvature of the forecast: $\Delta_i = \delta_{i+1} - \delta_i$
4: Make the decision: Decision = $\begin{cases} \Delta_i = 2 \rightarrow \text{Sell (indicates local maxima)} \\ \Delta_i = 0 \rightarrow \text{Hold (indicates no change)} \\ \Delta_i = -2 \rightarrow \text{Buy (indicates local minima)} \end{cases}$

However, in practice, this is unworkable when taking into account that different investors are interested in implementing different trading strategies and have different judgments on continuous trends for

stocks. Therefore, Wu et al. [4] formulated the "Auto-Labeling Data" algorithm which has the parameter ω for accounting for a certain proportion increase or decrease from the current highest or lowest price. Additionally, the paper also proves that models such as LSTM and GRU are more suitable for making predictions on financial time series data.

Algorithm 2: Auto-Labeling Data

Input: Original Time series data $X = [x_1, x_2, x_3, ..., x_N]^T$, $\omega > 0$, which represents the proportion threshold parameter of the trend definition

Output: Label vector $Y = [label_1, label_2, label_3, ..., label_N]^T$

Initialization: $FP = x_1$, the first price obtained by the algorithm;
$x_H = x_1$, the highest price; $HT = t_1$, time of xH;
$x_L = x_1$, the lowest price; $LT = t_1$, time of xL;
$Cid = 0$, current direction of labeling;
$FP_N = 0$, the index of the highest or lowest point obtained initially.

for $i = 1$ to N
  if $x_i > FP + x_1 * \omega$
    Set $[x_H, HT, FP\_N, Cid] = [x_i, t_i, i, 1]$ and end for
  if $x_i < FP - x_1 * \omega$ then
    Set $[x_L, LT, FP\_N, Cid] = [x_i, t_i, i, -1]$ and end for
for $i = FP\_N + 1$ to N
  if $Cid > 0$
    if $x_i > x_H$
      Set $[x_H, HT] = [x_i, t_i]$
    if $x_i < x_H - x_H * \omega$ and $LT \leq HT$
      for $j = 1$ to N
        if $t_j > LT$ and $t_j \leq HT$
          Set $Y_j = 1$
      Set $[x_L, LT, Cid] = [x_i, t_i, -1]$
  if $Cid < 0$
    if $x_i < x_L$
      Set $[x_L, LT] = [x_i, t_i]$
    if $x_i > x_L + x_L * \omega$ and $HT \leq LT$
      for $j = 1$ to N
        if $t_j > HT$ and $t_j \leq LT$
          Set $Y_j = -1$
      Set $[x_H, HT, Cid] = [x_i, t_i, 1]$

2.3 Model-Agnostic Meta-Learning

MAML, the meta-learning algorithm developed by Finn et al. [5] represents a significant advancement in meta-learning by providing a framework that seeks to prepare models for fast adaptation across a variety of tasks, requiring small amounts of new data. During image classification experiments that were carried out, it outperforms other meta-learning algorithms such as MANN [6] and the meta-learner LSTM. However, more experimentation is required in order to conclude if this technique can be applied to any problem and any model.

2.4 Incremental learning

A machine learning model can be either trained with the whole dataset from the start, or in small batches in cases where we have a real time stream of data that may be infinite. In the case of stock market data, the former case can not be applied as data continues to arrive in a sequential manner as time progresses and the market evolves. The process where the model learns and updates its parameters after each such small batch is known as incremental or online learning. Murphy [7] implements an online portfolio selection algorithm and emphasizes that online learning is even more significant for high frequency action strategies as the model needs to adapt faster to the new market conditions.

3 Methodology

3.1 Dataset

The data used in this project was obtained from Yahoo! Finance and is comprised of the Adjusted Closing historic prices for stocks of companies from NASDAQ with significant market capitalization, specifically companies with over $10 billion market capitalization. Therefore, the dataset encompasses stock market data from January 1st, 2011, to June 30th, 2023, the equivalent of 3143 trading days, covering 216 companies across 10 different industries. All data is normalized using the scikit-learn MinMaxScaler and labeled according to the desired trading strategy by using the auto-labeling data algorithm designed by Wu et al. [4].

3.1.1 Datasets Split

Out of the 3143 trading data days, the last 250 entries that represent a year of trading, were kept from the training dataset as this will be our test period. The remaining entries for each ticker have an 80% - 20% split between the training and validation sets for Stockbot* and ContextLSTM. For MetaContext, the tickers from each industry were split 80% - 20% to form the training and validation tasks datasets. From all models, the stock data for Apple Inc. was withheld in order to evaluate each model's ability to generalize on top of adapting to the new market conditions of the unseen 250 days of trading.

After the initial meta-training phase, online learning will be implemented for MetaContext for the remaining 250 test days by splitting the dataset into batches of 25 days. As before, the same 80%-20% ticker split across industries will be maintained.

Online learning batches:

- 1^{st} batch: data from the 1st-25th day of the test dataset
- 2^{nd} batch: data from the 26st-50th day of the test dataset
- ... etc.

3.2 Trading strategy

The value of ω used by Wu et al. [4] in their experiments when labeling the stock market data was 0.15, which accounts for a 15% change in stock price. Considering that mature companies such as the ones selected for our experiments usually have a single-digit goal for year-to-year % growth, the strategy was adjusted to a 2.5% change in order to formulate our ideal trading strategy for the year-long trading testing period.

3.3 StockBot*

Our baseline model StockBot* is the model developed by Mohanty et al. [3], with a slight modification to its decision making algorithm. The buy, sell, and hold decisions were generated using the auto-labeling data algorithm by Wu et al. [4] in order to align StockBot*'s output to our desired trading strategy.

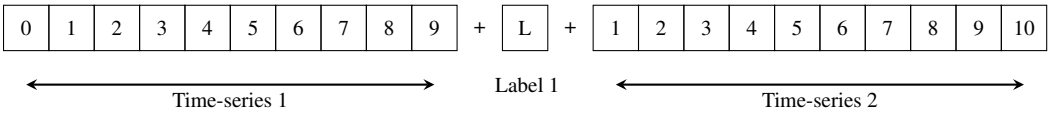
3.3.1 StockBot* Model Summary

The model architecture of Stockbot* was selected as the top-performing one from the experiments done by Mohanty et al. [3].

StockBot* Model Architecture		
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10, 20)	1760
lstm_1 (LSTM)	(None, 10, 20)	3280
lstm_2 (LSTM)	(None, 10, 20)	3280
lstm_3 (LSTM)	(None, 20)	3280
dense (Dense)	(None, 1)	21
		Total params: 11,621
		Trainable params: 11,621
		Non-trainable params: 0

3.4 "Context-awareness"

Inspired by the Few-Shot Classification input format, through simple feature engineering, the "context-aware" input was created. It incorporates the first time series of the specified rolling window size, concatenated with its corresponding label, and then concatenated with subsequent time series of the same widow size for which the label needs to be predicted. Example of "context-aware" input with a rolling window size of 10:



3.5 ContextLSTM

By reformulating the price prediction problem of StockBot* into a numerical data classification problem, ContextLSTM was formulated. It takes in the "context-aware" input and outputs a buy, sell or hold decision. For the "context-aware" input, the data was labeled using the same auto-labeling algorithm developed by Wu et al. [4] in order to train the model on the same 2.5% change desired trading strategy. The model architecture was kept mostly the same as for StockBot*, having the same number of LSTM layers, however, the Dense output layer was modified to accommodate the problem formulation change and therefore has an output shape of (None, 3).

3.6 MetaContext

Inspired by the Omniglot classification MAML meta-training experiment [8], the MetaContext model input was constructed in a similar fashion. While ContextLSTM treats all stocks as a single task, the MetaContext model keeps the model architecture of ContextLSTM, but treats each company's stock as a different classification task. Formulated as a 1-shot 3-way classification problem, the input consists of 3 support examples and 1 query from the same stock ticker. As highlighted by Antoniou et al. [9], MAML is quite unstable and in order to rectify this some of the proposed methods were applied to the model. Per-step batch normalization running statistics were implemented along with derivate-order annealing, which was done by reducing the number of inner loop steps to 1.

3.7 Evaluation

Both the baseline model Stockbot*, the ContextLSTM and MetaContext's performance will be evaluated on the return on investment (ROI), an often used return-based evaluation metric in financial settings [1].

$$\text{ROI} = \frac{\text{Sell Value of Stock} \ - \ \text{Buy Value of Stock}}{\text{Buy Value of Stock}} \times 100$$

Additionally, the performance of these models will be benchmarked against the *buy-and-hold* strategy. This strategy, widely regarded as a foundational benchmark in stock trading models, involves purchasing a stock at the outset of a specified period and retaining it until the period's conclusion [7].

Moreover, additional metrics for evaluation were gathered by using sklearn's classification report function. These metrics are accuracy, precision, recall, and f1-score, and were calculated by comparing our ideal trading strategy labels to the predicted labels by each model.

4 Experiments

4.1 Stockbot* Results

Our baseline model Stockbot* was trained for this experiment on the selected NASDAQ dataset with the 80% - 20% split between the training and validation sets and a rolling window size of 10. The stock AAPL was withheld during training, in order to evaluate the model's performance and adaptability on it. The predictions for the withheld stock were processed by the auto-labeling algorithm to generate buy, sell, or hold decisions. The ROI was plotted and benchmarked against the *"buy-and-hold"* strategy. The below plots represent the actual vs predicted normalized stock prices and the ROI % for the *buy-and-hold* strategy vs StockBot*'s predictions.

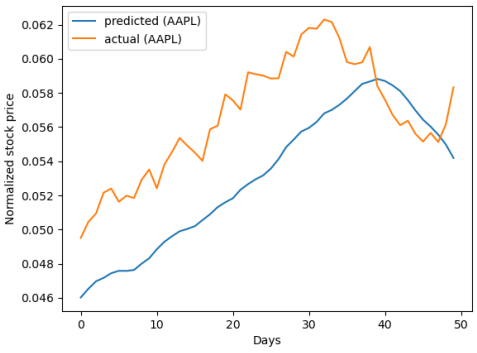


Figure 1: StockBot* price predictions

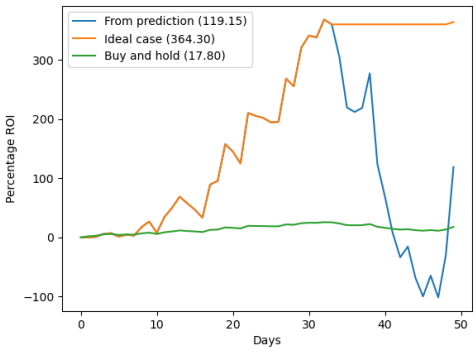


Figure 2: StockBot* generated decisions

As expected, Stockbot* was able to generalize to previously unseen data and make some accurate predictions, however as we can see from the metrics, the decision algorithm is actually unable to make any sell actions from the predicted prices. While the predictions generally follow the up and down trends, they are not accurate enough to implement a specific trading strategy as desired.

StockBot* Scores				
	precision	recall	f1-score	support
buy	0.82	0.94	0.88	34
sell	0.00	0.00	0.00	15
hold	0.09	1.00	0.17	1
accuracy			0.66	50
macro avg	0.30	0.65	0.35	50
weighted avg	0.56	0.66	0.60	50

4.2 ContextLSTM Results

Our classification model was trained both on the "context-aware" inputs with a rolling window size of 10 for its sequences, generated from the same NASDAQ dataset, as well as a "no-context" input which is the same input sequence that Stockbot* was trained on.

As we can observe in the plots below, the "context-aware" input significantly improves the performance of our classification model when predicting buy, sell, and hold labels. The ContextLSTM model trained on "context-aware" inputs has a 96% accuracy score, compared to 68% when it had the same input as Stockbot*. It is noted that we observe a 2% increase in accuracy by simply formulating the problem as classification, rather than performing regression and generating labels based on the predictions.

At first glance, it might be tempting to allocate the increase in accuracy between the two different inputs that ContextLSTM was tested on, to the additional features present in the "context-aware" input as the newly constructed input has 21 features as opposed to 10. However, on tests run with input sequences with a rolling window size of 21, the model's performance took a nose dive and obtained an accuracy score of 48% during experiments.

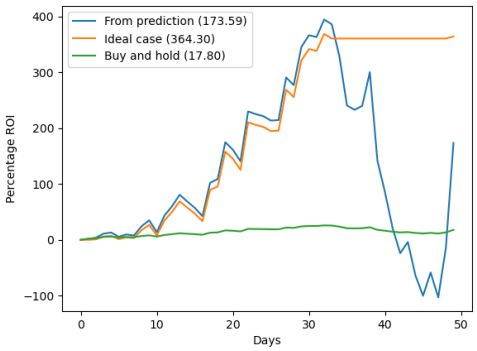


Figure 3: Predictions on sequence input

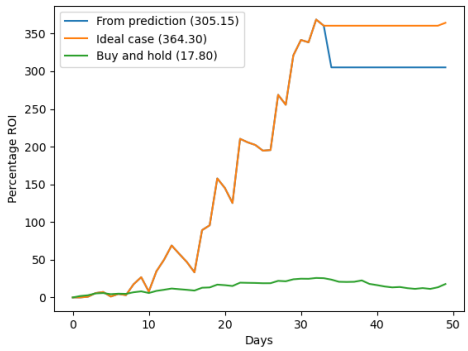


Figure 4: Predictions on "context-aware" input

ContextLSTM Scores - sequence input				
	precision	recall	f1-score	support
buy	0.68	1.00	0.81	34
sell	0.00	0.00	0.00	15
hold	0.00	0.00	0.00	1
accuracy			0.68	50
macro avg	0.23	0.33	0.27	50
weighted avg	0.46	0.68	0.55	50

ContextLSTM Scores - "context-aware" input				
	precision	recall	f1-score	support
buy	0.97	0.97	0.97	34
sell	0.93	0.93	0.93	15
hold	1.00	1.00	1.00	1
accuracy			0.96	50
macro avg	0.97	0.97	0.97	50
weighted avg	0.96	0.96	0.96	50

From comparing the plot in Figure 2 with the one in Figure 4, along with Stockbot*'s and ContextLSTM's metrics we observe that by reformulating it into a classification problem and by augmenting the input for the model, without making any significant modification to the architecture, we have obtained a significantly better-performing model when it comes to predicting a desired trading strategy. What both models still maintain in common is the inability to stick to the ideal trading strategy path past the 33rd trading day in the future from the last training date.

4.3 MetaContext Results

Three experiments were conducted regarding the task composition during the initial training of MetaContext. The model was trained during all tasks on the same training dataset with tasks split 80% - 20% between training and validation.

In **experiment A** after randomly sampling a task (stock) the query example is then selected at random from the task's dataset. After removing the query example, examples with the labels buy, sell, and

hold are selected at random from the remaining dataset. In **experiment B**, after randomly selecting the query example, the support examples are selected as the most recent 3 entries in the dataset. In **experiment C**, after randomly selecting the query example, the support examples are selected as the most recent entry for each buy, sell, and hold label.

Experiments with 3, 5 and 10 support examples were conducted for experiment B and it was concluded that by increasing the number of support examples does not improve model's performance, as in those experiments the training accuracy and validation accuracy were lower during training for the models trained on 5 and 10 support examples.

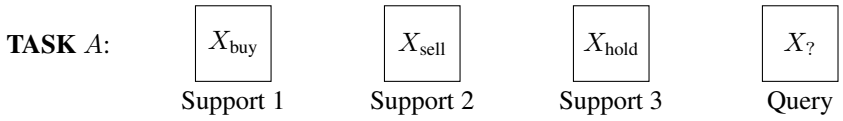


Figure 5: Task A composition

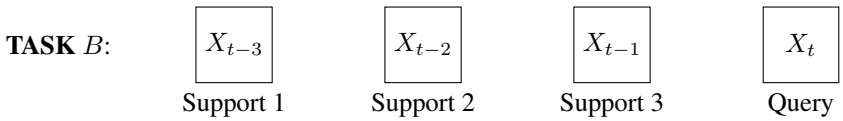


Figure 6: Task B composition

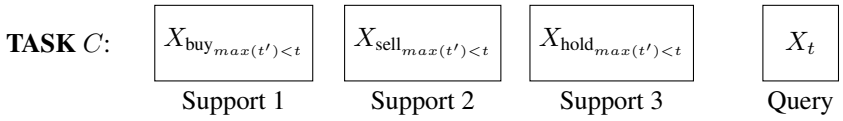


Figure 7: Task C composition

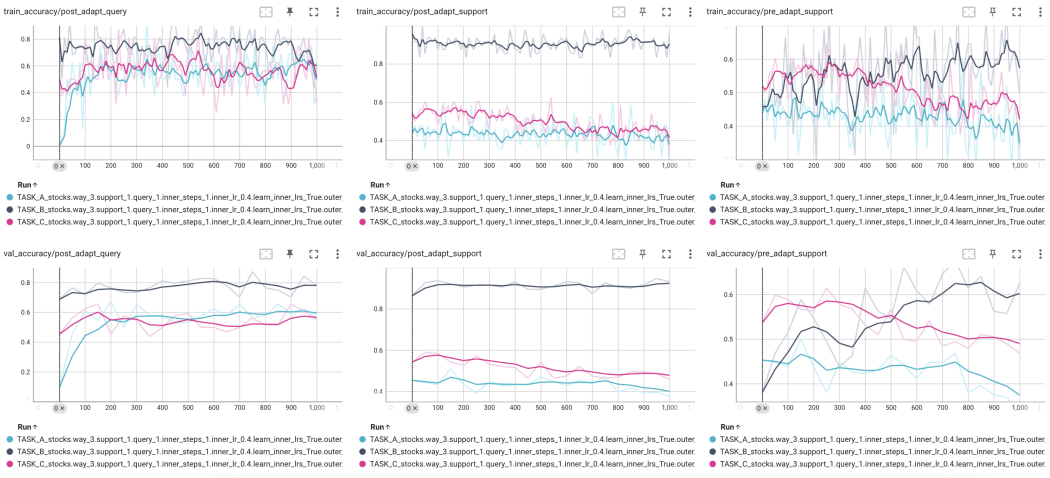


Figure 8: Initial training results for Task A (light blue), Task B (navy blue) & Task C (pink)

From the experiments, we observe that the task construction from experiment B has the best performance during training and validation despite not always being shown a support example for each buy, sell, and hold label, as in task B composition it is not guaranteed that the most recent 3 days all have different labels.

Moreover, by looking at Figure 9 and the outputted scores, we can remark that even though ContextLSTM and MetaContext had the same overall score, the MetaContext model was actually closer to the desired trading strategy than ContextLSTM, as indicated by the ROI obtained from the predicted actions. While it did not experience the same 33rd-day cut-off from correct predictions as the previous two models, it was however unable to identify the hold label. This might be due to some dataset label

imbalance across all tasks in the training set caused by the random sampling in conjecture with the task composition in experiment B.

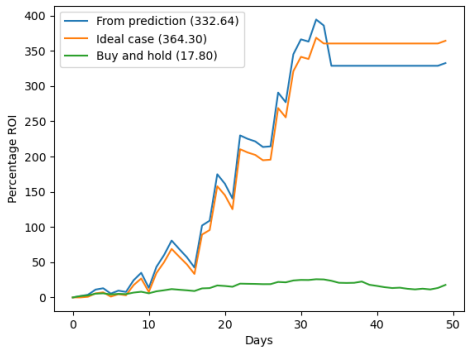


Figure 9: Predictions after initial training for Task B

MetaContext Scores Task B - initial training				
	precision	recall	f1-score	support
buy	0.94	1.00	0.97	34
sell	1.00	0.93	0.97	15
hold	0.00	0.00	0.00	1
accuracy			0.96	50
macro avg	0.65	0.64	0.65	50
weighted avg	0.94	0.96	0.95	50

4.4 Online learning results

As discussed in Section 3, the initial meta-training was followed by an online learning strategy in batches of 25 trading days. While the initial training lasted 1000 epochs with a batch size of 16, the online learning for each new batch was done for 200 epochs with a batch size of 8 and a lower learning rate for the outer loop.

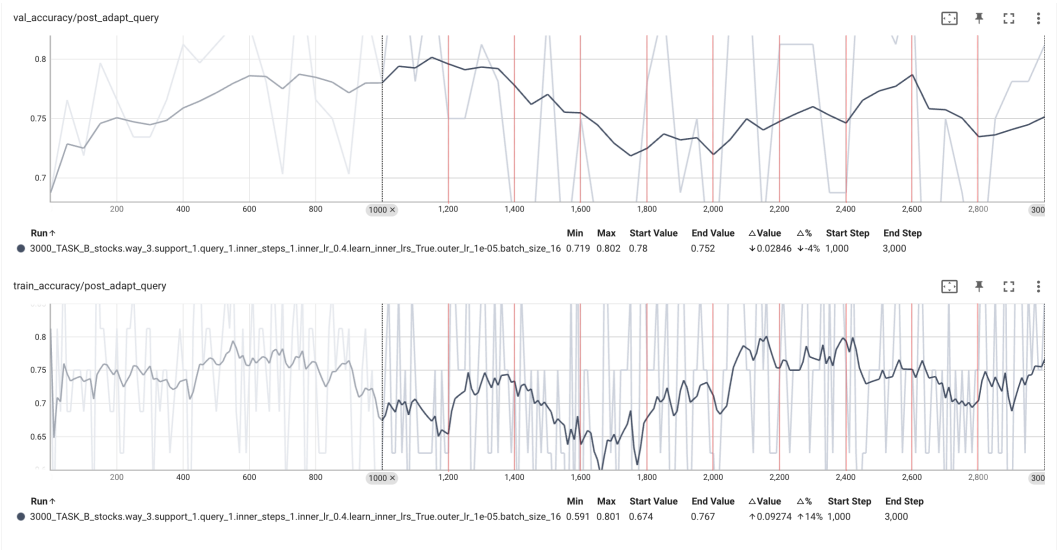


Figure 10: Online learning: validation & training accuracies

From Figure 10 it is apparent that despite the fact that some preventive measures have been taken in order to mitigate MAML's related issues, we can still observe significative instability during online training. During the initial online learning batches, MetaContext seems to improve its accuracy during the validation steps, however, it soon starts to overfit to the training dataset.

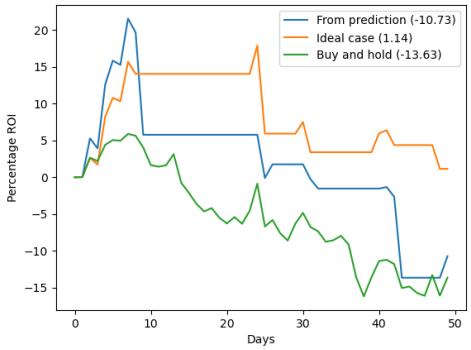


Figure 11: Predictions after batch 1

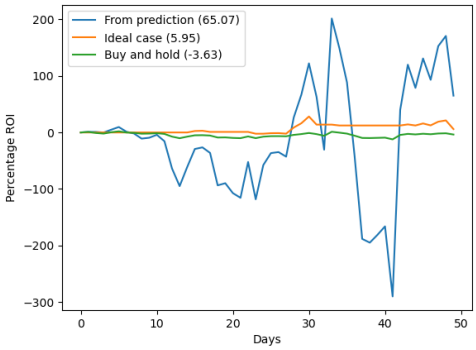


Figure 12: Predictions after batch 2

MetaContext Scores - after batch 1				
	precision	recall	f1-score	support
buy	0.65	0.69	0.67	16
sell	0.85	0.85	0.85	33
hold	0.00	0.00	0.00	1
accuracy			0.78	50
macro avg	0.50	0.51	0.51	50
weighted avg	0.77	0.78	0.77	50

MetaContext Scores - after batch 2				
	precision	recall	f1-score	support
buy	0.39	1.00	0.56	19
sell	1.00	0.03	0.06	30
hold	0.00	0.00	0.00	1
accuracy			0.40	50
macro avg	0.46	0.34	0.21	50
weighted avg	0.75	0.40	0.25	50

According to Lemos et al. [10] one of the main problems of a model when facing such a data stream is the change in the data distribution that occurs as time passes. "Concept drift" can be observed from Figure 11 and Figure 12 as well as the prediction metrics. After each online learning batch, MetaContext was asked to make predictions for the next 50-day period and we noticed that the prediction accuracy dramatically decreased from one batch to another. With a trading strategy and ROI nowhere near the ideal one by the second batch, we can conclude that the implemented incremental learning strategy was not successful on the current setup. In order to successfully implement online learning in an LSTM model, Lemos et al. [10] proposed an Incremental LSTM model that retains the hidden state and the cell state from every batch and provides it to the next as initialization instead of vectors of zeros.

5 Conclusion

The results indicate that while traditional LSTM models, exemplified by StockBot*, show promising generalization capabilities, they fall short in specific aspects of market prediction, particularly in implementing a realistic trading strategy. The introduction of ContextLSTM demonstrates a significant improvement in prediction accuracy by reformulating the problem as a classification task and augmenting the input data by creating the novel "context-aware" input. However, it still struggles with long-term prediction accuracy.

This research aimed to develop an algorithm with the capacity to adapt to diverse stock market conditions, a goal that stands at the forefront of financial predictive modeling. MetaContext, which treats each company's stock as a distinct classification task, shows potential in its adaptation ability, aligning closely with the desired trading strategy, as evidenced by its performance during experiments. Nonetheless, it encounters challenges in maintaining prediction accuracy over time, especially when subjected to concept drift during online training phases. This underscores the complexity of stock market prediction and the need for models to adapt continually to evolving market conditions. The research suggests that further refinements, such as those proposed by Lemos et al. for Incremental LSTMs, might be necessary to address the challenges of concept drift and improve the efficacy of online learning in dynamic market environments.

In conclusion, the pursuit of an algorithm capable of adeptly navigating various stock market conditions has been advanced by this research. The study not only contributes valuable insights into the application of meta-learning and incremental learning in finance but also sets the stage for future developments.

5.1 Future Work

1. Further mitigate MAML training instability and other related issues as proposed by Antoniou et al. [9].
2. Implement Incremental LSTM as proposed by Lemos et al. [10].
3. Conduct more experiments with "context-aware" inputs with different formats and different problems.

References

[1] Jinan Zou, Qingying Zhao, Yang Jiao, Haiyao Cao, Yanxi Liu, Qingsen Yan, Ehsan Abbasnejad, Lingqiao Liu, and Javen Qinfeng Shi. Stock market prediction via deep learning techniques: A survey. *arXiv*, 2022.

[2] David M. Q. Nelson, Adriano C. M. Pereira, and Renato A. de Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017.

[3] Shaswat Mohanty, Anirudh Vijay, and Nandagopan Gopakumar. Stockbot: Using lstms to predict stock prices, Jul 2022.

[4] Dingming Wu, Xiaolong Wang, Jingyong Su, Buzhou Tang, and Shaocong Wu. A labeling method for financial time series prediction based on trends. *Entropy*, 22(10):1162, 2020. PMCID: PMC7597331, PMID: 33286931.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[6] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA, 20–22 Jun 2016. PMLR.

[7] Nicholas John Murphy. *An Online Learning Algorithm for Technical Trading*. Dissertation, University of Cape Town, 2019.

[8] Chelsea Finn, Amelie Byun, Ansh Khurana, Alex Sun, Yoonho Lee, and Max Sobol Mark. Cs 330 autumn 2023 homework 2: Prototypical networks and model-agnostic meta-learning, 2023. Stanford University, Course Materials for CS 330.

[9] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2019. Published in ICLR 2019.

[10] Alvaro C. Lemos Neto, Rodrigo A. Coelho, and Cristiano L. de Castro. An incremental learning approach using long short-term memory neural networks. In *Proceedings of the Annual Conference on Neural Networks*, pages –, Belo Horizonte, MG, Brazil, 2021. Universidade Federal de Minas Gerais.