universität freiburg

Seminar in Business Analytics: Advanced Data Science

– Winter Term 2024/25 –

LLM for Stock Market Prediction: Using an Open Source Reproduced LLAMA Model for Multivariate Time Series Analysis

Submitted by:
Petar Zumbulev
Student-ID:
4362 512
Advisor:
Jannik Schäfer

Content

Abstract

The following study explores a Large Language Model (LLM) OpenLLaMA for binary classification of financial market data. By applying a zero-shot approach to forecasting, the model classifies market entry (1) or staying out of the market (0) for given financial instruments during time periods. Our necessary computational constraints allowed for only five features to be processed by the LLM. Overall, a 52.9% accuracy was achieved by the model, highlighting its current limitations. The main limitations of the study are the lack of fine-tuning, the unique problem setup of multivariate feature analysis, and lack of external information such as sentiment analysis. Our findings highlight the potential of OpenLLaMA for time series analysis as well as the drawbacks.

1. Introduction

In recent years, there has been growth in Large Language Model applications for the analysis of time series data. The LLM models have been used for natural language processing (NLP) and have later been used for sequential structures within data (Zhang et al., 2025). LLMs utilize sequential associations in text, and various authors have attempted to harness this structure for the sequential associations observed in time series data (Jin et al., 2024 and Tan et al., 2024). Although these sequential associations seem promising for time series data, recent research has shown that “LLMs do not have unique capabilities for representing sequential dependencies in time series” (Tan et al., 2024). Nevertheless, advancements in LLMs for time series such as the TIME-LLM, BloomberGPT, and LLAMA from Meta have promising capabilities (Jin et al., 2024 and Wu et al., 2023).

Financial markets and financial analysis are highly complex, and the increase in machine learning methods can be utilized in order to improve decision making processes, making this an important area for research. Although Autoregressive Integrated Moving Average (ARIMA) and Gradient Boosting Machines (GBM) are methods that have been used in the financial sector, LLMS promise a novel approach by using text based and tokenization based analysis for interdependencies among variables (Paliari et al., 2021).

Our study assesses the effectiveness of an open source LLM, OpenLLaMA from Meta, in order to predict binary decisions of market entry (1) or staying out of the market (0) based on financial time series instruments. The following research questions are addressed:

1. How effectively can OpenLLaMA classify market entry (1) and staying out of the market (0) based on various financial instruments (features)?

2. What limitations arise from the OpenLLaMA model and multivariate structure of our problem and data?

This study contributes to the growing research on LLMs for financial time series applications and increased machine learning applications. In addition, our study of the OpenLLaMA model provides empirical data for a specific application on a financial dataset, including accuracy and a confusion matrix analysis. Finally, we discuss the results and analyze limitations and future applications of the OpenLLaMA model.

2. Background

Large Language Models have been applied for various problem settings because of their ability to process text and analyze sequential dependencies in data. More recently, LLMs have been gaining attention in the financial sector and specifically with time series data (Jin et al., 2024 and Wu et al., 2023). Well known models such as BloomberGPT, Time-LLM, and the model used in this study: OpenLLaMA from Meta have been used within time series data and some show promising results (Jin et al., 2024, Wu et al., 2023, Tan et al., 2024).

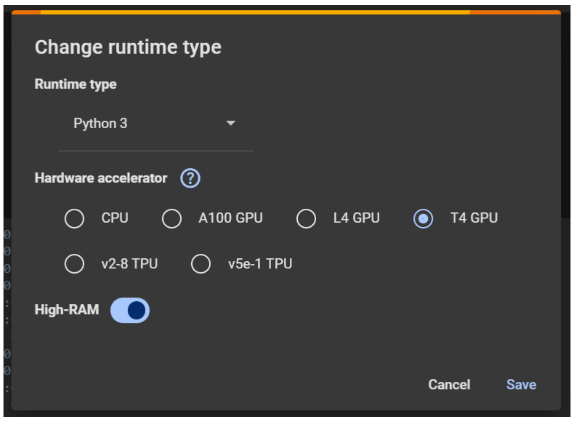
Many techniques for analyzing financial time series data exist: Autoregressive Integrated Moving Average (ARIMA) and Gradient Boosting Machines (GBM) for example. However, it has been shown that transformer-based architectures as used in LLMs show predictive results which are improvements upon previous analysis techniques (Jin et al., 2024 and Paliari et al., 2021). However, the area of LLMs for time series analysis still needs to be extensively studied and there is much room for improvement, especially within multivariate analysis.

3. Methodology

We selected the open source LLM called OpenLLaMA, reproduced by the author 'openlm-research' on the Hugging Face platform (OpenLM Research). There are a few considerations for selection which we took into account: open source nature, relatively manageable amounts of parameters, performance, and easily reproducible code. As can be seen in the description, OpenLLaMA is open source and offers a 3B, 3 billion parameter version. In addition to the lower computational load, OpenLLaMA is proven to perform well and the code is easily reproducible, as can be seen on the model card.

Furthermore, the zero-shot forecasting demo for Lag-Llama was used for setting up the virtual environment in Google Colab, framework for setting up the data, and predictions (Ashok et al., 2024). Within Google Colab, we used the 'T4 GPU' computing engine with high

RAM turned on (See Figure 1). This computing engine is a part of Colab Pro which costs around 11 Euros per month and was necessary for our computing requirements.



Change runtime type

Runtime type: Python 3

Hardware accelerator: CPU A100 GPU L4 GPU [✓ T4 GPU] v2-8 TPU v5e-1 TPU

High-RAM: [✓ on]

Figure 1: T4 GPU with High-RAM. Recommended for reproducible results.

Bloomberg is the data source for our financial dataset, and instruments such as indices, commodities, currencies, and government bonds are included. Our raw data consists of financial market data with 6943 rows/observations and 34 different features, including 1 last column as the target column. The target is either 1 or 0, meaning entering the market for 1 or staying out of the market for 0. This binary structure presents a unique challenge when asking the model to predict and will be discussed later. Finally, entering or staying out of the market represents investments in the S&P 500.

We could not work with the raw data directly, rather we first cleaned it. The pre-processing involved identifying the errors or inconsistencies in the data. Specifically, there were various data points which had a division error, resulting in the string '#DIV/0!'. A total of 10 cells contained this error and were resolved. To process these cells, we used pandas and removed the rows containing the '#DIV/0!' strings.

The most important libraries used are pandas, datasets and transformers. Pandas is needed for reading the data file and data processing. Datasets is used to convert our imported dataset into a Hugging Face dataset. When preparing the data, a train-test split of 90% training data and 10% test data was done. The 10% test data is data the model has never seen and will be tested on. Finally, the context window of OpenLLaMA has an additional limitation: the context window of the model allows only for 5 features/columns with 5 examples/rows from our dataset to be processed at one time, instead of the 34 original features. We discovered this after receiving an out of memory error when attempting to process all 6943 rows of data with all 34 features at once. After gradually reducing the number of columns and rows, we arrived at 5 features with 5 rows. This limitation will be discussed later, but for now it is important to note that after much experimentation this is what our current tools at our disposal allow us to do.

Loading the model is done by using the classes LlamaTokenizer and LlamaForCausalLM, from the Hugging Face Transformers library. In order to load the model and access the Hugging Face interface, a token is required, called “HF_TOKEN”. This token is given by Hugging Face and the user reproducing our study must obtain this token from the platform. After this, the token is put into the Google Colab ‘secrets’ section.

Due to the nature of our data, some of the inputs can be of different lengths. This is because our raw data uses floats, and this problem is solved by padding the tokens using ‘tokenizer.eos_token’. The ‘eos’ tokens stand for ‘end of sequence’ and allow the model to have inputs of equal lengths.

After our data pre-processing, we created the input structure for the model. The input structure follows a five feature input with a target. The target is the result: either entering the market or staying out of the market. A 1 stands for entering and 0 stands for staying out of the market. This 1 and 0 is the binary structure of our problem, and the binary encoding is done by feeding the data into the model. Due to the model’s capabilities, LLaMA sees the input structure and follows the input structure when predicting – no additional binary encoding necessary, the only addition in the code was to make the prediction into an integer to be completely sure of the output. The five features with the target are inserted into the model five times as the examples, and the sixth row of features is *without* the target being given. Therefore, the model has to predict 1 or 0, whether to enter or stay out of the market.

As can be seen in the previous paragraph, our study uses a pre-trained model with zero-shot forecasting. This means that no fine-tuning was done, only examples are given and the model is asked to predict. This limitation is due to the fact that there was not enough memory in Google Colab.

After running the model, we will judge our data based on a pie chart and confusion matrix. Within these graphics, there will be true positives: the model correctly identifying instances that belong to the positive category (1), true negatives: the model correctly identifying instances that belong to the negative category (0), false positives: the model incorrectly identifying instances that belong to the positive category when they should be negative, false negatives: the model incorrectly identifying instances that belong to the negative category when they should be positive (See Figure 2). The summation of the true positives and true negatives will give us all instances where the model predicted correctly, while the sum of false positives and false negatives will give the instances where the model did not predict correctly. Based on these values, we will identify the accuracy of the model: correct predictions / total predictions (See Figure 3).

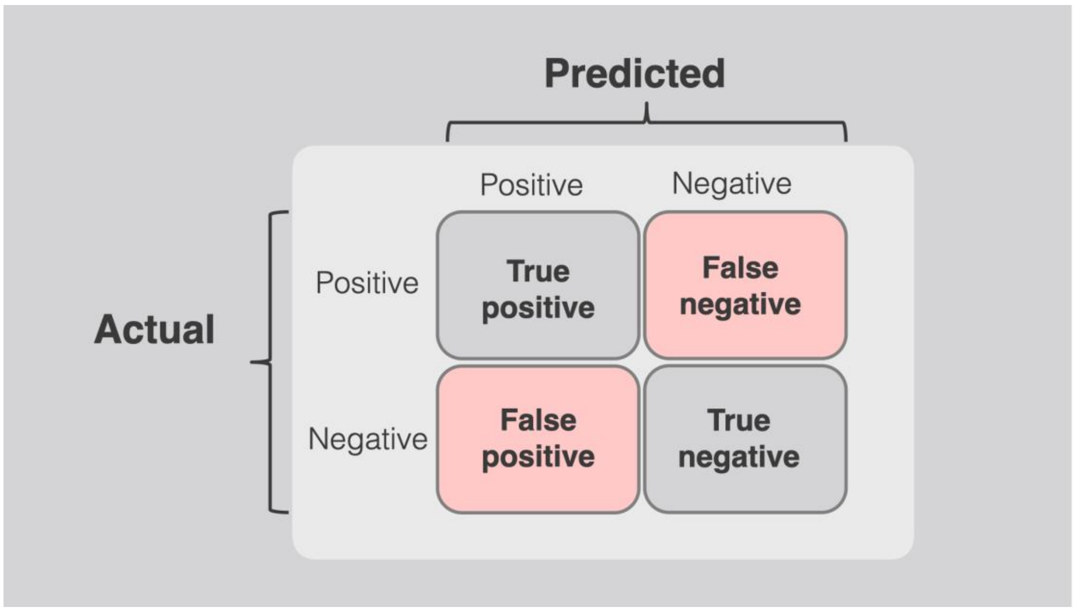


Figure 2 (Evidently AI): Framework for judging LLaMA's results with our financial dataset. To be applied in the data and results sections.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 3 (Evidently AI): Accuracy calculation, our method of judging how well the model performs. To be applied in the data and results sections.

4. Results

OpenLLaMA's ability to correctly classify data based on our binary outcomes is detailed in the following confusion matrix (See Figure 4), pie chart (See Figure 5), and accuracy calculation (See Figure 6). As can be seen in the confusion matrix, the model correctly predicted the outcome in the positive category (1) in 202 instances and incorrectly predicted staying out (0) when (1) was correct in 163 instances. Moreover, in 165 cases the model incorrectly predicted market entry (1) when the actual case was staying out of the market (0). Finally, the model predicted correctly to stay out (0) when that was the correct case in 166 instances. For our study, the 202 cases represent the true positives, 163 cases the false negatives, 165 cases the false positives, and 166 cases the true negatives. These results form the foundation for the pie chart and percentages.

The observations noted in the pie chart represent the outcomes based on percentages and they are derived from the confusion matrix in order to give a visual representation of the data. 29.0% of cases were true positive, 23.7% were false negative, 23.4% were false positive, and 23.9% were true negative. Using the data from the confusion matrix, which is related to the pie chart percentages, we calculated the accuracy of the OpenLLaMA model to be 52.9%. Model accuracy is computed as correctly classified cases (true positive + true negative) divided by the total number of cases. Note that 696 is 10% of the total rows/observations for our data, which corresponds to the test data from our test-training split.

Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	202	163
Actual Negative	165	166

Figure 4: Confusion matrix built from the results of OpenLLaMA. Corresponding data for true positives, true negatives, false positives, and false negatives is represented.

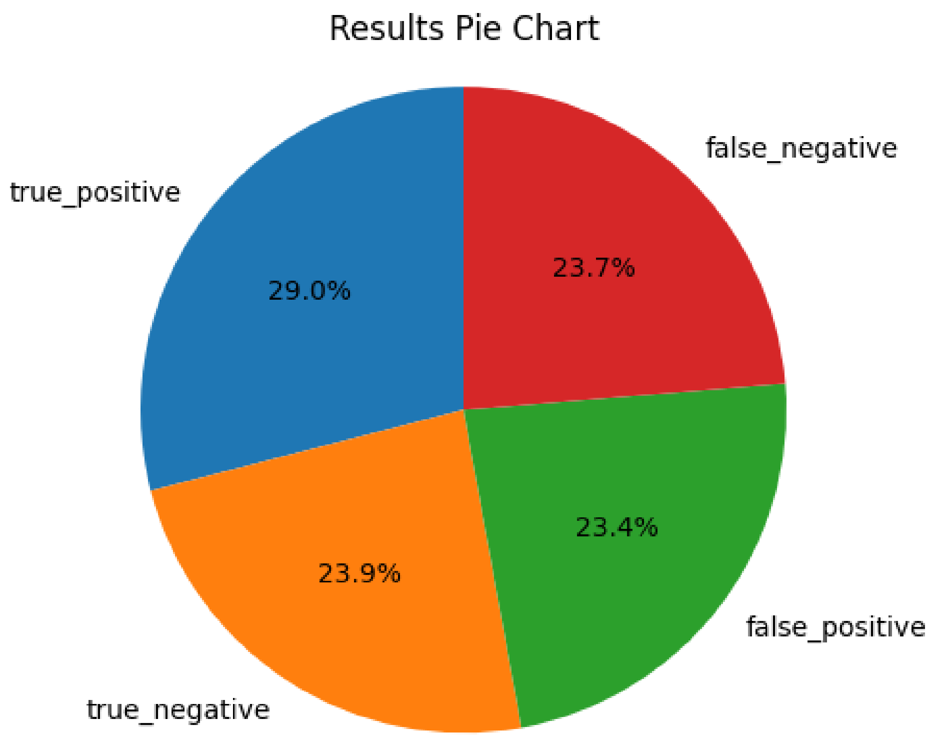


Figure 5: Pie chart based on the results of OpenLLaMA. The values of 29.0%, 23.9%, 23.4%, and 23.7% correspond to the true positives, true negatives, false positives and false negatives respectively.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
$$= \frac{202 + 166}{202 + 166 + 165 + 163} = \frac{368}{696} \approx 52.9\%$$

Figure 5: Accuracy calculations from the LLaMA results. As can be seen, the model we used achieved an accuracy of 52.9%.

5. Discussion

When tasked to predict market entry (1) or staying out of the market (0) based on 5 features, the OpenLLaMA model displayed a 52.9% accuracy which is not effective. Although our model correctly classified 368 out of 696 instances, about 47% of the total predictions (328 out of 696) - were not correct. The accuracy measure shows that our model performs just a few percentage points better than random guessing. Therefore, the predictive power is very limited and our model *as is currently* and with the given data is not reliable in financial decision-making.

Since our model only included a zero-shot method of prediction, the relatively low accuracy is understandable in this scenario. With fine-tuning and manageable computational load, the model could improve in accuracy and usability. This fine-tuning would include training on the financial time series data and changing relevant parameters instead of the current approach which is just giving the model examples of 5 rows and features. To increase the accuracy, we also suggest to increase the amount of features that the model is given: use 10, 15, or even all 34 features in the raw dataset instead of only the 5 features currently used.

The nature of our question was complex, involving binary classification of entering or staying out of the market based on multivariate feature analysis. In comparison, other LLM models such as the zero shot forecasting demo only use one feature and multiple rows or temporal data points (Ashok et al., 2024). This highlights a significant difference in the problem setup and requirements for memory, computational load, and actual expectations of the research question. Ashok et al. require their LLM to predict future time series values of a single financial asset, while our study required an analysis of various features – up to 34 – which proved more challenging for the LLM. Nevertheless, this is a worthwhile problem setup and should be studied and improved upon further.

As was seen, in 163 cases the model predicted staying out (0) incorrectly when entering the market (1) was correct. This is on the conservative side of underestimating opportunities for a possible trade in the S&P 500. Since our 5 selected features were the first 5 rows without further consideration, it is possible that these indicators do not have a large impact on the LLM's abilities to correctly predict. Moreover, the 165 cases where incorrect entry (1) was predicted instead of the correct exit (0) show that the model could also push for riskier trades and therefore losses when utilized. It is possible that with the 5 examples with 5 features given, the model was also not given the correct features that maximize its predictive power. Overall, we do observe a relatively equal distribution of total correct and total incorrect classifications (368 out of 696 correct), which shows some predictive power but not much.

Clearly, the model and the specific multivariate dataset that we are working with are not to be used in practice. Only after further adjustments to the model and an increased measure of accuracy as well as values in the confusion matrix should the model be considered for industry use.

6. Limitations

Possibly the biggest limitation of our study is the feature selection. Firstly, only 5 features were used for giving the model examples due to the memory and computational load. In addition, these features were selected as the first 5 columns, therefore making this a random selection. It is highly likely that these 5 features are not the most relevant nor the most impactful for OpenLLaMA's accuracy. Finally, it's important to mention that financial markets are not only influenced by financial instruments, but also by political events, geopolitical events, and investor sentiments. All of these factors are limitations and should be taken into account.

Another limitation is the binary nature of the decision making by the model. Financial instruments have a numerical dimension and simplifying this to only entering or staying out of the market is on one side creating complexities in the analysis of the LLM and it is simplifying the problem at hand. For example, there are many models such as BloomberGPT and the Ashok et al. model from 2024 which focus on predicting numerical values for financial instruments and not simplifying the decision process to a binary decision. Therefore, our study of this model should be approached with caution and knowledge of these limitations.

To reiterate a significant limitation: our study focused on a problem setup which focused on making a binary decision based on multivariate analysis of features. In comparison, well-known models such as BloomberGPT and the Ashok et al. model from 2024 focus on univariate analysis. That means focusing on one single feature and predicting its development over time instead of many features and predicting if market entry or staying out of the market should be done at a single time period.

In addition, this study is also limited by current memory and computational load capabilities. OpenLLaMA, especially if it is run at higher capabilities with more memory and features would result in significant CPU or in our case GPU resources. Since we worked with GoogleColab and the T4 GPU, our study was limited by the capabilities of GoogleColab, other researchers might have higher capabilities. This is a financial limitation as well and the balance between the GPU input and model output must be met.

At the moment, our model has limited real world applicability. This is due to the low accuracy score, the limited use of features and random feature selection, as well as the computational resources needed to run our model.

7. Future Outlook

Several areas of this work can be improved, and one of them is the fine-tuning. As is shown in previous studies, fine-tuning greatly improves LLM model predictions, and our model accuracy would benefit from training on the specific dataset we have (Jin et al., 2024 and Wu et al., 2023). Specifically, the challenges of memory errors and computational load in terms of using more than 5 features would have to be handled in these future scenarios. Feature expansion would allow for a broader range of connections between tokens within the model. If achieved, this would eliminate the problem this study has which is the 5 features were selected at random instead of analyzing which features best help model prediction. On the one side, the best features issue could be solved and on the other hand the features could be expanded. Expansion could include 10, 15, or even all 34 features. Future models could even include more than 34 features and balance model accuracy and computational load in terms of the amount of features used in training. If these improvements are made and OpenLLaMA's accuracy were improved, then this could be a model used in the financial industry.

One of the biggest differences between our model and others such as BloomberGPT and the Ashok et al., 2024 LLM is that our model requires multivariate feature analysis instead of univariate feature analysis. We suggest further research into leveraging LLM abilities to better manage multivariate feature analysis within the training and fine-tuning aspects. If this were to be achieved, OpenLLaMA could prove it's use in the financial sector. Although promising, it is important to mention that other model's and researchers would be improving their LLM's in parallel, possibly negating the benefits of utilizing multivariate feature analysis within our model.

One aspect that was not included in the dataset of our model is sentiment analysis. Since LLM's are inherently built to use text, sentiment analysis in the form of consumer reports, financial news, earnings call transcripts, and more could be utilized to improve OpenLLaMA's accuracy. Predictive power could then be improved to an even greater degree by combining the current 34 integer value features and the text based information from sentiment analysis.

8. Conclusion

The following study explored the potential of the LLM OpenLLaMA for financial market prediction based on binary classification. Our zero-shot approach classified market entry (1) and staying out of the market (0) and resulted in 52.9% accuracy, which is only a margin above random guessing. Overall, the model struggles with this approach to forecasting numerical data and the resulting confusion matrix values represent that. The low accuracy can be attributed to the lack of fine-tuning, random feature selection, as well as memory and computational power constraints. Although LLMs such as OpenLLaMA provide new approaches to time series analysis and forecasting, many improvements must be made. Future improvements could be fine-tuning, integrating more information such as text information in the form of sentiment analysis, and better selection of features once the memory issue is solved. Clearly, OpenLLaMA has both potential for financial time series forecasting as well as many challenges before it can be applied in the real world.

9. References

Ashok, A. et al. (2024). *Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting*. Google Colaboratory. Retrieved from https://colab.research.google.com/drive/1DRAzLUPxsd-0r8b-o4nlyFXrjw_ZajJJ

Evidently AI. (n.d.). Accuracy, precision, recall & F1 score. Evidently AI. https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall

Evidently AI. (n.d.). Confusion matrix. Evidently AI. https://www.evidentlyai.com/classification-metrics/confusion-matrix

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., & Wen, Q. (2024). *Time-LLM: Time series forecasting by reprogramming large language models*. arXiv. https://doi.org/10.48550/arXiv.2310.01728

Jin, M., Zhang, Y., Chen, W., Zhang, K., Liang, Y., Yang, B., Wang, J., Pan, S., & Wen, Q. (2024). *Position: What can large language models tell us about time series analysis*. arXiv. https://doi.org/10.48550/arXiv.2402.02713

OpenLM Research. (n.d.). *Open LLaMA 3B V2*. Hugging Face. Retrieved [20.12.2024], from https://huggingface.co/openlm-research/open_llama_3b_v2

Paliari, I., Karanikola, A., & Kotsiantis, S. (2021). *A comparison of the optimized LSTM, XGBOOST and ARIMA in time series forecasting*. In *2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)*. IEEE. https://doi.org/10.1109/IISA52424.2021.9555520

Tan, M., Merrill, M. A., Gupta, V., Althoff, T., & Hartvigsen, T. (2024). *Are language models actually useful for time series forecasting?* arXiv. https://doi.org/10.48550/arXiv.2406.16964

Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., & Mann, G. (2023). *BloombergGPT: A large language model for finance*. arXiv. https://doi.org/10.48550/arXiv.2303.17564

Zhang, J., Zhu, W., & Gao, J. (2025). *Adapting large language models for time series modeling via a novel parameter-efficient adaptation method*. arXiv. https://doi.org/10.48550/arXiv.2502.13725