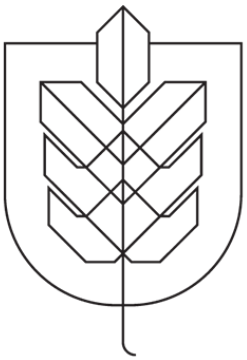


HULT
INTERNATIONAL
BUSINESS SCHOOL

Algorithmic Trading Proposal
Write-up by Awale Abdi

Write-up's Structure

1. The Proposal

1. The Sentiment Analysis

2. The Machine Learning Model

3. Backtesting the Strategies

4. The Final Wrap-up

5. References

6. Appendix

Word count excluding first two pages, references and appendix: 1,498

The Proposal

In the rapidly evolving financial markets, algorithmic trading has become essential for traders to leverage data and advanced analytics to enhance decision-making. This write-up outlines a proposed algorithmic trading framework that integrates sentiment analysis from social media platforms with a machine learning model to form a strategy for backtesting. Inspired by real-world cases, such as those discussed in recent articles by Tran et al. (2024) and HAARMK Infotech (2023), where Machine-Learning Models were effectively utilized to predict stock-prices, the author decided to apply an amateur form of this sort of framework themselves, combining it with sentiment data given the importance of social media today. Pfizer Inc. (PFE) was chosen to demonstrate identifying optimal times to buy or sell the stock based on historical data and sentiment indicators.

From the outset this framework is not expected to be profitable, hence some of the poor performance numbers you will observe going forward, as the author simply lacks the resources to make this concept work as it potentially could. The work outlined in this write-up simply serves as a framework for a company with the right resources to springboard off and how that company can do so will be outlined as the write-up progresses into subsequent sections.

The Sentiment Analysis

Everything springboards off the sentiment analysis which involves extracting and quantifying emotions from textual data to provide insights into market sentiments and be able to track when a stock is likely to go up or down and by how much based on the sentiments in the market found through data mining from social media. For this write-up, due to the author's limited resources and time, extremely limited but free sentiment data from Reddit was mainly focused on.

Methodology:

- **Data Collection:** Utilizing the Reddit API, posts from the subreddit "wallstreetbets" mentioning "Pfizer" were retrieved. The no more than 1,000 titles and texts of these posts available for free were combined to form a dataset.
- **Sentiment Scoring:** Each post was analyzed using the TextBlob library to compute a sentiment polarity score ranging from -1 (negative) to 1 (positive) with 0 representing neutrality.
- **Time Series Creation:** Whatever sentiment scores that were available were aggregated along a daily basis to create a time series that reflects daily market sentiment towards Pfizer.

ROC	Bollinger_High	Bollinger_Low	Reddit_Sentiment	Price_Percentage_Change	Lag_1
3.669190585	16.91855383	16.02072591	-0.75	0.281697543	0.002816975
1.222343854	16.92024719	16.04275158	0	-2.303381446	-0.023033814
4.497035395	16.95138141	16.05336312	0.5	1.552624929	0.015526249
2.321534873	16.96169377	16.09902801	-0.3	-0.169877603	-0.001698776
1.390504484	16.96745903	16.10275047	0.8	-0.737373736	-0.007373737

Visual representation of the write-up's model using Reddit Sentiment data

In a real-world scenario, a company could mine far more social media data over time with dedicated servers and employees. This write-up's scope was limited by the fact that it was utilizing freely available and highly limited reddit data. A similar issue was encountered when attempting to mine Twitter data.

In contrast, a company with the right resources could afford to acquire paid packages that offer them far more access to these platforms' data (Reddit Inc., 2023) (META, 2024) (X, 2024) whilst affording the servers required to mine extensive data over-time. They could also put teams of employees to work on developing more sophisticated, tailor-made sentiment analyses that account for financial jargon and slang, accurately classifying text mined as positive, neutral, or negative.

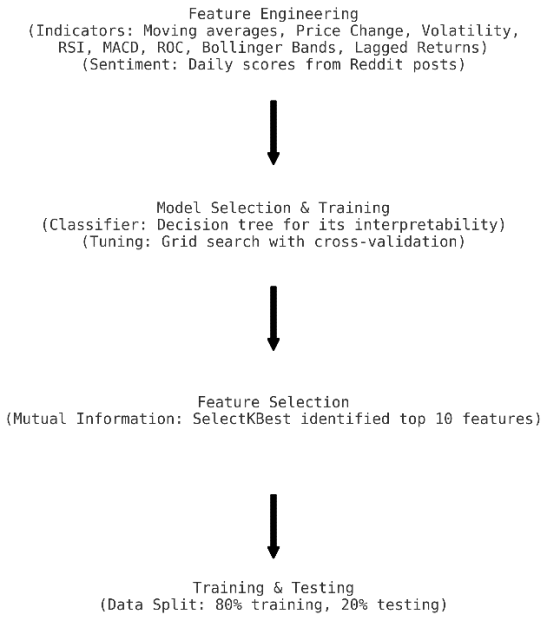
ROC	Bollinger_High	Bollinger_Low	Reddit_Sentiment	Google_Sentiment	Twitter_Sentiment	Facebook_Sentiment	Price_Percentage_Change	Lag_1
3.669190585	16.91855383	16.02072591	-0.75	0.2	0	5	0.281697543	0.002816975
1.222343854	16.92024719	16.04275158	0	-0.5	0.4	-0.1	-2.303381446	-0.023033814
4.497035395	16.95138141	16.05336312	0.5	0.3	-0.1	0	1.552624929	0.015526249
2.321534873	16.96169377	16.09902801	-0.3	0	0.7	0.2	-0.169877603	-0.001698776
1.390504484	16.96745903	16.10275047	0.8	0.2	-0.04	0.9	-0.737373736	-0.007373737

A small example of the sort of wider scope possible with the right resources

The Machine Learning Model

With the sentiment analysis completed, combining it with historical data on Pfizer's stock from Yahoo Finance and feeding this combined dataset into a Machine Learning Model could allow the model to use various features, such as sentiment data and stock price calculations like RSI and Sortino Ratio taught in class, to predict stock price movements. This could be used to make prudent trade decisions (long when the price is expected to rise and short when it's expected to fall), creating an applicable strategy that can be tested against a benchmark and backtested.

Simplified Outline of Modelling Process



(see appendix for more detail via the Python script)

In a real-world scenario, a well-resourced company could apply this machine-learning idea more effectively. Instead of a simple tree-classifier chosen for the author's own interpretability, they could develop a custom in-house ML model created by skilled professionals with finance and math degrees whilst finance experts, which the author of this write-up is certainly not, could refine the indicators utilized and how they are utilized.

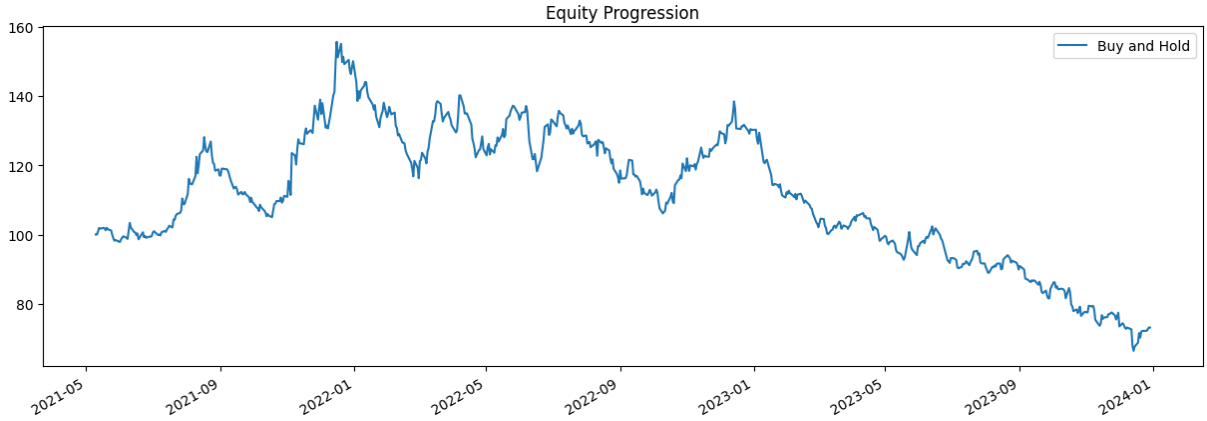
The overall stock tracking could also be refined to notice percentage changes by using a regression analysis, allowing the model to identify trends and predictors of price movements. This refined model could space its *Long and Short* decisions to realistic transaction frequencies that take advantage of significant percentage changes. With such fine-tuning, more funds, and increased computing power, the model could then run thousands to perhaps **millions** of loops to find effective instances applicable to the real world.

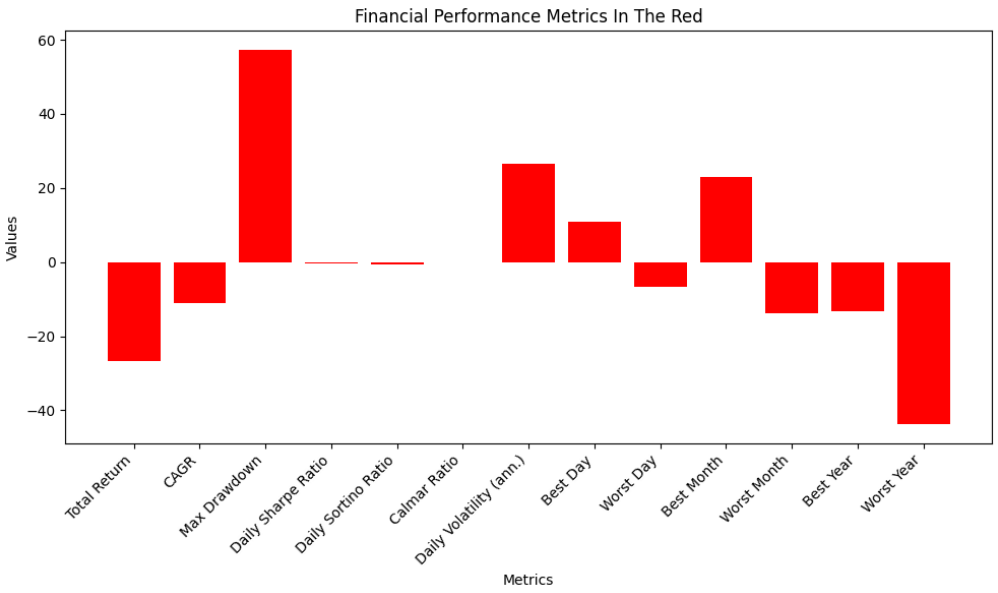
The applications of machine learning applied in this manner are endless if done right and with resources unavailable to the author of this write-up.

Backtesting the Strategies

The Benchmark Strategy

With the ML model ready for application, a benchmark was needed to evaluate it. The chosen benchmark strategy was a simple Buy and Hold, assuming an equal initial investment in Pfizer stock held throughout the testing period. This typically conservative baseline was backtested over the same period as the machine learning model (20% of time-series data not used for its own training set) through **1,000 loops**, generating the best result:



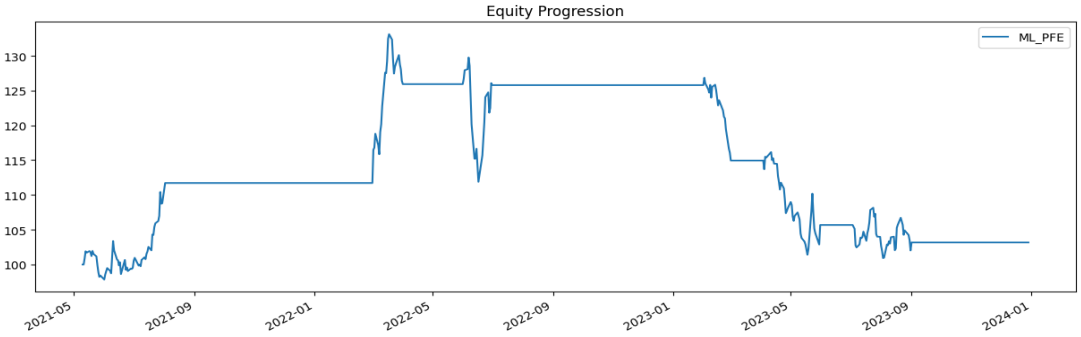


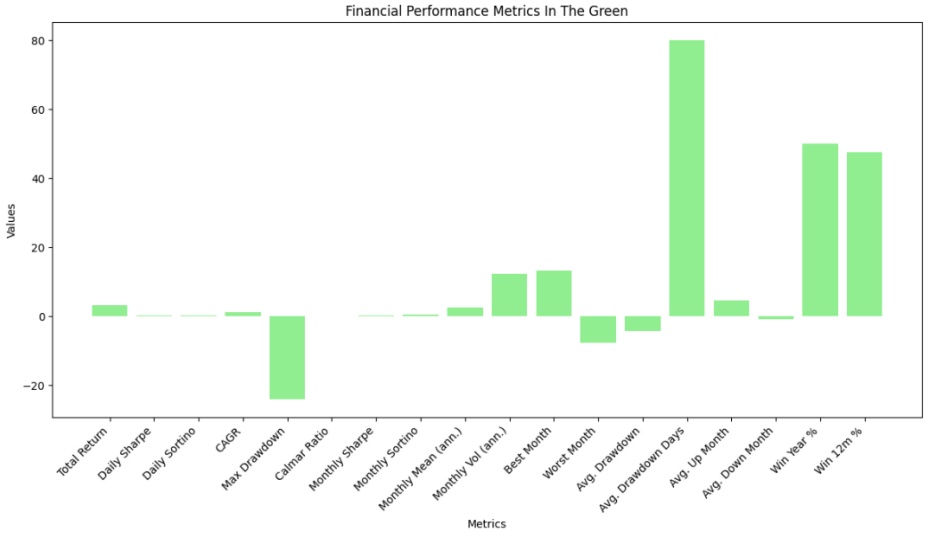
As shown, the benchmark "Buy and Hold" strategy significantly underperformed, resulting in substantial losses. It exhibited high volatility and significant drawdowns, with poor risk-adjusted returns as indicated by negative Sharpe and Sortino ratios. The total return and CAGR consistently declined, as corroborated by the equity progression graph.

Given these metrics, this strategy was ineffective during the period and would need substantial improvements for better performance. However, this makes it an ideal strategy for the machine learning model to be able to defeat.

The Machine Learning Strategy

The ML Model being ready was now ready to be introduced as an applicable strategy for backtesting via a custom algorithm class called "MLAlgo" using the bt library (see full script in appendix). Once it was ready for backtesting it was, like the benchmark, run through **1,000 loops** of backtests through the 20% of the time-series data that was not used for its modelling stage's training data and the best performing result was selected:



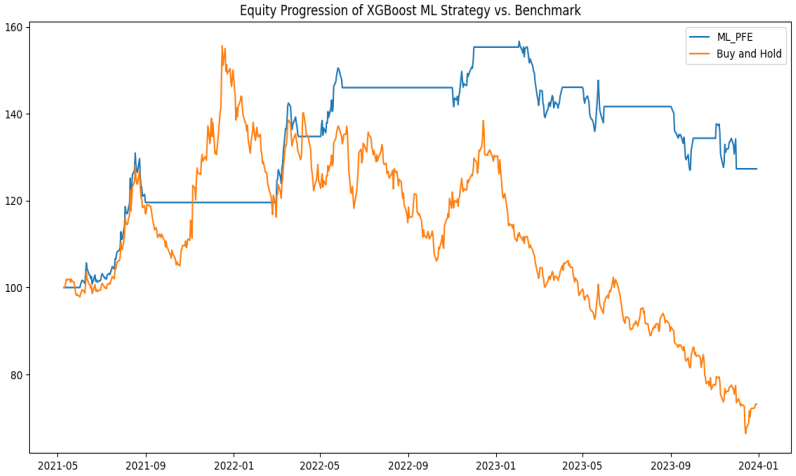


Despite this unrefined and simplistic Machine-Learning Model's accuracy being barely better than random chance (~51%), its strategy did manage to outperform the benchmark strategy in several of its loops across several key metrics. In this instance, the ML strategy achieved positive returns, managed drawdowns better, and provided better risk-adjusted returns. The benchmark strategy, on the other hand, suffered significant losses and higher volatility.

In a real-world scenario, a company with greater computing power could run millions of backtests, unlike the author's limited 1,000. This extensive testing would help identify optimal strategies using machine learning models, refine approaches, and ensure robust performance across different market conditions. They could also introduce and devise multiple benchmarks with expert input and backtest over various periods, adjusting the ML model's training period to avoid overlap.

The Final Wrap-up

Any individual reading this write-up or consulting the code scripts in the appendix's results may be underwhelmed in noticing that the models performed are only slightly better performing than random chance (51-52%) yet even that slight uptick in model performance over pure randomness (see histograms in appendix), in the author's opinion, shows this framework's potential. The fact that **anything** could be achieved at all with barely any reddit sentiment data to speak of and publicly available simple models such as a tree classifier or XGBoost highlights what far more serious and expansive resources could achieve.



For example, the slightly improved XGboost model compared to the tree classifier model (see appendix) which included better hyperparameter tuning and additional features such as learning rate and subsample adjustments, showed noticeable improvements in backtesting compared to the original tree classifier model, even if this performance was negligibly better.

Even though the improvements were minor, they highlighted the potential of the framework. In a real-world scenario, a more sophisticated model, developed by experts, could be significantly better. As stated prior, such a model could include advanced sentiment analysis techniques, years' worth of paid for data mined for sentiment analysis, more robust feature selection guided by financial experts, and sophisticated machine learning algorithms tailored by a dedicated team for financial data. Combining all of this with millions of loops would then refine the strategy further and bring its true potential into fruition.

In conclusion, while the current framework barely performs better than random, it is only meant as a proof of concept that acts as a guide for a promising path forward. For a concept that, with the right resources and time in real-world trading, could potentially bring a company promising returns, making it a compelling prospect for future applications the author of this write-up himself would like to someday be able to put into practice after refining their knowledge of the financial topics taught in class to better tune the indicators fed into models.

References

Tran, P., Anh, P.T.K., Tam, P.H., & Nguyen, C.V. (2024). Applying machine learning algorithms to predict the stock price trend in the stock market – The case of Vietnam. Humanities and Social Sciences Communications, 11, 393. https://doi.org/10.1057/s41599-024-02807-x

HAARMK Infotech. (2023, October 25). Mastering stock market forecasts: Harnessing machine learning for predicting stock prices. Medium. https://medium.com/@haarmkinfotech/introduction-bc6ecaf22f8b

Reddit Inc. (2023). Key dates for our API Terms and Services. https://www.redditinc.com/blog/apifacts#:~:text=Key%20dates%20for%20our%20API%20Terms%20and%20Services&text=Effective%20July%201%2C%202023%2C%20the,are%20not%20using%20OAuth%20authentication.

Meta for Developers. (2024). Accessing data with the Graph API. Retrieved from https://developers.facebook.com.

Twitter Developer. (2024). Getting Started with Twitter API. Retrieved from https://developer.x.com/en/docs/twitter-api/getting-started/about-twitter-api

Appendix

Link to Treeclassifier Model's Code

Link to XGBoost Model's Code

Link to metrics visualizations' Code

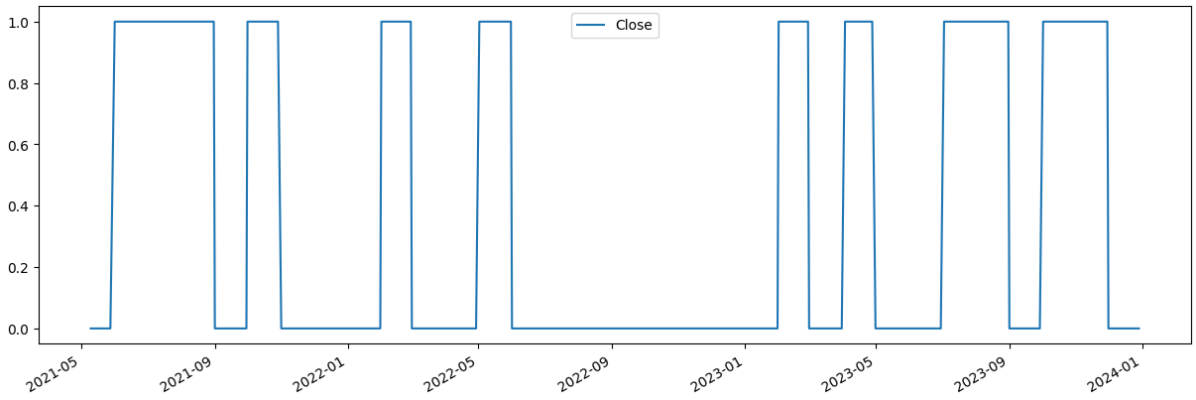
The cross-validated accuracies for the models were as follows:

0.5142956260708267 (tree)

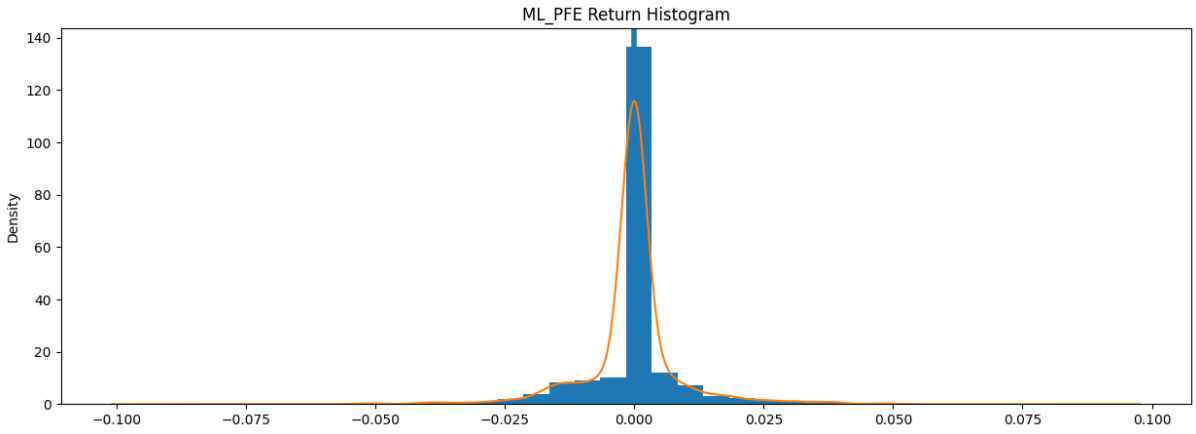
0.5142991659940812 (XG)

A completely negligible improvement but the author will take it.

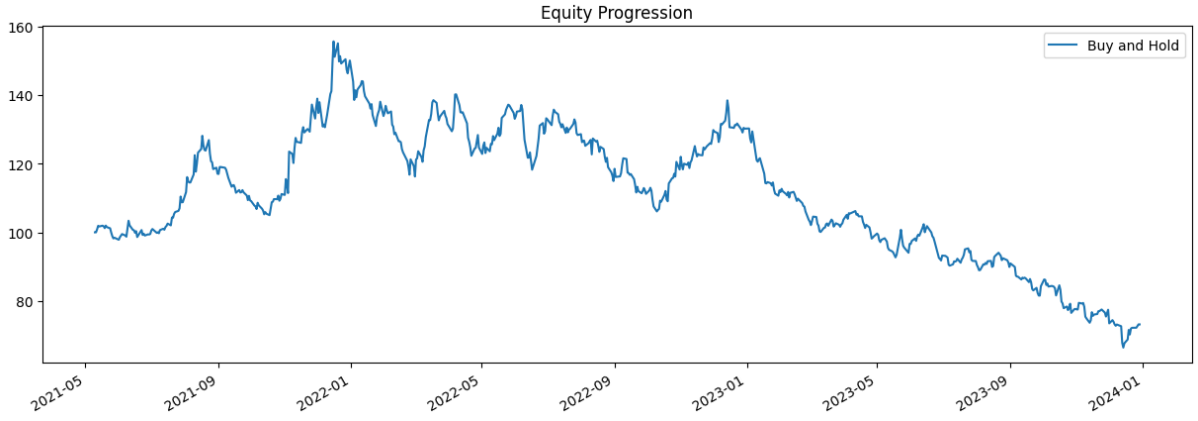
Other visualizations generated whilst backtesting through multiple loops:



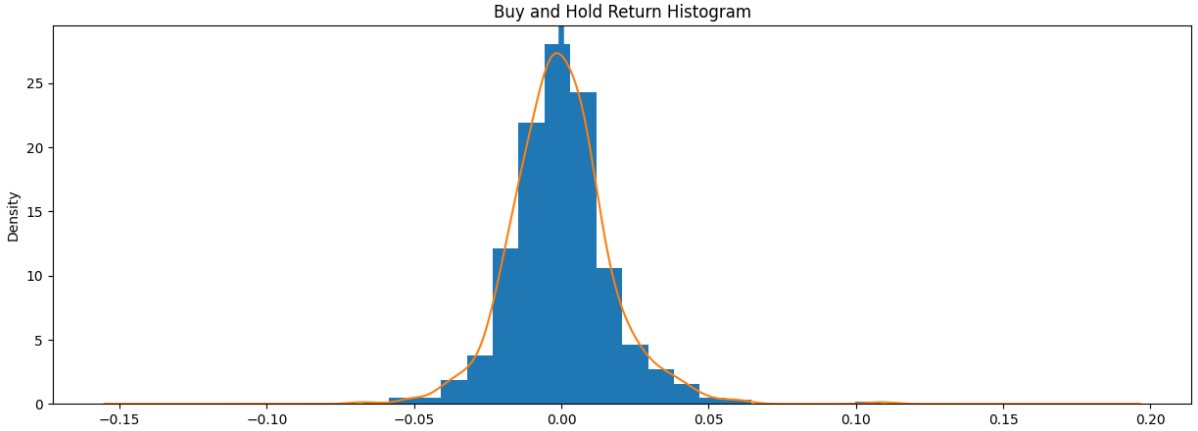
The graph shows the weights of Pfizer Inc. (PFE) in the tree classifier ML strategy over time. The weights fluctuate between 0 and 1, indicating the strategy alternates between fully investing and not investing in PFE at different periods.



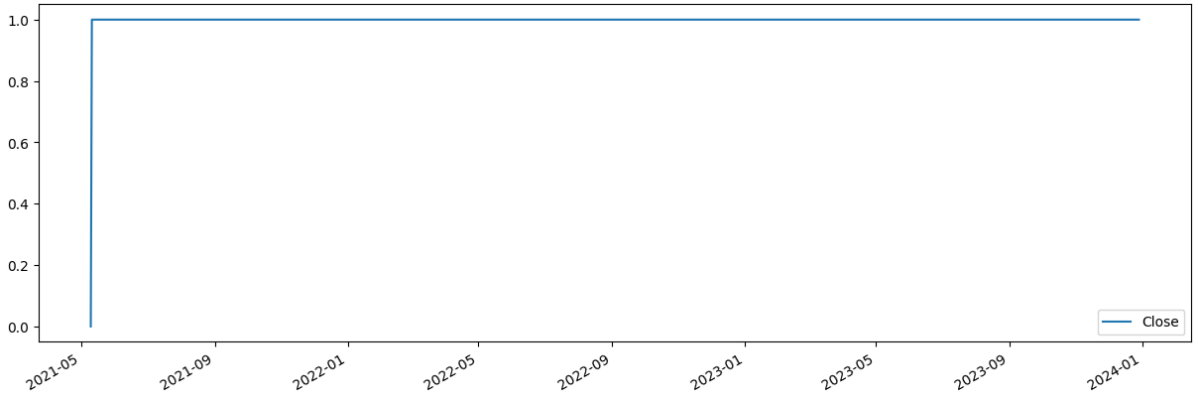
The histogram shows the return distribution of the tree classifier ML strategy, centered around zero with a slight peak of positive returns, indicating it was marginally more successful than random chance, demonstrating the framework's potential.



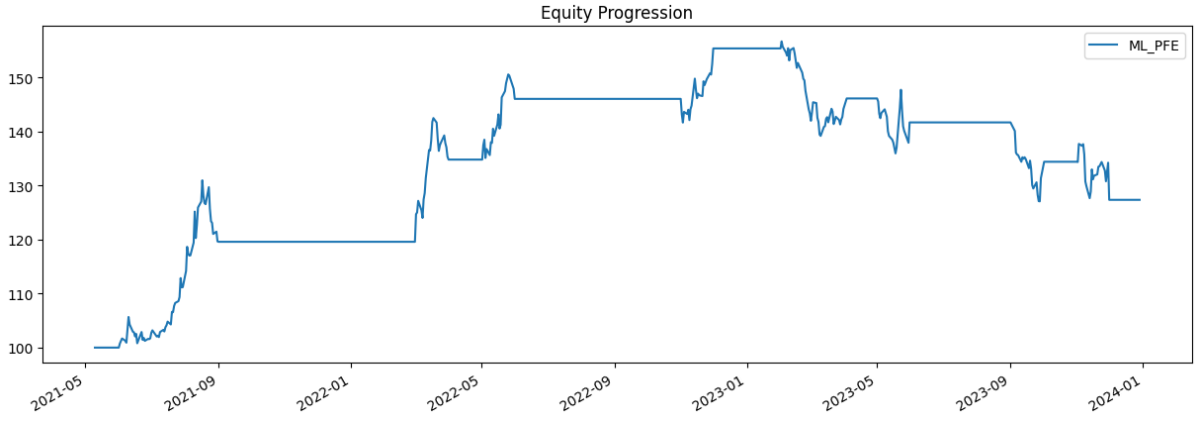
This graph shows the equity progression of the Buy and Hold strategy over time. The equity peaks in early 2022, followed by a steady decline through 2023, indicating underperformance in the later period.



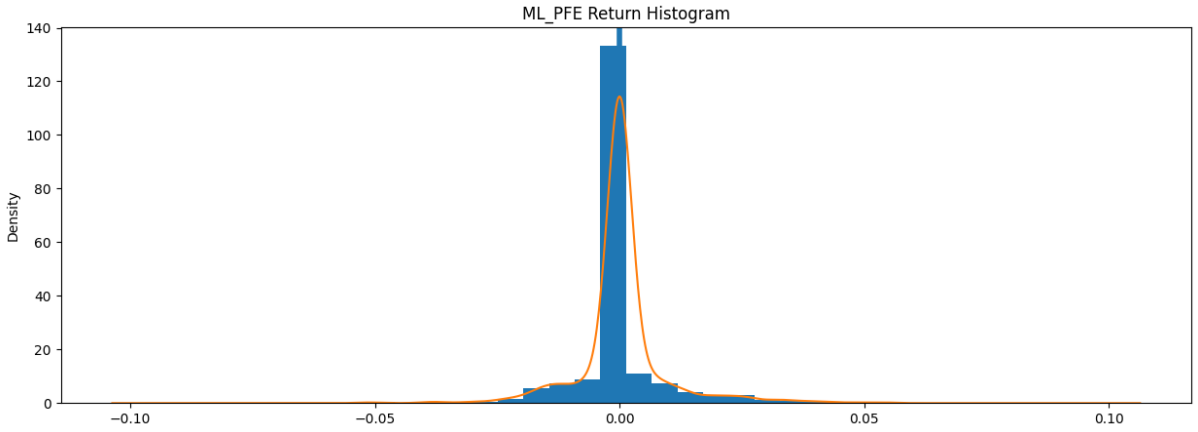
The histogram illustrates the return distribution of the Buy and Hold strategy, centered around zero with a normal distribution. There is a slight positive skew, indicating a higher frequency of small positive returns compared to negative returns.



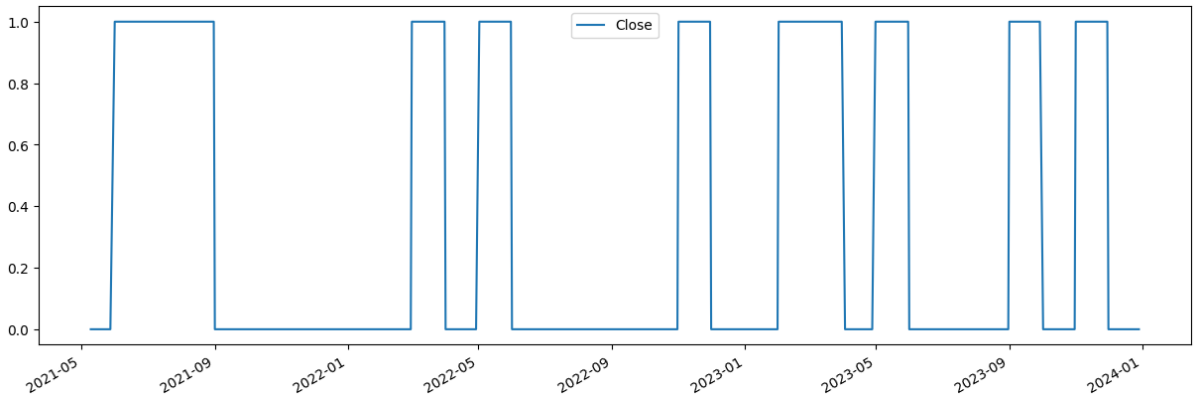
This graph shows the constant weight allocation of 1 (100%) to Pfizer Inc. (PFE) throughout the entire period. This indicates a static investment in PFE without any rebalancing or changes in allocation.



This graph shows the equity progression of the best XGBoost ML strategy over time. The equity sees significant increases in mid-2021 and early 2022, followed by periods of relative stability and some declines, indicating intermittent periods of strong performance.



The histogram displays the return distribution of the XGBoost ML strategy, which is centered around zero with a slight positive skew. This indicates that the strategy generally produces small positive returns more frequently than negative returns which is again, however slightly, a good start for this overall framework's viability.



This graph shows the fluctuating weights of Pfizer Inc. (PFE) in the XGBoost ML strategy over time. The strategy dynamically adjusts the weights, alternating between fully investing and not investing in PFE based on model predictions.