STOCK TRADING USING REINFORCEMENT LEARNING

PROJECT REPORT

Jonas Gottal

Mohamad Hagog

Viktoria Patapovich

Lorena Wemmer

Practical: Applied Reinforcement Learning

Ludwig-Maximilians-University Munich

September 2023

Contents

ABSTRACT

In the rapidly evolving landscape of stock trading, the integration of sentiment data from platforms like Twitter with traditional financial indicators offers a promising avenue for enhancing trading strategies. This report delves into the application of Reinforcement Learning to stock trading, leveraging both financial and sentiment data. We investigate the efficacy of five RL algorithms: A2C, PPO, SAC, DDPG, and TD3, and conduct a sensitivity analysis to understand the impact of various features on the decision-making process of the RL agent. Our findings highlight the significance of data normalization, the benefits of data smoothing, and the importance of transaction costs in influencing agent behavior. Furthermore, our models consistently outperform classical benchmarks, including Buy and Hold, Bollinger Bands, RSI, and MACD, even when these benchmarks incur zero transaction costs. The report emphasizes the potential of RL in stock trading and the importance of integrating sentiment data into decision-making.

1 Introduction

Stock trading, a cornerstone of global financial systems, has always been a complex interplay of numbers, strategies, and human emotions. From a macroeconomic perspective, stock trading is often viewed as a zero-sum game. Nevertheless, individuals engage in it. Psychologist Meir Statman provides an insightful analysis of this behavior, highlighting three primary reasons. Firstly, there is a prevalent belief among traders that they possess above-average skills, leading them to assume that their portfolio will outperform others. This overconfidence might be a driving force behind the plethora of algorithms developed to optimize stock market performance. Secondly, the inherent human love for games and challenges plays a role. Lastly, stock trading is seen as a pathway to social advancement, with successful trades bolstering one's social stature (43).

In the multifaceted domain of stock trading, Reinforcement Learning (RL) has risen as a valuable tool for refining trading strategies (2). RL is a machine learning paradigm where an agent learns by interacting with an environment, receiving feedback in the form of rewards or penalties (46). This iterative process of action and feedback allows the agent to develop strategies or policies that maximize its cumulative reward over time. RL provides a framework for making sequential decisions, making it particularly suited for dynamic environments like stock trading.

As technology and data analytics advance, the methods employed in stock trading have evolved. While traditional strategies lean heavily on financial data such as stock prices, volume, and technical indicators, the modern era emphasizes the growing influence of social media platforms. Platforms like Twitter, for instance, play a significant role in shaping the stock market's dynamics (53). Understanding the relationship between public sentiment and stock trends is essential in today's dynamic market environment. The primary question driving our research is: Can the integration of sentiment data with traditional financial indicators enhance the efficiency of RL in stock trading? To address this, our study employs five RL algorithms: A2C, PPO, SAC, DDPG, and TD3, focusing on using both financial and sentiment data. By using the power of RL and integrating diverse data sources, our goal is to develop trading strategies that are both insightful and adaptable.

Additionally, we delve deep into sensitivity analysis, an important component when working with neural networks. Sensitivity analysis, at its core, is a technique used to determine how variations in input features can influence the predictions or decisions made by the network. In our approach to sensitivity analysis, we employ gradient-based methods. By computing the gradient of the output with respect to each input feature, we can gauge the sensitivity

or importance of that feature, thus making the model explainable. Features with higher gradient values have a more pronounced impact on the model's output, while those with lower gradient values might be less influential. This understanding provides insights into the model's behavior and helps refine the input features, potentially leading to more robust and accurate models.

The structure of this report is organized as follows: In Section 2, we delve into existing literature, focusing on the application of RL in finance and methodologies of sensitivity analysis. Moving on to Section 3, we detail our data sources and the preprocessing steps, offering a deeper exploration into financial and sentiment data, technical indicators, data preprocessing techniques, and methods for noise reduction. Section 4 offers a comprehensive overview of our RL environment, detailing the state, agent's architecture and functionalities. Our approach to sensitivity analysis is detailed in Section 5, where we describe the implementation methods and the techniques used for assessing the analysis. In Section 6, we present classical strategies based on technical indicators like RSI, Moving Average, and Bollinger Bands. Section 7 presents the outcomes of our experiments, interpreting the data to draw meaningful insights and implications. Concluding the report, Section 8 offers reflections on our findings, discusses their implications, and considers potential avenues for future research and development.

2 Related Work

Our research draws upon both foundational and recent studies relevant to our objectives. This section is divided into two primary themes. In Subsection 2.1, we examine the intersection of RL and stock trading, looking into how RL techniques have been applied to enhance and innovate trading strategies. Subsequently, Subsection 2.2 provides an overview of sensitivity analysis methods, highlighting their importance in understanding and interpreting model behaviors, especially in the context of neural networks and RL.

2.1 RL and Stock trading

Integrating machine learning and deep learning techniques in the financial domain has encouraged the development of predictive and classification models for the stock market. These models primarily use fundamental and alternative data sources to extract investment patterns, leading to the generation of predictive alpha signals for stock selection. Nevertheless, they mainly emphasize choosing high-performing stocks without considering the allocation of trading positions among them. Deep Reinforcement Learning (DRL) fills this gap by offering a strategic approach for automated stock trading, facilitating dynamic

decision-making in the ever-evolving stock market. By navigating through the complexities of the stock market, which is flooded with rapid changes and diverse factors, DRL allows the creation of multi-factor models and algorithmic trading strategies that often surpass human capability.

In works (51), (25) the authors delve into the application of DRL to optimize stock trading. The former presents an ensemble strategy combining three actor-critic-based algorithms (PPO, A2C, DDPG). It tests them on the 30 Dow Jones stocks, revealing that the ensemble strategy outshines individual algorithms and other benchmarks in terms of risk-adjusted return. The latter explores the potential of DRL in maximizing investment returns on 30 selected stocks, and its results also display a superior performance against conventional benchmarks.

Paper (26) offers a toolkit named FinRL for newcomers in quantitative finance. The library assists in developing and comparing stock trading strategies by providing environments configured with stock market datasets and trading agents that are trained using neural networks. It accommodates trading constraints, simulates various stock markets, and has a modular structure incorporating state-of-the-art DRL algorithms. FinRL also features demonstrations for single-stock trading, multi-stock trading, and portfolio allocation.

Paper (6) examines the implementation of Deep Q-network and Deep Recurrent Q-network in stock trading. The goal is to automate daily stock trading decisions. Using the S&P500 ETF as the trading asset, the paper evaluates the agent's performance against traditional benchmarks like Buy and Hold and Random action-selected DQN trader. Results indicate that the DRQN trader, leveraging the recurrence framework, excels in recognizing and capitalizing on time-related profitable patterns, surpassing both the DQN trader and the other benchmarks.

To summarize, the integration of reinforcement learning in stock trading has shown significant promise in recent literature, with numerous approaches revealing potential advantages over traditional methods.

2.2 Sensitivity Analysis

Sensitivity analysis in the context of machine learning refers to the study of how the uncertainty in the output of a model can be split among different sources of uncertainty in its inputs (44). In simpler terms, it aims to determine how different input variations can influence the model's output. This is particularly crucial in finance, where models are often used to make decisions involving significant amounts of money. Understanding the

potential variability and uncertainty of predictions can be as important as the predictions themselves.

Several methods have been proposed and employed for sensitivity analysis in machine learning:

- **Gradient-based methods**: Gradient-based methods compute the gradient of the output concerning the input (33). The size or magnitude of this gradient signifies how sensitive the output is to variations in the input. In the context of neural networks, backpropagation is the standard technique for calculating these gradients. Such methods are particularly effective for continuous inputs. Notable gradient-based techniques include Gradient * Input (40), Integrated Gradients (45), Deep Taylor Decomposition (31), and Saliency Maps (41).

- **Perturbation Analysis**: This method involves introducing small changes to the input and observing the corresponding changes in the output (3). One can estimate the model's sensitivity to changes in each input by systematically perturbing each input variable and monitoring the output. Methods of perturbation analysis have been used in the domain of image classification (52), (54).

- **Variance-based methods**: These methods decompose the variance of the model's output into contributions from each input. The Sobol method (42) is a popular variance-based method that provides a global sensitivity measure, indicating the expected proportion of variance in the output due to each input. Other methods include (47) and (37).

- **Feature Ablation**: Especially pertinent to deep learning models, this technique entails the removal or "ablation" of individual features, one at a time, to observe its impact on the model's performance. Through this approach, one can discern the features that are pivotal for the model's predictions. Seminal works in this domain are referenced in (4) and (8).

In the realm of finance, sensitivity analysis is essential (32). Financial models, especially those based on reinforcement learning, often need to be revised and simplified. Understanding how these models respond to changes in input variables can provide insights into their robustness, reliability, and potential areas of vulnerability (27).

3 Data

In order to make statements about the stock market, a sufficient dataset is a prerequisite. In the following paragraph, an explanation of the utilized data and its preprocessing is provided.

3.1 Market Data

Accurate financial market data is crucial for quantitative analysis in finance, particularly when analyzing stock volatility and dynamics. The data's availability and richness make it a powerful tool for financial analysts, researchers, and stakeholders who rely on stock metrics to make decisions (39). The data in the stock market can be divided into different levels of detail based on its granularity. Granularity refers to the frequency at which the data points are captured, ranging from milliseconds to hours to days or even longer periods. Different use cases require varying levels of granularity in data. For example, high-frequency trading systems may rely on millisecond-level data, while longer-term investors or academic studies may utilize daily or monthly data.

This study focuses on examining hourly and daily stock market data. This level of granularity provides a well-rounded viewpoint, as it captures the intra-day movements crucial for specific strategies and reflects the overall daily trends that help to even out any short-lived market irregularities. The study's data was obtained from Alpha Vantage (48), a trustworthy API provider for real-time and historical stock market data. The data acquisition process involved the following steps:

- **API Registration**: Users must first register for an API key from the Alpha Vantage website. This key authorizes the user to fetch data from the platform.
- **Endpoint Selection**: Depending on the granularity needed, one selects either the 'TIME_SERIES_INTRADAY' endpoint for hourly data or the 'TIME_SERIES_DAILY' endpoint for daily data.
- **Data Request**: A request is made using the API key and chosen endpoint, and the returned data typically comes in JSON or CSV format.
- **Data Parsing and Storage**: After retrieval, the raw data is parsed and stored in a structured format for subsequent analysis. Depending on the study's scope, irrelevant metrics might be discarded.

When working with market data, it's important to consider the level of detail, or granularity, of the data. Too much detail can result in capturing irrelevant market movements, also

known as "noise" (16). This noise can be mistaken for real market changes, when in fact it's just random fluctuations in asset prices. Therefore, it's crucial to filter out this noise and focus on identifying patterns, trends, and anomalies that have practical and actionable implications for research and analysis purposes (49).

3.1.1 Technical Indicators

Technical indicators are used to express chart patterns for financial assets. They are implemented to derive measures for trends and its momentum. The indicators are calculated with the `stockstats` (18) package. And the first indicator is the Simple Moving Average (SMA). Basically, the average of the stock, calculated with a rolling window of a fixed number of time steps (here 30 and 60). (7; 13) The second indicator is the Bollinger Band as a trend-following indicator, made up of the upper and lower standard deviations of the Simple Moving Average. This means that the first step is to calculate an SMA (usually for 20 days) and then construct the upper band by adding a standard deviation (STD) and the lower band by subtracting the STD. The position within the bands is thought to indicate whether an asset is overbought or oversold. (13) The Moving Average Convergence Divergence (MACD) is also used for momentum and trend following. It uses exponential moving averages (EMA) – meaning more recent data has a higher weight in the calculation of the average. A 26-period EMA is subtracted from a 12-period EMA to calculate the final MACD. (13) The Relative Strength Index (RSI) is also a momentum indicator. It compares in a set time period the days of an asset with positive price changes and with the days of negative ones. Thus, this index gives an estimate weather the positive or negative price changes had a stronger impact. (13) The Commodity Channel Index (CCI) is used to spot new trends by comparing the current (moving average) price with the historical average, adjusted by mean deviation. (29) An Average Directional Index (ADX) estimates the strength of a trend by calculating both a positive and negative momentum indicator. (28)

For risk estimates, the turbulence index, inspired by KRITZMAN (20) is implemented. It is defined as a state in which asset prices behave in an abnormal manner compared to their historical pattern, e.g., extreme price fluctuations, separation or convergence of (un)related assets. Based on the Mahalanobis distance for the average vector μ of historical returns, vector of asset returns $\mathbf{y}_t$ for period t, covariance matrix Σ of historical returns, the financial turbulence for time period t is formally defined as:

$$(\mathbf{y}_t - \mu)\,\Sigma^{-1}\,(\mathbf{y}_t - \mu)'$$

Another Risk indicator is an implied volatility index on the largest index fund of our used data (S&P 500). It reflects the expected market fluctuations, thus also called the fear factor. While calculating historical volatility – via standard deviation – is straightforward, the calculation of implied volatility is more complex and cannot be done with our provided stock data. As the exact calculations are beyond the scope of this thesis, we will only provide more intuition through our example.

Ticker	Stock name	Spot price	Strike price	Option price	Time to maturity
AAPL	Apple Inc.	167.92	170.00	1.62	30 days
META	Meta Platforms, Inc.	168.53	170.00	3.45	30 days
NVDA	NVIDIA Corporation	169.86	170.00	1.92	30 days

Table 1: Let us consider three *call* options - entailing the right to *buy* the underlying stock - with comparable prices on the stock market (spot price). The strike price of an option is the price at which the underlying asset can be bought until the date of maturity. We have selected strike prices, that are comparable to spot prices (i.e. (*at-the-money* options), thereby rendering options devoid of intrinsic value. An instance may be a call option, with a strike price of 105.00 and a spot price of 125.00, possessing an intrinsic value of 20.00, consequent to immediate arbitrage opportunities.

While all three stocks are traded at around the same price and the options have the same strike price and time to maturity, their option prices differ significantly. These prices arise directly from assessments made by all market participants. Therefore, a higher option price (minus intrinsic value) results from increased uncertainty about the underlying asset's whereabouts until maturity. The implied volatility derived from option prices is a measure of uncertainty and is commonly referred to as the fear factor. (19) (17)

3.2 Alternative Data

The influence of social media on stock market dynamics is indisputable (53) . A recent illustration of this phenomenon was observed when Elon Musk announced his intention to purchase Twitter, leading to a surge in the price of Twitter stock (14). Recognizing the potential of such influences, we integrated sentiment data into our model to capture the nuances of social media's impact on stock prices. We subscribed to the Financial Modelling Prep platform (9) to procure the requisite sentiment data, which offers a specialized Social Sentiment API (10).

The Social Sentiment API from Financial Modelling Prep is a robust tool that tracks and combines sentiments from major social media platforms like Reddit, Yahoo, StockTwits, and Twitter. Importantly, this tool updates every hour and keeps data from the past two years. This two-year limit did set some boundaries for our study. We carefully downloaded hourly data for a selected list of 30 stocks during this timeframe. Since FMP does not provide daily sentiment summaries, we calculated daily averages from the hourly figures to get daily sentiment values. Our main sentiment measures were the number of posts, likes, impressions, and overall sentiment score for each stock at each time point. It is important to mention that some time points were missing data, so we used linear interpolation between the closest data points to fill these gaps.

Though the Financial Modelling Prep platform has other data types, like fundamental and senate trading data, we focused mainly on sentiment data. This specific focus lets us explore this area more deeply, ensuring a comprehensive and detailed comparison.

3.3 Data pre-processing

Before the models can be trained, all data needs to be accurately analysed and pre-processed. The first step is to plot the market data and detect anomalies. For example for Google, Amazon and Tesla we can observe sudden price changes of magnitudes. The reason behind these jumps are stock splits, where shares are split by an integer value to issue more stocks by the company. Shareholders do not lose value as the number of their stocks is multiplied by the same integer. This is why the volume and prices need to be adjusted.

Figure 1: Stock split for Tesla.

Furthermore, we focus on monitoring daily changes in interaction with various markets. Consequently, we calculate these changes as daily *logarithmic returns* for several reasons. The most significant one is that a stock market can be modelled using a geometric Brownian motion. This allows us to apply *Itô's lemma*[1] to obtain a log-normal distribution. As we aim

[1]Itô calculus is beyond the scope of this thesis, but is a recommendation for interested readers and enthusiasts for stochastic calculus.

to achieve normally distributed variables, we utilise the natural logarithm. Additionally, our approach normalises the data set. (19) (17) Also for general purposes the overall data set has been normalised. This will be demonstrated in the results. For better comparability and explain-ability a random number has been added. It is constructed once and not changed for each seed. This is an important step for the upcoming sensitivity analysis as an benchmark of noise. To ensure the data is in the correct order, it is sorted by date and symbol. The last step is separating the training data from the testing data with a split of 80/20.

3.3.1 Noise reduction

Unfortunately, financial data is noisy and possesses a low signal-to-noise ratio which makes it difficult to use as a basis for machine learning. (7; 17) One solution to overcome this problem as described in (22) is to use technical indicators to stabilise the state due to their inherent characteristic of a rolling window.

> *"Technical indicators can reflect the changes in the stock market from different perspectives and reduce the influence of noise."*
>
> – LI, *(22, p. 1)*

Hence, they are added to the data set. But this is only the first step to improve our signal-to-noise-ratio. Our market data of current stock prices can be described as "a small bird flying through dense jungle foliage at dusk"(36, p. 497). You only notice short, interrupted flashes of movement, which you use to guess the birds current and next position so that you won't loose it. This is the intuition behind the optimal Bayesian filter – the Kalman Filter – which estimates time dependant state variables from noisy observations. This can be the position and velocity of a bird or our stock prices. Without going into too much detail, the Kalman filter works iterative.

The first step is the initialization where the state and belief of the filter is initialized. The other two steps are recursive: predict and correct. Based on the past data, the state of the next time step is predicted and the belief is adjusted to account for uncertainty. For the update, the actual measurement and its associated belief about the accuracy are used to calculate the probability of both the measurement and prediction. Depending on those probabilities a state between both of them is adapted as "true" measurement and the belief in its accuracy is updated. This iterative process is superior to basic smoothing algorithms as it leverages the probabilities of outliers. However, the run time is extremely high. (50; 36; 21)

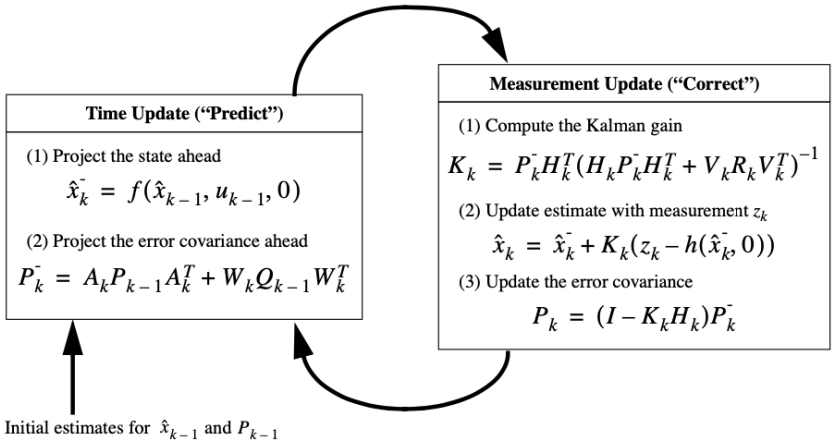

Figure 2: Recursive Process of predict and correct. (50, p. 6)

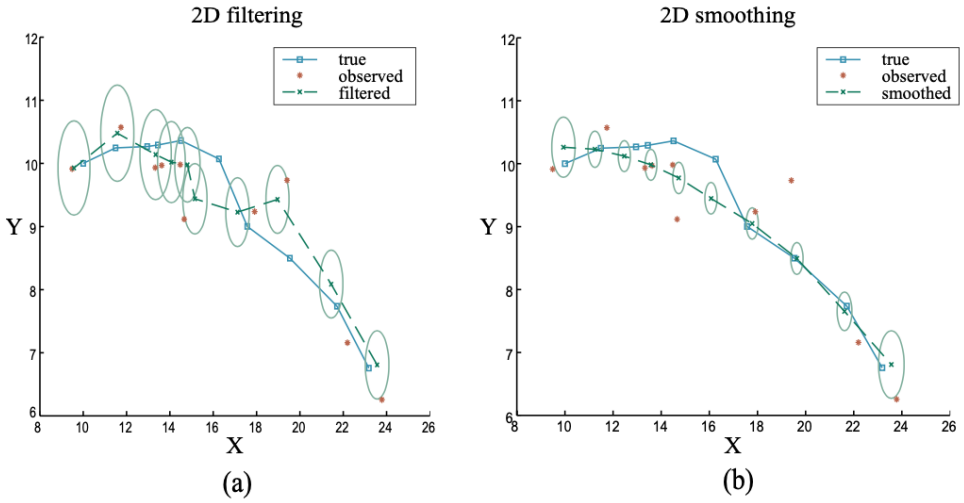

Figure 3: Results of Kalman filtering (a) and smoothing (b) for an arbitrary object. The true position, it's actual noisy measurements and the estimated states of the filter are displayed. Ovals are indicating the variance in the position estimates. (36, p. 502)

4 RL Approach: FinRL

FinRL is a framework designed for research in the field of reinforcement learning within the context of finance and trading. It is constructed upon the OpenAI Gym framework and is designed to simulate real stock markets using actual data. The aim of FinRL is to provide easy accessibility to this domain by offering various environments and fine-tuned Deep Reinforcement Learning (DRL) agents, among other resources (26). Figure 4 illustrates the structure and various functionalities of the FinRL library. In this thesis, we have opted for

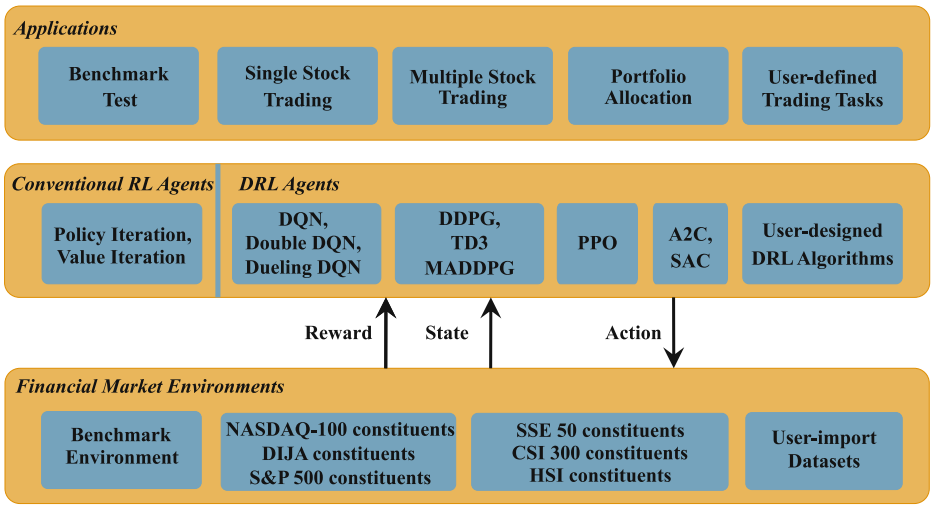

Figure 4: Structure of the FinRL Framework with visualization of options among three main modules (26).

the application of single-stock trading using the implemented agents A2C, DDPG, PPO, TD3, SAC, and have leveraged the opportunity to employ our own dataset.

4.1 Environment

The foundation for the environment used in this research is based on the FinRL Stock Trading Environment, which was modified to meet our specific requirements. As FinRL follows the OpenAI gym-style, every environment provides the functions `reset()`, `step()` and `reward()`. As the name *reset* implies, this function resets the environment, returning it to its initial state, `step()` takes an action from the environment and updates the state and `reward()` calculates the corresponding reward for the performed action (24). For stock trading, it is essential to have functions for buying and selling stocks. The former must assess how much budget is available for purchasing, taking into account transaction fees. The latter function has to check how many of the stocks to be sold are owned, ensuring that it does not sell stocks that are not in possession. If a turbulence threshold is provided, it is examined whether the turbulence indicator exceeds this threshold, as a high turbulence index corresponds to a heightened risk of a stock market crash. If the turbulence index surpasses the provided threshold, buying is stopped and all stocks are sold (26). The interface between the agent and the environment is the state space which is defined in the environment. Therefore functions to initiate and update the state are provided.

In the original environment, the state consists of the available budget, the price of each stock at the current timestep, the portfolio, i.e. how many stocks we own from each stock and all technical indicators. A graphic illustrating the state can be found in Figure 5.

Figure 5: Original state provided by the FinRL Stock Trading Environment. It consists of the current budget, the prices for each stock, the number of stocks owned of each stock and the technical indicators for each stock.

The first modification made to the environment was the addition of a sentiment list to the state, following the example of technical indicators. This allows providing the agent with additional sentiment data for decision-making. As in Section 3.3 elaborated it is useful to use daily changes instead of absolute prices. Replacing the absolute prices seemed less logical due to their necessity in calculations related to potential purchase quantities. Therefore, the rate of change was incorporated as additional information into the state. The literature highlights a correlation between stock prices and trading volume, making it sensible to include volume in the state as well (5)(12). Overall, a graphic illustrating the state of the modified environment can be found in Figure 6. Hereby, budget is of size one, the price-block is as big as the number of stocks which is also the case for the portfolio block as well as for the daily change rate and the traded volume. The technical indicator part is of size $number of technical indicators \cdot number of stocks$ and the sentiment data part is of size $number of sentiment data \cdot number of stocks$. As we are using 30 stocks that can be traded by the agent our state is of size 421 when not using sentiment data and of size 541 when including sentiment data. As this is already a state of big dimension, the plan of adding more than one timestep or more alternative or fundamental data had to be rejected.

Figure 6: Modified state consisting of the same components as the original state and additionally sentiment data for each stock, the change rate and the traded volume of each stock.

The original environment was designed exclusively for the use of daily data, which necessitated several adjustments to accommodate hourly data processing, including the calculation of the Sharpe ratio. The final modification made pertained to the seeding of the random function. To enhance result reproducibility, the seed is now provided within a range from zero to n. Consequently, the seed function in the environment had to be adjusted to meet this requirement. It involves passing the current seed to the environment, which then utilizes it to seed its random function.

4.2 Agents

Stable Baselines3 is a widely used library for training and evaluating reinforcement learning agents. It offers state-of-the-art algorithms, providing users with practical tools for their RL tasks (35). Our study involved 50,000 timesteps for each of the agents we used from the Stable Baselines3 library. Here is an overview of those agents:

- **A2C (Advantage Actor-Critic)**: A synchronous, deterministic variant of Asynchronous Advantage Actor-Critic (A3C) (30).

  Parameters: We've employed the standard model provided by the FinRL Stable Baselines3 library without any modification to the default parameters.

- **PPO (Proximal Policy Optimization)**: An on-policy algorithm which aims to alleviate issues related to large policy updates (38).

  Parameters: *n_steps:* 2048 - The number of steps to run for each environment per update. *ent_coef:* 0.01 - Entropy coefficient for exploration. *learning_rate:* 0.00025 - The learning rate of the optimizer. *batch_size:* 128 - Size of the batches for learning.

- **SAC (Soft Actor-Critic)**: An off-policy actor-critic deep RL algorithm based on the maximum entropy reinforcement learning framework (15).

  Parameters: *batch_size:* 128 - Size of the batches. *buffer_size:* 100,000 - Size of the replay buffer. *learning_rate:* 0.0001 - The learning rate of the optimizer. *learning_starts:* 100 - Number of steps before starting training. *ent_coef:* "auto_0.1" - Coefficient for the entropy regularizer, with automatic tuning.

- **TD3 (Twin Delayed Deep Deterministic Policy Gradient)**: An algorithm that addresses function approximation errors in DDPG (11).

  Parameters: *batch_size:* 100 - Size of the batches. *buffer_size:* 1,000,000 - Size of the replay buffer. *learning_rate:* 0.001 - The learning rate of the optimizer.

- **DDPG (Deep Deterministic Policy Gradient)**: A model-free, off-policy, actor-critic algorithm (23).

  Parameters: We've utilized the standard model from the FinRL Stable Baselines3 library without adjusting any default parameters.

  Each of these agents brings unique advantages to different scenarios, and their performances can be contingent upon both the nature of the task and the specific parameter configurations used.

5 Sensitivity Analysis

In our project, we opted for the gradient-based method for sensitivity analysis. Our decision was primarily influenced by two key attributes of this method: its simplicity and generalizability (33). The gradient-based method offers a straightforward approach to understanding how changes in input variables can influence the output of a model. By computing the gradient of the output with respect to the input, we can directly measure the model's sensitivity to each input variable. This method is not only intuitive but also generalizable across various types of models and datasets.

To facilitate this analysis, we employed the TensorFlow Python Library (1), a powerful tool renowned for its capabilities in deep learning and gradient computations. With TensorFlow, we can efficiently compute the gradients of our model's output with respect to its inputs.

Our primary focus will be on visualizing the absolute values of these gradients. Specifically, we are interested in the gradients of features that correspond to:

- **Different stocks**: This will provide insights into which stocks have the most influence on the model's predictions. By understanding the sensitivity associated with each stock, we can better gauge the importance and impact of individual stocks on our model's decision-making process.
- **Different days**: Analyzing the gradients across different days will allow us to discern any temporal patterns or anomalies that might exist. This can be crucial in understanding how the model's sensitivity varies over time, especially in the volatile world of finance where market dynamics can change rapidly.
- **Different features**: Beyond stocks and days, we will also delve into the gradients of other features, such as technical indicators and sentiment inputs, used in our model. This will shed light on which specific attributes or indicators are deemed most crucial by our model.

In summary, our gradient-based sensitivity analysis has several benefits. Firstly, it helps us understand the deeper workings of our model's decision-making process. We can determine the main factors influencing the model's predictions by visualizing and interpreting the gradients. This insight will lead to a more knowledgeable and sturdy use of our reinforcement learning model in finance.

Furthermore, this analysis helps us identify features that might not be very important, like certain technical parameters for stocks or specific sentiment data. By spotting and then leaving out these less impactful features, our model becomes more streamlined. As a result, it can make predictions faster and train more efficiently. On the other hand, by finding the most impactful features or stocks, we can give them more importance in our model. This ensures that the most crucial parts influencing our predictions are highlighted. This combined strategy of reducing unnecessary features and focusing on the important ones not only improves the model's accuracy but also makes it run more efficiently.

6 Backtesting

To verify the models and rank them by their capabilities, a holistic backtesting approach is defined. The basic idea of backtesting is trading on the test data, which was separated before. The results include the total returns (corresponds to $alpha$), but also various types of other metrics, mostly focused on risk (corresponds to $beta$). To ensure that random effects are not influencing the underlying generating process, we not only repeat the training on multiple seed numbers, but also include all these models in our back-tests simultaneously. The most important biases to keep in mind are the survivor-ship bias of only including the companies that "survived" and are still traded on the financial markets. The look ahead bias of unconsciously allowing the model to access future data. And most importantly the optimisation bias – the over-fitting of an model. Also the sample selection bias is of importance: training and testing data that has been selected can include underlying trends that will not be repeated and are an anomaly. Especially for our problem statement we have to include trading specific obstacles, like transaction costs, slippage, volume and liquidity. Slippage describes the price differences between expected price due to market data while ordering and the final price, that is filled, during the execution of the trade. Volume and liquidity are not relevant, since we only include the largest stocks of the S&P500 and test with 1mn $ of cash and a maximum of 100 units per trade. The transaction costs and slippage are controlled in the environment by a conservative estimate of 1% for each trade. (24; 17; 7; 13)

For a fair comparison, all three dimensions of strategies, which are based on different approaches, are included: trading on classical signals, portfolio theory and buy and hold. This includes strategies based on technical indicators like RSI, Moving Average and Bollinger Bands. But also more conservative strategies like Mean Variance Optimization – one of the standard approaches in modern portfolio theory. It calculates the variances as a measure of risk to allocate a weight for the final trade-off between risk and return. Thus the final portfolio maximizes the return per risk. Also included is the straight forward buy and hold approach, which is only allowed to hold the same stocks our RL models have been trained on.

As the last building block further metrics have been calculated. This is based on the `PyFolio` (34) package and includes risk metrics like max draw-down (describing the highest amount lost – from peak to trough – in a portfolio), and its matching return-per-risk metric, the calmar ratio. Also the sharp ratio gives an risk adjusted return metric: the returns above an benchmark are adjusted to its standard deviation to estimate the risk-adjusted performance.

In total this allows a holistic estimate to research an optimal trading strategy given our demands. And with these backtests, the models parameters can be optimized, and the sensitivity analysis conducted to obtain the best performing, yet still explainable models. Thus, a randomly trading model with higher excess returns ($alpha$) with identical volatility ($beta$) is not preferred over a model whose actions are transparent.

7 Results

In this section, we present the findings and outcomes of our research. The data provides insights into the patterns and implications discovered during the study.

7.1 Optimizing Model Hyperparameters

In the quest for optimal model performance, hyperparameter tuning is an essential step. It involves fine-tuning various parameters to find the best combination that maximizes the model's efficiency and accuracy. For our setup, we considered four primary hyperparameters:

- **Data Normalization**: Neural networks often benefit from normalized data, especially when the range of input values is vast. In our raw dataset, the highest values reach up to while the lowest values dip below zero. Such a broad spread can potentially hinder the model's learning capability. To investigate the impact

of normalization, we tested two agents, A2C and PPO, with both normalized and non-normalized data.

- **Data Smoothing**: Smoothing data can help in reducing noise and capturing the underlying trend in the data. The idea behind this is to provide the model with a clearer signal, potentially aiding in better decision-making. We evaluated the effect of data smoothing using the A2C and PPO agents.
- **Transaction Costs**: Introducing transaction costs can influence an agent's trading frequency. By setting a higher transaction cost (0.01) versus a lower one (0.001), we hypothesized that the agent would trade more conservatively, leading to more stable performance. The rationale is that higher costs would deter the agent from making frequent trades unless the predicted rewards are substantial.
- **Turbulence Threshold**: Turbulence in the stock market can lead to erratic price movements. By introducing a threshold for turbulence, we aimed to make the agent more cautious. If a stock's turbulence index exceeds this threshold, the agent refrains from buying or selling it. This strategy was designed to mitigate potential losses during highly volatile periods. We tested this approach using the A2C and SAC agents.

7.1.1 Data Normalization

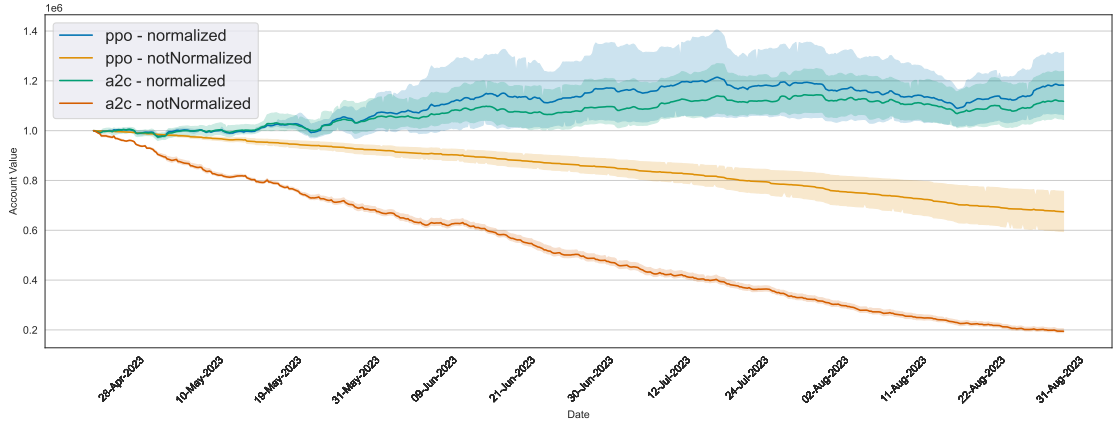

Figure 7: Performance comparison of A2C and PPO agents with normalized vs. non-normalized data.

Figure 7 offers a comprehensive visualization of the influence of data normalization on the performance of A2C and PPO agents. In the non-normalized scenario, both agents experience a decline in account value, indicating a loss in their trading activities. This

downward trajectory is almost linear, suggesting a consistent underperformance over time. Conversely, when trained with normalized data, both agents demonstrate a positive return on investment. Specifically, the A2C and PPO agents manage to achieve gains of approximately 10% and 20%, respectively. This stark contrast underscores the importance of data normalization in enhancing the efficency of reinforcement learning agents in stock trading.

7.1.2 Data Smoothing

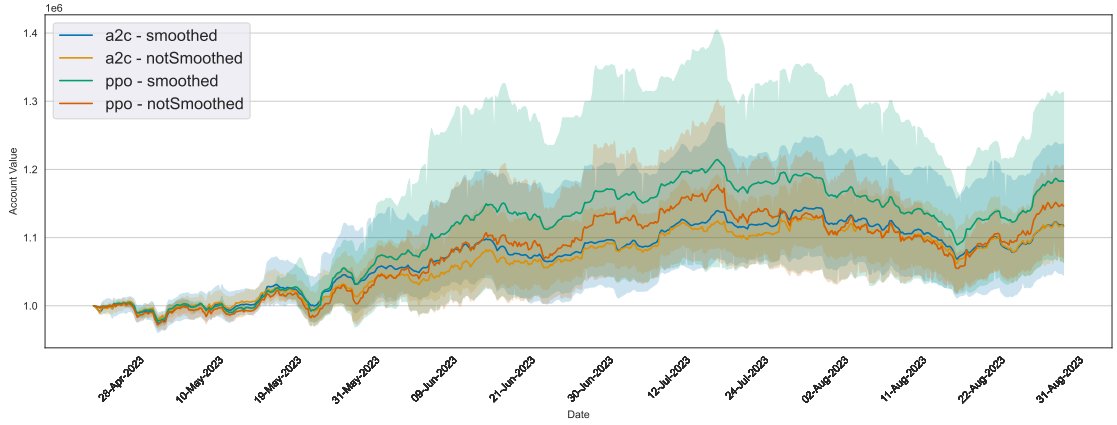

Figure 8: Performance comparison of A2C and PPO agents trained on smoothed versus non-smoothed data.

In Figure 8 it is proven that both A2C and PPO agents trained on smoothed data outperform their counterparts trained on non-smoothed data. This suggests that employing a Kalman filter for noise reduction, thereby smoothing the data, can be beneficial for reinforcement learning agents in stock trading.

7.1.3 Transaction Costs

In Figure 9 it is shown that both SAC agents manage to generate profits - regardeless of the transaction cost settings. However, there are clear distinctions in their trading behaviors. The agent trained with low transaction costs tends to fully utilize its capital to purchase stocks, resulting in a higher final account value. Yet, this aggressive trading strategy comes at a cost: its performance exhibits a standard deviation that is five times larger than that of the agent trained with higher transaction costs. This heightened volatility suggests a less stable trading behavior. Given the importance of stability and predictability in stock trading, we opted for the higher transaction costs setting to ensure a more consistent and reliable performance from our agent.

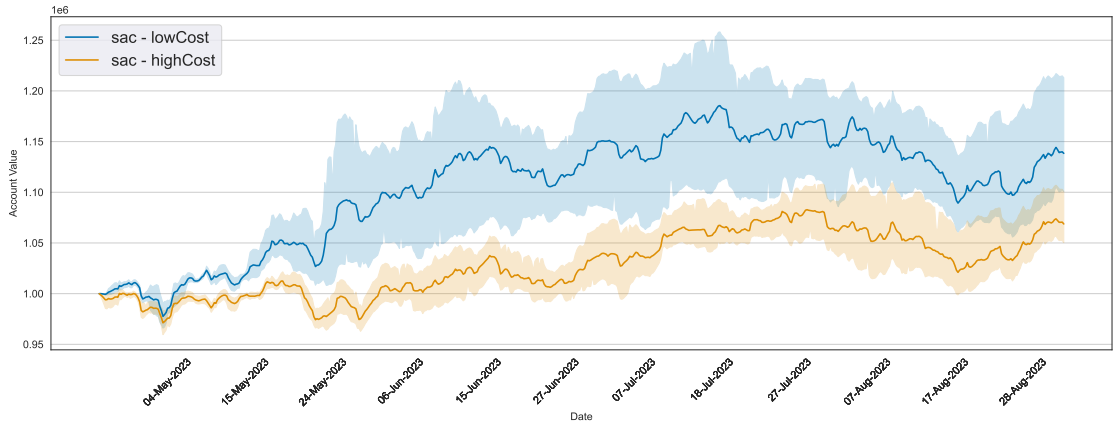

Figure 9: Performance comparison of the SAC agent under varying transaction costs.

7.1.4 Turbulence Threshold

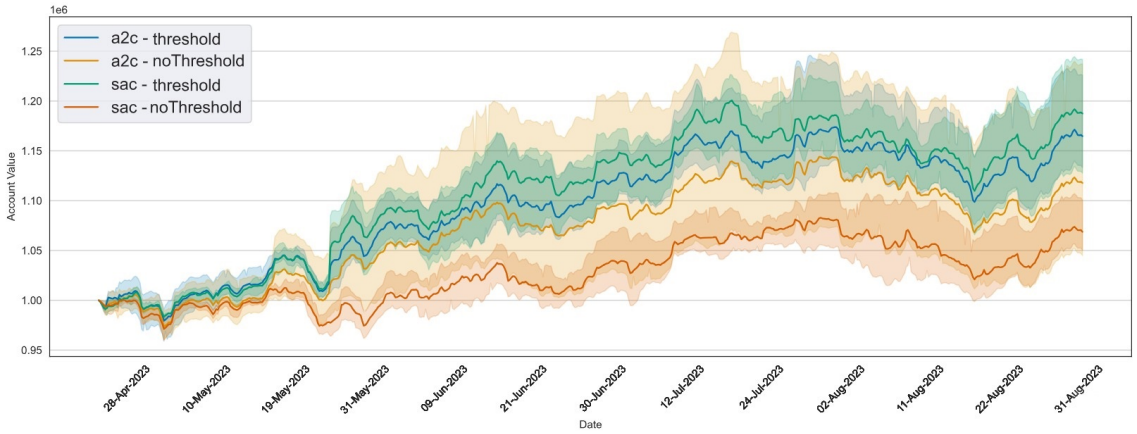

Figure 10: Performance comparison of A2C and SAC agents with and without setting a turbulence threshold.

Figure 10 illustrates a clear performance distinction between agents trained with setting a turbulence threshold and those without. Both A2C and SAC agents exhibit superior performance when utilizing the turbulence threshold. Moreover, the agents without the turbulence threshold demonstrate a higher deviation, indicative of their unstable trading behavior. Given the evident benefits of stability and improved performance, we opted to incorporate the turbulence threshold in our subsequent experiments.

7.2 Evaluating the Impact of Sentiment Data Integration

From Figures 12 and 11 it is evident that agents trained on a dataset including sentiment data consistently outperform those trained without sentiment data, irrespective of whether

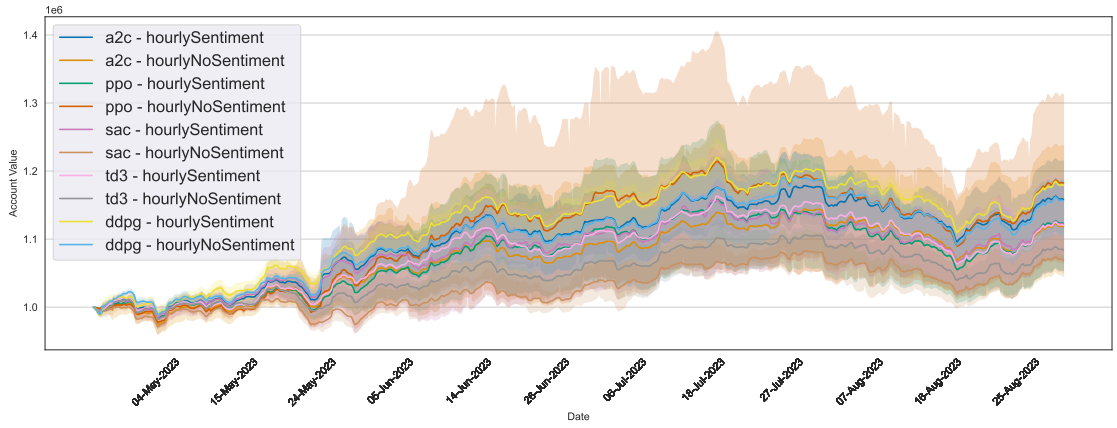

Figure 11: Performance comparison of agents trained with hourly data, with and without sentiment integration.

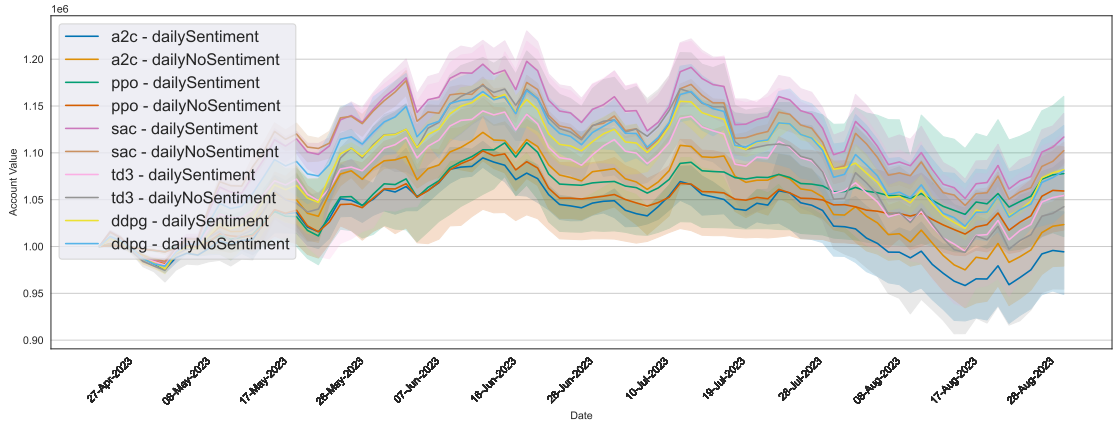

Figure 12: Performance comparison of agents trained with daily data, with and without sentiment integration.

the data is hourly or daily. This underscores the potential value of integrating sentiment data into the training process, suggesting that sentiment provides crucial insights that enhance the predictive capabilities of the agents.

For a detailed pairwise comparison of each agent's performance, considering both daily and hourly data, with and without sentiment integration, please refer to the appendix.

7.3 Comparing Daily and Hourly Data Models

It's important to note that the time scales for hourly and daily data are inherently different, making a direct comparison between the two challenging. However, when we evaluate

the end account values of the agents, a discernible pattern emerges. Agents trained on hourly data consistently outperform those trained on daily data. This suggests that the finer granularity of hourly data provides agents with more detailed insights, allowing them to make more informed decisions and, consequently, achieve better performance in stock trading.

7.4 Benchmarking Against Traditional Models

In our quest to evaluate the efficacy of our reinforcement learning agents, we benchmarked them against four classical trading strategies: Buy and Hold, Bollinger Bands, RSI, and MACD.

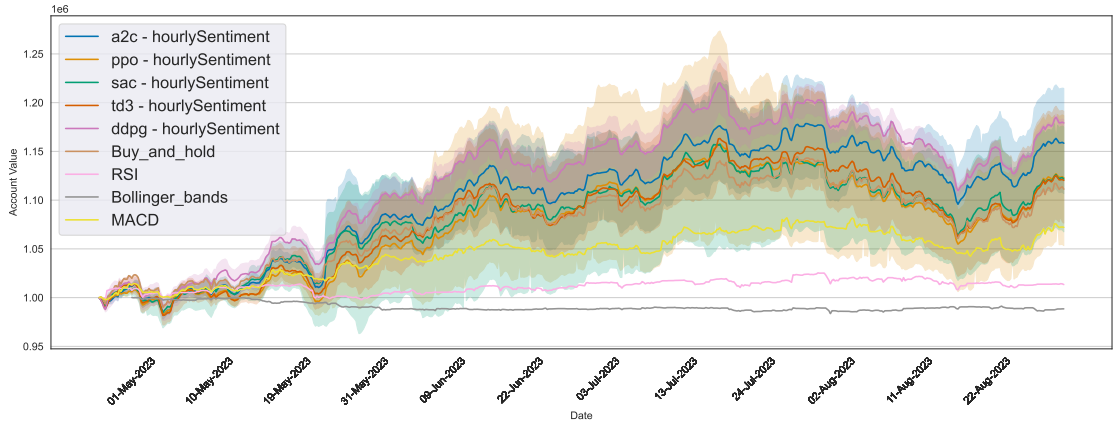

Figure 13: Performance comparison of RL agents trained with hourly data against classical benchmarks. Notably, the classical benchmarks operate without transaction costs.

From Figure 13, it's evident that our RL agents consistently outperform all classical benchmarks, even when considering that the classical strategies are not burdened with transaction costs. Among the classical strategies, Buy and Hold comes closest in performance to our agents. A more detailed comparison between our agents and the Buy and Hold strategy is presented in Figure 14.

These results underscore the potential of RL in stock trading, indicating its ability to adapt and optimize trading strategies in a dynamic market environment, outpacing traditional, static strategies.

7.5 Analysis of Model Sensitivity

In this section, we present the outcomes of our comprehensive sensitivity analysis.

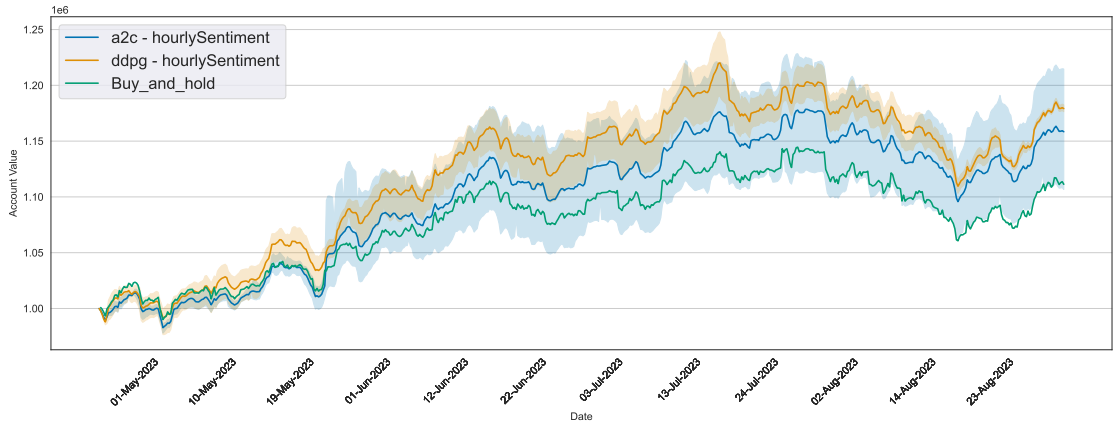

Figure 14: Performance comparison of RL agents trained with hourly data against the Buy and Hold strategy.

7.5.1 Visualizing Gradients: A Heatmap Approach

In our attempt to understand the model's decision-making process, we visualized the gradients using a heatmap, as depicted in Figure 15. This heatmap represents the gradients for the Soft Actor Critic (SAC) model. A notable observation from the heatmap is the zero gradients for features in the range 330-360. At first glance, this might raise concerns about the model's functionality. However, a deeper inspection reveals that these features correspond to random numbers generated for each stock, amounting to 30 random numbers in total. The zero gradients indicate that the SAC model has effectively learned to disregard this random information, recognizing its irrelevance in the decision-making process. This behavior underscores the model's capability to discern and prioritize meaningful features, filtering out the noise or irrelevant data.

Following the heatmap visualization for the SAC model trained on hourly data, we further examined the gradients for the SAC model trained on daily data, as illustrated in Figure 16. This heatmap reveals that, similar to the hourly model, the agent tends to ignore the random features (330 - 360). Interestingly, there's also a discernible pattern where the agent seems to disregard certain stocks, as represented by features 30 - 60. This behavior suggests that the model might find these specific stocks less influential or irrelevant for its decision-making process when trained on daily data.

In Figure 17, we present the heatmap of gradients for the PPO model trained on daily data with sentiment incorporation. A striking observation from this heatmap is the presence of higher gradients on specific days, irrespective of the features. This indicates that the agent

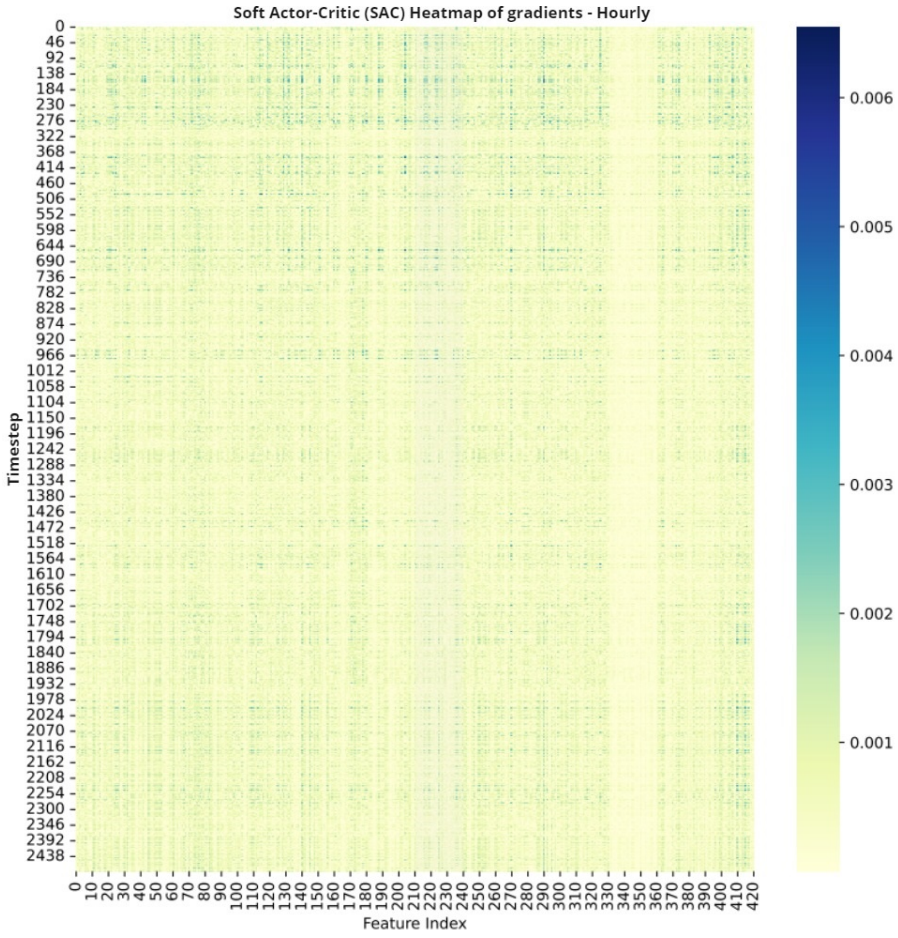

Figure 15: Heatmap of gradients for the SAC model: The agent tends to ignore random features (330 - 360). Additionally, one of the technical indicators (features 210 - 240, correspond to dx 30) exhibits a slightly larger gradient than others. The agent was trained on hourly data without incorporating sentiment.

assigns more importance to these particular days, possibly due to significant market events or shifts in sentiment on those days.

Another interesting observation emerges from the heatmap of the SAC model trained on hourly data with sentiment incorporation, as shown in Figure 18. This heatmap distinctly showcases the agent's ability to prioritize certain features, as evidenced by the higher gradients. Such behavior indicates that the model has identified these features as particularly influential in its decision-making process.

However, it's worth noting that the agent does not exhibit any clear prioritization for specific days, suggesting that the temporal aspect might not be as influential for this model configuration. Additionally, the presence of gradients for random variables is a reminder

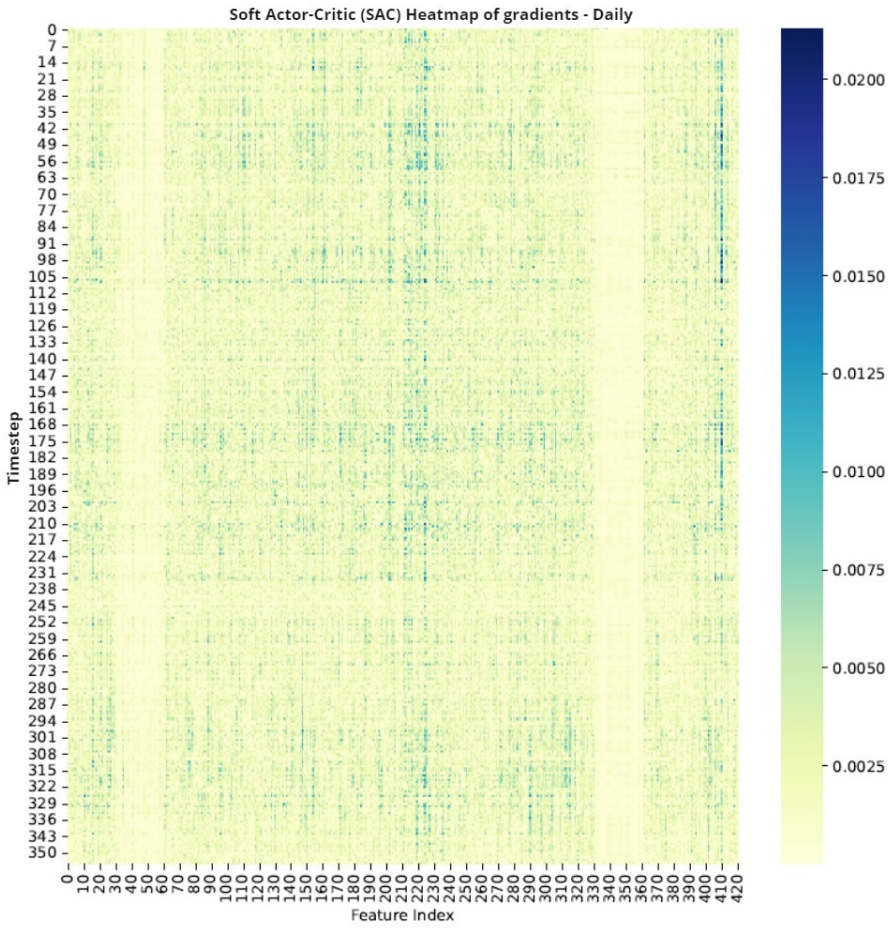

Figure 16: Heatmap of gradients for the SAC model: The agent tends to ignore random features (330 - 360) as well as certain stocks, represented by features 30 - 60. The agent was trained on daily data without incorporating sentiment.

that while the model can discern and prioritize meaningful features, it doesn't always disregard noise or irrelevant data entirely. This behavior underscores the importance of further refining the model or possibly introducing regularization techniques to minimize the influence of such noise.

7.5.2 Evaluating Temporal Effects on Predictions

As observed in Figure 17, certain agents demonstrate a distinct behavior where specific days hold more significance than others in their decision-making process. The PPO model, in particular, exhibits this tendency consistently for both daily and hourly data configurations. To delve deeper into this temporal effect, we computed the sum of gradients across all

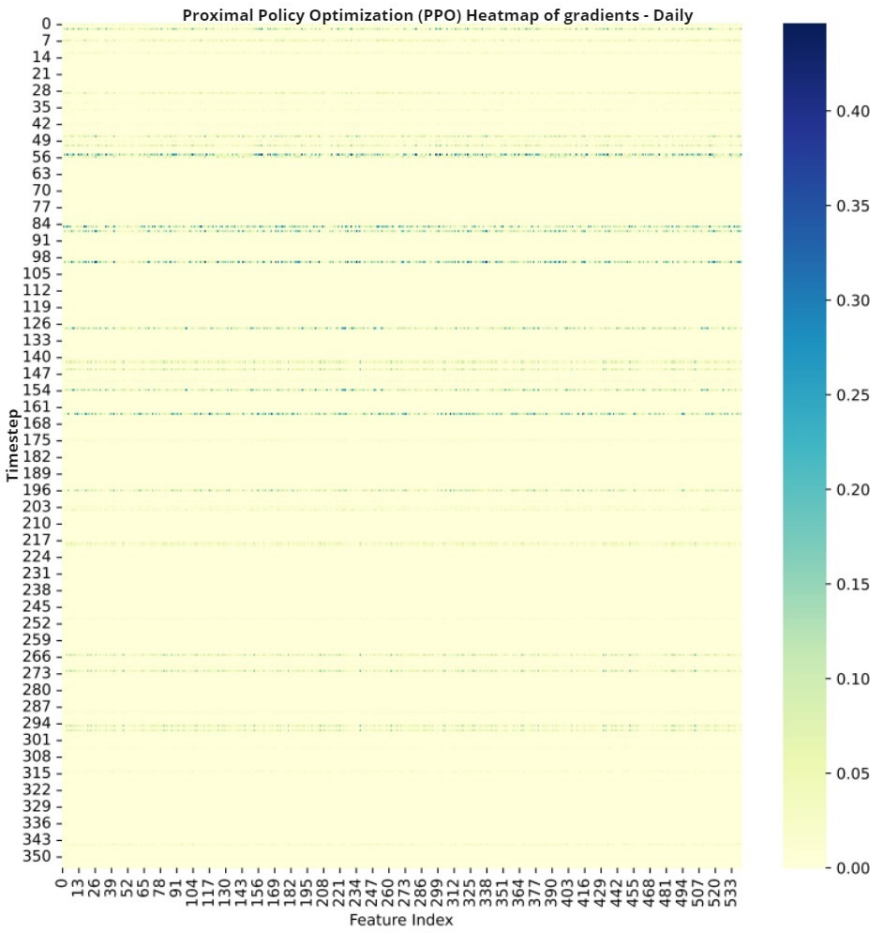

Figure 17: Heatmap of gradients for the PPO model: The agent has a higher gradient on some days, but not depending on features. The agent was trained on daily data incorporating sentiment.

features for each date within the training range. The results of this computation are visualized in the subsequent figure.

The visualization in Figure 19 presents the gradients over time for the PPO agent trained on daily data without sentiment. A striking observation is the pronounced spike in gradients at the beginning of 2022. This heightened sensitivity correlates with the onset of the military conflict in Ukraine, which had substantial repercussions on global stock markets. Such events often lead to increased market volatility, with stocks reacting sharply to news and geopolitical developments. The PPO agent's heightened gradient during this period indicates its increased sensitivity and adaptability to these market shifts, emphasizing the importance of certain days or events in its decision-making process. This behavior

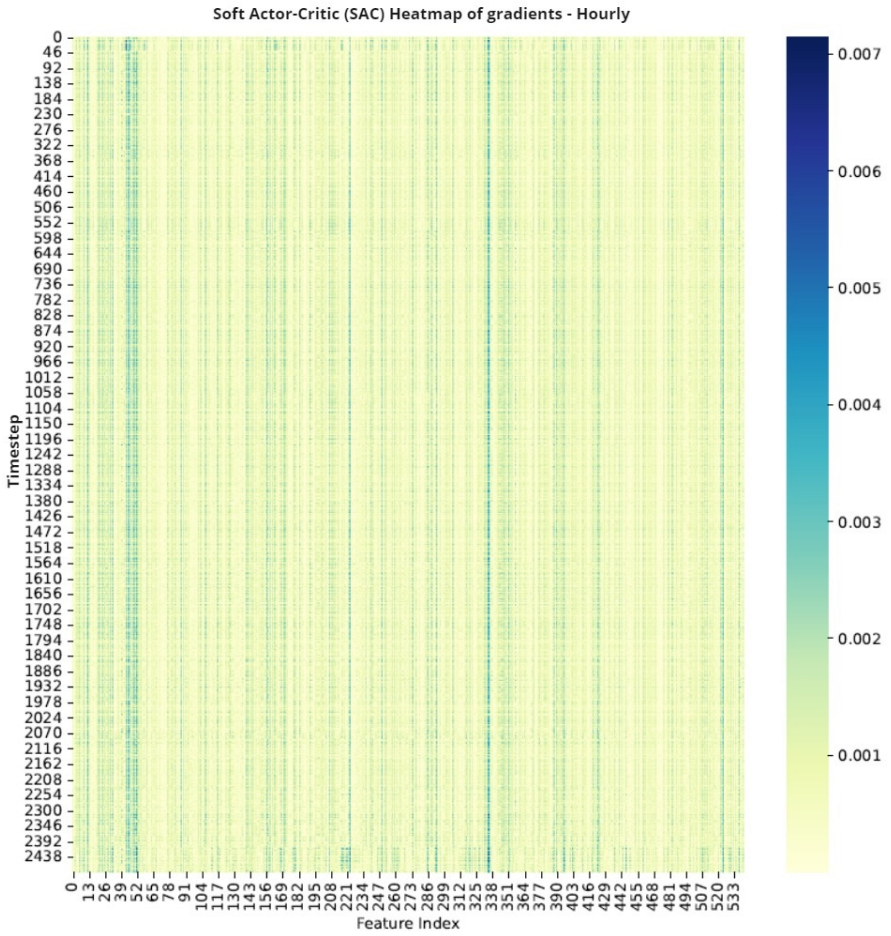

Figure 18: Heatmap of gradients for the SAC model trained on hourly data with sentiment: The agent exhibits clear feature prioritization, as indicated by higher gradients (features 40 - 52, 222, 337). However, there's no discernible prioritization for specific days, and gradients are also observed for random variables, suggesting the model's sensitivity to noise or irrelevant data.

underscores the capability of RL models like PPO to dynamically adjust their strategies in response to significant market events, even when not explicitly trained on sentiment data.

Figures 20 and 21 provide a visual representation of feature importance for the SAC agent trained on daily and hourly data, respectively, without sentiment.

From the plots, it's evident that the agent effectively disregards the random variable, as indicated by its low gradient. This behavior is consistent with our earlier observations from the heatmap, underscoring the model's ability to filter out noise or irrelevant data. Additionally, the agent seems to downplay the significance of certain stocks, suggesting that not all stocks contribute equally to its decision-making process.

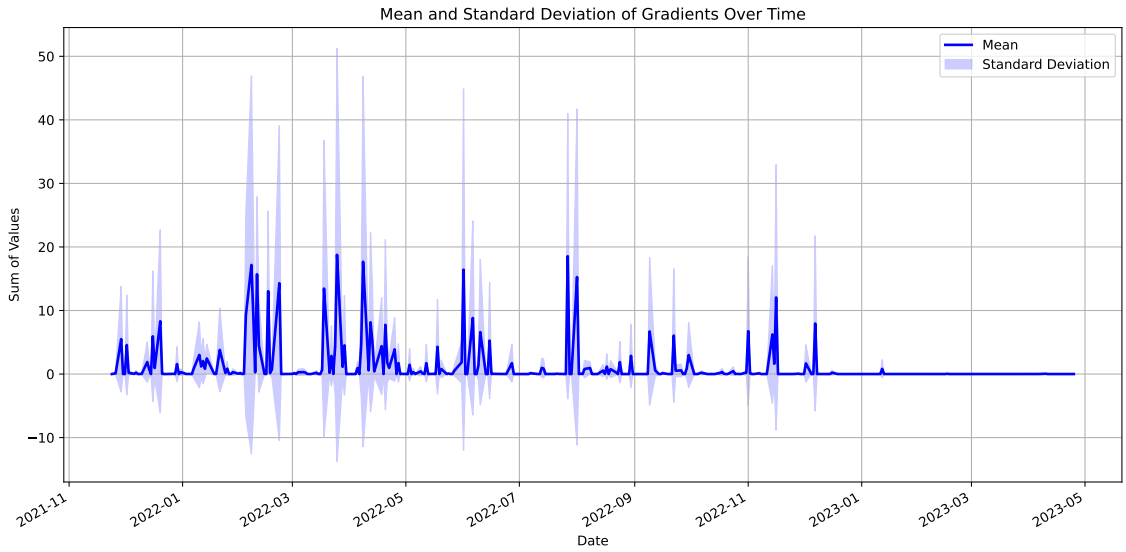

Figure 19: Temporal Gradient Distribution for PPO Agent Trained on Daily Data Without Sentiment

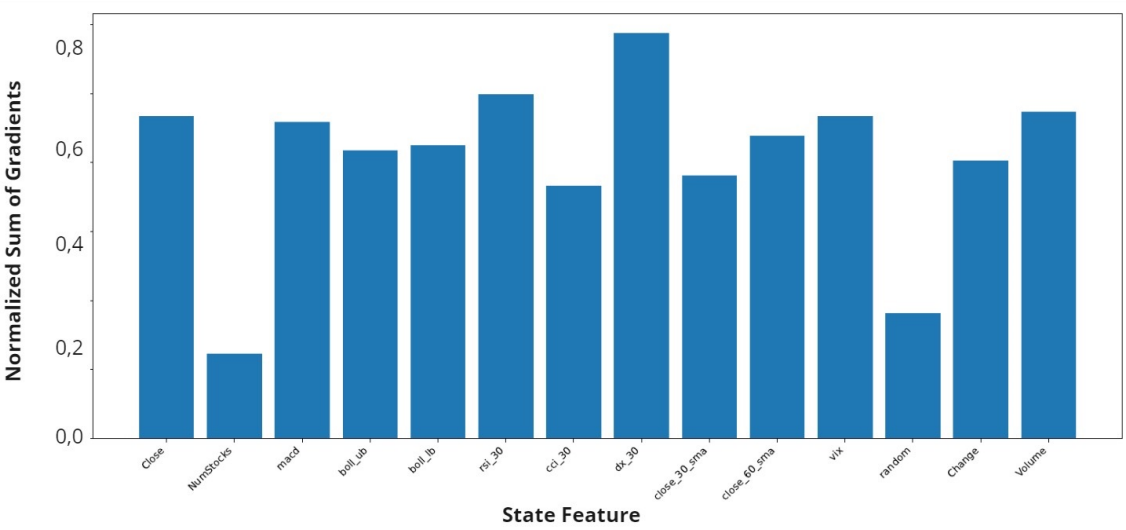

Figure 20: Feature Importance Visualization for SAC Agent on Daily Data without Sentiment: Emphasis on the ‘dx 30‘ Technical Indicator and Disregard of Random Variables

Among the features, the technical indicator dx 30 stands out with a noticeably higher gradient, especially in the daily data plot. This indicates that the SAC agent attributes a higher importance to this particular technical indicator compared to others. While this distinction is more pronounced in the daily data, a similar trend, albeit less distinct, is observable in the hourly data plot. The elevated gradient for dx 30 suggests that this

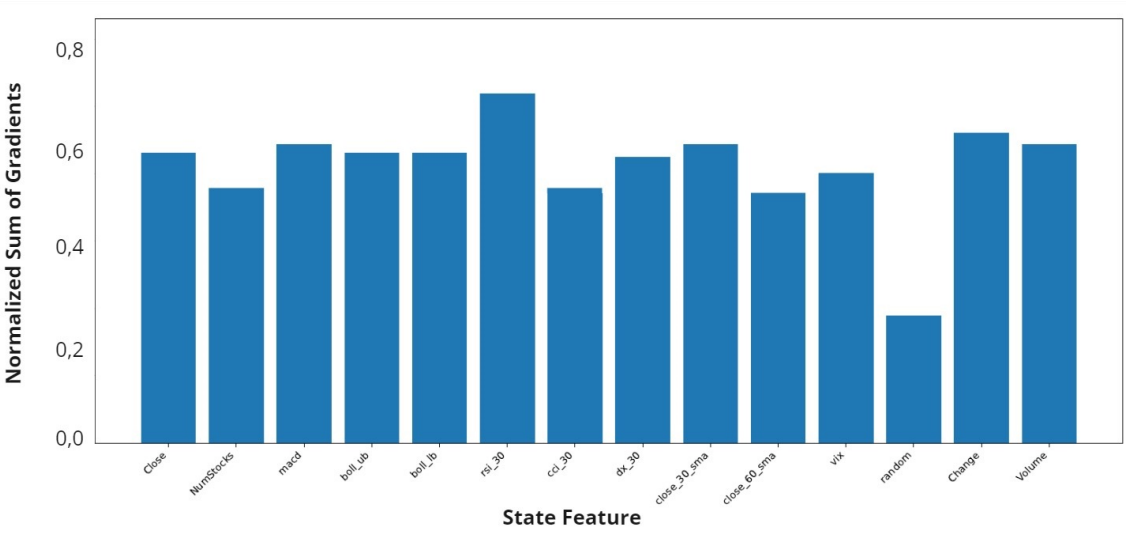

Figure 21: Feature Importance Visualization for SAC Agent on Hourly Data without Sentiment: Subtle Emphasis on the ‘dx 30‘ Technical Indicator and Disregard for Random Variables

technical indicator might be a more influential factor in the agent's decision-making process, potentially due to its ability to capture specific market dynamics or trends.

These observations emphasize the nuanced behavior of the SAC agent, highlighting its capability to discern and prioritize specific features while filtering out less relevant information. Such insights are invaluable for refining the model further and tailoring its training to emphasize key indicators.

The figure 22 provides an intriguing perspective on the SAC agent's behavior when trained on daily data with sentiment. While most features exhibit a relatively uniform gradient, suggesting a balanced consideration by the agent, there are distinct peaks for the number of stocks, random variables, and notably, the dx 30 technical indicator.

The prominence of random variables in the gradient distribution is particularly interesting. Given the extensive state length of 541, it's plausible that the agent is overwhelmed with information, leading to a diffusion of feature importance. This "information saturation" might cause the agent to inadvertently assign significance to even the random numbers, mistaking noise for potentially valuable data. Such behavior underscores the challenges of training models on high-dimensional data and emphasizes the need for effective feature selection and engineering to guide the agent towards more meaningful patterns.

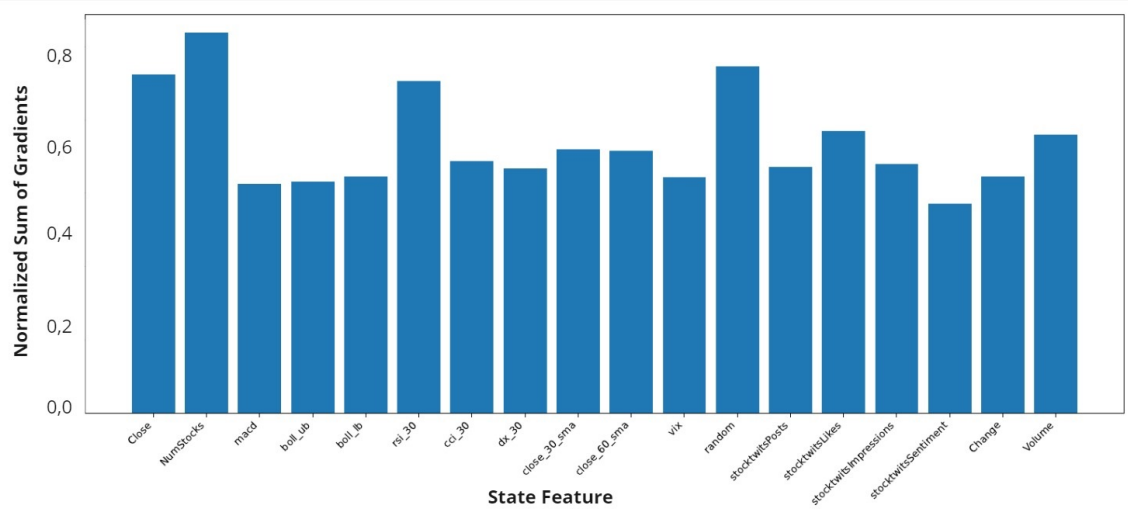

Figure 22: Feature Importance Visualization for SAC Agent on Daily Data with Sentiment: A relatively uniform gradient distribution with notable peaks for the number of stocks, random variables, and the ‘dx 30‘ technical indicator.

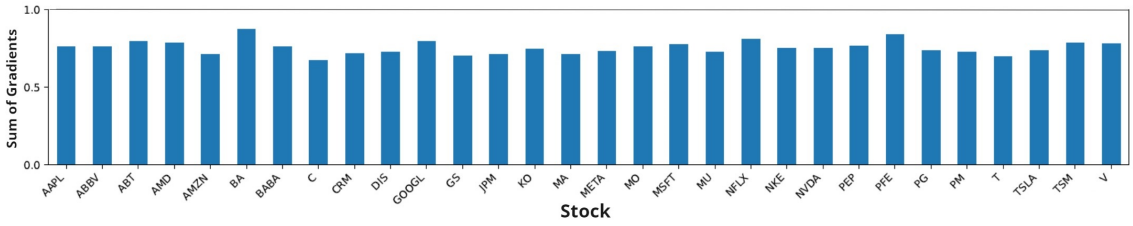

Figure 23: Stock Importance Visualization for SAC Agent on Daily Data without Sentiment: Consistent Gradient Distribution Across Different Stocks.

7.5.3 Assessing the Impact of Individual Stocks

The Figure 23 offers a perspective on the stock-specific importance as perceived by the SAC agent when trained on daily data without sentiment. Interestingly, the gradient sums for individual stocks appear to be uniformly distributed. This suggests that the agent does not exhibit a strong preference or bias towards any particular stock, treating them with relatively equal importance. While this uniformity ensures a balanced consideration of stocks, it also raises questions about the agent’s ability to discern nuanced patterns or trends specific to individual stocks.

8 Discussion and Future Work

In summary the reinforcement learning approach for stock trading was very successful. Even the proven method of buy and hold and portfolio theory could be outperformed while remaining in the same risk thresholds regarding the calmar ratio and sharp ratio. Additionally, the developed pipeline from data collection, pre-processing and feature engineering to model training and holistic backtesting can be applied on all various alternative data sets. As the state of our model was already very large, the other data sets had to be omitted. But this approach – as it looks very promising – should be replicated with fundamental data, senate trading or other alternative data. Due to the explainability of sensitivity analysis the risk exposure can be reduced additionally since obvious randomly acting agents can be rejected. Nonetheless, our approach was expected to give the opportunity to reduce the dimensions of our state. However, since the sensitivity analysis could not exclude single features with significant confidence, the models could not be re-trained and benchmarked with a reduced state. Similarly our efforts in terms of model parameter optimisation resulted in stagnating improvements. The most likely reason is the low signal-to-noise ratio. To overcome this obstacle, one should focus even stronger on data cleaning and feature engineering. While the Kalmar filter showed significant improvements, one could follow completely different data representation: instead of time bars, one could adapt volume or tick bars and other novel approaches to produce stable signals from noisy data. However, these strategies are all outside of reinforcement learning and therefore not relevant for this thesis.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Samit Ahlawat. *Reinforcement Learning for Finance: Solve Problems in Finance with CNN and RNN Using the TensorFlow Library*. Apress, Berkeley, CA, 1 edition, 2022. 1 b/w illustrations, 84 illustrations in colour.

[3] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks, 2017.

[4] Andre Biedenkapp, Marius Lindauer, Katharina Eggensperger, Frank Hutter, Chris Fawcett, and Holger Hoos. Efficient parameter importance analysis via ablation with surrogates. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017.

[5] John Y Campbell, Sanford J Grossman, and Jiang Wang. Trading volume and serial correlation in stock returns. *The Quarterly Journal of Economics*, 108(4):905–939, 1993.

[6] Lin Chen and Qiang Gao. Application of deep reinforcement learning on automated stock trading. In *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, pages 29–33, 2019.

[7] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley Publishing, 1st edition, 2018.

[8] Chris Fawcett and Holger Hoos. Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22, 01 2015.

[9] FMP. Financial modeling prep api documentation, 2023. Accessed on September 2023.

[10] FMP. Social sentiment api, 2023. Accessed on September 2023.

[11] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018.

[12] A Ronald Gallant, Peter E Rossi, and George Tauchen. Stock prices and volume. *The Review of Financial Studies*, 5(2):199–242, 1992.

[13] Mauricio Garita. *Applied Quantitative Finance*. Springer International Publishing, 2021.

[14] Julia Groth. Das bedeutet musks kaufplan für twitter-aktionäre, 2022. Accessed on September 2023.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[16] Peter R Hansen and Asger Lunde. Realized variance and market microstructure noise. *Journal of Business & Economic Statistics*, 24(2):127–161, 2006.

[17] John C. Hull. *Options, Futures, and other Derivatives*. Pearson, Boston, 11th edition, 2021.

[18] Jealous. Stockstats package. Accessed on 17.09.2023.

[19] Mark S. Joshi. *The Concepts and Practice of Mathematical Finance*. Mathematics, Finance and Risk. Cambridge University Press, 2nd edition, 2008.

[20] Mark Kritzman and Yuanzhen Li. Skulls, financial turbulence, and risk management. *Financial Analysts Journal*, 66(5):30–41, 2010.

[21] Roger R. Jr Labbe. Filterpy – kalman and bayesian filters in python, 2020. Accessed on 17.09.2023.

[22] Yawei Li, Peipei Liu, and Ze Wang. Stock trading strategies based on deep reinforcement learning. *Scientific Programming*, 2022:1–15, March 2022.

[23] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[24] Xiao-Yang Liu, Ziyi Xia, Jingyang Rui, Jiechao Gao, Hongyang Yang, Ming Zhu, Christina Wang, Zhaoran Wang, and Jian Guo. Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1835–1849, 2022.

[25] Xiao-Yang Liu, Zhuoran Xiong, Shan Zhong, Hongyang Yang, and Anwar Walid. Practical deep reinforcement learning approach for stock trading, 2022.

[26] Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, and Christina Dan Wang. Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance. *arXiv preprint arXiv:2011.09607*, 2020.

[27] Owen Lockwood and Mei Si. A review of uncertainty for deep reinforcement learning, 2022.

[28] Cory Mitchell. Average directional index (adx): Definition and formula. Accessed on 17.09.2023.

[29] Cory Mitchell. What is the commodity channel index (cci)? how to calculate. Accessed on 17.09.2023.

[30] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016.

[31] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[32] Dayakar Naik and Ravi Kiran. A novel sensitivity-based method for feature selection. *Journal of Big Data*, 8, 10 2021.

[33] Ian E. Nielsen, Dimah Dera, Ghulam Rasool, Nidhal Bouaynaya, and Ravi Prakash Ramachandran. Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks. *CoRR*, abs/2107.11400, 2021.

[34] Quantopian. pyfolio package. Accessed on 17.09.2023.

[35] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[36] Stuart Jonathan Russell and Peter Norvig. *Artificial Intelligence – A Modern Approach*. Pearson, 4th edition, 2021.

[37] Andrea Saltelli, Stefano Tarantola, and K. Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41, 03 2012.

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[39] Jingyi Shen and M. Omair Shafiq. Short-term stock market price trend prediction using a comprehensive deep learning system. *Journal of Big Data*, 7(1):66, Aug 2020.

[40] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *CoRR*, abs/1605.01713, 2016.

[41] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

[42] I.M Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55:271–280, 02 2001.

[43] Meir Statman. Lottery players/stock traders. *Financial Analysts Journal*, 58(1):14–21, 2002.

[44] Bas Van Stein, Elena Raponi, Zahra Sadeghi, Niek Bouman, Roeland C. H. J. Van Ham, and Thomas Bäck. A comparison of global sensitivity analysis methods for explainable ai with an application in genomic prediction. *IEEE Access*, 10:103364–103381, 2022.

[45] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th*

International Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 06–11 Aug 2017.

[46] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[47] Stefano Tarantola, Debora Gatelli, and Thierry Mara. Random balance designs for the estimation of first order global sensitivity indices. *Reliability Engineering System Safety*, 91:717–727, 06 2006.

[48] Alpha Vantage. Alpha vantage - free apis for realtime & historical stock, forex (fx), cryptocurrency data, technical analysis, charting and more!, 2023.

[49] Rahul Verma and Priti Verma. Noise trading and stock market volatility. *Journal of Multinational Financial Management*, 17(3):231–243, 2007.

[50] Greg Welch and Gary Bishop. Welch & bishop , an introduction to the kalman filter. 1994.

[51] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. Sep 2020. Available at SSRN: https://ssrn.com/abstract=3690996 or http://dx.doi.org/10.2139/ssrn.3690996.

[52] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.

[53] Hua Zhang, Yuanzhu Chen, Wei Rong, Jun Wang, and Jinghua Tan. Effect of social media rumors on stock market volatility: A case of data mining in china. *Frontiers in Physics*, 10, 2022.

[54] Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. *CoRR*, abs/1702.04595, 2017.

A Pairwise Comparision - Daily

B Pairwise Comparision - Hourly

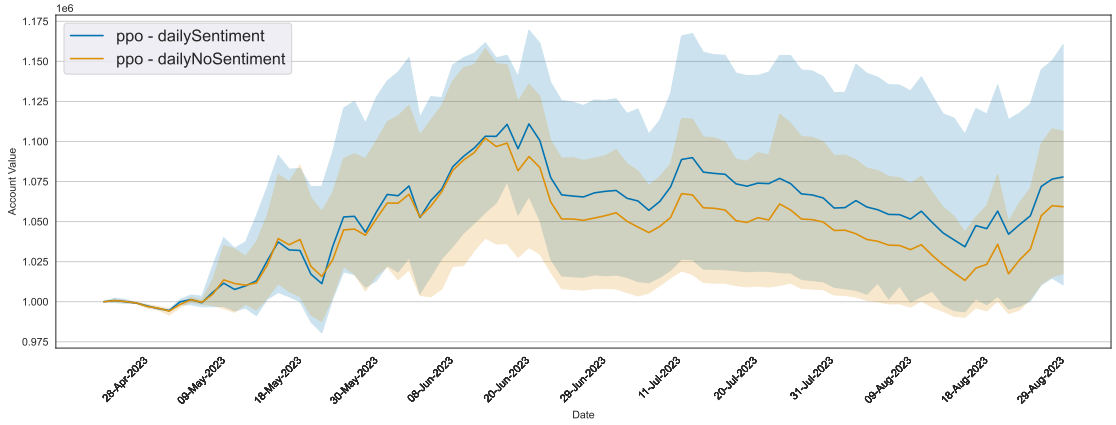

Figure 24: Performance comparison of PPO agent trained on daily data with and without sentiment integration.

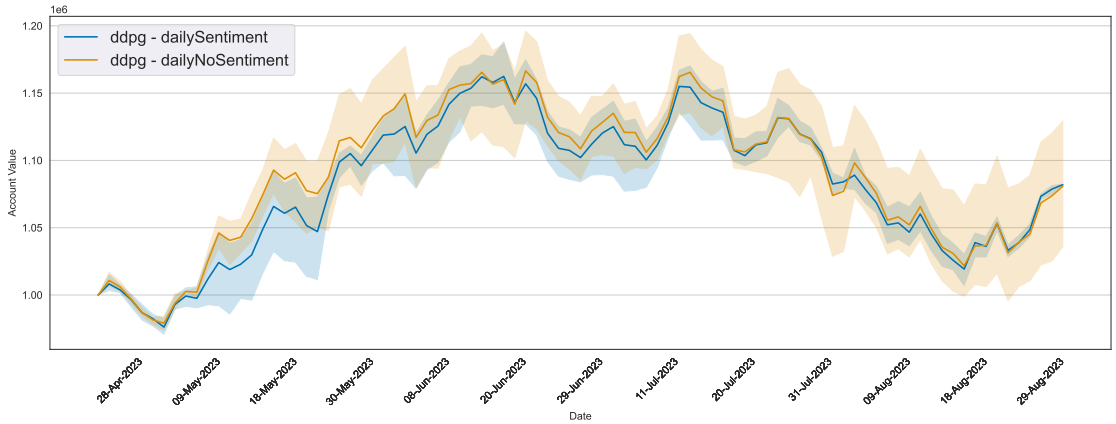

Figure 25: Performance comparison of DDPG agent trained on daily data with and without sentiment integration.

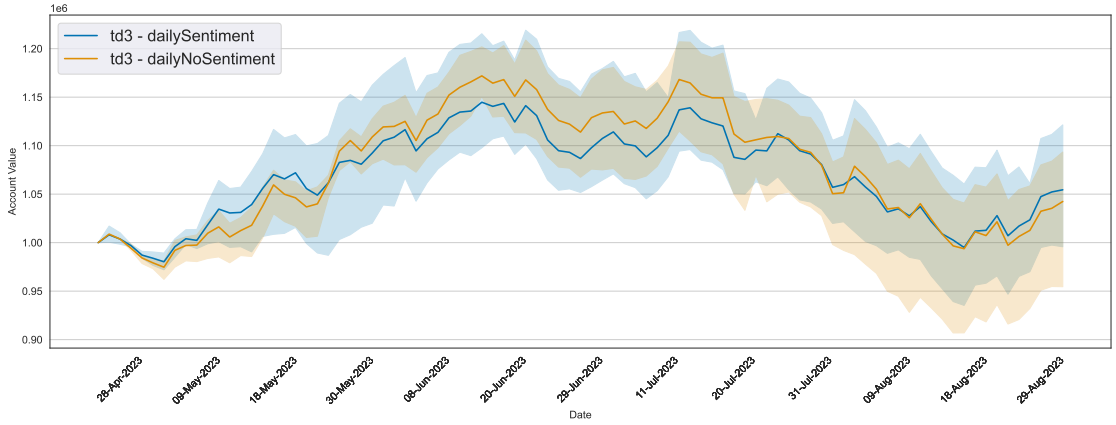

Figure 26: Performance comparison of TD3 agent trained on daily data with and without sentiment integration.

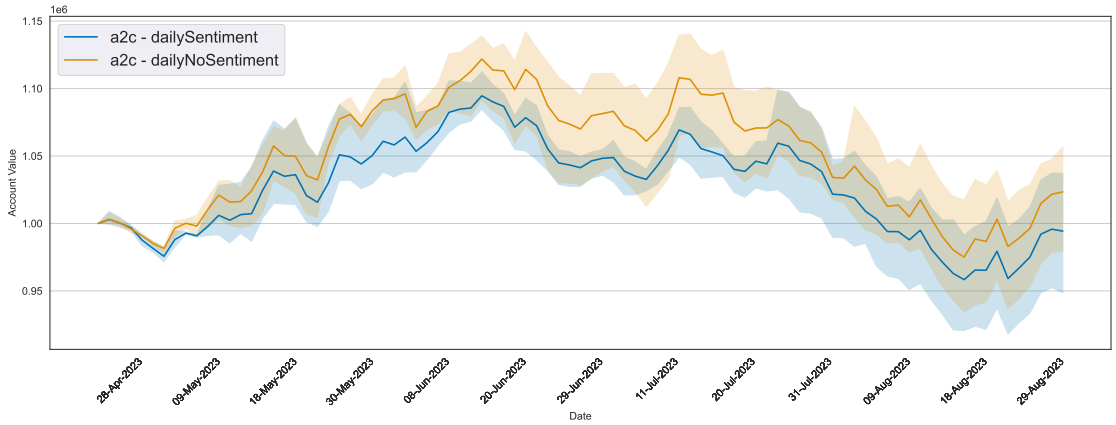

Figure 27: Performance comparison of A2C agent trained on daily data with and without sentiment integration.

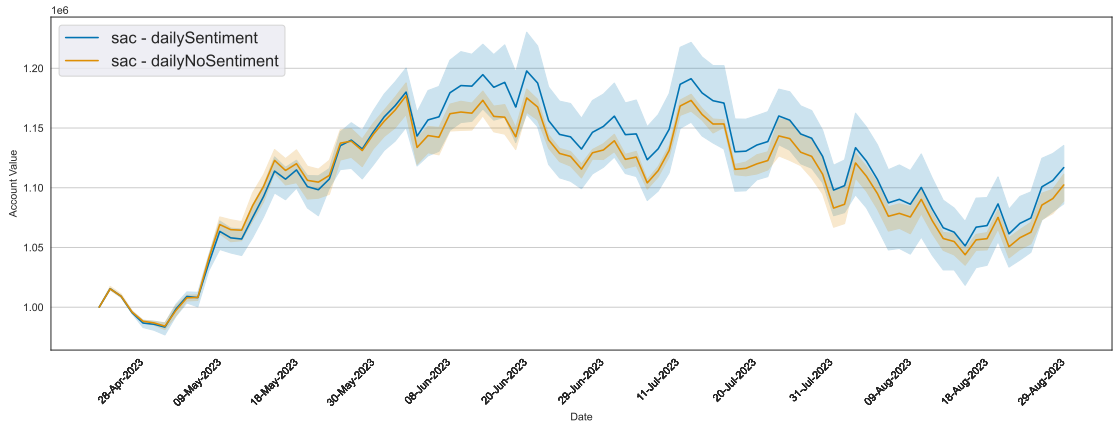

Figure 28: Performance comparison of SAC agent trained on daily data with and without sentiment integration.

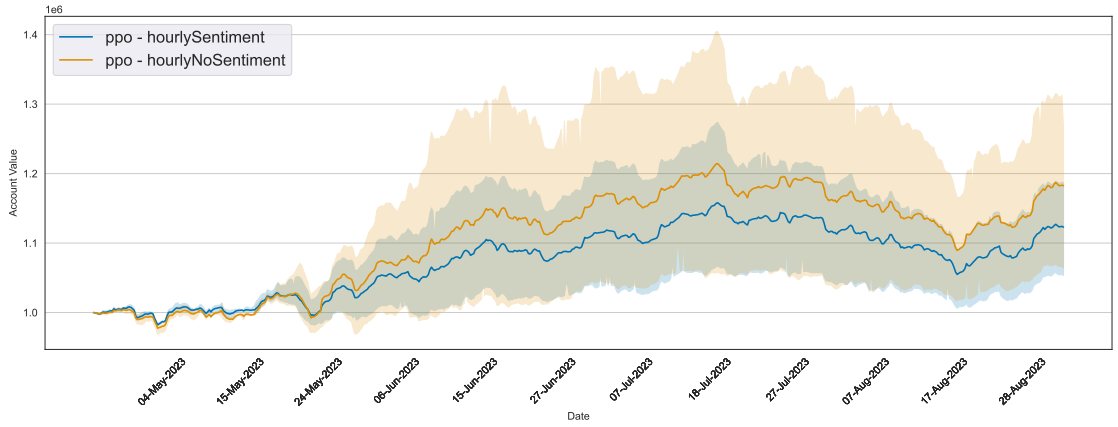

Figure 29: Performance comparison of PPO agent trained on hourly data with and without sentiment integration.

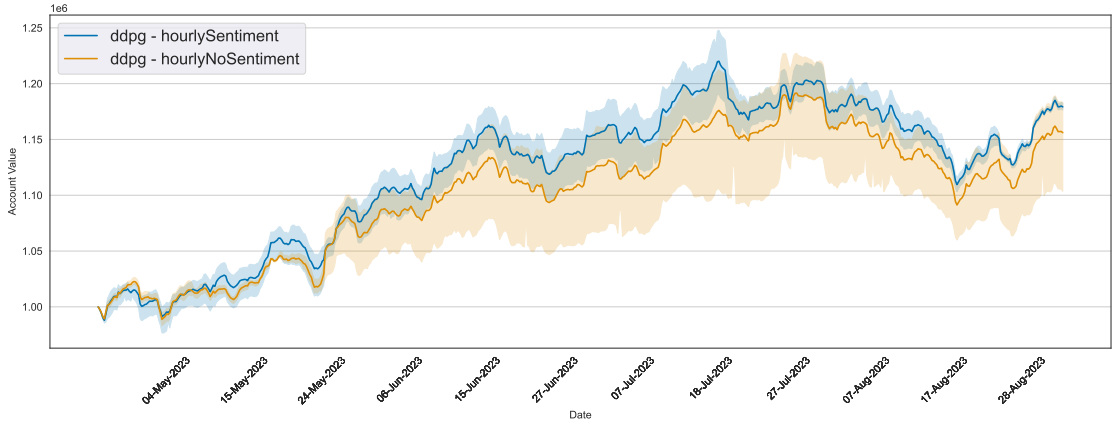

Figure 30: Performance comparison of DDPG agent trained on hourly data with and without sentiment integration.

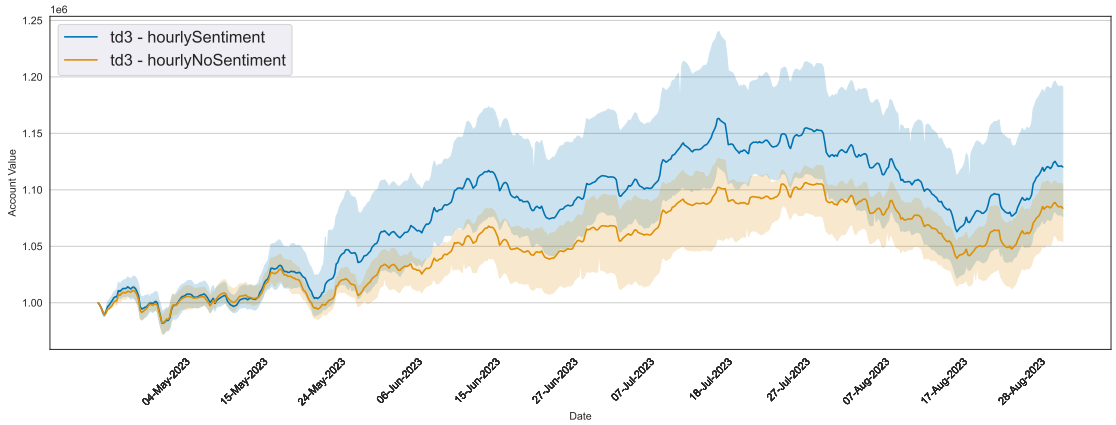

Figure 31: Performance comparison of TD3 agent trained on hourly data with and without sentiment integration.

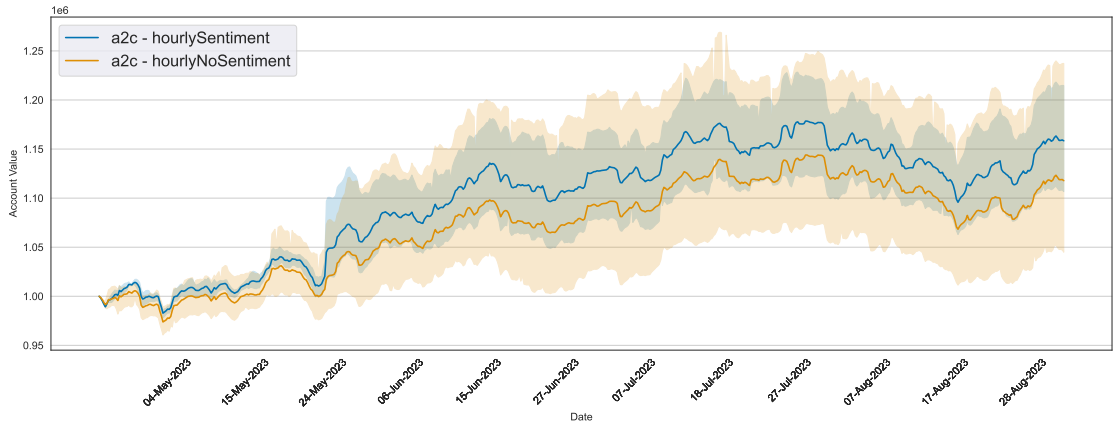

Figure 32: Performance comparison of A2C agent trained on hourly data with and without sentiment integration.

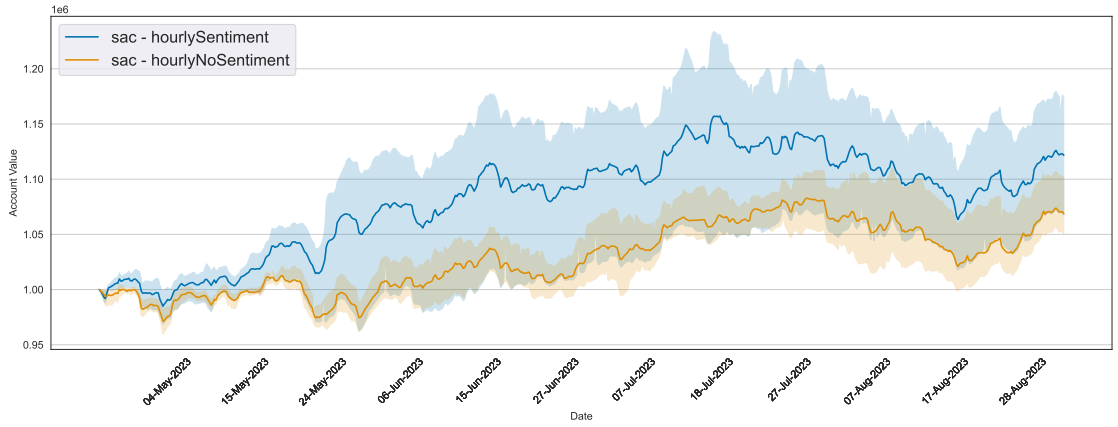

Figure 33: Performance comparison of SAC agent trained on hourly data with and without sentiment integration.