Executive Summary of the Thesis

Reinforcement Learning-based Trading Model for US Sectors

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Giulia Mulattieri

Advisor: Prof. Marcello Restelli

Co-advisor: Antonio Riva, Luca Sabbioni

Academic year: 2021-2022

1. Introduction

In recent years there has been growing interest in applying artificial intelligence (AI) techniques to assist investors in navigating financial markets and perform effective investment strategies. As highlighted in [3], AI has the potential to revolutionize trading and portfolio management by enabling investors to analyze huge amounts of data. Therefore the integration of AI represents a promising opportunity to achieve optimal investment objectives.
In this thesis, our focus is on applying AI techniques to find the best investment strategy for US Sector Indices. Therefore we aim to build a *trading model* to properly identify buy and sell signals for US Sectors. The mission of this project, that goes far beyond the contributions of this thesis, is to construct a *market-neutral* portfolio accordingly to these signals. A portfolio is considered *market-neutral* if the size of the long positions equals the size of the short positions, resulting in a null net exposure against the market.
First, we consider the main financial indicators that drive the market. We consider both technical and fundamental indicators, and through feature selection algorithms, we find the most relevant ones. Then we try to handle the non-stationarity of the data in order to increase the predictive power. Finally we decide to employ the Fitted Q-Iteration (FQI, [6]) as reinforcement learning algorithm. To train our model, we used daily data from 2008 to 2019. We evaluated the model's performance through a backtest between 2020 and 2022. We achieve positive performance for most of the sectors for some iterations of FQI. However, some of the obtained strategies exhibit excessive volatility. One way to overcome these instability issues is to implement effective volatility control techniques which could make our trading strategy a promising approach for producing positive and stable results.

2. Reinforcement Learning

Reinforcement learning [9] is a subfield of Artificial Intelligence that aims to develop and train an *agent* to interact with a given system, called *environment*, in order to maximize a reward signal. The agent-environment interaction is meant as a sequence of decisions, or actions, each of which is taken by the agent after observing the environment state.

2.1. Markov Decision Process

The reinforcement learning theory is built on the framework of Markov Decision Processes which are stochastic mathematical systems able to model the interaction between an agent and an environment. A Markov Decision Process (MDP) is a tuple $< \mathcal{S}, \mathcal{A}, P, R, \gamma, \mu >$, where: $\mathcal{S}$ is the set of all possible states called *state space*, $\mathcal{A}$ is the set of all possible actions called *action space*, P is the stationary transition probability matrix defining P(s'|s,a), R is the reward function, γ is the discount factor such that $\gamma \in [0, 1]$, μ is the distribution of the initial state of the environment. In RL, the agent selects its action based on a policy, $\pi(\cdot|s)$, which assigns a distribution over the action space $\mathcal{A}$ to each state s. Almost all RL algorithms are based on estimating the action-value function Q_π, which describes the expected future reward for taking a particular action a in a given state s following a certain policy π. It can be expressed as:

$$Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t R_t | S_t = s, A_t = a]. \quad (1)$$

The goal of the agent is to find the optimal action-value function, defined as:

$$Q^*(s, a) = \max_\pi Q_\pi(s, a), \quad (2)$$

which allows to determine the optimal policy.

2.2. Fitted Q-Iteration

Fitted Q-Iteration (FQI, [6]) is a RL algorithm that uses an approximate model of the system dynamics in order to approximate the optimal action-value function for each action in a given state. The model is iteratively trained on a dataset in the form (S, A, S', R) with the aim of obtaining the optimal policy. Then a regression model, such as Extra Trees or XGBoost, is employed to estimate the Q-values for each state-action pair. This iterative process updates the model based on a batch of experiences collected from the environment, allowing it to learn from the past and improve its predictions of future rewards in order to obtain a new policy. The final result is an approximation of the optimal policy that maximizes the long-term reward.

3. Related Works

Reinforcement Learning has been shown to be effective in dealing with dynamic and uncertain environments, making it well-suited for financial markets. A commonly employed approach is by means of deep RL techniques, which use neural networks to approximate the agent's decision-making process. [10] adopts deep reinforcement learning to generate buy and sell signals for various financial instruments including stocks, currencies and cryptocurrencies. The proposed method is tested on real-world financial daily data and achieved superior performance compared to other state-of-the-art models in learning trading rules specific to individual assets. However there is no analysis of which financial indicators should be used to better train the agent and improve the models performance.
An alternative approach that does not rely on neural networks is the FQI algorithm, which can be used for single-asset trading with action discretization. Rather than using neural networks, FQI typically adopts regressors such as Extra Trees or XGBoost. This approach enables a more transparent decision-making process and mitigates the instability issues frequently encountered in continuous framework models. [2] and [8] develop an effective high frequency FX trading strategy by training an agent via Fitted Q-Iteration. They take into account both the non stationarity of the data and the transaction costs. However they do not incorporate any financial indicators as features.
In this thesis, we propose an approach that manages daily signals to perform weekly trading without any simplified hypothesis. To achieve this goal, we incorporate effective techniques for managing data non-stationarity and selected financial indicators as input data in order to better handle the dynamics and complexity of the financial market. We adopt the FQI algorithm with action discretization as Reinforcement Learning model.

4. Problem Formulation

In order to effectively apply reinforcement learning algorithms to find the best investment strategy, the first step is to model the trading dynamic as a Markov decision process and define the concepts of state, action, and reward.
Our goal is to create an agent that utilizes RL

techniques to generate trading signals. Therefore, we need to introduce fundamental concepts related to price time series and define the underlying dynamics.

4.1. Financial Preliminaries

The *closing price* of an asset is the price of the last transaction made on that security during a trading session.
The *opening price* of an asset is the price of the first transaction made on that security during a trading session.
The performance of an asset is evaluated by calculating its *return*, which can be determined in various ways depending on the time horizon under consideration. For instance, if we consider daily returns, they are computed as:

$$r_t = \frac{ClosePrice_t}{ClosePrice_{t-1}} - 1 \tag{3}$$

Investors can buy, sell or hold financial assets, depending on their investment objectives and market conditions.
'Buy' means purchasing a financial asset, hoping that its value will increase in the future.
'Sell' involves disposing of a financial asset and selling it, either to realize a profit or to cut losses.
'Hold' refers to maintaining the current position in a financial asset.

4.2. Reward

Our objective is to identify the optimal investment strategy for every sector. Therefore we have to properly define a reward function in order to reach this goal.
Let us consider a set Ω of financial assets. A *portfolio* is a combination of such assets, in a way that each asset $i \in \Omega$ at a specific timestep t is associated to a weight denoted as $\omega_{i,t} \in [0,1]$ and $\sum_i \omega_{i,t} = 1$.
The value of such *portfolio* can be expressed as:

$$\begin{aligned} V_t = \sum_{i \in \Omega} \omega_{i,t-1}\left(\frac{OpenPrice_{i,t}}{ClosePrice_{i,t-1}} - 1\right) \\ + \omega_{i,t}\left(\frac{ClosePrice_{i,t}}{OpenPrice_{i,t}} - 1\right), \end{aligned} \tag{4}$$

where *OpenPrice* and *ClosePrice* are respectively the opening price and the closing price of the asset i.
Assuming that the closing price at time t-1 is equal to the opening price at time t, we obtain:

$$V_t = \sum_{i \in \Omega} \omega_{i,t}\left(\frac{ClosePrice_{i,t}}{ClosePrice_{i,t-1}} - 1\right), \tag{5}$$

which can be rewritten as:

$$V_t = \sum_{i \in \Omega} \omega_{i,t} r_{i,t}, \tag{6}$$

where $r_{i,t}$ is the return of the asset i at time t, introduced in Equation 3.
In real markets, when an investor buys or sells an asset, a transaction cost must be paid. Therefore, the value of a portfolio including transaction cost at time t can be defined as:

$$V_t = \sum_{i \in \Omega} \omega_{i,t} r_{i,t} - fee \cdot |\omega_{i,t} - \omega_{i,t-1}|, \tag{7}$$

where fee is the transaction cost associated with each operation.
Given this result, we can define the reward function of the asset i as the difference between the return and the transaction cost associated with the trading operation:

$$R_{i,t} = \omega_{i,t} r_{i,t} - fee \cdot |\omega_{i,t} - \omega_{i,t-1}|. \tag{8}$$

In our case, we decide to work with discrete actions instead of continuous weights ω, therefore the final reward function of asset i at time t becomes:

$$R_{i,t} = a_{i,t} r_{i,t} - fee \cdot |a_{i,t} - a_{i,t-1}|, \tag{9}$$

where $a_{i,t} \in \mathcal{A}$ is the action performed by the agent as described in (10).
Our approach involves using a reward function that evaluates the performance of each asset individually, rather than treating the entire portfolio as a single entity. Specifically, we construct a single asset optimal trading strategy for each sector in which the goal of the agent is to maximize the reward expressed in Equation 9 over time.

4.3. Environment Formulation

In order to construct an environment suitable for a trading framework, we need to properly define the reward function (introduced in Section 4.2), the action space and the state space.

Action Space The action consists of the allocation the agent chooses for the week, therefore the set of possible actions is defined as:

$$\mathcal{A} = \{-1, 0, 1\}, \tag{10}$$

which corresponds respectively to the three different possible actions described in Section 4.1: Sell, Hold and Buy.

Feature Space The state in an MDP contains all information needed to select the best action to maximize future rewards. While the current state of an asset is trivially represented by the current price, we can include a series of financial indicators that may provide useful information to better understand the underlying dynamics of the dynamic process. The most significant financial indicators we select are Fundamental Indicators (Price to Book (PB), Price over Earnings (PE), Dividend Yield, Earnings Growth, Earnings Yield) and Technical Indicators (Volatility, Bollinger Bands, Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Signal Line, Volume, Volume Oscillator). However the large number of financial indicators can lead to computational and sample complexities issues, as well as the possibility of having correlated features. Therefore a feature selection process will be conducted in the next chapter (5) to identify the most relevant and informative features. Starting from the results of this analysis, we define the state of our environment composed of the technical indicators and the returns (daily, weekly, and weekly lagged). Specifically daily returns are the ones introduced in Equation 3. Similarly the weekly returns are computed considering a window of 5 days. The lagged returns we consider are the weekly returns of 2,3, and 4 previous weeks. Additionally we consider the allocation at the previous step. This last feature is crucial because when the agent selects an action, he needs to take into account the previous position because of transaction costs. In particular, if the price changes by an amount smaller than the transaction fee, the cost of the operation will be higher than the return, resulting in a negative reward. Adding the position of the previous step enables the agent to choose the 'hold' action if he predicts that the costs of a potential transaction exceed the returns.

4.4. Algorithm Selection

In order to properly choose the more suitable RL algorithm we opt for a value-based approach over a policy-based approach. While the latter is more sample efficient, it tends to be less stable and more sensitive to the quality of the function approximator and the choice of hyperparameters, which can lead to suboptimal solutions.
We choose FQI algorithm because it is more robust than other value-based methods, such as DQN algorithm which is more sensitive to noise or inaccuracies in the data. Additionally, FQI is more interpretable since it uses XGBoost or Extra Trees instead of neural networks, which are a black box approach.
Therefore, we train the FQI algorithm considering daily features as input to perform a weekly trading strategy for each sector individually. To evaluate our model, we simulate the optimal trades according to the predictions of the model and compute the cumulative returns over the testing period.

5. Data Analysis

We focus on analyzing the sectors using the Global Industry Classification Standard (GICS) classification. The sectors we consider are: Communication Services, Consumer Discretionary, Consumer Staples, Energy, Financial, Health Care, Industrial, Information Technology, Materials, Real Estate, and Utilities.
Instead of considering sectors prices 'alone', we decide to use sector performance data relative to the market, which means that we construct an historical time series of sector minus market. In order to do this we compute the performance of each sector, and the performance of the market and take the difference. We then recalculate the prices. By doing this, we are able to identify specific sector risks: by removing the market component, we eliminate the market risk, leading to a market neutral time series. We will refer to this new time series as *sectors vs market*.
Then we choose the most significant financial indicators, both fundamental and technical, as explained in Section 4.3. Like for sector prices, we consider these indicators relative to the market, constructing for each of them a new historical time series of *Indicators vs Market*.

5.1. Feature Selection

In this section, we perform a feature selection procedure to detect the most informative indicators, that is, which indicators have an effective predictive power on asset returns.
In order to do that we choose ExtraTrees Regressor [7] and XGBoost [4], which are supervised learning algorithms that have been shown to be effective in producing accurate predictions for a wide range of regression problems.

Extra Trees Regressor Extra Trees Regressor is a supervised learning method that provides an estimator that fits multiple extra-trees on various sub-samples of the dataset, followed by averaging the predictions to improve predictive accuracy and prevent over-fitting. The fact that the cut points are chosen randomly for each tree makes the trees diversified and uncorrelated.

XGBoost Extreme Gradient Boosting (XGBoost) is a decision-tree-based algorithm that combines multiple decision trees. It works in a gradient boosting framework, which is a technique that uses gradient descent to optimize the loss function of the model, providing a parallel tree boosting. It starts by fitting an initial predictor model (e.g. a tree) to the data. Then the algorithm works by iteratively adding predictors to the model while optimizing a given objective function. Each successive model attempts to correct for the shortcomings of the combined boosted ensemble of all previous models.

In order to assess the accuracy of the forecast, we employ the adjust R^2 statistic and the *Accuracy*. The adjusted R^2 statistic is a modified version of the R^2 that takes into account the number of independent variables in the model and better manage overfitting. To compute the accuracy, since the return values are continuous, we have considered their sign. By doing this, we have a measure of how well the model can predict gains (positive returns) and losses (negative returns) correctly.

Year	p-value	Stationarity
2019	8.747e-7	stationary
2020	2.594e-4	stationary
2021	1.044e-6	stationary
2022	1.743e-5	stationary

Table 1: ADF Test - Bollinger Up

5.2. Stationarity

We attempt to make the data more stationary while preserving their predictive power. To achieve this, we employ an approach called *Fractional Differentiation*, which was first introduced by Lopez de Prado in [5].
The most common method used to remove non-stationarity from data is to make the some integer order difference or the logarithm. However, these approaches erase a significant portion of the data memory, dramatically reducing their predictive power. Fractional differentiation overcomes this problem by finding the fraction d such that the data become sufficiently stationary while preserving as much information as possible. In particular, each past value of the time series is assigned a weight ω such that:

$$\omega = \{1, -d, \frac{d(d-1)}{2!}, ..., (-1)^k \prod_{i=0}^{k-1} \frac{d-i}{k!}\} \quad (11)$$

When d is equal to k, the memory beyond that point is removed. The goal is to find d such that stationarity is achieved and the maximum volume of memory of the time series is preserved.
We obtained good results in terms of stationarity by setting $d = 0.2$. In Table 1 we present the result of The Augmented Dickey-Fuller (ADF) test on the historical data from 2019 to 2022 of the technical indicator *Bollinger Up*. Similar results were obtained for the other features and for the other years. In particular we perform an ADF test on each feature in the dataset after applying the *Fractional Differentiation* method for some value of d. The ADF test is a type of unit root test that examines how strongly a time series is defined by a trend. To interpret the results of this test, we use the p-value. If the p-value is below a defined threshold α (set to 0.05), we conclude that the time series is stationary. Conversely, if the p-value is above the threshold, we

R^2 Train	0.68
R^2 Test	-0.03
Accuracy Train	0.87
Accuracy Test	0.56

Table 2: R^2 and *Accuracy* - Rolling Approach 2018-2021

conclude that the time series is non-stationary. Therefore, analyzing the p-value, we find that with d = 0.2 the data became stationary.

5.3. Rolling Approach

We decide to employ a rolling approach in order to avoid a net division between train and test set: considering a rolling window allows us to properly evaluate the predictive power of the features over different periods, leading to a more stable analysis. In order to find the best model parameters for each train-test period we decide to use Optuna [1], which is a commonly employed open-source hyperparameter optimization framework for machine learning.
We consider as training set 3 years of daily data and validation set the following year. In order to find the best parameters we set the Optuna objective function as the adjusted R^2 computed on the weekly returns of the validation set. After identifying the optimal parameters, we train the model on 3 years of daily data to predict weekly returns for the following month. We then shift the dataset by one month and repeat this process. At the end of each year, i.e. after 12 shifts, we collect and evaluate the predictions.
We present the results for sector Consumer Discretionary, considering Extra Trees Regressor. In Table 2 we can see the results in terms of adjusted R^2 and *accuracy* with training set from 2018 to 2020 and test set 2021. Unfortunately, the results at the adjusted R^2 level are negative, despite achieving a reasonable level of *accuracy*. Through various years of train-test, we have always obtained negative R^2 values. However, our accuracy score has yielded some good results (>55%).
One of the reasons we identified for this behavior is that the adjusted R^2 may be too sensitive as a measure to evaluate prediction in such a volatile context, and already achieving discrete *accuracy* results may indicate that some features are more significant than others.
Therefore, we have considered the feature importance results obtained from Extra Tree Regressor, which revealed that Fundamental Indicators are not relevant. The importance of a feature derived from Extra Trees is computed using the Mean Decrease Impurity (MDI) method considering as impurity metric the Gini impurity. As a consequence of feature importance results, the set of features selected to perform further analysis includes only technical indicators and returns.

6. Experimental Results

We trained the FQI model during the period from 2008 to 2015, using the 2016-2019 data as a validation set to obtain optimal parameters for the FQI regressor through Optuna Optimizator. We experiment XGBoost and Extra Trees as regressors, with XGBoost being selected due to its superior performance in computational efficiency and accuracy. To ensure the robustness of the optimal parameters, the Optuna objective function is set to maximize the average cumulative reward during the validation period across three different independent runs. Finally, the validation set is included again in the training period and the model is re-trained using the previously obtained parameters on daily data from 2008 to 2019, with performance evaluation conducted on weekly out-of-sample data from 2020 to 2022 with 5 FQI iterations. We assume transaction costs of 0.0005 for each operation. Therefore, in our problem, we have to consider for each transaction a cost of 0.001 as our historical series are *sector vs market*, which requires two transactions each time a signal is generated. In fact when we have, for instance, a buy signal, it results in double transaction costs since we have to buy the sector index and sell the market.
To test the stability and robustness of the model, we perform backtesting simulations for the same sector using various seeds to observe the strategy volatility. We present the result for the sector Consumer Discretionary with 5-day window and XGboost as regressor. Similar results were obtained for other sectors. We obtain positive performance in backtesting for certain iterations of FQI, such as the 3th iteration which yielded a

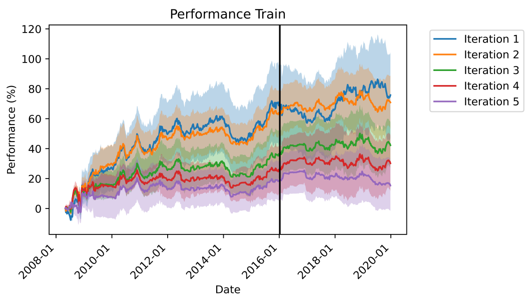

Figure 1: Performance Train - Consumer Discretionary

mean return of +7.8% across various seeds (Figure 2). However, the results from validation set (Figure 3) indicates that the iteration to choose in the test is the 1th, since it is the iteration that leads to the better performance (+10.6%). Therefore if we consider the first iteration of FQI for the test we get -5.3% (Figure 4). One reason we identified for this negative results it that the choice of the iteration to consider is biased due to our use of Optuna which optimizes the parameters only in the first iteration of FQI. Specifically we train the model and we use Optuna to find the parameters that maximize the cumulative return on the validation set, but only for the first iteration of FQI. This is because the Optuna research is computationally really expensive therefore the optimization procedure is limited to the first iteration. However selecting the best parameters for each iteration during the validation procedure would lead to a more stable and correct choice of what FQI iteration to consider, instead of relying only on the optimal parameters of the initial iteration.
However, concerns regarding the volatility and instability of the resulting strategies persist, especially when we compare the results obtained with different independent runs (Figures 1, 3 and 2). In conclusion, while the tested FQI iterations have shown promising performance for some US sectors, further investigation is needed to address the identified issues and ensure the reliability of the proposed trading model.

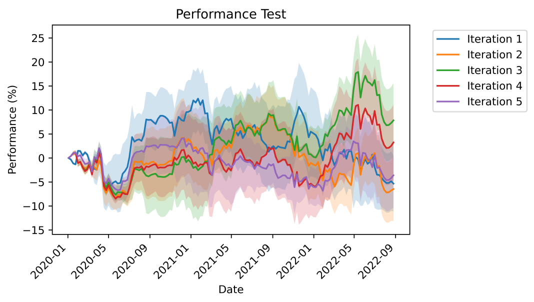

Figure 2: Performance Test - Consumer Discretionary

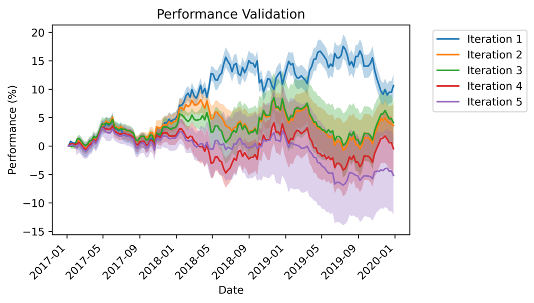

Figure 3: Performance Validation - Consumer Discretionary

7. Conclusions and future developments

The aim of this work is to use reinforcement learning to construct a trading model for US market sector indices. We begin by selecting and analyzing a set of financial indicators in order to identify the most relevant ones in terms of predictive power on asset returns. We employ the *Fractional Differentiation* approach to remove non-stationarity in the data while preserving their predictive power. In order to evaluate the predictive power of our features we decide to employ a rolling approach with Extra Trees Regressor. Although the adjusted R^2 statistic did not indicate a strong predictive power among the features, we obtained a discrete level of accuracy. Therefore we selected the most relevant features for further analysis according to the Extra Trees feature importance.
Finally we trained the FQI algorithm on the selected features. Although the robustness of the algorithm, we notice that the resulting strategies exhibit high volatility and instability issues. Therefore there are several future developments that could enhance this work, including:

- investigating additional financial indicators

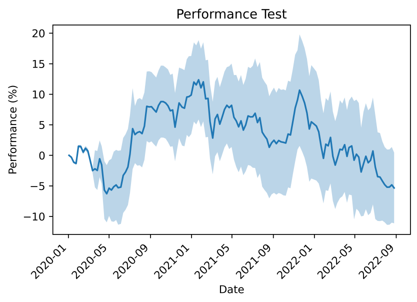

Figure 4: Performance Test, First Iteration - Consumer Discretionary

in order to find other significant features which can improve the predictive power on asset returns;

- exploring alternative approaches for managing the non-stationarity of data;
- exploring alternative approaches to perform feature selection, e.g. discretize the features using quantiles and employ a classifier rather than a regressor;
- developing effective techniques to manage the volatility, e.g. set a volatility target and normalize the historical time series of the sectors and the market by this target.

In conclusion, we developed an effective trading model with the final goal of constructing a market-neutral portfolio based on the obtained strategies. One way to do this can be to consider the Q-value Function of the FQI agent's buy action for each sector to derive the optimal weights since assets with lower Q-values should have lower weights. The Q-value, in fact, represents the expected return when buying the asset, enabling us to assign lower weights to sectors with lower expected returns. Additionally, we must perform volatility controls to penalize assets with high expected returns that also have higher volatility.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[2] Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Gianmarco Reho, Nico Montali, Marcello Restelli, and Cristiana Corno. Foreign exchange trading: A risk-averse batch reinforcement learning approach. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

[3] Vittorio Carlini. L'intelligenza artificiale aiuta gli investimenti, ma serve più conoscenza. `https://www.quotidiano.ilsole24ore.com/sfoglio/aviator.php?newspaper=S24&edition=SOLE&issue=20230311&startpage=2&displaypages=2&articleId=1867716`, 2023.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[5] Marcos Lopez de Prado. Advances in financial machine learning, 2018.

[6] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

[7] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

[8] Antonio Riva, Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Edoardo Vittori, Marco Pinciroli, Michele Trapletti, and Marcello Restelli. Learning fx trading strategies with fqi and persistent actions. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.

[9] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. The MIT Press, 2018.

[10] Mehran Taghian, Ahmad Asadi, and Reza Safabakhsh. Learning financial asset-specific trading rules via deep reinforcement learning. *Expert Systems with Applications*, 195:116523, 2022.