Facultad de Ciencias Económicas y Empresariales Grado en Administración y Dirección de Empresas y Business Analytics

Deep Reinforcement Learning for Optimizing Fixed-Income Portfolios

Autor: Jorge González de San Román Sánchez

Director: Ignacio Cervera Conte

MADRID | Junio 2025

Acknowledgements

I would like to express my deepest gratitude to my family for their unwavering support throughout every stage of this journey, as well as to the faculty at Comillas for their valuable academic guidance and personal support along the way, particularly to Eduardo Garrido Merchán for his assistance in developing the code and for generously sharing his expertise.

INDEX

List of Figures

List of Equations

Abstract

This study explores the application of Deep Reinforcement Learning (DRL) to optimize the management of fixed-income portfolios, with a particular focus on U.S. Treasury bonds. Using the Proximal Policy Optimization (PPO) algorithm, a trading agent was trained in a custom environment that simulates bond market dynamics based on real historical data. The agent's performance was benchmarked against two reference strategies: a passive hold policy and a random trading policy.

Results indicate that the PPO agent achieved superior cumulative returns, supported by stable training diagnostics such as explained variance, KL divergence, and clip fraction. Although the limited number of training runs reduces the statistical power of the inference tests, the evidence suggests that the learned policy consistently outperforms the baselines. These findings support the rejection of the null hypothesis and validate PPO as a promising tool for active fixed-income portfolio optimization.

Key words: Fixed Income, Treasury Bonds, Deep Reinforcement Learning (DRL), Proximal Policy Optimization (PPO), Duration, Convexity, Portfolio Optimization, Policy Gradient.

Resumen

Este estudio explora la aplicación del Aprendizaje por Refuerzo Profundo (DRL) para optimizar la gestión de carteras de renta fija, con un enfoque particular en los bonos del Tesoro de los Estados Unidos. Utilizando el algoritmo Proximal Policy Optimization (PPO), se entrenó un agente de negociación en un entorno personalizado que simula la dinámica del mercado de bonos a partir de datos históricos reales. El desempeño del agente se comparó con dos estrategias de referencia: una política pasiva de mantenimiento y una política de negociación aleatoria.

Los resultados indican que el agente PPO logró rendimientos acumulados superiores, respaldados por métricas de entrenamiento estables como la varianza explicada, la divergencia KL y la fracción de recorte. Aunque el número limitado de ejecuciones de entrenamiento reduce la potencia estadística de las pruebas de inferencia, la evidencia sugiere que la política aprendida supera consistentemente a los referentes. Estos hallazgos respaldan el rechazo de la hipótesis nula y validan a PPO como una herramienta prometedora para la optimización activa de carteras de renta fija.

Palabras clave: Renta Fija, Bonos del Tesoro, Aprendizaje por Refuerzo Profundo (DRL), Optimización Proximal de Políticas (PPO), Duración, Convexidad, Optimización de Carteras, Gradiente de Política.

1. INTRODUCTION AND SCOPE OF THE STUDY

1.1. Research Objectives

The main objective of this research is to evaluate the viability of using a Deep Reinforcement Learning (DRL) model, specifically the Proximal Policy Optimization (PPO) algorithm, for optimizing the returns of fixed income portfolios. The study aims to determine whether such a model can outperform two benchmark strategies: a random trading policy (random) and a passive strategy that does not engage in market activity (hold), selected to provide a baseline for performance evaluation. The random policy represents uninformed trading behavior, while the hold strategy serves as a minimal-activity reference, allowing the model's added value through active decision-making to be clearly measured. By assessing the PPO agent's performance in a simulated U.S. Treasury bond environment, the study intends to contribute to the growing body of work that explores the application of DRL in portfolio management.

As discussed in the literature review, numerous reinforcement learning models have been proposed for portfolio management and trading. However, most of these applications focus predominantly on stocks and currencies, with comparatively limited attention given to fixed-income assets. This gap highlights the need for tailored approaches that address the unique characteristics and challenges of fixed-income portfolio optimization.

Additionally, this research is conceived as a steppingstone, providing empirical evidence and a reproducible framework that can motivate future developments toward more sophisticated models capable of operating in real-world public fixed income markets.

1.2. Hypotheses

To test the objective of the research, we formally define the following hypotheses:

Null Hypothesis (H_0):
The PPO agent does not achieve significantly higher cumulative returns R_{PPO} than the benchmark strategies R_{random}, R_{hold}.

$$R_{PPO} \leq \max\left(R_{random}, R_{hold}\right)$$

Alternative Hypothesis (H_1):
The PPO agent achieves significantly higher cumulative returns than both benchmarks.

$$R_{PPO} \geq \max\left(R_{random}, R_{hold}\right)$$

These hypotheses are tested using statistical inference methods, specifically t-tests, to assess whether the observed differences in performance can be attributed to the learning capabilities of the PPO agent rather than random variation.

1.3. Methodology

To test the hypothesis and structure the investigation, the methodology is divided into the following steps, combining theoretical review and experimental validation:

1. Literature Review. An examination of existing research on the use of DRL for portfolio management and market trading strategies. This also includes foundational work on classical portfolio theory.
2. Understanding Fixed Income Markets. A detailed explanation of the fixed income market is provided, covering the scale of U.S. Treasury instruments and the mathematical construction of bond pricing. Key pricing variables such as yield, duration, and coupon structures are identified to ensure a clear understanding of the domain where the PPO agent is applied.
3. Reinforcement Learning and PPO Foundations. The study introduces the conceptual basis of Reinforcement Learning (RL), followed by a focused explanation of the PPO algorithm, its clipped objective function, and associated metrics which are used to monitor training stability and performance.
4. Model Design and Dataset Construction. The paper outlines the construction of a custom OpenAI Gym environment tailored to U.S. Treasury bonds. This includes selecting relevant bond identifiers, structuring observation and action spaces, and detailing the state variables used. The dataset is derived from Bloomberg terminal exports, filtered for consistent structure and date coverage, and split into training and evaluation sets.
5. Performance Evaluation. The PPO agents are trained on the training dataset and evaluated on a holdout test set. Their cumulative returns are compared against random and passive benchmarks. Statistical testing is used to evaluate the significance of the results, supported by training diagnostics that include explained variance, KL divergence and clip fraction, to ensure both the stability and efficacy of learning.

2. LITERATURE REVIEW

2.1. Portfolio Theory

Modern portfolio theory provides a systematic framework for constructing investment portfolios that optimize returns relative to risk. The foundational premise of the theory is that investors should not focus on individual assets in isolation, but rather consider how the entire portfolio performs in terms of expected return and risk, measured by the variance of returns (Markowitz, 1952). A key insight is that the combination of assets can reduce overall portfolio risk through diversification, as long as the returns of the assets are not perfectly correlated. The efficient frontier, a central concept in this framework, represents the set of portfolios offering the highest expected return for a given level of risk. This concept is formalized through quadratic optimization techniques to identify portfolios that minimize variance subject to an expected return constraint (Markowitz, 1952).

Markowitz (1952) demonstrates that diversification is not merely a heuristic but a mathematically grounded outcome of optimizing the trade-off between expected return and variance. The set of efficient portfolios is derived from the feasible combinations of assets, where no other allocation offers higher expected return without also increasing risk.

The Capital Asset Pricing Model (CAPM) extends this framework by introducing a linear relationship between the expected return of an asset and its systematic risk, quantified by beta (Sharpe, 1964). CAPM assumes the existence of a risk-free rate and a market portfolio comprising all investable assets. In equilibrium, all investors hold a combination of the risk-free asset and the market portfolio, with the expected return of any asset determined by its sensitivity to market movements. This implies that only systematic risk is priced, while unsystematic risk can be diversified away. The model yields the Security Market Line, which represents the equilibrium trade-off between risk and return for any individual asset or portfolio (Sharpe, 1964).

Later refinements of Modern Portfolio Theory (MPT) have focused on the practical difficulties of implementation, particularly in estimating input parameters such as expected returns, volatilities, and correlations. Fabozzi et al. (2002) illustrate that relying solely on historical performance to derive these estimates can lead to significant inaccuracies, as asset class returns and risk measures vary considerably across time periods. This variability challenges the reliability of mean-variance optimization outcomes. To address this issue, practitioners often adjust inputs based on forward-looking expectations or impose constraints on certain asset classes to reflect the degree of confidence in their estimates. These adjustments aim to improve the robustness of the optimization process and reflect the growing sophistication of investment professionals in applying MPT principles in real-world portfolio management (Fabozzi et al., 2002).

2.2. Rl and DRL for Portfolio Management

Deep Reinforcement Learning (DRL) has emerged as a powerful paradigm for portfolio optimization, enabling autonomous agents to learn optimal asset allocation strategies by interacting with dynamic financial environments. Jiang (2017) propose a fully end-to-end DRL framework using policy-gradient methods, allowing agents to directly learn cryptocurrency portfolio weights from historical market data. This dynamic approach enables real-time adaptation to evolving market conditions, outperforming traditional static strategies.

Guo et al. (2018) propose a robust reinforcement learning-based strategy for portfolio management over constituent stocks of the CSI 300 index, which combines a log-optimal strategy with a convolutional neural network. Their method aims to maximize long-term capital growth by optimizing the expected logarithmic rate of return, and it demonstrates empirical robustness to estimation noise. Although transaction costs are assumed to be zero and volatility is not directly modeled, the design includes regularization to help mitigate overfitting in non-stationary financial environments.

Huang (2018) formulates financial trading as a Markov Decision Process (MDP) and applies a modified Deep Recurrent Q-Network (DRQN) to model it as a sequential decision-making task under uncertainty. The agent is trained to maximize cumulative currency portfolio returns while accounting for transaction costs, using techniques like action augmentation and longer training sequences to handle the complexity of financial markets. The framework is entirely model-free and does not rely on classical Markowitz-type assumptions.

Soleymani (2020) present a deep reinforcement learning framework, DeepBreath, which integrates online learning and restricted stacked autoencoders to tackle the non-stationarity of stocks financial markets. Their system performs dimensionality reduction to extract informative latent features from high-dimensional financial data, which are then used to train the portfolio policy. The agent adapts effectively to evolving market conditions, improving rebalancing efficiency and achieving superior returns compared to traditional expert strategies.

Yang et al. (2021) propose an ensemble DRL approach that combines three actor-critic algorithms. By aggregating the outputs of these distinct agents, their method improves both generalization and robustness. Applied to stock markets, the performance of the trading agent with different reinforcement learning algorithms is evaluated and compared with the Dow Jones Industrial Average index. Empirical results show that the ensemble model outperforms each individual algorithm as well as traditional benchmarks.

2.3. Rl and DRL for Trading

Huang (2018) reformulates financial trading as a Markov Decision Process and applies a modified Deep Recurrent Q-Network (DRQN) to the foreign exchange market. The study introduces several innovations, including a reduced replay memory, longer sequence sampling for recurrent training, and a novel action augmentation technique that allows the agent to learn effectively without relying on random exploration. These modifications enable the model to operate in a fully online learning setting, making real-time deployment more feasible. The proposed Financial DRQN agent achieves positive returns on 12 currency pairs under transaction costs, demonstrating strong empirical performance and robustness. The author also reports competitive Sharpe and Sortino ratios, with some strategies showing low correlation with traditional baselines.

Yang et al. (2021) emphasize the value of ensemble models in trading systems, showing that the combination of multiple DRL agents significantly outperforms both individual strategies and market indices. Their method demonstrates robustness across different market regimes, a crucial advantage in volatile financial environments.

Ozbayoglu et al. (2020) conduct a comprehensive review of deep learning applications in finance. They observe that these models can capture temporal dependencies, regime shifts, and nonlinear patterns more effectively than traditional machine learning methods.

Sarlakifar et al. (2024) propose a novel deep reinforcement learning architecture for automated stock trading, combining xLSTM networks with Proximal Policy Optimization. Their model outperforms classic LSTM-based agents across several metrics, including Sharpe ratio and cumulative return, when applied to volatile stock markets.

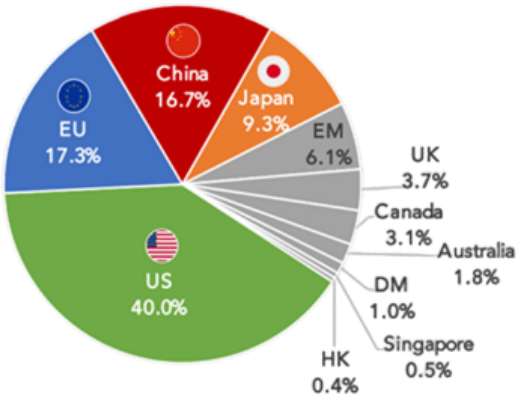
Nabipour et al. (2020) present a comprehensive comparative analysis of nine machine learning models and two deep learning architectures (RNN and LSTM) for stock market trend prediction. Utilizing ten years of historical data from the Tehran Stock Exchange and ten technical indicators, they evaluate performance under two input settings: continuous and binary data. Their empirical findings reveal that deep learning models, particularly LSTM and RNN, consistently outperform traditional machine learning techniques in terms of accuracy. The study highlights the effectiveness of deep learning in capturing temporal dependencies in financial time series, especially when enhanced through data preprocessing techniques like binarization.

3. FIXED INCOME MARKET

Fixed income refers to a category of financial instruments that provide investors with regular, predetermined payments and the return of principal upon maturity. These instruments are most commonly issued by governments, corporations, or financial institutions to raise capital for funding operations or public initiatives. The name "fixed income" originates from the consistent cash flows these assets typically generate, usually in the form of fixed interest (coupon) payments made at regular intervals (Bolsas y Mercados Españoles [BME], n.d.).

The fixed income market is also one of the largest and most liquid segments of the global financial system. As of early 2024, the U.S. fixed income market represents approximately 40% of the more than $130 trillion in outstanding global debt securities, making it more than twice the size of the next largest market, the European Union. Combined, the bond markets of the U.S., EU, China, and Japan account for over 80% of all outstanding global fixed income securities (MUFG, 2024).

Figure 1. Global fixed income securities outstanding



Source: MUFG

Given their global scale and systemic relevance, fixed income markets play a fundamental role in capital formation, monetary policy transmission, and financial stability. Their predictable cash flows and structured features make them a cornerstone of both public and private investment strategies.

Developing an optimization model for fixed income portfolios represents a strategic opportunity within the financial domain. Although debt instruments typically offer lower returns compared to equities, this difference is justified by their reduced risk profile, stemming from their senior position in a company's capital structure. In the event of bankruptcy, creditors have legal priority over shareholders in recovering their investments, which grants bonds a higher likelihood of repayment. This hierarchy within the capital structure explains the lower yields demanded by the market for debt securities relative to equity. Nevertheless, this stability creates a window for deploying optimization

strategies aimed at maximizing risk-adjusted returns in one of the largest and most essential sectors of the global financial system.

However, fixed income securities are not free from financial risk. Several forms of risk may impact the performance or valuation of these instruments:

- Inflation Risk: Inflation refers to the general increase in price levels over time. Since most fixed income securities offer nominal (non-inflation-adjusted) payments, rising inflation diminishes the real purchasing power of future cash flows, reducing the instrument's attractiveness to investors.
- Interest Rate Risk: If interest rates rise after a bond is issued, newly available bonds may offer higher yields, making existing lower-yield bonds less valuable in the secondary market.
- Default Risk: Refers to the possibility that the issuer may fail to meet its financial obligations, either by missing interest payments or defaulting on the principal. Securities with higher perceived default risk typically offer higher yields to compensate for this uncertainty.
- Exchange Rate and Capital Control Risk: When fixed income instruments are denominated in a foreign currency, investors face the additional risk that exchange rate fluctuations could erode returns. In some cases, governments may impose capital controls that restrict investors' ability to repatriate funds or liquidate investments (Corporate Finance Institute, n.d.).

Long-term government debt instruments, such as bonds with maturities of up to 30 years, pay fixed semiannual interest. Their extended duration makes them highly sensitive to changes in interest rates, increasing both their return potential and associated risk. In this study, we focus specifically on bonds due to their standardized structure, deep market liquidity, and long maturities, which enable consistent valuation across investment horizons. These characteristics make bonds particularly suitable for modeling within our reinforcement learning framework.

3.1. Bonds Structure

Bond valuation relies on the discounted cash flow approach, which accounts for the time value of money. A bond generates a stream of future payments, typically fixed periodic coupons and a lump-sum principal repayment at maturity, which are discounted back to their present value using the prevailing market yield. This yield, often referred to as the required rate of return, incorporates the bond's risk profile and reflects investor expectations (Petitt et al., 2015).

In the case of a standard fixed-rate bond, its price is determined by summing the present values of all coupon payments and the face value repayment. When the bond's coupon rate is below the current market rate, it sells at a discount; if it exceeds the market rate, it trades at a premium; and if the rates are equal, it trades at par. This pricing dynamic is fundamental to fixed income valuation (Petitt et al., 2015).

The general pricing formula is:

(1)

$$PV = \frac{PMT}{(1+r)^1} + \frac{PMT}{(1+r)^2} + \frac{PMT + FV}{(1+r)^N}$$

Where:

PV: The current value of the bond, representing its market price
PMT: The fixed interest payment made by the bond at each period
FV: The amount the bondholder receives at maturity
R: Market discount rate or required rate of return per period
N: Total number of equal time intervals until the bond matures

This framework applies across global markets, regardless of whether coupons are paid annually or semiannually.

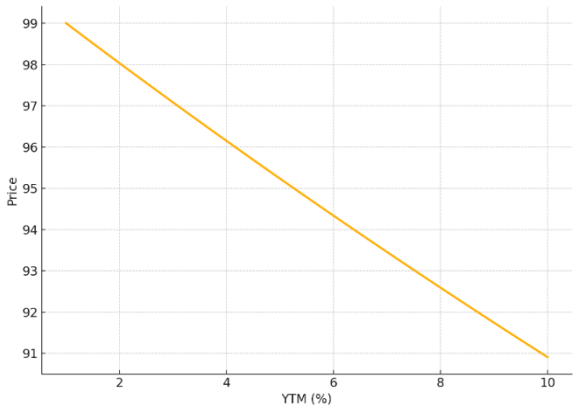
3.2. Bond Metrics and Characteristics

In fixed income analysis, evaluating the performance and sensitivity of bonds requires the use of specific analytical metrics. Among the most critical are Yield to Maturity (YTM), Duration, and Convexity, which together provide a comprehensive understanding of a bond's return profile and its reaction to interest rate changes.

1. Yield to Maturity (YTM)

Yield to Maturity is the most widely referenced yield metric in bond valuation. It represents the internal rate of return that equates the present value of all expected future cash flows of a bond, including periodic coupon payments and the repayment of principal at maturity, with its current market price. YTM is also referred to as the redemption yield, and it is typically expressed on an annualized basis. It offers a theoretical projection of the yield an investor would receive by buying the bond at its present market price and holding it until maturity, assuming all payments occur as planned (Petitt et al., 2015).

Figure 2. Bond price vs. Yield to Maturity



Source: Own elaboration

There exists an inverse relationship between bond prices and their yield to maturity. When YTM increases, the present value of future cash flows declines, leading to a lower bond price. This relationship plays a central role in market pricing and return expectations, especially in anticipation of interest rate movements.

The sensitivity of the inverse relationship between a bond's price and its yield is primarily determined by two key metrics: duration and convexity. These tools estimate how pronounced price changes will be in response to shifts in yield. For this reason, they are considered critical variables in the construction of the PPO model, as they provide structural information on interest rate risk and enable the reinforcement learning agent to learn more effective policies to maximize risk-adjusted returns within the fixed income environment.

2. Duration and Modified Duration

Duration represents the weighted average time over which a bondholder receives the bond's expected cash flows. It serves as a key indicator of a bond's sensitivity to interest rate changes, quantifying the estimated change in price resulting from a marginal shift in yield. For zero-coupon bonds, duration matches the bond's maturity, as all cash flows occur at the end. In contrast, bonds that pay periodic coupons exhibit shorter durations, since portions of the investment are repaid before maturity (Hull, 2018).

The basic idea behind duration is that it calculates a weighted average of all payment dates, using the present value of each payment as its weight. This makes duration a useful approximation of the first-order price change of a bond due to changes in interest rates. This is mathematically (continuously compounded) expressed as:

(2)

$$D = \frac{\sum_{i=1}^{n} t_i C_i e^{-y t_i}}{B}$$

Where:

D: Macaulay duration
n: total number of cash flows
t_i: time (in years) to the i th cash flow
C_i: amount of the i th cash flow
y: continuously compounded yield
B: current bond price
e: Euler's number (base of natural logarithms)

For a small change in yield Δy, the change in bond price ΔB can be approximated as:
(3)

$$\Delta B \approx \frac{dB}{dy}\,\Delta y$$

Where:
ΔB: approximate change in bond price
$\frac{dB}{dy}$: derivative of the bond price with respect to the yield
Δy: small change in yield

To reflect the relationship between price and yield more practically, we use the duration-based estimate under continuous compounding:
(4)

$$\Delta B \approx -BD\Delta y$$

Here, D is the Macaulay duration under continuous compounding, and the minus sign reflects the inverse relationship between price and yield.

However, since most yields are expressed with periodic compounding rather than continuous compounding, we often use modified duration, which adjusts standard duration to reflect the bond's compounding frequency.
(5)

$$\Delta B \approx \frac{-BD\Delta y}{1+y}$$

And more generally, if the compounding frequency is m times per year:
(6)

$$\Delta B \approx \frac{-BD\Delta y}{1+\frac{y}{m}}$$

To simplify this expression, the Modified Duration $D^*$is defined as:
(7)

$$D^* = \frac{D}{1+\frac{y}{m}}$$

This allows us to express the price change more compact, when y is expressed with a compounding frequency of m times per year, as:
(8)

$$\Delta B \approx -BD^*\Delta y$$

Modified duration gives a more practical estimate of the percentage change in bond price for a 1% change in yield. For example, if a bond has a modified duration of 2.5, a 1% (or 100 basis points) increase in yield would result in an approximate 2.5% decrease in the bond's price, all else being equal. This makes modified duration a critical tool for interest rate risk management (Hull, 2018).

3. Convexity

While duration is a useful approximation, it is most accurate only for small changes in yield. For larger shifts, the bond price-yield relationship becomes nonlinear. This curvature is captured by a second-order measure called convexity. Convexity quantifies how the duration of a bond changes as yields change, offering a correction to the duration-based estimate of price sensitivity (Hull, 2018).

From Taylor series expansions we obtain:

(9)

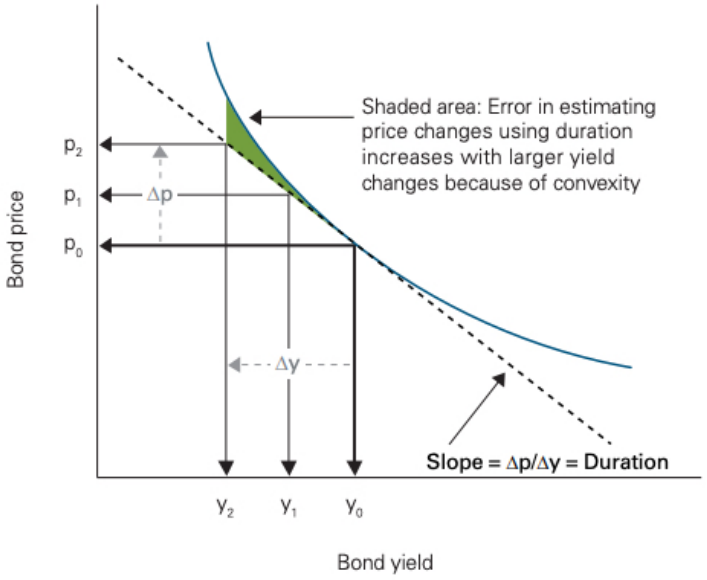
$$\Delta B = \frac{dB}{dY}\,\Delta y + \frac{1}{2}\frac{d^2B}{dy^2}\Delta y^2$$

This leads to:

(10)

$$\frac{\Delta B}{B} = -D\,\Delta y + \frac{1}{2}C(\Delta y)^2$$

Figure 3. Bond Price-Yield Relationship: Duration and Convexity



Source: Asymmetry Observations (2019)

Higher convexity implies a bond is less sensitive to interest rate risk than a bond with lower convexity, assuming the same duration. It becomes particularly important in scenarios involving significant yield changes, as it allows for a more accurate prediction of price fluctuations. Bonds with evenly distributed payments over time tend to exhibit higher convexity than those with concentrated cash flows.

3.3. Metrics in Portofolios

In portfolio management, the duration of a bond portfolio is calculated as the weighted average of the durations of its component bonds, with weights based on each bond's market value within the portfolio. This composite duration provides a measure of the portfolio's overall price sensitivity to minor, consistent shifts in interest rates. However, this method relies on the assumption of a parallel yield curve movement, where interest rates across all maturities change by roughly the same magnitude. This assumption holds reasonably well when bonds have similar maturities but may not be reliable when the portfolio contains instruments with highly diverse durations (Hull, 2018).

To mitigate interest rate risk, financial institutions often structure portfolios where the duration of assets matches that of liabilities, thereby achieving what is known as net duration neutrality. In this case, the portfolio becomes immune to small parallel shifts in interest rates, although it remains exposed to non-parallel or large shifts. Moreover, convexity also plays a critical role: portfolios with evenly distributed cash flows tend to exhibit higher convexity, enhancing protection against more substantial parallel movements. When both net duration and net convexity are set to zero, a portfolio may achieve a higher level of immunization, although full protection against all yield curve shifts remains theoretically unattainable (Hull, 2018).

4. REINFORCEMENT LEARNING

Before delving into the structure and functioning of the PPO algorithm, it is essential to establish the theoretical foundations upon which the model is built. In this context, Reinforcement Learning (RL) and its extension, Deep Reinforcement Learning (DRL), provide the conceptual and computational framework that enables the agent to learn optimal portfolio allocation strategies through interaction with a dynamic financial environment. The following section explores these paradigms in detail, laying the groundwork for understanding how PPO operates within this framework to enhance fixed-income portfolio performance.

Reinforcement Learning (RL) is a field of machine learning in which an agent learns by interacting with an environment. Unlike supervised learning, where models are trained on pre-labeled datasets, RL does not require such datasets. Instead, it depends on feedback signals generated by the agent's actions within the environment. This characteristic allows RL to be applied to a broader range of problems where labeled data may not be available (Plaat, 2022).

However, it also introduces challenges. Without guidance, an agent may become overly reliant on actions that yield immediate rewards, avoiding actions that might lead to better long-term outcomes. Balancing exploration and exploitation is a crucial aspect of RL (Plaat, 2022).

The following section is based entirely on the work of Gosavi (2014):

Reinforcement Learning (RL) addresses the Markov Decision Problem (MDP) or its variant, the Semi-Markov Decision Problem (SMDP), by leveraging the theory of dynamic programming (DP) and artificial intelligence (AI). The framework of the MDP consists of several key elements: (1) state, (2) actions, (3) transition probabilities, (4) transition rewards, (5) policy, and (6) performance metric. These components are typically modeled using a Markov chain, a stochastic process that forms the basis for MDPs.

5.1. MDP and SMDP framework

1. State (S)
   The state represents the system's configuration at any given time, formally denoted as $s_t \in S$, where S is the state space. It is a parameter (or set of parameters) that fully describes the current condition of the system. In dynamic systems, the state evolves over time, meaning the system's state changes in response to various factors.
2. Actions (A)
   The action $a_t \in A(s_t)$ is the decision or control applied at state s_t. Each state may allow multiple actions, influencing the system's transition to the next state. The set $A(s_t)$ defines the possible actions for each state s_t. The goal is to select actions that optimize the long-term performance of the system.

3. Transition Probabilities (P)
   The transition probability $P(s_{t+1}|s_t, a_t)$ represents the likelihood of moving from state s_t to s_{t+1} after taking action a_t. This function defines the stochastic dynamics of the system, where each action has an associated probability of leading to a subsequent state.
4. Immediate Rewards (R)
   The reward $r(s_t, a_t, s_{t+1})$ is the immediate feedback or benefit obtained after transitioning from state s_t to state s_{t+1} under action a_t. The reward quantifies how favorable or unfavorable a particular state transition is, often representing costs or benefits.
5. Policy (π)
   A policy π is a mapping that specifies the action to be taken at each state. It can be deterministic ($\pi(s_t) = a_t$) or stochastic ($\pi(a_t|s_t)$), defining the agent's strategy for making decisions based on the current state. The objective is to find an optimal policy π^* that maximizes performance over time.
6. Performance metric (ρ) and time of transition
   a) Average Reward:

   The average reward is used to evaluate the long-term effectiveness of a policy π over an infinite time horizon, measuring the reward accumulated per transition.

   - In MDPs, since the time between transitions is constant, the average reward is calculated by summing the immediate rewards over an infinite sequence of transitions and dividing by the total number of transitions. The formula for the average reward is:

(11)

$$\rho_i = \lim_{k\to\infty} \frac{1}{K} \mathbb{E}\left[\sum_{s=1}^{k} r(x_s, \pi(x_s), x_{s+1}) \mid x_1 = i \right]$$

   where $r(x_s, \pi(x_s), x_{s+1})$ represents the immediate reward for transitioning from state x_s to state x_{s+1} under action $\pi(x_s)$. The $\mathbb{E}$ operator denotes the expected value of the sum of rewards, considering the stochastic nature of the transitions.
   - In SMDPs, where the time between transitions is variable, the average reward is modified to account for the time spent in each transition. The formula for the average reward in SMDPs becomes:

(12)

$$\rho_i = \lim_{k\to\infty} \frac{\mathbb{E}\left[\sum_{s=1}^{k} r(x_s, \pi(x_s), x_{s+1}) \mid x_1 = i \right]}{\mathbb{E}\left[\sum_{s=1}^{k} t(x_s, \pi(x_s), x_{s+1}) \mid x_1 = i \right]}$$

   Here, $t(x_s, \pi(x_s), x_{s+1})$ represents the time spent in each transition, and the $\mathbb{E}$ operator accounts for the expected value of both the rewards and the transition times. This formula ensures that the reward is evaluated in relation to the time

taken for each transition, which is critical in systems where transition durations are not uniform.

b) Discounted Reward

The discounted reward metric is used when future rewards are less valuable than immediate rewards, reflecting the time value of rewards in decision-making.

- In MDPs, the discounted reward is calculated by applying a discount factor $\gamma \in (1,0)$ to future rewards. The formula for the discounted reward in MDPs is:

(13)

$$\psi_i = \lim_{k\to\infty} \mathbb{E}\left[\sum_{s=1}^{k} \gamma^{s-1} r(x_s, \pi(x_s), x_{s+1}) \mid x_1 = i\right]$$

where γ^{s-1} discounts future rewards, with $\mathbb{E}$ representing the expected value of the sum of rewards over time. The expectation ensures that the rewards are averaged according to the probabilistic nature of state transitions.

- In SMDPs, the discounted reward is adjusted to consider the variability in transition times. The formula for the discounted reward in SMDPs is:

(14)

$$\psi_i = \lim_{k\to\infty} \mathbb{E}\left[r(x_1, \pi(x_1), x_2) + \sum_{s=2}^{k} r(x_s, \pi(x_s), x_{s+1}) \int_{\tau_s}^{\tau_s+1} e^{-\mu\tau} d\tau \mid x_1 = i\right]$$

Here, $e^{-\mu\tau}$ represents the continuously compounded discount factor applied over the transition time τ. The expectation operator $\mathbb{E}$ accounts for the expected value of the rewards and the time-dependent discounts, reflecting the random nature of transitions and their durations.

In MDPs, where transition times are assumed to be constant, the performance metrics (average reward and discounted reward) are calculated using simple summation of rewards over time. However, in SMDPs, where transition times vary, the performance metrics are adjusted to account for the variable transition times. This is done by incorporating time directly into the average reward and discounted reward formulas, ensuring that the evaluation of policies in SMDPs accounts for the duration of each transition.

In this text, s_t and x_s both denote the system's state. They differ only by indexing: t for time steps and s for transitions, but are otherwise equivalent.

5.2. Reinforcement Learning Framework

The Reinforcement Learning (RL) can be applied to both Semi-Markov Decision Processes (SMDPs) and Markov Decision Processes (MDPs). In SMDPs, transition times are variable, meaning the time spent in each state transition depends on the action taken. However, if $t(i, a, j)$ for all values of i, j, and a, the problem simplifies to a MDP, where the transition times are constant. Therefore, the algorithm described can easily be adapted to both SMDPs and MDPs by setting $t(i, a, j) = 1$.

An important aspect of RL is that transition probabilities and rewards are not required in certain cases. Specifically, if:

1. The system can be directly interacted with, where actions are chosen and the resulting rewards are observed.
2. A simulator of the system is available, which can model the system dynamics based on easily accessible parameters (such as distribution functions for inter-arrival times or service times).

In RL, the key idea is to store a Q-factor for each state-action pair in the system. The Q-factor, denoted as $Q(i, a)$, represents the expected cumulative reward for taking action a in state i. Initially, these Q-factors are set to small values, typically zero, and they are updated as the system interacts with the environment.

At each state visited, an action is selected and the system transitions to the next state. The immediate reward and transition time generated from this transition are recorded as feedback. The feedback is then used to update the Q-factor for the action selected in the previous state. The Q-factor is adjusted through a reward-punishment mechanism, where:

- If the feedback is favorable (higher rewards or shorter transition times), the Q-factor for the selected action is increased (rewarded).
- If the feedback is unfavorable (lower rewards or longer transition times), the Q-factor is decreased (punished).

This updating process follows the Relaxed-SMART algorithm, which refines the Q-factor by incorporating both the immediate rewards and the transition times observed. The learning process continues through multiple state transitions, with the Q-factors being gradually refined.

After a large number of transitions, the Q-factors converge to the optimal values. The optimal policy is then determined by selecting the action with the highest Q-factor for each state, ensuring that the agent maximizes its cumulative rewards. Importantly, this RL strategy does not require knowledge of transition probabilities, making it adaptable to real-time systems or simulations where such probabilities may be unknown or difficult to model. This concludes the overview adapted from Gosavi 2014.

5. DEEP REINFORCEMENT LEARNING

Deep Reinforcement Learning (DRL) represents the integration of deep learning and reinforcement learning into a unified framework. The principal aim of DRL is to determine optimal actions that maximize cumulative rewards across all possible states an environment can assume. This process requires the agent to interact with complex and high-dimensional environments, experimenting with different actions and refining its behavior based on the feedback received. Deep learning, as a field, focuses on approximating functions in contexts where the dimensionality and complexity of the problem make traditional, tabular methods infeasible. It employs deep neural networks to construct function approximations capable of handling large-scale and intricate datasets, as seen in domains such as image classification and natural language processing (Plaat, 2022).

Reinforcement learning, in contrast, centers on learning through feedback, utilizing trial-and-error methods. Unlike supervised learning, reinforcement learning does not depend on a pre-compiled dataset for training. Instead, it autonomously selects actions and updates its strategies based on the rewards or penalties provided by the environment. During this exploration process, agents inevitably commit errors; however, learning to succeed often stems from effectively processing both positive and negative experiences. In recent developments, deep learning and reinforcement learning have merged, giving rise to new algorithms that leverage deep neural networks to approximate solutions to high-dimensional problems, based on the feedback from the agent's interactions, this convergence has advanced several subfields, including policy optimization (Plaat, 2022).

5.1. Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a reinforcement learning algorithm designed to improve policy performance while maintaining training stability (Schulman et al., 2017). It was introduced by researchers at OpenAI in response to the limitations of earlier policy gradient methods such as Trust Region Policy Optimization (TRPO), which required complex second-order computations. PPO offers a more accessible alternative by using a simpler objective function that still limits drastic changes to the agent's behavior between updates (Schulman et al., 2017).

One of the main contributions of PPO is the use of a probability ratio to compare the new and old policies.This ratio reflects how much the new policy changes the likelihood of taking a specific action in a given state compared to the previous policy (Such Ballester, 2024).

The probability ratio is defined as:
(15)

$$r_t(\theta) = \frac{\pi_{(\theta)}(a_t \mid s_t)}{\pi_{(\theta old)}(a_t \mid s_t)}$$

where $\pi_{(\theta)}(a_t \mid s_t)$ is the probability of taking action a_t in state $s_t$under the new policy, and $\pi_{(\theta old)}(a_t \mid s_t)$ is the same under the old policy (Schulman et al., 2017).
This ratio allows the algorithm to detect whether the policy is changing too much in a single update, which could harm learning stability. To prevent this, PPO introduces a clipping mechanism that restricts how far the probability ratio can deviate from 1. The clipped surrogate objective function ensures that updates remain within a safe region around the previous policy (Schulman et al., 2017). This function is written as:
(16)

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta)\,\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t]$$

Where:
θ: is the vector of policy parameters
$\hat{E}_t$: denotes the empirical expectation over a batch of timesteps
$r_t(\theta)$: is the probability ratio under the new and old policies, respectively
$\hat{A}_t$: is the estimated advantage at timestep t, representing how favorable an action was
ϵ: s a small positive hyperparameter (commonly 0.1 or 0.2) used to clip the ratio range

This formulation prevents the policy from being updated too aggressively, even if the advantage estimate suggests a strong reward signal. In this way, PPO avoids overfitting to recent experiences and reduces the risk of policy collapse (Schulman et al., 2017).

González Oviedo (2023) notes that PPO is especially valued in practical applications because it combines good learning performance with ease of implementation. The algorithm has been widely used in training agents for video games, control environments, and simulations requiring either continuous or discrete actions. Its balance between computational simplicity and training robustness makes it suitable for both academic research and applied reinforcement learning projects (González Oviedo, 2023; Such Ballester, 2024).

Finally, PPO does not rely on complex mathematical constraints like KL divergence penalties or Lagrangian multipliers, which were necessary in earlier algorithms like TRPO (Schulman et al., 2017). Instead, it controls policy change using a clipping approach that is both intuitive and effective (Schulman et al., 2017). This simplicity has contributed to its status as one of the most popular algorithms in Deep Reinforcement Learning (González Oviedo, 2023).

6. MODEL TRAINING FRAMEWORK

6.1. Data Preprocessing

The dataset used in this study consists of 33 fixed-income instruments issued by the United States government. This sample was obtained from the iShares U.S. Treasury Bond ETF, an exchange-traded fund managed by BlackRock that invests exclusively in U.S. Treasury securities. The objective of this ETF is to replicate the performance of an index composed of public debt instruments issued by the U.S. Department of the Treasury, thus providing diversified exposure to the government fixed-income market. Although this fund was not used as a performance benchmark in the study, it served as a reference source for selecting a representative and realistic portfolio of sovereign bonds with diverse characteristics. The selected instruments are bonds with maturities ranging from 20 to 30 years. This choice responds to a methodological necessity: fixed-income instruments have a limited lifespan, and to ensure consistency and completeness of the observations over time, it was essential to use data from assets with continuous availability throughout the entire analysis period, which spans from January 2017 to February 2024.

Each row in the dataset represents a trading day, and each column contains specific information about a variable corresponding to a given bond. In this way, for each date, the values of all relevant metrics for the 33 bonds are collected sequentially, allowing the reinforcement learning model to observe the full state of the market.

The variables included in the dataset are as follows:

- PX_LAST: Represents the last market price of the bond on the corresponding date. It reflects the price at which the asset was traded on the secondary market and is expressed in U.S. dollars.
- YLD_YTM_MID: Corresponds to the yield to maturity (YTM), calculated based on the mid-market price. This indicator reflects the effective return rate an investor can expect if the bond is held to maturity. It is expressed as an annualized percentage.
- DUR_MID: Indicates the bond's duration, a measure of price sensitivity to interest rate changes. Duration is fundamental for assessing interest rate risk and is expressed in years.
- CNVX_MID: Measures the bond's convexity, a characteristic that refines the duration estimate by considering the curvature in the price-yield relationship. This variable is dimensionless and improves the estimation of the impact of interest rate changes on bond prices in high-volatility environments.
- CPN: Represents the bond's coupon rate, the percentage of the face value periodically paid by the issuer as interest. It is also expressed as an annual percentage and is a fixed characteristic of the bond.

- ISSUE_DT: Indicates the bond's issue date, when it was introduced to the market. This variable is recorded in standard date format (DD/MM/YYYY).
- MATURITY: Corresponds to the bond's maturity date, marking the moment the issuer must repay the face value to the holder. It is also represented in standard date format.

The selection of these variables is based on technical criteria aimed at capturing the fundamental dynamics of fixed-income instruments. Since a bond's price is inversely correlated with interest rates, as is it well-documented during this study, the explanatory variables included in the dataset are designed to reflect each bond's level of exposure to interest rate movements. Metrics such as duration and convexity are therefore essential for enabling the agent to learn and anticipate the impact of macroeconomic changes on the value of the assets comprising the portfolio.

6.2. Training Agent

To simulate the investment process over time, a custom environment named FixedIncomeEnv was developed by Eduardo Garrido. This environment models the trading of a fixed-income portfolio composed of U.S. Treasury bonds, leveraging the historical dataset described above as its market data source.

6.2.1. Environment Design and State Representation

State Observation

At each time step, the observation returned by the environment is a numerical array representing recent prices of a subset of bonds. In this implementation, the portfolio is restricted to three specific Treasury bonds, so the state focuses on those three instruments. For each of the three bonds, the state includes its current price and a one-day lagged price, the previous trading day's price. By including lagged prices, the state provides the agent with a notion of short-term price momentum or trend. This results in an observation vector of size 6 (for three bonds, each with two price values). All static bond attributes such as coupon rate, issue date, or maturity date are excluded from the state. These features were originally considered but later removed in the code because they did not improve learning performance. The observation is purely numerical and is cleaned of any invalid values: any NaNs or infinite values in the price data are sanitized before use, ensuring the agent always receives well-defined numeric inputs.

Portfolio Variables

The environment maintains an internal state of the agent's portfolio, consisting of a cash balance and holdings of each of the three bonds. At the start of an episode, the agent is given an initial cash balance, with no bonds held initially. The initial cash provides capital that the agent can deploy to buy bonds. Throughout the episode, the environment continuously updates the portfolio state based on the agent's actions: bond holdings can increase or decrease and the cash balance is debited or credited accordingly. The total portfolio value at any time is calculated as the sum of remaining cash plus the market

value of all bond holdings, computed as number of units of each bond held multiplied by that bond's current price.

6.2.2. Action Space and Reward Mechanism

At every trading day (time step), the agent must decide how to manage the portfolio by choosing an action for each of the three bonds. The action space is defined as a discrete choice for each bond, and we uses a *Gym MultiDiscrete* space to represent the joint action. Specifically, the action space is *MultiDiscrete([3] * n_bonds)*, which in this case is *MultiDiscrete()* for the three bonds. This means the agent outputs a vector of three actions $[a_1, a_2, a_3]$, one for each bond, and each component can take one of three integer values:

- 0 = BUY: Purchase one unit of the bond. If this action is selected for a bond, the environment will attempt to buy one unit of that bond on the market. The bond's price, from the current day, is subtracted from the cash balance, and the holding count for that bond is increased by one. The code ensures that a buy action is only effective if there is sufficient cash available; otherwise, if the agent tries to buy with insufficient cash, the environment may prevent the trade.
- 1 = HOLD: Take no action for that bond. This means the agent neither buys nor sells the bond on that day. The portfolio remains unchanged for that instrument, holdings and cash are unaffected, aside from any market-driven price change that will be reflected in the portfolio value at the next step. A hold action effectively means the agent is satisfied with its current position in that bond for the day.
- 2 = SELL: Sell one unit of the bond. If this action is chosen, the environment will sell one unit from the agent's holdings of that bond, if the agent currently holds at least one. The bond's current price is added to the cash balance, and the holding count for that bond decreases by one. The environment enforces a no short selling rule, so it will not allow the agent's holdings to go negative.

By using a *MultiDiscrete* action space, the agent makes simultaneous decisions for all three bonds at each step. The environment processes all three sub-actions and updates the portfolio accordingly.

<u>Reward Function</u>

The reward at each time step is designed to encourage the agent to increase the total portfolio value. The reward is defined as the change in the portfolio's total value from the previous day to the current day. Formally, if V_t is the total portfolio value (cash + market value of all bond holdings) at time t, then the reward $r_t = V_t - V_{t-1}$. This reward is essentially the daily profit or loss of the portfolio.

Positive rewards are earned when the portfolio's value grows and negative rewards are incurred when the portfolio value falls. There are no transaction costs in this simulation, so buying or selling does not incur any fee. Moreover, since short selling is not allowed, the agent's strategy is constrained to hold long positions or cash. Over an episode, the

cumulative reward will equal the total profit achieved from start to finish. The episode concludes at the end of the dataset after the last trading day, at which point the final portfolio value can be compared to the initial value to assess overall return.

6.2.3. Training Setup with Proximal Policy Optimization (PPO)

To train the trading agent, the implementation employs the Proximal Policy Optimization (PPO). In this code, two independent PPO agents are trained with identical parameters but different random seeds to ensure robustness of results. Each agent interacts with the FixedIncomeEnv environment configured on the historical data excluding the final testing period and learns a trading policy over many iterations.

Training Episodes and Timeline

The total training length is specified in terms of time steps: each PPO agent is trained for 300,000 time steps. A time step corresponds to one day and one set of actions for the three bonds, so 300k steps covers many passes through the dataset. During training, PPO optimizes the policy neural network which decides actions given state observations and a value neural network which estimates future returns to maximize expected rewards.

During training, the algorithm interacts with the environment as follows: at each time step, the agent produces an action vector $[a_1, a_2, a_3]$ given the current observation, current and previous prices for each bond. The environment applies these actions, transitions to the next day's state, and returns a reward, the change in portfolio value. PPO collects these (state, action, reward, next state) transitions in a rollout buffer.

6.2.4. Evaluation and Benchmarking Performance

After training the PPO agents, we evaluate their performance on a held-out test dataset. This simulates how the agent would perform on new, unseen market data, and it serves to validate the generalization of the learned trading strategy. The evaluation process is as follows:

1. Testing Environment

   A new instance of the FixedIncomeEnv is initialized for the test period. The initial portfolio conditions, initial cash and zero bond holdings are reset. From that point, the agent will step through each trading day of the test year exactly once, without any learning updates, pure evaluation mode.

   Each trained PPO agent is run on the test environment. In evaluation, the agent can operate in deterministic mode, meaning it selects the action with highest probability, instead of sampling randomly from its policy, to produce a consistent, greedy strategy.

2. Benchmark Strategies

   To put the PPO agent's performance in context, the evaluation includes two baseline strategies run on the same test environment:

   - Random Policy Benchmark

     This baseline involves an agent that takes random actions for each bond at each time step. In practice, the code samples actions uniformly from the discrete action space for each bond: 0, 1, or 2 with equal probability each day, independently for each bond. This simulates a naive trader that makes uninformed buy/hold/sell decisions.

   - Always-Hold Policy Benchmark

     This strategy represents a no-trading policy. The agent chooses the HOLD action (1) for all bonds at every step, meaning no transactions occur throughout the episode. In the environment as defined, the portfolio is initialized with all cash and zero bond holdings. Since the HOLD action does not initiate any purchases, the agent retains the full cash position without ever entering the bond market. As a result, the always-hold policy leads to zero exposure to bond price movements and yields no return. It serves as a lower bound on performance, against which the effectiveness of dynamic strategies like PPO can be compared.

3. Performance Comparison

   Once the PPO agents and the two benchmarks have been run on the test set, we compare their cumulative returns over the evaluation period. The results from the two PPO agents that are trained with a different random seed are averaged to provide a single representative performance metric. By aggregating the PPO outcomes, the evaluation accounts for variability due to stochastic training, offering a more robust estimate of the algorithm's effectiveness. The hold policy will have a single deterministic outcome on the test the return is straightforwardly 0.

   We perform a statistical analysis to determine if the PPO agent's performance is significantly better than the baselines. In particular, it conducts t-tests to compare the returns:

   - A t-test between the PPO agent's returns and the random policy's returns.
   - A t-test between the PPO agent's returns and the hold policy's returns.

   The t-test assesses whether the difference in mean return between PPO and the baseline is statistically significant, not attributable to random chance. A low p-value from these t-tests would indicate that the PPO agent yields significantly different returns than the baseline strategies at a 95% confidence level.

7. TRAINING MONITORING AND CONCLUSIONS

7.1. Results of the research

Throughout training, detailed diagnostics were logged to track the PPO agent's learning progress and stability. These metrics were visualized with Matplotlib to assess performance over time. Evaluation of the trained agent on a held-out test dataset revealed superior performance compared to basic reference strategies While the p-values obtained from statistical tests were close to the 5% threshold, the limited sample size (n = 2 agents) reduces the statistical power of these results. Thus, although the differences are not definitively significant, they do suggest a meaningful advantage in favor of the trained agent, warranting further investigation with a larger set of runs.

	Strategy	Accumulated Return (%)	T-test vs PPO
0	PPO	2.744141	None
1	Random	-2.128125	0.052861
2	Uniform	0.000000	[0.0573890]

To further verify the robustness of the learning process, key training metrics, explained variance, KL divergence, and clip fraction, were analyzed. These diagnostics reinforce the stability and effectiveness of the training, as described below:

- Explained Variance: Explained variance is a metric that quantifies the extent to which a machine learning model captures the variability of the target variable through its predictions. It indicates how effectively the model accounts for fluctuations in the actual data. A value of 1.0 signifies perfect predictive performance, where the model explains all variations in the outcomes. A value of 0 suggests the model is no better than predicting the average of the target variable, while negative values reveal performance worse than this baseline. This metric is especially valuable for assessing how well the model reflects the underlying patterns in the data (KoshurAI, 2024).

In our training, the explained variance of the value function increased steadily over the first 150,000 training steps, eventually surpassing 40% and stabilizing thereafter. This indicates that the critic network learned to estimate future returns with increasing accuracy, which is essential for computing reliable advantage values in PPO.

- KL Divergence: In the context of Proximal Policy Optimization (PPO), the Kullback–Leibler (KL) divergence measures how much the new policy deviates from the previous one during each update. This divergence acts as a surrogate for trust-region constraints, with PPO implicitly defining a "safe" update region

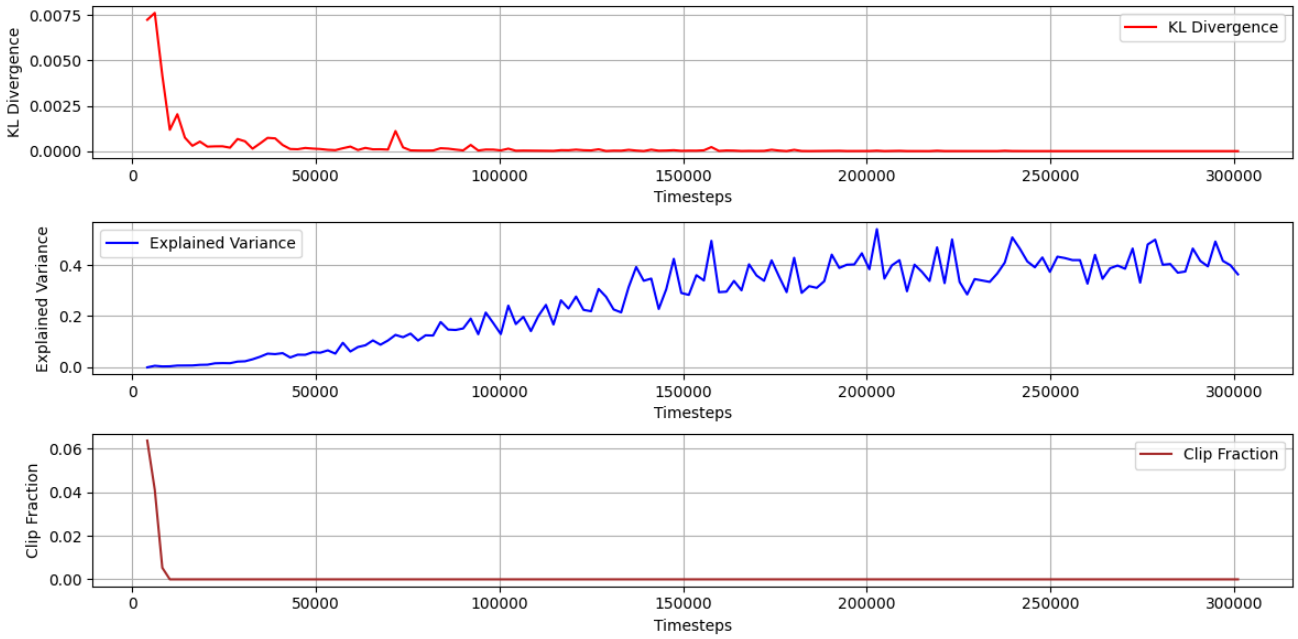
through clipping or by monitoring the KL distance between policies (Hsu et al., 2021). A low KL value indicates that policy updates are conservative and may slow learning, whereas high values suggest that updates are too aggressive, risking instability or performance collapse. Therefore, maintaining KL divergence within a moderate range is essential to ensuring training stability and preserving the effectiveness of the learning process (Hsu et al., 2021).

In our experiments, the KL divergence remained low and stable from early in the training process. No spikes were observed, indicating that policy updates stayed moderate and well within the safety margin defined by the algorithm. This supports the overall stability of the learning process.

- Clip Fraction**:** In the context of Proximal Policy Optimization (PPO), the clip fraction is a metric that indicates how often the algorithm's clipping mechanism is activated during training. PPO updates policies based on a probability ratio between the new and old policies, and this ratio is constrained within a specified range to prevent overly large and potentially destabilizing updates. The clip fraction measures the proportion of update steps in a batch where the policy change exceeded this range and was consequently clipped. This metric provides insight into the degree of constraint imposed by the algorithm and is monitored to evaluate whether the policy updates are appropriately scaled (Schulman et al., 2017).

In our training runs, the clip fraction dropped rapidly to zero in the early stages and remained at that level throughout the remainder of training. This behavior suggests that most policy updates fell within the acceptable range and did not require clipping. It confirms that the learning rate and exploration parameters were well balanced, preventing the agent from making overly large updates.

Figure 4. Training PPO metrics



7.2. Conclusions

Altogether, these findings indicate that the training process was stable, gradual, and free from collapse or overfitting. The learned policy consistently aligned with the overarching objective of maximizing portfolio returns within a fixed income investment context. The smooth evolution of performance metrics, further reinforces the notion of a well-behaved optimization process. Although the limited sample size constrains the statistical power of the hypothesis tests and cautions against overgeneralization, the observed results and the trajectory of training indicators support the conclusion that deep reinforcement learning, specifically through the PPO algorithm, represents a viable and promising methodology for the active management of U.S. Treasury securities portfolios.

Therefore, based on the empirical evidence collected, we reject the null hypothesis (H_0) that there are no significant differences in performance between PPO and the benchmark strategies. The superior returns obtained by PPO suggest that the agent learned a non-trivial policy capable of adapting to dynamic market conditions and outperforming both random and passive hold strategies in a consistent manner.

7.3. Perspectives for future research

Future work in this area can build upon the results obtained by exploring several key extensions:

1) Research should progressively expand the portfolio by incrementally introducing more bonds, which increases the dimensionality of the action and state spaces and thus demands longer training horizons to ensure proper exploration.

2) Additionally, evaluating alternative reward functions that incorporate transaction penalties and risk-adjusted metrics, such as the Sharpe and Sortino ratios, could promote more stable and realistic trading behavior.

3) Finally, benchmarking PPO against other advanced Deep Reinforcement Learning algorithms like DDPG, SAC, and TD3 would provide valuable comparative insights into their suitability for fixed income portfolio optimization.

Broadening the methodological scope will also help assess the full potential and limitations of reinforcement learning in this financial domain.

8. ARTIFICIAL INTELLIGENCE DECLARATION

Declaration of Use of Generative Artificial Intelligence Tools:
I, Jorge González de San Román Sánchez, a student of E2 + Business Analytics at Universidad Pontificia Comillas, hereby declare that in the submission of my Bachelor's Thesis entitled *" Deep Reinforcement Learning for Optimizing Fixed-Income Portfolios "*, I have used the Generative Artificial Intelligence tool ChatGPT or similar GAI-based tools solely in the context of the activities described below:

1. Language and style editor: Used to improve the linguistic and stylistic quality of the text.
2. Preliminary diagram and content generator: Used to draft initial flowcharts and visual content.
3. Summarizer and explainer of complex books: Used to understand and condense difficult literature.
4. Reviewer: Used to receive suggestions on how to improve and refine the thesis with varying levels of academic rigor.

I affirm that all the information and content presented in this thesis are the result of my individual research and effort, except where otherwise indicated and proper credit has been given (I have included appropriate references in the thesis and have clearly stated where ChatGPT or similar tools were used). I am aware of the academic and ethical implications of submitting non-original work and accept the consequences of any violation of this declaration.

Date: 16/06/2025

9. BIBLIOGRAPHY

Asymmetry Observations. (2019). *Asymmetry vs. convexity*. https://asymmetryobservations.com/2019/07/31/asymmetry-vs-convexity/

Bolsas y Mercados Españoles. (n.d.). *What is fixed income?* BME Exchange. https://www.bolsasymercados.es/bme-exchange/en/faqs/what-is-fixed-income. Last visit 18/06/2025

Corporate Finance Institute. (n.d.). *Fixed income securities*. CFI. https://corporatefinanceinstitute.com/resources/fixed-income/fixed-income-securities/. Last visit 18/06/2025

Fabozzi, F. J., Gupta, F., & Markowitz, H. M. (2002). The legacy of modern portfolio theory. *The Journal of Investing, 11*(3), 7–22.

González Oviedo, R. J. (2023). *Análisis de dos algoritmos de Reinforcement Learning aplicados a OpenAi Gym Retro: DQN y PPO aplicado entrenamiento de game agents en Ice Climbers*. Universidad de Los Andes.

Gosavi, A. (2014). *Simulation-based optimization: Parametric optimization techniques and reinforcement learning* (2nd ed.). Springer.

Guo, Y., Fu, X., Shi, Y., & Liu, M. (2018). *Robust log-optimal strategy with reinforcement learning*. *arXiv preprint*, arXiv:1805.00205. https://arxiv.org/abs/1805.00205

Hsu, C.-H., Tang, Y., Krishnan, S., & Singh, A. (2021). *Revisiting design choices in proximal policy optimization*. arXiv preprint, arXiv:2009.10897. https://arxiv.org/abs/2009.10897

Huang, C.-Y. (2018). *Financial trading as a game: A deep reinforcement learning approach*. arXiv preprint, arXiv:1807.02787. https://arxiv.org/abs/1807.02787

Hull, J. C. (2018). *Options, futures, and other derivatives* (10th ed.). Pearson.

Jiang, Z., Xu, D., & Liang, J. (2017). *A deep reinforcement learning framework for the financial portfolio management problem*. arXiv preprint, arXiv:1706.10059. https://arxiv.org/abs/1706.10059

KoshurAI. (2024). *Understanding explained variance score: A key metric for regression analysis*. Medium. https://koshurai.medium.com/understanding-explained-variance-score-a-key-metric-for-regression-analysis-17b31f50108c

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance, 7*(1), 77–91. https://doi.org/10.2307/2975974

MUFG. (2024). *Largest and most liquid capital markets in the world*. https://www.mufgamericas.com/sites/default/files/document/2024-01/Chart-of-the-Day_1_8_Largest-Most-Liquid-Capital-Markets-in-the-World.pdf

Nabipour, M., Nayyeri, P., Jabani, H., Shahab, S. S., & Mosavi, A. (2020). *Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data: A comparative analysis*. IEEE Access, 8, 150199–150212. https://doi.org/10.1109/ACCESS.2020.3015966

Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). *Deep learning for financial applications: A survey*. Applied Soft Computing, 93, 106384. https://doi.org/10.1016/j.asoc.2020.106384

Plaat, A. (2022). *Deep reinforcement learning*. Springer.

Petitt, B. S., Pinto, J. E., & Pirie, W. L. (Eds.). (2015). *Fixed Income Analysis* (3ª ed.). John Wiley & Sons.

Sarlakifar, F., Mohammadzadeh Asl, M., Rezvani Khaledi, S., & Salimi-Badr, A. (2025). *A deep reinforcement learning approach to automated stock trading, using xLSTM networks*. arXiv preprint, arXiv:2002.05786.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms.* arXiv preprint, arXiv:1707.06347. https://arxiv.org/abs/1707.06347

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance, 19*(3), 425–442. https://doi.org/10.2307/2977928

Soleymani, F., & Paquet, E. (2020). Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder: DeepBreath. *Expert Systems with Applications, 157*, 113456. https://doi.org/10.1016/j.eswa.2020.113456

Such Ballester, I. (2024). *Agente inversor para acciones de small cap mediante el uso de Reinforcement Learning*. Universitat Oberta de Catalunya.

Yang, H., Zhong, S., Liu, X.-Y., & Walid, A. (2021). *Deep reinforcement learning for automated stock trading: An ensemble strategy*. In Proceedings of the 1st ACM International Conference on AI in Finance (ICAIF '20). https://doi.org/10.1145/3383455.3422540

10. APPENDIX

Python Model Code by Eduardo Garrido Mérchan

In [ ]:

```
import gym
import numpy as np
import pandas as pd
from gym import spaces
import stable_baselines3 as sb3
from stable_baselines3 import PPO, DQN
from stable_baselines3.common.logger import configure
from scipy import stats

class FixedIncomeEnv(gym.Env):

    def seed(self, seed=None):
        np.random.seed(seed)
        return

    def augment_dataset(self, df, num_synthetic_points=4):
        augmented_data = []
        for i in range(len(df) - 1):
            base_row = df.iloc[i].copy()
            next_row = df.iloc[i + 1]
            augmented_data.append(base_row)
            for j in range(1, num_synthetic_points + 1):
                synthetic_row = base_row.copy()
                fraction = j / (num_synthetic_points + 1)
                synthetic_row[self.price_columns] = base_row[self.price_columns] * (1 - fraction) + next_row[self.price_colu
                synthetic_row[self.price_columns] += np.random.normal(0, 0.002, len(self.price_columns))  # Add small noise
                augmented_data.append(synthetic_row)
            augmented_data.append(df.iloc[-1])
        return pd.DataFrame(augmented_data).reset_index(drop=True)

    def __init__(self, data, initial_cash=1000000, seed=None):
        super(FixedIncomeEnv, self).__init__()

        self.seed(seed)
        self.data = data.reset_index(drop=True)
        self.selected_assets = ["912810FB", "912810FE", "912810PU"]  # Seleccionar solo 3 activos
        excluded_vars = ["CPN", "ISSUE", "MATURITY"]  # Variables constantes a excluir
        self.price_columns = [
            col for col in self.data.columns
            if any(asset in col for asset in self.selected_assets)
            and not any(excl in col.split("_") for excl in excluded_vars)  # Check all parts of the column name
        ]
        self.data = self.data[["Date"] + [col for col in self.data.columns if any(asset in col for asset in self.selected_as
        #self.data = self.augment_dataset(self.data)  # Apply augmentation, de momento no hacemos esto.
        self.n_bonds = len(self.selected_assets) # Número de bonos seleccionados
        self.data = self.data[self.price_columns]

        self.px_last_columns = [col for col in self.price_columns if "PX_LAST" in col]
        for col in self.px_last_columns:
            self.data[f"{col}_LAST"] = self.data[col].shift(1, fill_value=self.data[col].iloc[0])

        self.current_step = 0
        self.initial_cash = initial_cash
        self.cash = initial_cash
        self.holdings = np.zeros(self.n_bonds)  # Posición en cada bono

        # Espacio de observación (todas las variables excepto la fecha)
        self.price_columns.extend([f"{col}_LAST" for col in self.px_last_columns])
        self.observation_space = spaces.Box(low=-np.inf, high=np.inf, shape=(self.data.shape[1],), dtype=np.float32)

        # Espacio de acción: Buy, Hold, Sell para cada bono
        self.action_space = spaces.MultiDiscrete([3] * self.n_bonds)

    def reset(self):
        self.current_step = 0
        self.cash = self.initial_cash
        self.holdings = np.zeros(self.n_bonds)
        return self._next_observation()

    def _next_observation(self):
        obs = self.data.iloc[self.current_step, :].values
        obs = np.nan_to_num(obs, nan=0.0, posinf=0.0, neginf=0.0).astype(np.float32)  # Eliminar NaNs e infinitos
        return obs
```

```
    def step(self, action):
        prev_portfolio_value = self._get_portfolio_value()

        # Procesar acciones
        for i, act in enumerate(action):
            bond_price = self.data.iloc[self.current_step][self.price_columns[i]]  # Precio del bono
            if act == 0:  # BUY
                if self.cash >= bond_price:
                    self.holdings[i] += 1
                    self.cash -= bond_price
            elif act == 2:  # SELL
                if self.holdings[i] > 0:
                    self.holdings[i] -= 1
                    self.cash += bond_price

        self.current_step += 1

        # Calcular nueva recompensa
        new_portfolio_value = self._get_portfolio_value()
        reward = new_portfolio_value - prev_portfolio_value

        done = self.current_step >= self.data.shape[0] - 1
        return self._next_observation(), reward, done, {}

    def _get_portfolio_value(self):
        bond_values = self.holdings * self.data.iloc[self.current_step][self.price_columns[0:len(self.holdings)]].values
        return np.sum(bond_values) + self.cash

    def render(self):
        print(f"Step: {self.current_step}, Cash: {self.cash}, Holdings: {self.holdings}, Portfolio Value: {self._get_portfol

def lin_schedule(initial_value):
    def func(progress_remaining):
        return progress_remaining * initial_value
    return func

# Entrenar 2 agentes PPO
#def train_ppo_agents(env, num_agents=2, timesteps=100000):
def train_ppo_agents(env, logger, num_agents=2, timesteps=1000000):
    models = []
    log = True
    for i in range(num_agents):
        model = PPO("MlpPolicy", env, verbose=1, ent_coef=0.02, target_kl=0.05, learning_rate=lin_schedule(2.5e-4), seed=i)
        if log: #Just log the first agent.
            model.set_logger(logger)  # Attach Logger
            log = False
        model.learn(total_timesteps=timesteps)
        models.append(model)
    return models

# Implementar 5 benchmarks aleatorios
def random_policy(env, num_runs=5):
    results = []
    for _ in range(num_runs):
        env.reset()
        done = False
        total_return = 0
        while not done:
            action = env.action_space.sample()  # Política aleatoria
            _, reward, done, _ = env.step(action)
            total_return += reward
        results.append(total_return)
    return results

# Implementar benchmark de política uniforme (HOLD)
def uniform_policy(env):
    env.reset()
    done = False
    total_return = 0
    uniform_action = [1] * env.n_bonds  # Siempre mantener posiciones equilibradas
    while not done:
        _, reward, done, _ = env.step(uniform_action)
        total_return += reward
    return total_return
```

```python
# Función para evaluar agentes PPO
def evaluate_drl_agents(agents, env, num_runs=1):
    returns = []
    for agent in agents:
        for _ in range(num_runs):
            obs = env.reset()
            done = False
            total_return = 0
            while not done:
                action, _ = agent.predict(obs)
                obs, reward, done, _ = env.step(action)
                total_return += reward
            returns.append(total_return)
    return returns

# Convertir fechas a valores numéricos
def process_dates(df):
    for col in df.columns:
        if "ISSUE_DT" in col or "MATURITY" in col:
            df[col] = pd.to_datetime(df[col], errors='coerce')
            df[col] = (df[col] - pd.Timestamp("1970-01-01")).dt.days.fillna(0)
    return df

# Cargar y dividir los datos en train y test
df = pd.read_excel("dataset.xlsx")
ultimo_anio = df["Date"].max().year
fecha_corte = pd.Timestamp(year=ultimo_anio, month=1, day=1)
df_train = df[df["Date"] < fecha_corte].reset_index(drop=True)
df_eval = df[df["Date"] >= fecha_corte].reset_index(drop=True)
#df_train = process_dates(df_train)
#df_eval = process_dates(df_eval)

# Crear el entorno con los datos de entrenamiento y evaluación
env_train = FixedIncomeEnv(df_train, seed=0)
env_eval = FixedIncomeEnv(df_eval, seed=0)

# Configurar el logger para rastrear métricas de entrenamiento
log_dir = "./ppo_logs"
logger = configure(log_dir, ["stdout", "csv", "tensorboard"])

# Entrenar 2 agentes PPO
#timesteps = 100000
timesteps = 300000
ppo_agents = train_ppo_agents(env_train, logger, num_agents=2, timesteps=timesteps)

# Evaluar benchmarks
random_results = random_policy(env_eval)
uniform_result = uniform_policy(env_eval)

# Evaluar agentes PPO en el entorno de evaluación
ppo_returns = evaluate_drl_agents(ppo_agents, env_eval)

# Prueba de hipótesis: Comparar medias entre PPO y Benchmarks
t_stat_ppo_random, p_value_ppo_random = stats.ttest_ind(ppo_returns, random_results, equal_var=False)
t_stat_ppo_uniform, p_value_ppo_uniform = stats.ttest_1samp(ppo_returns, [uniform_result])

'''
t_stat_ppo_random, p_value_ppo_random = stats.ttest_ind(ppo_returns, random_results, equal_var=False)
t_stat_ppo_uniform, p_value_ppo_uniform = stats.ttest_1samp(ppo_returns, [uniform_result])
'''

# Presentar los resultados en DataFrame
comparison_results = pd.DataFrame({
    "Strategy": ["PPO", "Random", "Uniform"],
    "Accumulated Return": [np.mean(ppo_returns), np.mean(random_results), uniform_result],
    "T-test vs PPO": [None, p_value_ppo_random, p_value_ppo_uniform]
})

print(comparison_results)

import pandas as pd
import matplotlib
matplotlib.use("Agg")  # Use a non-interactive backend
import matplotlib.pyplot as plt

# Function to monitor and plot training metrics
def plot_training_metrics(log_path="./ppo_logs/progress.csv", output_path="ppo_training_metrics.png"):
    df_logs = pd.read_csv(log_path)
```

```python
# Function to monitor and plot training metrics
def plot_training_metrics(log_path="./ppo_logs/progress.csv", output_path="ppo_training_metrics.png"):
    df_logs = pd.read_csv(log_path)

    plt.figure(figsize=(12, 8))

    plt.subplot(4, 1, 1)
    plt.plot(df_logs["time/total_timesteps"], df_logs["train/approx_kl"], label="KL Divergence", color="red")
    plt.xlabel("Timesteps")
    plt.ylabel("KL Divergence")
    plt.legend()
    plt.grid()

    plt.subplot(4, 1, 2)
    plt.plot(df_logs["time/total_timesteps"], df_logs["train/explained_variance"], label="Explained Variance", color="blue")
    plt.xlabel("Timesteps")
    plt.ylabel("Explained Variance")
    plt.legend()
    plt.grid()

    plt.subplot(4, 1, 3)
    plt.plot(df_logs["time/total_timesteps"], df_logs["train/loss"], label="Loss", color="purple")
    plt.xlabel("Timesteps")
    plt.ylabel("Loss")
    plt.legend()
    plt.grid()

    plt.subplot(4, 1, 4)
    plt.plot(df_logs["time/total_timesteps"], df_logs["train/clip_fraction"], label="Clip Fraction", color="brown")
    plt.xlabel("Timesteps")
    plt.ylabel("Clip Fraction")
    plt.legend()
    plt.grid()

    plt.tight_layout()
    plt.savefig(output_path)
    plt.close()
    print(f"Training metrics saved to {output_path}")

plot_training_metrics()
```

Extract from the employed database

Date	912810FB_PX_LAST	912810FE_PX_LAST	912833XN_PX_LAST	9128335A_PX_LAST	9128337E_PX_LAST	912810PU_PX_LAST	912810PW_PX_LAST	912810PX_PX_LAST	912810QA_PX_LAST	912810QB_PX_LAST
2017-01-02 00:00:00	134,1328125	129,3671875	71,422	57,134	55,137	134,1015625	124,078125	126,09375	109,1875	121,25
2017-01-03 00:00:00	134,140625	129,3984375	71,547	57,212	55,2	134,234375	124,25	126,296875	109,3828125	121,4765625
2017-01-04 00:00:00	134,2109375	129,5078125	71,617	57,359	55,376	134,4453125	124,5234375	126,59375	109,6171875	121,7578125
2017-01-05 00:00:00	135,3125	130,640625	72,474	58,371	56,412	136,203125	126,265625	128,359375	111,265625	123,546875
2017-01-06 00:00:00	134,4765625	129,8046875	71,899	57,831	55,878	135,140625	125,234375	127,2890625	110,2734375	122,4453125
2017-01-09 00:00:00	135,1015625	130,4609375	72,439	58,449	56,495	136,1484375	126,1953125	128,2734375	111,1953125	123,453125
2017-01-10 00:00:00	135,0234375	130,359375	72,41	58,335	56,374	135,9375	126,015625	128,0859375	111	123,21875
2017-01-11 00:00:00	135,0859375	130,4296875	72,449	58,442	56,493	136,109375	126,1875	128,28125	111,1875	123,4140625
2017-01-12 00:00:00	135,1640625	130,5078125	72,532	58,307	56,334	135,9375	126,0625	128,109375	111,0390625	123,25
2017-01-13 00:00:00	134,796875	130,15625	72,286	57,996	56,022	135,390625	125,5234375	127,5703125	110,5234375	122,7109375
2017-01-16 00:00:00	134,796875	130,15625	72,286	57,996	56,022	135,390625	125,5234375	127,5703125	110,5234375	122,7109375
2017-01-17 00:00:00	135,5625	130,9453125	72,923	58,592	56,64	136,5	126,625	128,6875	111,546875	123,8125
2017-01-18 00:00:00	134,3984375	129,7578125	72,004	57,692	55,75	134,8359375	125,0078125	127,046875	110,046875	122,21875
2017-01-19 00:00:00	133,8984375	129,2421875	71,616	57,36	55,401	134,2265625	124,40625	126,4375	109,484375	121,59375
2017-01-20 00:00:00	133,9375	129,2890625	71,684	57,275	55,33	134,1015625	124,3125	126,3203125	109,3984375	121,5
2017-01-23 00:00:00	134,7109375	130,0703125	72,207	57,959	56,026	135,3203125	125,4921875	127,546875	110,53125	122,703125
2017-01-24 00:00:00	133,9609375	129,2578125	71,626	57,331	55,365	134,1484375	124,3359375	126,375	109,46875	121,5546875
2017-01-25 00:00:00	133,4296875	128,7265625	71,241	56,752	54,755	133,1328125	123,3671875	125,3515625	108,53125	120,5234375
2017-01-26 00:00:00	133,5234375	128,8203125	71,289	56,892	54,886	133,34375	123,578125	125,5625	108,7578125	120,75
2017-01-27 00:00:00	133,7421875	129,0390625	71,5	57,198	55,17	133,7578125	123,984375	125,9921875	109,125	121,171875
2017-01-30 00:00:00	133,6953125	129,0078125	71,484	57,035	54,978	133,4921875	123,703125	125,703125	108,859375	120,875
2017-01-31 00:00:00	134,0859375	129,40625	71,8	57,285	55,216	133,9296875	124,1484375	126,1484375	109,265625	121,296875
2017-02-01 00:00:00	133,9453125	129,2578125	71,699	57,269	55,231	133,8828125	124,09375	126,078125	109,171875	121,203125
2017-02-02 00:00:00	133,9375	129,1953125	71,645	57,201	55,13	133,7578125	123,96875	125,9453125	109,0546875	121,0703125
2017-02-03 00:00:00	134,015625	129,265625	71,708	57,147	55,143	133,7421875	123,953125	125,9140625	109,03125	121,046875
2017-02-06 00:00:00	134,6640625	129,9296875	72,174	57,696	55,692	134,7109375	124,8828125	126,8671875	109,8828125	122,0078125
2017-02-07 00:00:00	134,8203125	130,125	72,372	58,002	56,008	135,1796875	125,359375	127,375	110,359375	122,53125
2017-02-08 00:00:00	135,4765625	130,8203125	72,924	58,742	56,711	136,390625	126,5390625	128,5546875	111,5	123,78125
2017-02-09 00:00:00	134,6796875	130,0078125	72,365	58,128	56,135	135,265625	125,46875	127,546875	110,5078125	122,671875
2017-02-10 00:00:00	134,53125	129,84375	72,236	58,025	55,998	135,015625	125,234375	127,265625	110,296875	122,4453125
2017-02-13 00:00:00	134,1953125	129,5078125	71,989	57,671	55,638	134,421875	124,671875	126,6953125	109,75	121,890625
2017-02-14 00:00:00	133,8203125	129,1171875	71,701	57,346	55,344	133,828125	124,1015625	126,125	109,2421875	121,3125
2017-02-15 00:00:00	133,546875	128,8515625	71,512	57,079	55,062	133,3359375	123,6171875	125,6328125	108,8046875	120,828125
2017-02-16 00:00:00	134,078125	129,359375	71,895	57,428	55,364	133,921875	124,1953125	126,2109375	109,328125	121,4140625
2017-02-17 00:00:00	134,3984375	129,71875	72,223	57,715	55,675	134,4609375	124,71875	126,7578125	109,8125	121,9375
2017-02-20 00:00:00	134,3984375	129,71875	72,223	57,715	55,675	134,4609375	124,71875	126,7578125	109,8125	121,9375
2017-02-21 00:00:00	134,25	129,5390625	72,118	57,63	55,557	134,2109375	124,4765625	126,484375	109,5625	121,6640625
2017-02-22 00:00:00	134,4765625	129,765625	72,234	57,709	55,647	134,3671875	124,6484375	126,671875	109,734375	121,8359375
2017-02-23 00:00:00	134,9296875	130,21875	72,565	57,966	55,909	134,90625	125,1484375	127,1796875	110,1640625	122,3046875
2017-02-24 00:00:00	135,59375	130,8984375	73,105	58,579	56,543	135,9453125	126,1953125	128,21875	111,171875	123,375
2017-02-27 00:00:00	135,015625	130,328125	72,68	58,339	56,315	135,421875	125,6640625	127,6640625	110,640625	122,8203125
2017-02-28 00:00:00	134,7421875	130,03125	72,523	58,132	56,168	135,078125	125,34375	127,3671875	110,375	122,5234375
2017-03-01 00:00:00	134,0078125	129,2734375	71,957	57,391	55,403	133,7734375	124,09375	126,0859375	109,1796875	121,234375
2017-03-02 00:00:00	133,7265625	128,9921875	71,785	57,234	55,233	133,4296875	123,765625	125,7578125	108,8984375	120,9375
2017-03-03 00:00:00	133,7265625	129,0078125	71,861	57,421	55,4	133,609375	123,921875	125,90625	109,0390625	121,078125
2017-03-06 00:00:00	133,46875	128,734375	71,655	57,139	55,077	133,078125	123,4375	125,3984375	108,5859375	120,6015625
2017-03-07 00:00:00	133,2578125	128,5390625	71,496	57,025	54,955	132,84375	123,1953125	125,1484375	108,3203125	120,2890625
2017-03-08 00:00:00	132,8046875	128,0546875	71,14	56,637	54,53	132,1171875	122,546875	124,46875	107,703125	119,640625
2017-03-09 00:00:00	132,2734375	127,5	70,749	56,044	53,928	131,078125	121,546875	123,4921875	106,8203125	118,7109375
2017-03-10 00:00:00	132,6015625	127,8046875	70,983	56,145	54,02	131,28125	121,765625	123,7109375	107,046875	118,9453125
2017-03-13 00:00:00	132,0390625	127,2421875	70,56	55,724	53,602	130,4140625	120,9375	122,890625	106,3203125	118,140625
2017-03-14 00:00:00	132,34375	127,5625	70,819	56,099	54,039	131,1015625	121,609375	123,5625	106,9453125	118,8125
2017-03-15 00:00:00	133,4609375	128,7265625	71,688	56,949	54,84	132,5859375	123,03125	125	108,2890625	120,21875
2017-03-16 00:00:00	132,953125	128,1796875	71,252	56,45	54,427	131,8203125	122,296875	124,2734375	107,5703125	119,46875
2017-03-17 00:00:00	133,3671875	128,640625	71,605	56,969	54,99	132,6640625	123,1015625	125,1015625	108,3515625	120,3046875
2017-03-20 00:00:00	133,8046875	129,0859375	72,011	57,367	55,364	133,3125	123,7265625	125,703125	108,9375	120,921875
2017-03-21 00:00:00	134,28125	129,5859375	72,406	57,837	55,815	134,140625	124,5	126,515625	109,703125	121,7421875
2017-03-22 00:00:00	134,4296875	129,7421875	72,548	58,067	56,012	134,4921875	124,8671875	126,8828125	110,03125	122,09375
2017-03-23 00:00:00	134,28125	129,59375	72,447	58,009	56,011	134,3671875	124,6875	126,734375	109,875	121,9296875
2017-03-24 00:00:00	134,359375	129,6796875	72,582	58,197	56,272	134,625	124,9609375	126,9921875	110,1328125	122,203125
2017-03-27 00:00:00	134,734375	130,0625	72,89	58,455	56,486	135,1015625	125,4140625	127,4375	110,59375	122,6875
2017-03-28 00:00:00	134,2734375	129,6015625	72,55	58,038	56,061	134,3359375	124,65625	126,703125	109,90625	121,953125
2017-03-29 00:00:00	134,7578125	130,0703125	72,88	58,433	56,452	135,0078125	125,3125	127,375	110,5390625	122,6328125
2017-03-30 00:00:00	134,3125	129,6015625	72,531	57,979	55,953	134,1796875	124,5078125	126,546875	109,78125	121,828125
2017-03-31 00:00:00	134,6796875	129,984375	72,897	58,299	56,293	134,6796875	124,9765625	127,046875	110,234375	122,3046875
2017-04-03 00:00:00	135,4140625	130,7421875	73,446	58,977	56,995	135,859375	126,109375	128,1953125	111,3125	123,453125
2017-04-04 00:00:00	134,96875	130,3203125	73,134	58,62	56,572	135,109375	125,3671875	127,421875	110,5625	122,65625
2017-04-05 00:00:00	135,2421875	130,6015625	73,341	58,843	56,733	135,4375	125,671875	127,7109375	110,84375	122,9609375
2017-04-06 00:00:00	135,1796875	130,5546875	73,325	58,762	56,651	135,2890625	125,5546875	127,5859375	110,75	122,8671875
2017-04-07 00:00:00	134,7265625	130,1484375	73,071	58,633	56,573	134,921875	125,171875	127,203125	110,3984375	122,46875
2017-04-10 00:00:00	134,8984375	130,3203125	73,247	58,717	56,767	135,2265625	125,4921875	127,546875	110,6953125	122,8125
2017-04-11 00:00:00	135,65625	131,1015625	73,82	59,363	57,348	136,2890625	126,5546875	128,625	111,7265625	123,921875
2017-04-12 00:00:00	136,2734375	131,75	74,302	59,853	57,847	137,1171875	127,359375	129,453125	112,4765625	124,7578125
2017-04-13 00:00:00	136,2734375	131,7421875	74,348	59,844	57,782	137,0390625	127,2578125	129,3671875	112,421875	124,640625
2017-04-14 00:00:00	136,2734375	131,7421875	74,348	59,844	57,782	137,0390625	127,2578125	129,3671875	112,421875	124,640625
2017-04-17 00:00:00	136,09375	131,546875	74,223	59,692	57,622	136,7578125	127	129,0703125	112,140625	124,359375
2017-04-18 00:00:00	137,0078125	132,484375	74,934	60,5	58,517	138,2578125	128,453125	130,546875	113,4921875	125,828125
2017-04-19 00:00:00	136,4921875	131,9765625	74,546	60,124	58,144	137,5703125	127,7734375	129,8671875	112,84375	125,109375
2017-04-20 00:00:00	136,28125	131,7734375	74,425	60,074	58,093	137,4140625	127,609375	129,6953125	112,6875	124,953125
2017-04-21 00:00:00	136,1015625	131,5703125	74,306	59,936	58,018	137,109375	127,328125	129,390625	112,421875	124,6796875
2017-04-24 00:00:00	135,7890625	131,2578125	74,1	59,622	57,67	136,53125	126,7578125	128,828125	111,9375	124,125
2017-04-25 00:00:00	135,125	130,59375	73,597	59,049	57,074	135,4609375	125,734375	127,78125	110,9453125	123,046875
2017-04-26 00:00:00	135,453125	130,9140625	73,899	59,411	57,416	136,015625	126,25	128,3125	111,4375	123,5625