LUCAS GABRIEL BASSAN DE MORAES

Precificação de Ativos utilizando Cadeias de Markov e Aprendizagem por Reforço: Uma Possível Abordagem para Previsão Financeira

São Carlos

2025

LUCAS GABRIEL BASSAN DE MORAES

Precificação de Ativos utilizando Cadeias de Markov e Aprendizagem por Reforço: Uma Possível Abordagem para Previsão Financeira

Trabalho Final de Curso apresentado ao Departamento de Física da Universidade Federal de São Carlos como requisito para a obtenção do título de Bacharel em Engenharia Física.

Orientador:

Prof. Dr. Claudio Antonio Cardoso

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Física

São Carlos

2025

Moraes, Lucas Gabriel Bassan

Precificação de ativos utilizando cadeias de Markov e aprendizagem por reforço: uma possível abordagem para previsão financeira / Lucas Gabriel Bassan Moraes -- 2025.
53f.

TCC (Graduação) - Universidade Federal de São Carlos, campus São Carlos, São Carlos
Orientador (a): Claudio Antonio Cardoso
Banca Examinadora: Claudio Antonio Cardoso, Pedro Augusto Franco Pinheiro Moreira, Leonardo Kleber Castelano
Bibliografia

1. Cadeias de Markov. 2. Aprendizado por reforço. 3. Previsão Financeira. I. Moraes, Lucas Gabriel Bassan. II. Título.

Ficha catalográfica desenvolvida pela Secretaria Geral de Informática (SIn)

DADOS FORNECIDOS PELO AUTOR

Bibliotecário responsável: Arildo Martins - CRB/8 7180

LUCAS GABRIEL BASSAN DE MORAES

PRECIFICAÇÃO DE ATIVOS UTILIZANDO CADEIAS DE MARKOV E APRENDIZAGEM POR REFORÇO: UMA POSSÍVEL ABORDAGEM PARA PREVISÃO FINANCEIRA

Trabalho Final de Curso apresentado ao Departamento de Física da Universidade Federal de São Carlos como requisito para a obtenção do título de Bacharel em Engenharia Física.

Banca Examinadora:

Prof. Dr. Claudio Antonio Cardoso
Universidade Federal de São Carlos - Departamento de Física

Prof. Dr. Pedro Augusto Franco Pinheiro Moreira
Universidade Federal de São Carlos - Departamento de Física

Prof. Dr. Leonardo Kleber Castelano
Universidade Federal de São Carlos - Departamento de Física

Dedico este trabalho aos meus pais, Glaucia Bassan e Rafael Moraes, cuja paciência e apoio tornaram possível cada passo desta jornada acadêmica.

Agradecimentos

Após anos de esforço e dedicação neste curso, é difícil encontrar palavras que expressem o quanto é emocionante estar concluindo este importante ciclo da minha vida. Como forma de gratidão, quero registrar meu agradecimento a algumas pessoas que foram essenciais para que eu chegasse até aqui, de cabeça erguida.

Primeiramente, quero dedicar de forma profunda este trabalho e tambem toda a minha jornada até este momento à minha família, especialmente ao meu pai, Rafael, e à minha mãe, Glaucia. Eles estiveram presentes em todas as minhas conquistas e dificuldades desde sempre, oferecendo apoio, incentivo e amor incondicional, que foram fundamentais para que eu alcançasse este objetivo. Claro que tambem não poderia deixar de agradecer aos meus amigos Matheus, José e Davi, que estiveram ao meu lado durante toda a graduação. Suas companhias tornou os momentos difíceis mais leves e os bons momentos ainda mais divertidos.

Expresso ainda minha sincera gratidão ao meu orientador, professor Cláudio Cardoso, pela oportunidade de realizar este trabalho, pelo apoio na escolha do tema e pela orientação cuidadosa durante todo o desenvolvimento da pesquisa. Agradeço, sobretudo, por acreditar no meu esforço e na concretização deste projeto.

Por fim, agradeço especialmente à minha namorada e parceira Maiza, que tem sido meu porto seguro ao longo de todo o desenvolvimento deste trabalho. Sua presença, paciência, carinho e inspiração foram fundamentais e possibilitaram que eu me mantivesse firme e desse o meu melhor para a construção desta pesquisa.

"Quando tudo tiver parecendo ir contra você,
lembre-se que o avião decola contra o vento,
e não a favor dele."
(Henry Ford)

Resumo

As cadeias de Markov e os métodos de Aprendizado por Reforço têm se destacado como ferramentas promissoras para a modelagem de sistemas estocásticos complexos, especialmente em ambientes financeiros marcados por incertezas e volatilidade. Nesse contexto, este trabalho propõe a aplicação conjunta dessas abordagens na precificação e previsão de ativos financeiros, utilizando como estudo de caso as ações da Petrobras (PETR4) e da Vale (VALE3). O modelo desenvolvido integra a dinâmica probabilística das cadeias de Markov em tempo discreto com a capacidade adaptativa do algoritmo Q-Learning pela aprendizagem por reforço, permitindo ao agente ajustar suas decisões de compra e venda com base nas recompensas acumuladas definidas. A metodologia foi implementada em linguagem Python e avaliada com base na métrica Raiz Quadrada do Erro Médio (RMSE), comparando os resultados simulados com os valores reais de cada ativo em períodos estabelecidos e parâmetros definidos pelo algoritmo. Os resultados obtidos demonstraram que o modelo é capaz de representar de forma consistente a evolução do preço dos ativos, apresentando melhor desempenho em horizontes de treinamento mais curtos, dependendo do ativo. Esse comportamento reforça o potencial da integração entre técnicas estocásticas e aprendizado de máquina para previsão financeira em mercados emergentes.

Palavras-chave: Cadeias de Markov; Aprendizado por Reforço; Q-Learning; Precificação de Ativos; Previsão Financeira.

Abstract

Markov chains and Reinforcement Learning methods have emerged as promising tools for modeling complex stochastic systems, especially in financial environments characterized by uncertainty and volatility. In this context, the present work proposes the joint application of these approaches to the pricing and forecasting of financial assets, using Petrobras (PETR4) and Vale (VALE3) stocks as case studies. The developed model integrates the probabilistic dynamics of discrete-time Markov chains with the adaptive capabilities of the Q-Learning algorithm through reinforcement learning, enabling the agent to adjust its buy and sell decisions based on the defined accumulated rewards. The methodology was implemented in the Python programming language and evaluated using the Root Mean Squared Error (RMSE) metric, comparing the simulated results with the real values of each asset over predefined periods and algorithm-defined parameters. The obtained results demonstrated that the model is capable of consistently representing the evolution of asset prices, showing better performance over shorter training horizons depending on the asset. This behavior reinforces the potential of integrating stochastic techniques and machine learning methods for financial forecasting in emerging markets.

Keywords: Markov Chains; Reinforcement Learning; Q-Learning; Asset Pricing; Forecasting.

Sumário

Lista de Figuras

Lista de Quadros

Lista de Abreviações

MDP	Markov decision process
RL	Reinforcement Learning
RMSE	Root Mean Square Error
MTD	Mixture Transition Distribution

Introdução

As cadeias de Markov, inicialmente estudadas pelo matemático russo Andrei Andreyevich Markov (1856–1922), surgiram como uma forma de descrever fenômenos que envolvem aleatoriedade. Esses modelos se baseiam nos chamados processos estocásticos, que buscam compreender como sistemas sujeitos ao acaso evoluem ao longo do tempo. O interesse de Markov por esses sistemas surgiu por volta de 1900. Na época ele estudava compreender a probabilidade de ocorrência de uma consoante em determinada posição de uma palavra — considerando apenas se a letra anterior era vogal ou consoante. Esse tipo de abordagem, em que o estado futuro depende apenas do estado atual, caracteriza processos markovianos (HASHIOKA, 2018).

Desde então, as ideias desenvolvidas por Markov e as suas aplicações dos modelos se expandiram amplamente, abrangendo campos como física atômica, teoria quântica, biologia, genética, comportamento social e, de modo especial, a economia e as finanças. O *Markov decision process* (MDP), traduzido como processos de decisão de Markov, vem sendo explorado como uma poderosa ferramenta matemática para resolver problemas de controle e otimização em sistemas complexos (GU; WANG; ZHANG, 2025), visto que podem descrever a evolução de sistemas sujeitos a incertezas, permitindo a modelagem probabilística de fenômenos dinâmicos.

No universo financeiro, esse processo têm se mostrado particularmente útil para modelar e compreender a evolução temporal de preços de ativos, bem como para apoiar a tomada de decisões estratégicas. Nas últimas décadas, diversos estudos demonstraram a viabilidade de modelar preços de ações e séries temporais financeiras discretas com cadeias de Markov. Tais modelos se demonstram especialmente úteis para capturar a dinâmica de mercados que apresentam alternância de regimes, como períodos de alta, baixa ou estabilidade, oferecendo um arcabouço conceitual que alia simplicidade e capacidade preditiva (GHEZZI; PICCARDI, 2003).

Com base nisso, este trabalho propõe a utilização de cadeias de Markov em tempo discreto como modelo para previsão de preços de ativos financeiros, tomando como estudo de caso as ações da Vale S.A. e da Petrobras S.A.. A escolha desses dois ativos de grande representatividade no mercado brasileiro busca avaliar a aplicabilidade e o desempenho do modelo em um mercado emergente, caracterizado por alta volatilidade, incertezas

macroeconômicas e assimetrias informacionais, condições que tornam a previsão de preços uma tarefa especialmente desafiadora.

Entretanto, embora as cadeias de Markov sejam eficazes para descrever a dinâmica probabilística de um sistema, elas não possuem um mecanismo intrínseco de tomada de decisão adaptativa. Em outras palavras, o modelo markoviano é capaz de estimar as probabilidades de transição entre estados, mas não de determinar quais ações devem ser executadas diante dessas mudanças para maximizar um resultado esperado. Essa limitação motiva a integração do modelo de Markov com abordagens mais recentes, capazes de incorporar aprendizado e adaptação em tempo real.

Nesse contexto, torna-se valioso o estudo de uma extensão moderna ao da modelagem puramente markoviana, combinando-a com métodos baseados em aprendizado por reforço (*Reinforcement Learning* – RL). O RL, um ramo da aprendizagem de máquina, utiliza o formalismo do MDP para modelar a interação entre um agente e um ambiente, permitindo o desenvolvimento de políticas adaptativas de decisão baseadas em recompensas (SUTTON; BARTO, 2018). A combinação entre Cadeias de Markov e Aprendizado por Reforço representa, portanto, uma estratégia híbrida capaz de unir robustez estatística e capacidade de aprendizado dinâmico, ampliando o potencial de previsão e análise de séries financeiras (SUTTON; BARTO, 2018).

1 Fundamentos Teóricos

1.1 Processos Estocásticos

Em diversas situações da vida real, os sistemas frequentemente operam sob condições de incerteza, influenciados por inúmeras variáveis aleatórias, eventos imprevisíveis ligados ao comportamento humano, fatores ambientais e possíveis falhas técnicas. Embora modelos determinísticos sejam úteis para obter uma compreensão inicial dos sistemas dinâmicos, eles acabam sendo insuficientes diante da complexidade e da aleatoriedade observadas em cenários reais. Nesse contexto, os processos estocásticos se destacam como ferramentas matemáticas fundamentais, permitindo modelar e analisar a evolução temporal de sistemas sujeitos à incerteza. Essas abordagens são amplamente aplicadas em áreas como economia, engenharia e finanças, onde a compreensão dos riscos e das flutuações do mercado, são essenciais para a tomada de decisões fundamentadas (DOOB, 1990).

Os processos estocásticos são definidos formalmente como uma coleção de variáveis aleatórias $\{X(t),\ t \in T\}$, classificadas geralmente por uma variável como a de tempo t, onde T representa o conjunto de instantes em que o sistema é observado (podendo ser discreto ou contínuo). Cada variável aleatória $X(t)$ representa o estado do sistema no tempo t, e o conjunto de todos os possíveis valores que $X(t)$ pode assumir é chamado de *espaço de estados* (ALVES; DELGADO, 1997).

Segundo Alves e Delgado (1997),

> "Estabelecendo o paralelismo com o caso determinístico, onde uma função $f(t)$ toma valores bem definidos ao longo do tempo, um processo estocástico toma valores aleatórios ao longo do tempo. Aos valores que $X(t)$ pode assumir chamam-se estados e ao seu conjunto X espaço de estados."

Podem-se citar como exemplos típicos de processos estocásticos:

- O estado de uma máquina industrial (ligada/desligada) no instante t;
- O número de clientes em uma loja a cada dia;
- O número de componentes defeituosos ao fim de um turno de produção;
- A cotação de uma ação no fechamento do mercado;

- O nível de estoque de um produto ao fim de cada semana.

Nesses exemplos, observa-se que tanto o tempo quanto o espaço de estados podem assumir diferentes naturezas. Sendo assim, pode-se dizer que a variável aleatória X é definida como uma função que associa a cada resultado de um experimento aleatório um número real. Essa definição fornece a base para quantificar a incerteza e permite a modelagem matemática de fenômenos aleatórios.

O conjunto definido pelo espaço de estados e o tempo correspondente, representa o conjunto de todos os valores que a variável aleatória $X(t)$ pode assumir ao longo do tempo. Esse espaço pode ser classificado como:

- **Discreto**, quando é formado por valores distintos e enumeráveis, como o número de clientes em uma fila ou o total de componentes defeituosos em uma linha de produção;
- **Contínuo**, quando os valores pertencem a um intervalo contínuo de números reais, como a temperatura em um dado instante ou velocidade durante um percurso.

A distinção entre espaços de estados discretos e contínuos é fundamental, pois influencia diretamente a escolha dos modelos matemáticos e das ferramentas estatísticas utilizadas na análise dos processos já definidos. Modelos com espaço discreto frequentemente empregam *cadeias de Markov* ou distribuições como a *binomial* e a de *Poisson*, enquanto modelos com espaço contínuo utilizam processos como o *movimento browniano* ou *equações diferenciais estocásticas*.

A utilização de processos estocásticos na modelagem de sistemas incertos possibilita não apenas a descrição de sua evolução ao longo do tempo, mas também a previsão de comportamentos futuros com base em distribuições de probabilidade. Essa abordagem é particularmente valiosa em áreas nas quais decisões estratégicas dependem fortemente da análise quantitativa da incerteza (ALVES; DELGADO, 1997).

1.1.1 Comportamento Estacionário

Um importante comportamento a ser estudado dentro dos processos estocásticos, devido ao escopo desse trabalho, é o chamado *estacionário*, que descreve quando seu comportamento probabilístico não depende do tempo. Em outras palavras, diz-se que o

processo é estacionário se a função de distribuição das variáveis aleatórias $X(t)$ que o compõem permanece inalterada ao longo do tempo.

Partindo da definição sobre comportamento estacionário, temos que se o processo for desse tipo e possuir a chamada propriedade de Markov, também conhecida como "propriedade da memória curta", ele será classificado como markoviano, ou como um processo de Markov. Essa propriedade implica que o comportamento futuro do processo depende apenas do estado atual, sendo indiferente ao histórico completo de estados anteriores (ALVES; DELGADO, 1997).

1.2 Processos de Markov

As cadeias de Markov podem ser representadas em dois tipos principais: as cadeias em tempo discreto e as cadeias em tempo contínuo. Ambas compartilham a propriedade markoviana, isto é, a dependência exclusiva do estado atual para a determinação do próximo estado do processo. Porém, se diferem quanto ao tempo em que está sendo tratado no processo estocástico, de maneira discreta, por etapas sucessivas, ou de forma contínua, ao longo de um intervalo real.

As modelagens como as de tempo contínuo são fundamentais para o uso de princípios como o de Black-Scholes (movimento browniano geométrico), verificando o comportamento intradiário de ativos. No entanto, neste trabalho, considerando a análise diária em que serão tratadas as modelagens, será adotada a abordagem em tempo discreto (ALVES; DELGADO, 1997)

O uso desse tipo de cadeia tem se mostrado como ferramenta eficaz na modelagem de preços de ativos financeiros, como ações, ao permitirem a descrição probabilística das transições entre diferentes estados de crescimento de dividendos ou variações de preços. A tese de Riccardo De Blasis da Universidade de Wollongong demonstra aplicações dessa abordagem na avaliação de ações e na descoberta de preços, utilizando tanto modelos univariados quanto multivariados. Em seu estudo, o autor mostra como a discretização de séries temporais de retornos e dividendos pode ser empregada para estimar probabilidades de transição e calcular estimativas robustas do preço fundamental de ativos, bem como métricas de risco associadas. A estrutura em tempo discreto permite, ainda, a aplicação de modelos de cadeia de Markov com Distribuição de Transição Mista (*Mixture Transition Distribution – MTD*), que facilitam a análise de múltiplas séries correlacionadas sem a

explosão no número de parâmetros (BLASIS, 2019).

1.2.1 Cadeias de Markov em tempo discreto

Uma cadeia de Markov em Tempo Discreto é definida pela variável aleatória $X(t)$, para tempos $t = 0, 1, 2, 3, \ldots$, onde $X(0), X(1), X(2), ..., X(n)$ representam os sucessivos sistemas em cada instante discreto no tempo, onde $X(n)$ é a variável aleatória que representa o estado do sistema no n-ésimo tempo (HASHIOKA, 2018). Esse processo satisfaz a propriedade de Markov, ou seja, a probabilidade de transição para o próximo estado depende apenas do estado atual, sendo independente dos estados anteriores.

Essa propriedade pode ser expressa da seguinte forma:

$$\begin{aligned} P_{ij} &= P(X(t+1) = j \mid X(t) = i, X(t-1) = k_{t-1}, \ldots, X(0) = k_0) \\ &= P(X(t+1) = j \mid X(t) = i), \end{aligned} \tag{1.1}$$

$$\forall t = 0, 1, 2, \ldots, \quad \forall i, j, k_0, \ldots, k_{t-1} \in \mathcal{X}.$$

onde $\mathcal{X}$ é o espaço de estados do processo.

Para o estudo das cadeias de Markov em tempo discreto, serão consideradas as seguintes características:

1. O espaço de estados $\mathcal{X}$ é finito ou enumerável (isto é, trata-se de um conjunto de estados discretos);

2. As probabilidades de transição entre estados são constantes ao longo do tempo, caracterizando um processo *estacionário*, Equação 1.1.

Para representar uma Cadeia de Markov pode-se utilizar duas formas: A gráfica, através de um 'diagrama de transições' ou partindo de uma matriz quadrada $\mathbf{P}$, a qual irá conter as probabilidades de transição ao longo de um período de tempo.

Dessa forma, seguindo o modelo de um diagrama de transições, temos que cada estado do processo é representado por um vértice, e as possíveis transições entre estados são indicadas por setas que estão direcionadas entre esses vértices. Cada seta representa uma transição de um estado para outro em um dado instante de tempo e está associada a uma certa probabilidade de ocorrência, que é indicada como um valor atribuído à própria seta.

Outra forma da qual as probabilidades podem ser organizadas é na chamada matriz de transição P, onde cada linha i representa o estado atual do sistema e cada coluna j representa um possível estado futuro. Dessa forma, o elemento p_{ij} da matriz P expressa a probabilidade de o sistema passar do estado i para o estado j em um único passo de tempo (ALVES; DELGADO, 1997).

Essa matriz P é uma matriz estocástica, ou seja, todos os seus elementos são não negativos ($p_{ij} \geq 0$) e, para cada linha i, a soma das probabilidades deve ser igual a 1:

$$\sum_j p_{ij} = 1, \quad \forall i. \tag{1.2}$$

Isso garante que, a partir de qualquer estado atual, o processo transita com certeza para algum estado no próximo período, mesmo que de forma probabilística. De maneira muito ilustrativa, (ALVES; DELGADO, 1997) apresenta um exemplo clássico que ajuda a compreender a essência de uma cadeia de Markov:

> "No pequeno país de Verdscamp'sfloridos, a bebida tradicional, um peculiar licor de flores silvestres, é produzida apenas por duas famílias concorrentes: a família Florescente e a família Petalado. Ao longo das gerações, cada família introduziu diferentes modificações na fórmula original, buscando conferir ao seu licor um sabor especial. Embora as receitas sejam mantidas em absoluto segredo, descobriu-se que uma porção extra de pétalas de malmequer realmente "faz a diferença", garantindo à família Florescente a preferência da maioria dos verdscamp'sfloridenses.
>
> Verificou-se que, para cada pessoa que compra o licor Florescente, há 90% de probabilidade de que a próxima compra também seja desse licor. Já para cada pessoa que adquire o licor Petalado, há 80% de probabilidade de que a inconfundível essência de miosótis o torne tão memorável que o próximo licor escolhido seja novamente o Petalado.
>
> Com base nesses dados, pode-se construir uma cadeia de Markov que represente as transições de preferência entre os dois licores, levando em consideração as probabilidades de fidelização e de mudança entre as marcas."

A partir do problema suposto, vamos dizer que $X(t)$ represente o tipo de licor que será comprado. Sendo assim, criaremos uma cadeia de Markov da qual definirá os seguintes

valores: **F**, para quando o último licor comprado for o florescente, e **P**, para quando o último licor comprado for o petalado.

Dessa forma, é possível representar o diagrama de transições (Figura 1) da seguinte forma.

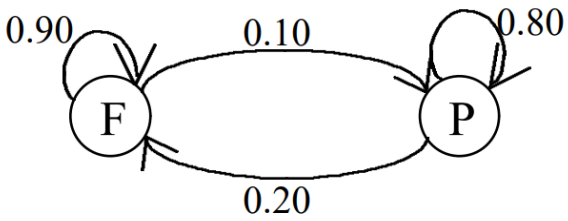


Figura 1 – Representação do diagrama de transições para o entre licores florescente (F) e petalado (P) (ALVES; DELGADO, 1997).

onde, observa-se que a cada vértice representa um estado da cadeia (F ou P), e as setas entre os estados são as probabilidades de transição. Já as setas curvas voltadas para o próprio estado indicam a probabilidade de permanência.

A partir dessa representação gráfica, é possível expressar o modelo de forma matricial, por meio da matriz de transição de estados, conforme apresentado a seguir:

$$\mathbf{P} = \begin{array}{c} \\ (\mathrm{F}) \\ (\mathrm{P}) \end{array} \begin{array}{c} \begin{array}{cc} (\mathrm{F}) & (\mathrm{P}) \end{array} \\ \begin{bmatrix} 0.90 & 0.10 \\ 0.20 & 0.80 \end{bmatrix} \end{array}$$

Do qual, a primeira linha representa as transições do estado F, possuindo 90% de chance de continuar comprando o licor florescente (F $\rightarrow$ F), e 10% de chance de mudar para o licor petalado (F $\rightarrow$ P). Já a segunda linha, representa as transições do estado P, onde possui 20% de chance de mudar para o licor florescente (P $\rightarrow$ F), e 80% de chance de continuar com o licor petalado (P $\rightarrow$ P) (ALVES; DELGADO, 1997).

1.2.2 Probabilidade de transição

A probabilidade de transição, do qual é observada ao analisar o diagrama de transições da Figura 1, é definida como a probabilidade de $X(t+1)$ evoluir do estado $X(t) = i$ para o estado j em um único passo de tempo (HASHIOKA, 2018).

$$P(X(t+1) = j | X(t) = i) = P_{ij}^{t,t+1}. \tag{1.3}$$

1.2.3 Distribuição de Probabilidade

E como forma de modelar e compreender o comportamento de variáveis aleatórias, como preços e retornos de ativos financeiros, é fundamental analisar o comportamento das distribuições de probabilidade. Por natureza, uma distribuição de probabilidade é representada por uma função matemática que atribui probabilidades a diferentes resultados possíveis em um experimento aleatório, permitindo quantificar a incerteza inerente a diversos fenômenos econômicos.

Essas ferramentas são essenciais para a modelagem de preços de ativos, avaliação de riscos e previsão financeira, sendo particularmente úteis no contexto de cadeias de Markov, em que a evolução futura de um ativo depende de estados discretos ou contínuos e de probabilidades de transição entre esses estados.

Dentre as diversas distribuições aplicáveis no conceito estudado, as distribuições *normal* e *log-normal* se destacam por suas características matemáticas, das quais permitem uma modelagem consistente dos retornos e preços dos ativos (STERN et al., 2020).

1.2.3.1 Normal e Log-Normal

A *distribuição normal* ou também conhecida popularmente por Gaussiana, é uma das distribuições contínuas mais conhecidas em estudos probabilísticos. Seu gráfico, como exibido pela Figura 2 é descrito como uma "curva em forma de sino", devido à sua simetria em torno de sua média, sendo assim amplamente utilizado para modelar os log-retornos de ativos financeiros (STERN et al., 2020). Partindo de uma variável aleatória (retorno do ativo) x, uma média da distribuição μ, e o desvio padrão σ, que mede a volatilidade, é possível calcular a densidade de probabilidade:

$$f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{1.4}$$

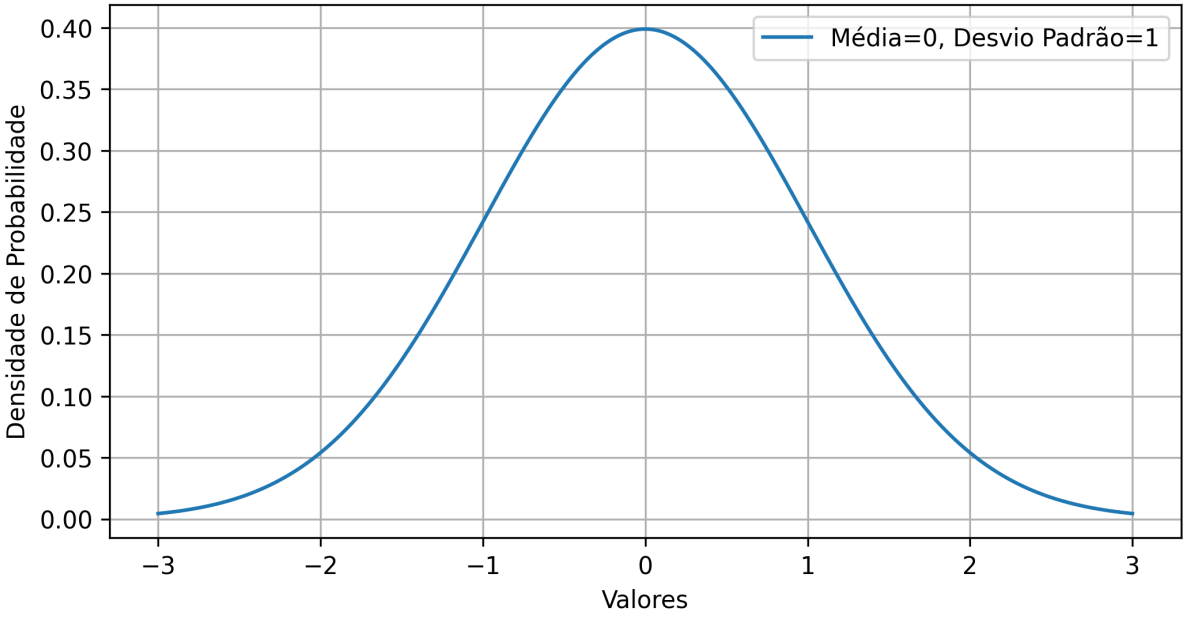


Figura 2 – Exemplo de função de densidade normal. (Autoria própria)

Já a *distribuição log-normal* é definida de forma que, se uma variável aleatória S segue uma distribuição log-normal, então o seu logaritmo natural, $\ln(S)$, segue uma distribuição normal. Essa propriedade a torna uma classe particularmente interessante de distribuição, adequada para modelar preços de ativos financeiros, visto que garante que os valores permaneçam sempre positivos, uma característica realista para preços de ativos. Sua função densidade de probabilidade é dada por:

$$f(s|\mu,\sigma) = \frac{1}{s\sigma\sqrt{2\pi}} \exp\left[-\frac{(\ln s - \mu)^2}{2\sigma^2}\right], \quad s > 0, \tag{1.5}$$

onde:

- s é o preço do ativo,
- μ e σ são os parâmetros da distribuição normal associada ao logaritmo de s.

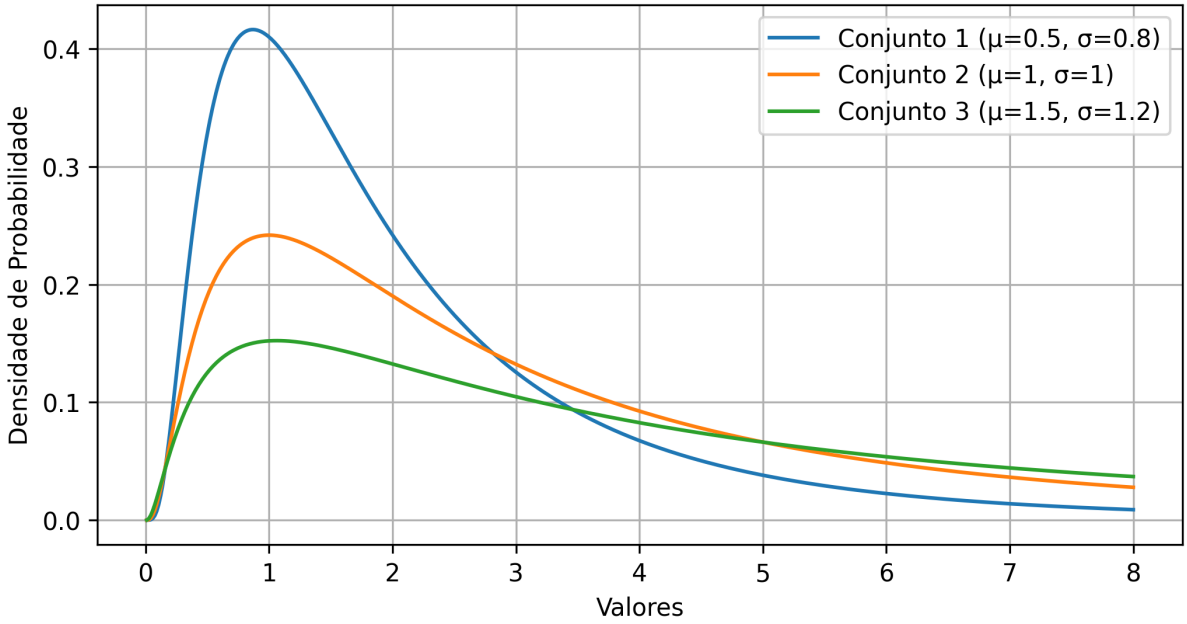


Figura 3 – Exemplo de funções de densidade utilizando variáveis diferentes para uma distribuição log-normal. (Autoria própria)

1.3 Aprendizado por Reforço

O Aprendizado por Reforço (ou *Reinforcement Learning* – RL) consiste em uma abordagem de aprendizado de máquina baseada na interação contínua entre um agente e um ambiente dinâmico definido para análise. Diferente da maneira como funcionam os métodos supervisionados, onde o modelo aprende a partir de pares de entrada e saída previamente rotulados, no RL o processo ocorre por tentativa e erro, sendo guiado por recompensas acumuladas ao longo do tempo. Isso significa, em termos práticos, que o agente observa o estado atual do ambiente, executa uma ação e recebe em retorno um sinal de recompensa, que pode ser positivo ou negativo, como exemplificado na Figura 4. Esse ciclo permite ao agente ajustar gradualmente sua estratégia, ou política, a fim de maximizar o retorno esperado. ((AWS), 2024)

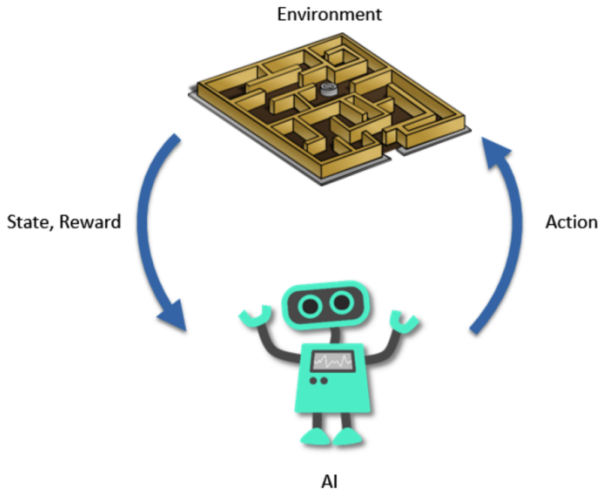


Figura 4 – Modelo de aprendizagem de Reforço (GUSMãO, 2020).

Para o atual contexto, utilizando RL com o Processo de Decisão de Markov (MDP), ocorre uma dinâmica da qual o sistema é caracterizado por estados, ações, probabilidades de transição e recompensas. O objetivo central é encontrar uma política ótima que maximize a soma esperada de recompensas futuras. Essa formulação é particularmente adequada para o mercado financeiro, onde as decisões de investimento frequentemente produzem efeitos diferidos, fenômeno conhecido como atraso de recompensa. (MUREL; KAVLAKOGLU, 2024)

Sendo assim, cada dia de negociação é classificado em estados (baixa, estável ou alta), e o agente deve escolher entre duas possíveis ações: *comprar* (posição *long*) ou *vender* (posição *short*). A recompensa é dada pelo retorno do próximo dia ponderado pela ação tomada: ganhos são positivos quando a decisão está alinhada ao movimento do mercado e negativos quando estão em desacordo. O objetivo do agente, portanto, é aprender uma política de decisão capaz de mapear estados de mercado em ações de compra ou venda para maximizar o retorno acumulado.

A integração entre o modelo markoviano e o aprendizado por reforço representa um avanço significativo em relação ao uso isolado de Cadeias de Markov. Enquanto a cadeia de Markov descreve a dinâmica probabilística dos retornos entre estados, o RL acrescenta uma camada de tomada de decisão adaptativa, permitindo ao agente aprender, a partir da experiência histórica, quais ações financeiras são mais vantajosas em cada cenário de mercado. Dessa forma, a modelagem proposta combina a robustez estatística de Markov com a capacidade de aprendizado dinâmico do RL, produzindo uma ferramenta mais eficaz para simulação e avaliação de estratégias de investimento.

1.3.1 Aplicações de Aprendizagem por Reforço na Precificação de Ativos Financeiros

Conforme descrito pela proposta abordada neste projeto, o uso de métodos de *Reinforcement Learning* (RL) junto ao modelo markoviano traz inúmeros resultados satisfatórios quando introduzidos no contexto da precificação de ativos financeiros, visto que a recompensa associada pode ser modelada pelo retorno obtido ou por métricas ajustadas ao risco (como índice de *Sharpe* ou *drawdown*), ajustando o modelo de previsão quando comparado ao real.

No entanto, o desempenho quanto à tomada de decisão escolhida pelo modelo, ao atribuir uma recompensa durante a previsão, depende fortemente da janela de treinamento utilizada pelo processo. Isso porque, diferentemente de contextos estáticos, o mercado financeiro é marcado por alta volatilidade, mudanças rápidas nas dinâmicas de preços e possíveis quebras de regime. Sendo assim, períodos de treinamento mais longos podem, em princípio, capturar padrões históricos mais robustos, mas também podem introduzir ruídos e informações defasadas, que não refletem as condições recentes do mercado. Por outro lado, períodos curtos permitem que o modelo se adapte melhor a tendências recentes, mas podem levar ao *overfitting* (sobreajuste) e à perda de capacidade de generalização.

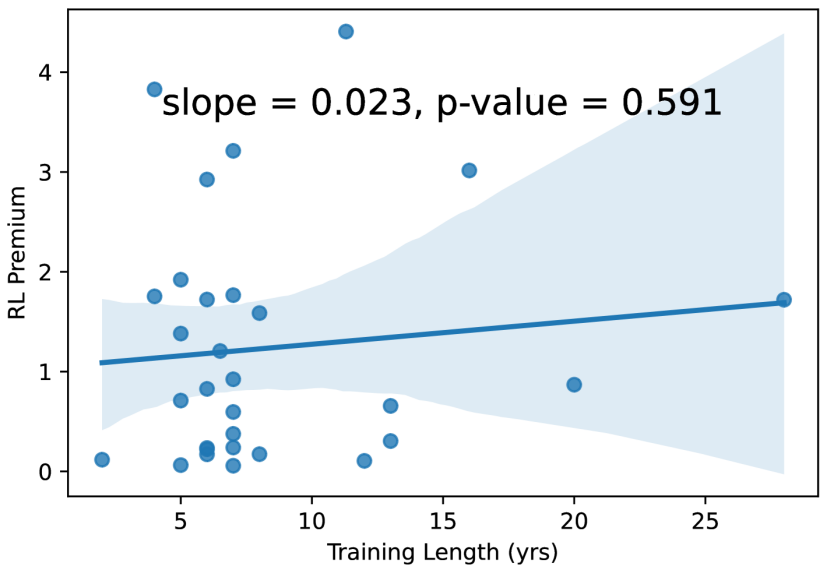


Figura 5 – Uso da regressão linear para analisar a relação entre o prêmio atribuído pelo RL e os anos do período de treinamento. A regressão indica uma inclinação (*slope*) positiva muito pequena (0.023), sugerindo ganho marginal no desempenho. Contudo, o *p-value* (0.591) mostra que essa relação não é estatisticamente significativa (BAI et al., 2024).

Estudos recentes de 2024 (BAI et al., 2024) apontam que, quando os dados incluem fases de recessão ou choques abruptos, a dificuldade de desempenho do modelo aumenta, pois o caráter não estacionário do mercado pode comprometer a estabilidade do aprendizado. Como forma de ilustrar mais claramente essa análise (Figura 5), é possível atribuir a extensão do período de treinamento com o prêmio obtido pelo modelo de Aprendizagem por Reforço, por meio da subtração da razão de *Sharpe* e da linha de base da razão de *Sharpe* obtida pelo método de RL, normalizando essa diferença pela linha de base. Assim, o RL fornece uma medida relativa do ganho proporcionado pelo modelo, permitindo identificar se ele, de fato, supera estratégias tradicionais.

O resultado obtido pela Figura 5 reforça a hipótese levantada por Yahui Bai. (BAI et al., 2024), de que, em ambientes financeiros altamente voláteis e não estacionários, períodos muito extensos podem introduzir ruído ou informações obsoletas, que pouco agregam ao processo decisório do agente de RL. Assim, a escolha do horizonte de treinamento deve ser feita com cautela, considerando não apenas a extensão temporal, mas também as condições econômicas predominantes, uma vez que não existe padrão estabelecido que assegure ganhos de performance à medida que se ampliam os anos analisados.

1.4 Raiz Quadrada do Erro-Médio (RMSE)

Tendo em vista as possíveis variações de desempenho que podem ser analisadas ao considerar diferentes horizontes de treinamento do modelo de Aprendizado por Reforço (RL), como descrito anteriormente, torna-se necessário adotar uma métrica quantitativa capaz de avaliar objetivamente a qualidade das previsões geradas. Uma métrica estatística amplamente utilizada para essa finalidade é a **Raiz Quadrada do Erro Médio** (*Root Mean Square Error — RMSE*), que mede a diferença entre os valores estimados e os valores reais observados (JAMES et al., 2021).

Sua aplicação é de extrema relevância em análises de séries temporais e contextos financeiros, uma vez que permite quantificar o erro médio de modelos de simulação, em relação ao comportamento real do ativo analisado. Segundo (BUENO, 2012), métricas como o RMSE são essenciais na comparação de previsões, pois traduzem o erro em unidades diretamente associadas à variável estudada, facilitando a interpretação dos resultados.

Matematicamente, o RMSE é definido como:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{1.6}$$

onde:

- $\hat{y}_i$ representa o valor previsto pelo modelo;
- y_i representa o valor real observado; e
- n é o número total de observações comparadas.

O RMSE expressa o erro médio em unidades da variável original (por exemplo, o preço do ativo em reais), permitindo uma interpretação direta da precisão do modelo. Valores menores indicam maior aderência entre as previsões e os valores observados, sugerindo melhor capacidade preditiva. Já valores elevados sinalizam maior dispersão dos resultados em relação aos dados reais (JAMES et al., 2021).

De acordo com (NIELSEN, 2021), o RMSE apresenta eficácia e simplicidade, que o tornam amplamente aplicável em diferentes metodologias de previsão, assim como os que combinam aprendizado estatístico e técnicas de machine learning. Essa versatilidade faz do RMSE uma métrica particularmente útil na modelagem de séries financeiras, como as desenvolvidas neste trabalho, permitindo comparar o desempenho entre diferentes abordagens preditivas, partindo desde cadeias de Markov puras até modelos aprimorados por aprendizado por reforço (RL), sob uma base quantitativa comum e consistente.

2 Métodos Utilizados

Com o objetivo de desenvolver múltiplos pipelines de dados que permitissem analisar o comportamento do mercado de ativos financeiros brasileiro e, ao mesmo tempo, aplicar as metodologias de Cadeias de Markov e Aprendizado por Reforço (RL) apresentadas anteriormente, foi utilizada a plataforma *Google Colab* em conjunto com a linguagem *Python*. Essa combinação possibilitou a realização das análises de forma escalável, reprodutível e aplicável a diferentes ativos e períodos.

2.1 Coleta e Tratamento dos Dados

Para a construção do modelo e observação do seu comportamento durante o tempo, foi realizada a coleta de dados históricos dos preços de fechamento diário das ações da Petrobras (PETR4) e Vale (VALE3), ambas negociadas na B3. Os dados foram obtidos por meio da biblioteca `yfinance` do *Python*, com período de análise entre 01/01/2023 e 31/12/2024.

Com o objetivo de analisar a variação real dos ativos e avaliar o desempenho das simulações ao longo do tempo, utilizou-se o ano de 2023 como conjunto de treinamento do modelo, empregando técnicas de aprendizagem por reforço para identificar padrões e otimizar decisões com base nos estados dos ativos. Aqui foram definidos três intervalos de datas para a execução das simulações. Dois deles correspondem ao ano de 2023, utilizados para a construção da matriz de transição e o treinamento dos modelos de Cadeias de Markov e Aprendizado por Reforço (RL). Já o terceiro intervalo, referente ao ano de 2024, do qual foi destinado à simulação e previsão do comportamento dos preços das ações nesse período.

A combinação entre Cadeias de Markov e RL aprimora a capacidade do modelo em representar a evolução temporal dos ativos. Enquanto o modelo de Markov descreve o comportamento dos retornos e as transições entre estados, sem tomar decisões, o RL introduz uma camada de decisão adaptativa, que aprende a partir da experiência histórica qual ação é mais vantajosa em cada situação. Essa integração resulta em uma representação mais dinâmica e inteligente do sistema, permitindo que o modelo não apenas compreenda as transições de mercado, mas também adapte suas estratégias com base em padrões aprendidos.

2.2 Desenvolvimento do Modelo e Projeções

2.2.1 Análise Inicial

Como primeira etapa, implementou-se a cadeia de Markov em tempo discreto usando a Equação 1.1 para calcular as probabilidades de transição. Os retornos foram segmentados em três categorias com base no comportamento do ativo em relação ao dia anterior:

- Estado 0: Retorno negativo significativo;
- Estado 1: Retorno estável (variação próxima de zero);
- Estado 2: Retorno positivo significativo.

Os limites que separam esses três regimes foram determinados a partir dos tercis (33% e 66%) da distribuição empírica dos retornos do período de treino. Esse procedimento fornece uma divisão estatisticamente consistente dos dados, permitindo que cada ativo seja categorizado de acordo com sua própria dinâmica histórica. Tal abordagem é amplamente utilizada em modelagens baseadas em cadeias de Markov, conforme discutido em (BARBU; D'AMICO; BLASIS, 2017). A discretização adequada dos retornos é fundamental, pois permite representar a dinâmica estocástica do ativo por meio de um espaço de estados bem definido, etapa necessária para a construção da matriz de transição e do correspondente diagrama de estados (Figura 1).

A partir da série discretizada, construiu-se a matriz de transição de cada ativo por meio da contagem das mudanças observadas entre estados consecutivos. Para cada par de estados sucessivos (i, j), contabilizou-se a ocorrência da transição, que posteriormente foi normalizada pela soma total das transições originadas do estado i. O resultado é a matriz P_{ij}, na qual cada elemento representa a probabilidade de o ativo mover-se do estado i para o estado j em um passo temporal, conforme estabelecido na Equação 1.3.

Quadro 1 – Construção da probabilidade e matriz de transição (Autoria própria).

```
returns_matrix = data_matrix.pct_change().dropna()
returns_train  = data_train.pct_change().dropna()

quantiles = {
    ticker: returns_train[ticker].quantile([0.33, 0.66]).values
    for ticker in tickers
}

def classify_return_value(r, q1, q2):
    return 0 if r < q1 else (2 if r > q2 else 1)

states_train = pd.DataFrame({
    ticker: returns_train[ticker].apply(lambda r: classify_return_value(r,
    ↪  *quantiles[ticker]))
    for ticker in tickers
})

states_matrix = pd.DataFrame({
    ticker: returns_matrix[ticker].apply(lambda r: classify_return_value(r,
    ↪  *quantiles[ticker]))
    for ticker in tickers
})

def estimate_transition_matrix(states, n_states=3):
    P = np.zeros((n_states, n_states))
    for (i, j) in zip(states[:-1], states[1:]):
        P[i, j] += 1
    row_sums = P.sum(axis=1, keepdims=True)
    P = np.divide(P, row_sums, where=row_sums!=0)
    return P

matrices = {ticker: estimate_transition_matrix(states_matrix[ticker].values) for
↪  ticker in tickers}
```

2.2.2 Simulação markoviana

Partindo do uso da matriz de transição, como segunda etapa, é realizado o desenvolvimento da simulação markoviana. Essa etapa segue diretamente a abordagem apresentada por D'Amico e De Blasis no artigo *A Multivariate Markov Chain Stock Model*, que demonstra como cadeias de Markov podem ser utilizadas para modelar a evolução temporal de ativos financeiros por meio de estados discretizados (D'AMICO; BLASIS, 2020). Assim

como no modelo proposto pelos autores, assume-se que a dinâmica futura do ativo depende exclusivamente do estado atual, em conformidade com a propriedade de Markov.

O Quadro 2 apresenta a implementação utilizada. Primeiramente, constroem-se os retornos médios por estado, armazenados no objeto `returns_map`. Em seguida, define-se a função `simulate_markov_prices`, responsável por executar a lógica da simulação: dado um estado inicial, a função sorteia sucessivos estados de acordo com a distribuição de probabilidades da matriz de transição, aplica o retorno médio correspondente e atualiza o preço ao longo do tempo. Ao final, o processo é executado para cada ativo, gerando uma série simulada de preços para o período de teste.

Essa metodologia, alinhada à proposta de D'Amico e De Blasis, fornece uma articulação clara entre a estrutura probabilística dos estados e a evolução dos preços, garantindo coerência entre a modelagem teórica e os resultados empíricos obtidos.

Quadro 2 – Função criada para construir a cadeia de Markov (Autoria própria).

```
returns_map = {
    ticker: returns_train[ticker].groupby(states_train[ticker]).mean().to_dict()
    for ticker in tickers
}

np.random.seed(0)
def simulate_markov_prices(P, initial_price, n_days, initial_state, ticker):
    prices = [initial_price]
    state = initial_state
    for _ in range(n_days):
        next_state = np.random.choice(len(P), p=P[state])
        r = returns_map[ticker].get(next_state, 0)  # usa retorno médio do estado
        price = prices[-1] * (1 + r)
        prices.append(price)
        state = next_state
    return prices

n_days = len(data_test)
simulated_prices = {}
for ticker in tickers:
    last_price = data_matrix[ticker].iloc[-1]
    last_return = returns_matrix[ticker].iloc[-1]
    q1, q2 = quantiles[ticker]
    initial_state = classify_return_value(last_return, q1, q2)
    simulated_prices[ticker] = simulate_markov_prices(matrices[ticker], last_price,
    ↪  n_days, initial_state, ticker)
```

2.2.3 Aprendizado por Reforço

Em continuidade à análise, foi aplicado a Aprendizado por Reforço (*Reinforcement Learning – RL*), por meio do treinamento de um agente baseado no algoritmo *Q-Learning*. Esse algoritmo utiliza uma tabela de valores $Q(s, a)$, como definida pela abordagem descrita por (BAI et al., 2024), do qual associa cada par estado–ação a uma estimativa de recompensa esperada. A cada iteração, o agente observa o estado atual (classificação do retorno diário do ativo), escolhe uma ação (comprar ou vender) e recebe como recompensa o retorno subsequente resultante dessa decisão.

Nesse contexto, os estados correspondem às categorias discretizadas dos retornos (queda, estabilidade e alta) e as ações, representam decisões de investimento simplificadas: 0 = posição vendida (short) e 1 = posição comprada (long). A cada iteração, o agente observa o estado atual do ativo, escolhe uma ação e recebe uma recompensa proporcional ao retorno subsequente, consistente com a estrutura caracterizada.

A atualização da função de valor segue a regra:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \Big[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \Big] \tag{2.1}$$

em que:

- α é a taxa de aprendizado,
- γ é o fator de desconto de recompensas futuras,
- r é a recompensa imediata recebida,
- s' é o próximo estado após a ação a.

A política de escolha das ações segue o método ε-*greedy*: o agente explora ações aleatórias com probabilidade ε e, com probabilidade $1 - \varepsilon$, escolhe a melhor ação conhecida até o momento. O Quadro 3 exibe a função desenvolvida com o algoritmo *Q-Learning*.

Quadro 3 – Função de desenvolvimento por RL do agente de Q-Learning (Autoria própria).

```python
n_states, n_actions = 3, 2  # estados e ações (0 = short, 1 = long)
Q = np.zeros((n_states, n_actions))
alpha, gamma, epsilon = 0.1, 0.9, 0.1
episodes = 5000
np.random.seed(0)

def choose_action(state):
    if np.random.rand() < epsilon:
        return np.random.choice(n_actions)
    return np.argmax(Q[state])

# --- Treinamento ---
for ep in range(episodes):
    for ticker in tickers:
        states = states_train[ticker].values
        rets = returns_train[ticker].values
        for t in range(len(states)-1):
            s = states[t]
            a = choose_action(s)
            r = rets[t+1] if a == 1 else -rets[t+1]
            s_next = states[t+1]
            Q[s,a] = Q[s,a] + alpha * (r + gamma*np.max(Q[s_next]) - Q[s,a])
```

Assim, a integração entre a simulação markoviana e o aprendizado por reforço permite que o agente não apenas observe a dinâmica estocástica dos preços, mas também desenvolva políticas de investimento adaptativas. Dessa forma, ele pode testar e ajustar decisões de compra e venda em cenários probabilísticos.

O fluxograma da Figura 6 resume essa arquitetura, de como é desenvolvida pelo Quadro 4. A simulação markoviana fornece, para cada dia do período de teste, a evolução dos preços e dos estados discretizados segundo as probabilidades da matriz de transição. Esses estados representam diferentes condições de mercado (queda, estabilidade ou alta) e constituem a *entrada* do ambiente no qual o agente de RL atua.

Após receber o estado atual, o agente aplica sua política aprendida representada por $Q(s, a)$ e escolhe a ação que maximiza a recompensa esperada. Essa ação corresponde a uma decisão de investimento (manter posição comprada ou vendida). Em seguida, observa-se o retorno real do ativo naquele dia, que é utilizado para atualizar o valor acumulado de um portfólio hipotético. Dessa forma, cada decisão tomada pelo agente influencia diretamente

o desempenho final do portfólio.

O processo é iterativo: a cada novo retorno, o estado é recalculado por meio da mesma função de discretização utilizada no período de treino, garantindo coerência entre a simulação markoviana, o ambiente e a política aprendida. Assim, a cada passo, o agente navega pela cadeia de Markov, tomando decisões com base nos estados observados e experimentando seus efeitos sobre o portfólio.

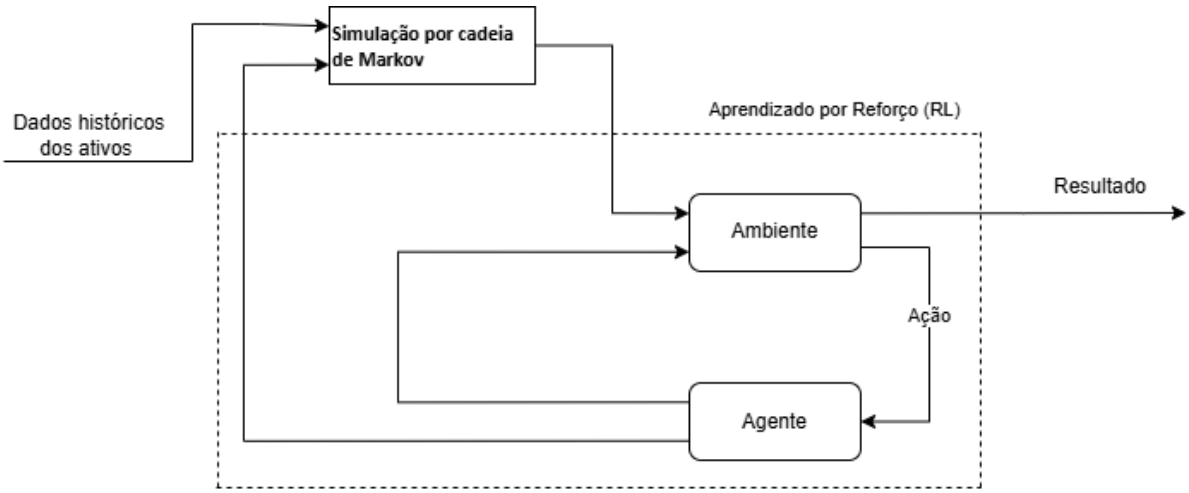


Figura 6 – Fluxograma da simulação do modelo. (Autoria própria)

Quadro 4 – Função de integração unindo a simulação markoviana e o agente de Q-Learning (Autoria própria).

```
positions, portfolios = {}, {}
for ticker in tickers:
    prices = [data_train[ticker].iloc[-1]]
    last_return = returns_train[ticker].iloc[-1]
    q1, q2 = quantiles[ticker]
    state = classify_return_value(last_return, q1, q2)
    pos_list, portfolio = [], [prices[0]]
    real_prices = data_test[ticker].values

    for i, date in enumerate(data_test.index):
        action = np.argmax(Q[state])  # política aprendida
        pos_list.append(action)
        price = real_prices[i]
        prices.append(price)
        ret = (prices[-1]-prices[-2])/prices[-2] if len(prices) >= 2 else 0
        portfolio.append(portfolio[-1] * (1 + action*ret))
        state = classify_return_value(ret, q1, q2)

    positions[ticker] = pos_list
    portfolios[ticker] = portfolio[1:]  # remove inicial
```

2.3 Verificação dos Resultados por meio de RMSE

Como etapa final do modelo, para ser possível mensurar a magnitude média dos desvios entre as previsões e os dados reais de cada ativo, foi utilizada a métrica estatística **Raiz Quadrada do Erro Médio** (*Root Mean Square Error — RMSE*), como definida pela Equação 1.6. Para utilizá-la pelo pipeline construído foi importada a função *mean_squared_error* da biblioteca *sklearn.metrics* do python.

O RMSE foi calculado individualmente para cada ativo e para cada horizonte de treinamento (3 meses, 6 meses, 1 ano e 2 anos), permitindo avaliar como a quantidade de dados de treino influencia a capacidade preditiva do modelo. A partir do Quadro 5 é possível verificar o uso da função para cada ativo, definido já anteriormente pelo "ticker".

Quadro 5 – Implementação da métrica de RMSE por meio da biblioteca em python *sklearn.metrics* (Autoria própria).

```python
from sklearn.metrics import mean_squared_error

def calcular_rmse(real, previsto):
    """Calcula a raiz quadrada do erro médio (RMSE)."""
    return np.sqrt(mean_squared_error(real, previsto))

# Dicionário para armazenar resultados
rmse_resultados = {}

for ticker in tickers:
    y_real = data_test[ticker].values
    y_rl = np.array(portfolios[ticker])  # valores previstos pelo RL

    rmse_rl = calcular_rmse(y_real, y_rl)
    rmse_resultados[ticker] = rmse_rl

for ticker, rmse in rmse_resultados.items():
    print(f"{ticker}: RMSE = {rmse:.4f}")
```

3 Resultados

3.1 Matriz e Diagrama de estados

Com base no modelo proposto anteriormente, a fim de realizar a previsão do comportamento dos retornos diários das ações da Petrobras (PETR4) e da Vale (VALE3) no ano de 2024, foi realizada a discretização dos dados em três estados, considerando o ano de 2023, definidos a partir dos tercis da distribuição empírica dos retornos diários:

- Estado 0: **Baixa**, correspondente aos retornos abaixo do primeiro tercil;
- Estado 1: **Estável**, correspondente aos retornos entre o primeiro e o segundo tercil;
- Estado 2: **Alta**, correspondente aos retornos acima do segundo tercil.

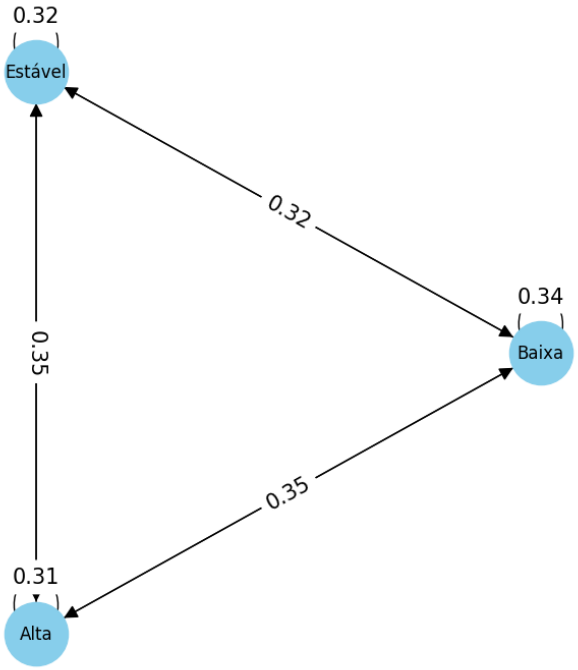
Essa categorização permitiu a construção das matrizes de transição de Markov, partindo de uma matriz 3x3 onde cada linha/coluna representa os estados, e que descreve a probabilidade do ativo mudar de um para outro em dias consecutivos. Cada elemento p_{ij} da matriz representa a probabilidade de transição do estado i para o estado j, onde a soma de cada linha é igual a 1 (100%).

$$P_{\text{PETR4}} = \begin{array}{c} \\ \text{Baixa (0)} \\ \text{Estável (1)} \\ \text{Alta (2)} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Baixa (0)} & \text{Estável (1)} & \text{Alta (2)} \end{array} \\ \begin{pmatrix} 0.337 & 0.325 & 0.337 \\ 0.317 & 0.317 & 0.366 \\ 0.345 & 0.345 & 0.310 \end{pmatrix} \end{array}$$

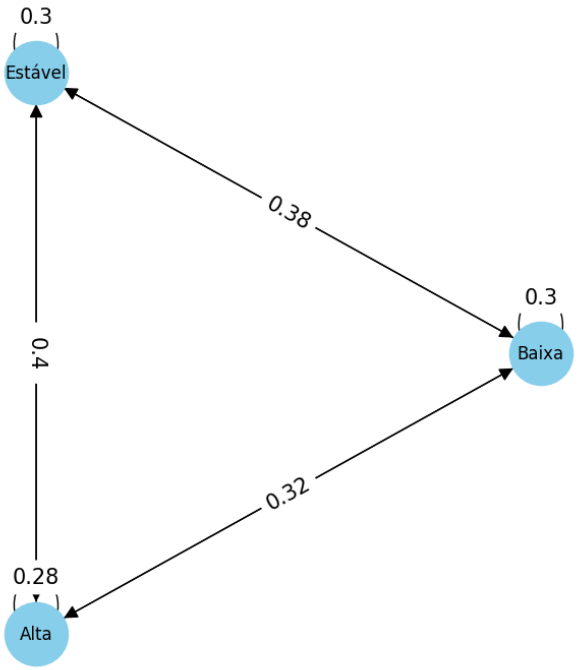
$$P_{\text{VALE3}} = \begin{array}{c} \\ \text{Baixa (0)} \\ \text{Estável (1)} \\ \text{Alta (2)} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Baixa (0)} & \text{Estável (1)} & \text{Alta (2)} \end{array} \\ \begin{pmatrix} 0.301 & 0.277 & 0.422 \\ 0.383 & 0.296 & 0.321 \\ 0.318 & 0.400 & 0.282 \end{pmatrix} \end{array}$$

A interpretação dessas matrizes revela uma tendência de persistência nos estados em ambos os ativos: retornos altos tendem a ser seguidos por novos retornos altos, e retornos baixos tendem a ser seguidos por novos retornos baixos. Entretanto, há também uma probabilidade considerável de transição para o estado intermediário (“Estável”), indicando que movimentos extremos são frequentemente suavizados.

Além disso, para ilustrar melhor as transições entre os estados, é possível construir diagramas em forma de grafos para cada ativo, como apresentado nas Figuras 7a e 7b. Nesses diagramas, os nós representam os estados discretizados e as setas indicam as transições possíveis, ponderadas pela probabilidade de ocorrência, facilitando a identificação dos ciclos predominantes e da intensidade relativa dos caminhos entre os estados.



(a) PETR4



(b) VALE3

Figura 7 – Diagramas de estados dos ativos PETR4 e VALE3 (Autoria própria).

3.2 Simulação Markov e RL

A partir das matrizes de transição obtidas para PETR4 e VALE3 anteriormente, foi então possível construir a primeira versão de modelo de previsão. Partindo da probabilidade de transição (Equação 1.3), foi realizada a aplicação de Markov durante o ano de 2024, realizando uma comparação entre os valores reais observados no ano e os caminhos simulados pelo modelo, como visualizados pela Figura 8.

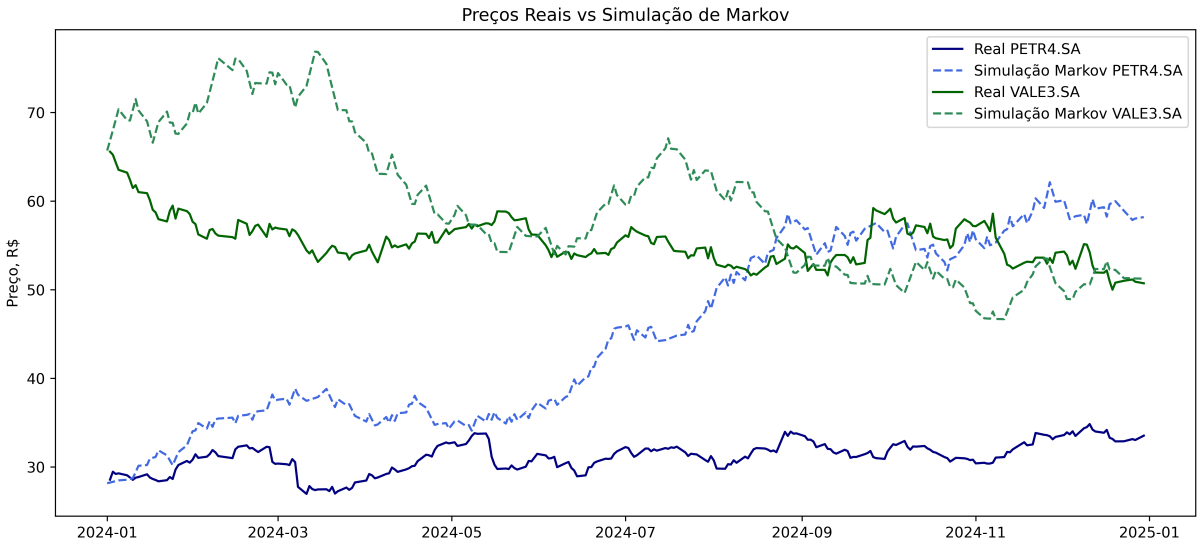
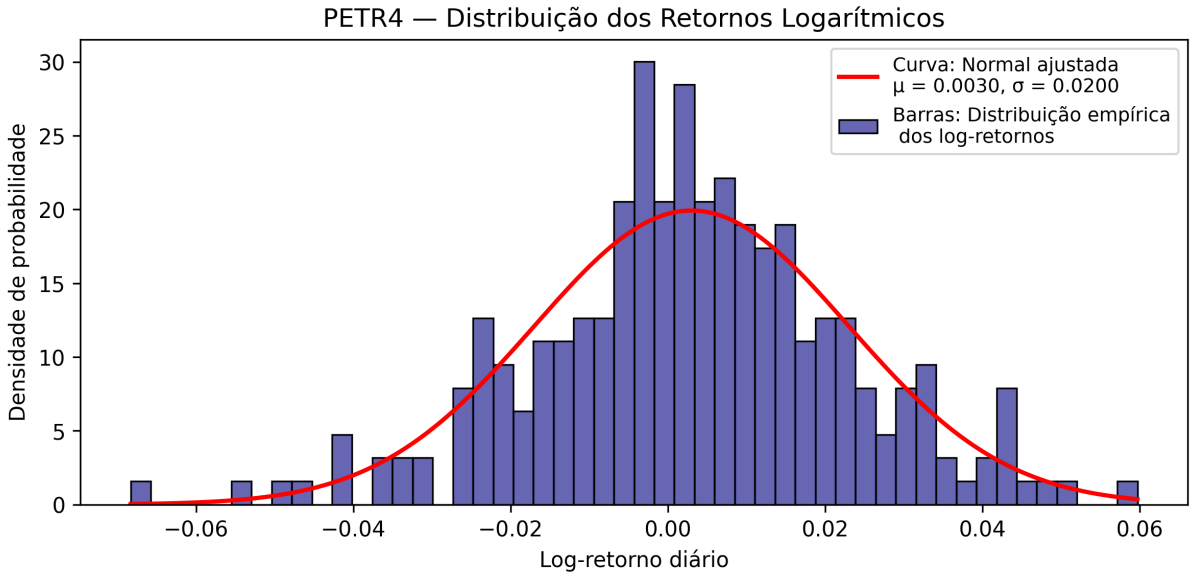


Figura 8 – Comparativo entre a simulação por Markov e os valores reais do ativo VALE3 e PETR4 em 2024. (Autoria própria)

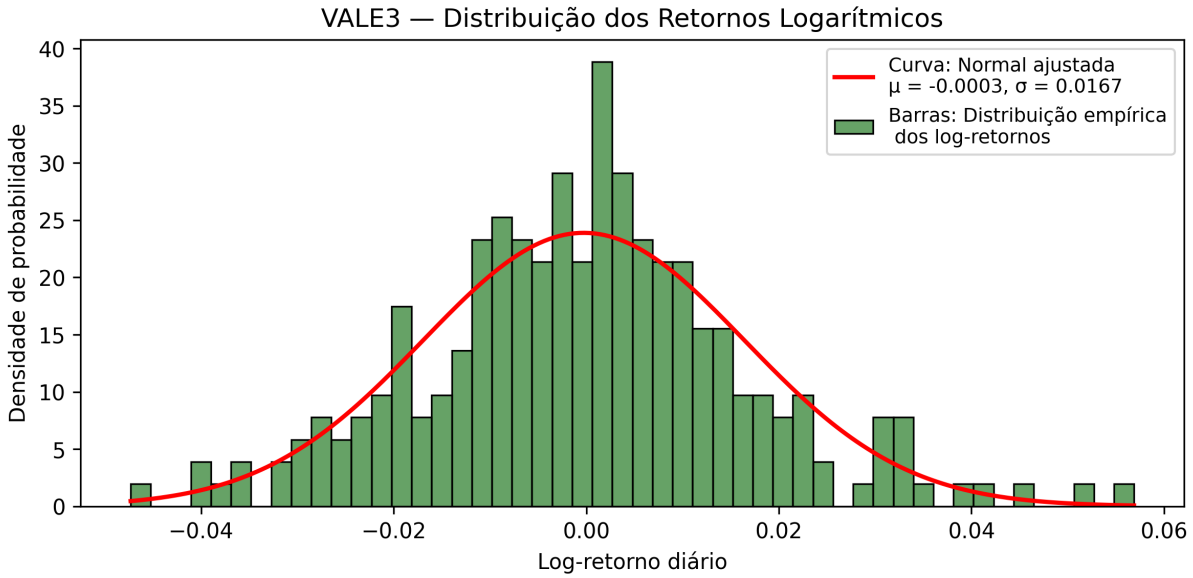
Ao analisar o gráfico, entende-se que o modelo de Markov reproduz parcialmente a tendência geral dos preços reais ao longo do ano de 2024, especialmente nos períodos de estabilidade. No entanto, percebe-se uma defasagem nas variações mais abruptas, indicando que o modelo se perde ao tentar capturar movimentos súbitos de alta ou queda, uma limitação do proprio modelo.

Quando observados os dois ativos, o ajuste foi ligeiramente melhor para a PETR4, possivelmente por apresentar uma dinâmica de preços mais consistente com a hipótese de Markov, ou seja, transições entre estados menos voláteis e mais previsíveis. Já a VALE3 apresentou maior dispersão e mudanças mais abruptas, o que pode explicar a maior divergência entre o valor real e o simulado. Essas diferenças sugerem que o modelo de Markov, por depender apenas da informação do estado anterior, tende a representar melhor ativos com comportamento mais estável e menos sujeitos a choques externos de grande magnitude.

Outra análise interessante para entender a presença dessas variações é a distribuição dos retornos logarítmicos, como na Figura 9. Essa representação permite avaliar de forma mais clara a frequência relativa dos diferentes níveis de retorno, destacando a dispersão e possíveis assimetrias da série temporal, visto que, enquanto a matriz de transição evidencia a dinâmica de mudança entre estados discretos (baixa, estável e alta), a distribuição dos retornos oferece uma visão agregada do comportamento estatístico do ativo, possibilitando comparar o padrão empírico observado com aquele simulado pelo modelo.



(a) PETR4



(b) VALE3

Figura 9 – Distribuição dos retornos logarítmicos simulados para as ações da PETR4 e VALE3 (Autoria própria).

Dessa forma, é possível verificar que a cadeia de Markov reproduz de maneira satisfatória

as transições de curto prazo e parte da variabilidade global observada nos dados históricos. No entanto, os resultados também evidenciam a presença de outliers expressivos em determinados períodos do ano de 2024. Esses desvios podem ser atribuídos a fatores externos ao mercado financeiro que impactaram diretamente as ações analisadas.

No caso da PETR4, por exemplo, eventos relacionados à volatilidade dos preços internacionais do petróleo e a decisões sobre a política de combustíveis da Petrobras influenciaram fortemente o comportamento do ativo ao longo do ano. Já para a VALE3, as oscilações estiveram associadas principalmente às variações na demanda chinesa por minério de ferro e às instabilidades nos preços das commodities metálicas no mercado global.

Sendo assim, como forma de aprimorar o modelo, considerando as variações abruptas que podem ocorrer durante o período estudado, foi acionada uma camada de decisão estratégica junto à simulação puramente Markoviana utilizando Aprendizado por Reforço (RL), por meio do algoritmo Q-Learning. Nela, é desenvolvido um agente que aprende, a partir de experiências passadas (comportamento dos ativos no ano de 2023), quais ações (comprar ou vender) maximizam a recompensa acumulada.

As Figuras 10 e 11 apresentam a aplicação do método seguindo a Equação 2.1, do qual descreve a adaptação dos valores de cada par estado-ação (s, a) conforme as recompensas observadas e as estimativas futuras.

Para esta simulação, adotou-se uma taxa de aprendizado $\alpha = 0{,}1$, responsável por determinar o peso das novas experiências na atualização dos valores da política, e um fator de desconto/recompensa $\gamma = 0{,}9$, que define a importância relativa das recompensas futuras em relação às imediatas. Esses parâmetros foram definidos de modo a equilibrar a velocidade de aprendizado e a estabilidade do modelo, permitindo que o agente aprendesse gradualmente sem perder sensibilidade às variações de longo prazo no comportamento dos ativos.

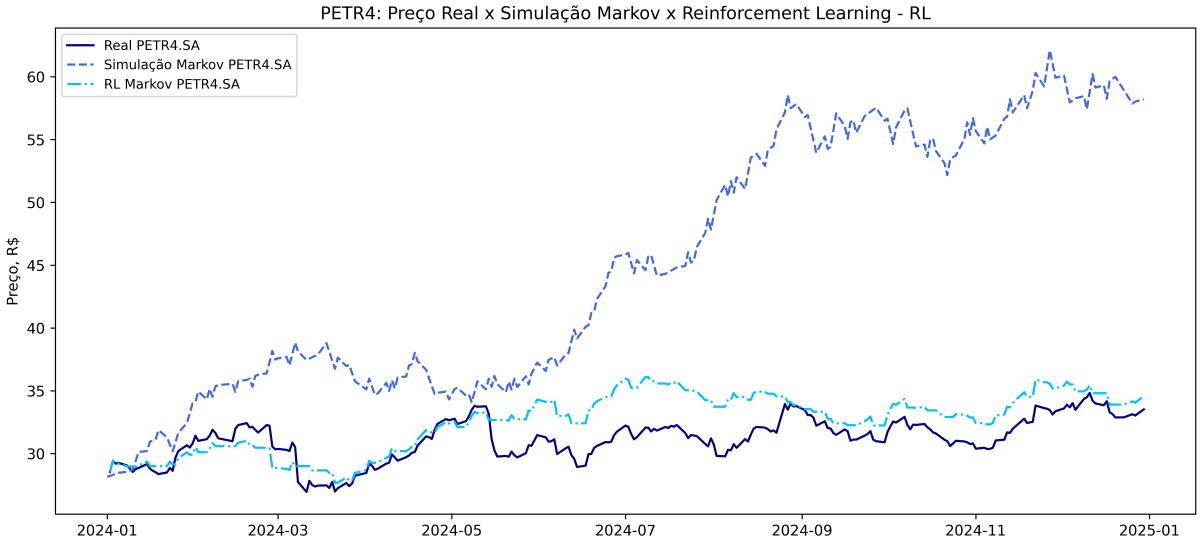


Figura 10 – Comparativo entre valores reais, simulação markoviana e por Reinforcement Learning do ativo PETR4 em 2024. (Autoria própria)

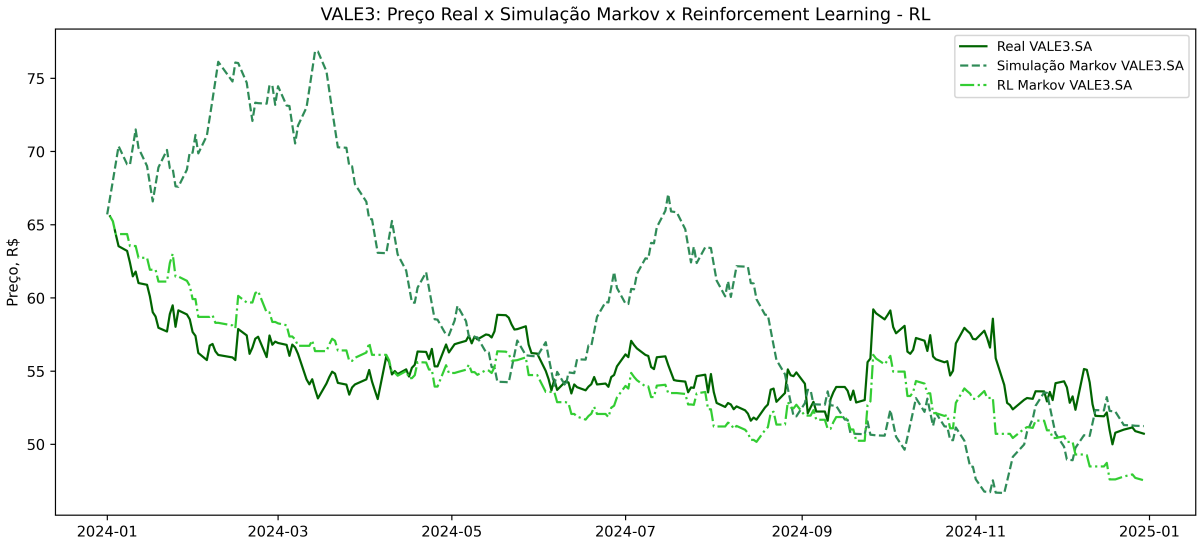


Figura 11 – Comparativo entre valores reais, simulação markoviana e por Reinforcement Learning do ativo VALE3 em 2024. (Autoria própria)

Como é possivel ver após uma analise dos dois gráficos, em alguns momentos, o modelo de RL conseguiu acompanhar mais de perto a dinâmica real do ativo em comparação à simulação de Markov, sugerindo que a incorporação de uma política adaptativa oferece maior flexibilidade e sucesso na modelagem, quando comparado ao comportamento real do ativo. Ainda assim, observam-se discrepâncias relevantes, especialmente em períodos de forte oscilação, o que evidencia tanto a natureza estocástica do mercado quanto as limitações do modelo.

Observa-se, a partir das Figuras 10 e 11, que nas simulações puramente markovianas, as trajetórias geradas apresentaram uma tendência de seguir o comportamento médio do

ativo, porém com desvios cumulativos ao longo do tempo, especialmente em períodos de oscilação abrupta. Isso geralmente ocorre porque o modelo de Markov, embora capture as probabilidades de transição entre estados (baixa, estável e alta), não incorpora um mecanismo de correção dinâmica.

Já com a adição do Aprendizado por Reforço, nota-se nos gráficos que o modelo passa a "corrigir-se" gradualmente em direção aos valores reais, reagindo melhor às mudanças de tendência. Essa adaptação ocorre pois o agente de RL ajusta suas ações a partir dos erros passados, aprendendo políticas que maximizam o retorno, corrigindo assim o erro acumulado e melhorando a consistência temporal da previsão.

3.2.1 Intervalos de tempo

Dessa forma, uma vez que se observa uma melhoria quantitativa dos resultados ao aplicar o modelo de RL, faz-se necessária a discussão sobre à maneira como o período utilizado pelos dados de treino do modelo influencia a eficiência do resultado, quando comparado ao valor real. Sendo assim, tomaram-se como mudanças analisar não apenas 1 ano como treino para o modelo, mas também os intervalos de 2 anos, 6 meses e 3 meses.

Como já discutido anteriormente, é de se esperar que ao considerar diferentes intervalos de treinamento, ocorram impactos na acurácia das simulações. Isso porque, intervalos mais longos abrangem uma maior variedade de condições econômicas e comportamentais do mercado, o que pode enriquecer o aprendizado do modelo e torná-lo mais robusto frente às variações reais do ativo. Por outro lado, essa mesma diversidade pode introduzir ruídos ou informações defasadas, reduzindo a sensibilidade do modelo a mudanças recentes e comprometendo sua capacidade de gerar previsões mais precisas.

Como comparação dessas simulações, considerando o ativo PETR4, a Figura 12 apresenta a modelagem via RL para os diferentes horizontes de treinamento (2 anos, 6 meses e 3 meses).

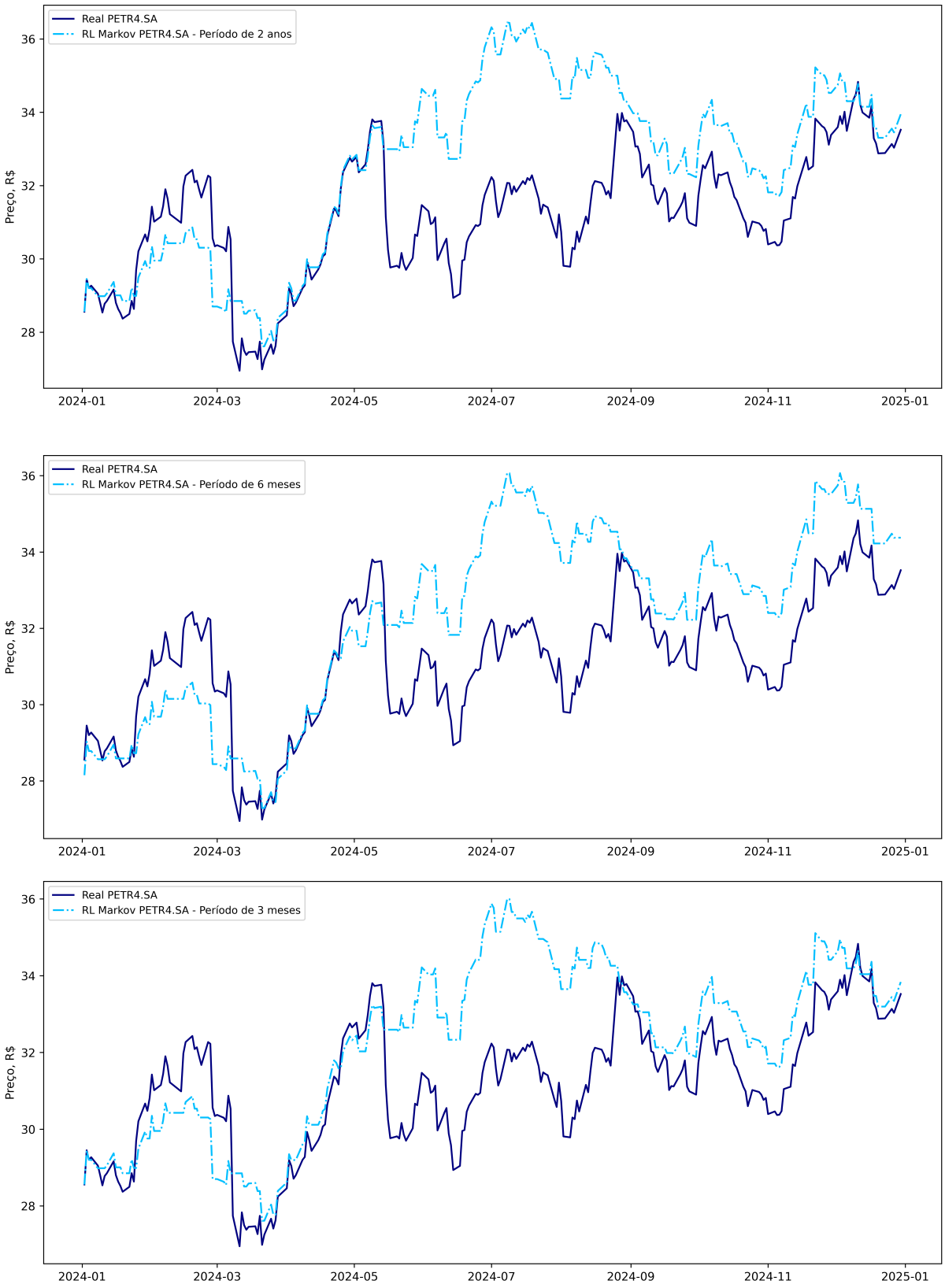


Figura 12 – Preço Real x Reinforcement Learning (RL) para o ativo PETR4 em diferentes intervalos de tempo como base de treino para o RL (Autoria própria).

De maneira análoga, considerando o ativo VALE3 e aplicando os mesmos intervalos de treinamento (2 anos, 1 ano, 6 meses e 3 meses), obtêm-se os resultados ilustrados na

Figura 13.

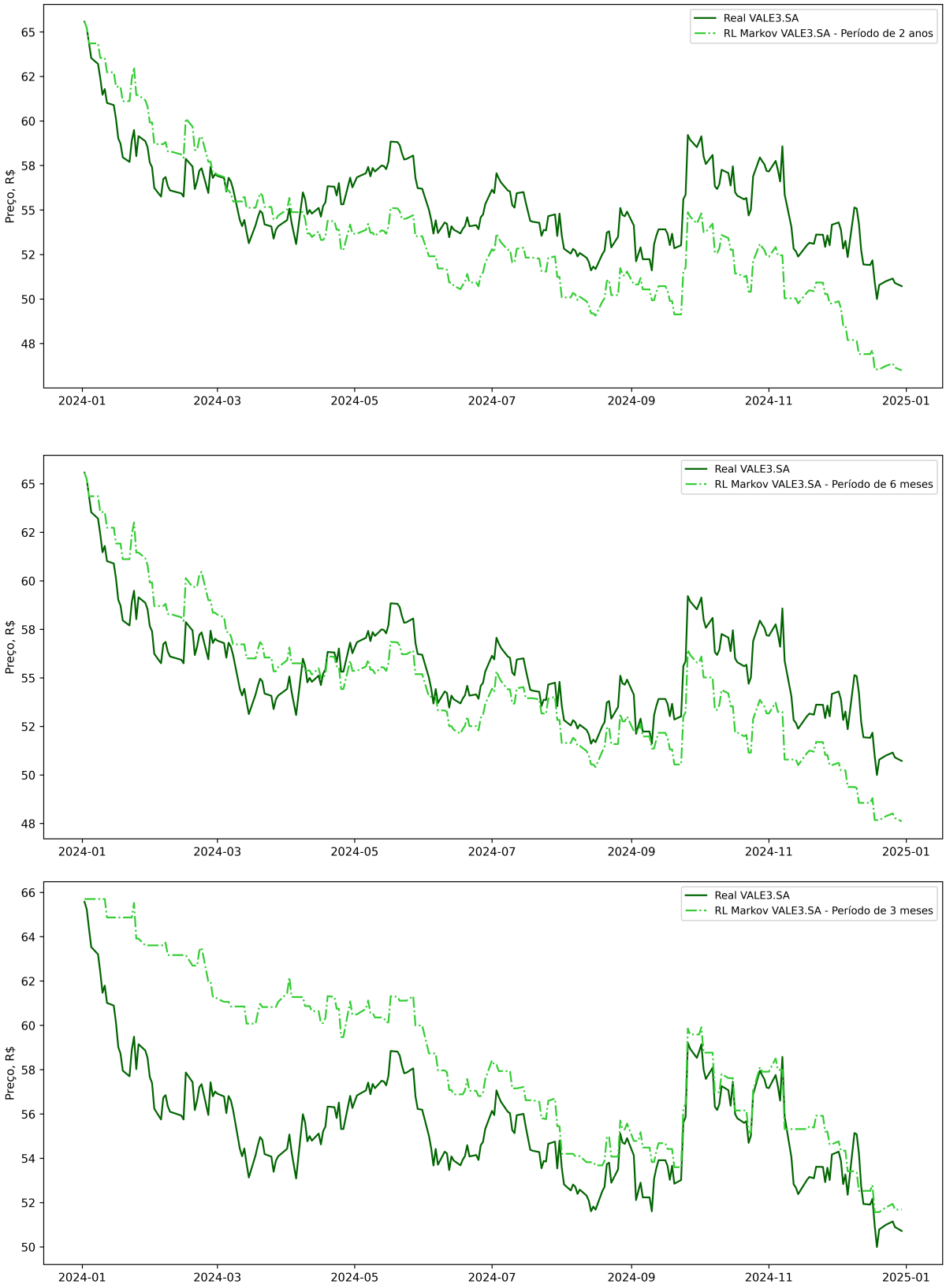


Figura 13 – Preço Real x Reinforcement Learning (RL) para o ativo VALE3 em diferentes intervalos de tempo como base de treino para o RL (Autoria própria).

Por fim, com base nos resultados qualitativos obtidos e considerando os diferentes períodos de análise, tornou-se necessário complementar a avaliação com uma métrica quantitativa capaz de mensurar a acurácia e a eficiência do método aplicado. Para isso, foi utilizada a **Raiz Quadrada do Erro Médio** (*Root Mean Square Error — RMSE*), conforme definido na metodologia deste trabalho. Essa métrica permite avaliar de maneira objetiva a proximidade entre os valores previstos pelos modelos e os valores reais observados, fornecendo um indicador numérico da precisão alcançada em cada cenário de treinamento. A Tabela 1 exibe essa comparação dos 2 ativos estudados, levando em consideração 4 diferentes horizontes de treinamento.

Tabela 1 – Valores de RMSE do modelo de RL (Q-Learning) em diferentes horizontes de treinamento.

Ativo	**3 meses**	**6 meses**	**1 ano**	**2 anos**
PETR4.SA	1.912%	1.983%	2.060%	2.222%
VALE3.SA	3.670%	2.274%	2.443%	3.155%

Fonte: Elaborado pelo autor.

Observa-se, ao analisar os valores de RMSE em cada período, que, para a PETR4, o menor valor ocorre no horizonte de 3 meses, indicando que o modelo de Q-Learning apresenta melhor desempenho em períodos de treinamento mais curtos. Esse resultado é coerente com o comportamento da PETR4, cuja a variação de preços tende a refletir oscilações de curto prazo influenciadas por fatores conjunturais, como mudanças no preço do petróleo, decisões governamentais e variações cambiais. Dessa forma, períodos menores de observação capturam melhor as dinâmicas recentes do mercado, evitando que o modelo incorpore ruídos ou padrões antigos que não representam o cenário atual.

Por outro lado, a VALE3 apresenta o menor RMSE no período de 6 meses, o que sugere que esse horizonte de treinamento permite um equilíbrio mais adequado entre a captura de tendências estruturais e a suavização de flutuações pontuais do mercado. Como a VALE é fortemente influenciada por fatores externos, como a demanda global por minério de ferro, preços internacionais de commodities e políticas ambientais, períodos ligeiramente mais longos oferecem ao modelo uma base mais robusta para identificar padrões consistentes sem superajustar-se às variações diárias.

3.2.2 Parâmetros de Q-Learning

De forma complementar, ao analisar os gráficos obtidos, observou-se que o modelo de Aprendizado por Reforço (RL) tende a apresentar maior dificuldade de adaptação diante de variações muito abruptas no comportamento do ativo, demorando alguns períodos para se realinhar ao valor real. Diante dessa limitação, foi proposta a variação dos parâmetros de aprendizagem (α) e de desconto (γ), uma vez que ambos exercem influência direta sobre a capacidade de aprendizado do agente.

Teoricamente, valores mais elevados de α tendem a acelerar a atualização dos valores da função $Q(s, a)$, permitindo uma adaptação mais rápida às mudanças recentes do mercado. Por sua vez, o fator de desconto γ controla o peso atribuído às recompensas futuras em relação às imediatas.

Por exemplo, valores próximos de 1 fazem com que o agente considere horizontes mais longos de decisão, o que pode ser vantajoso para ativos com comportamento mais previsível e estruturado, como a VALE3.SA. Já valores menores de γ priorizam retornos de curto prazo, sendo mais adequados para ativos com alta volatilidade e maior sensibilidade a fatores conjunturais, como a PETR4.SA.

Dessa forma, partindo dos melhores resultados de RMSE obtidos, 3 meses para a PETR4 e 6 meses para a VALE3, foi realizado o ajuste dos parâmetros α e γ em cada simulação, conforme definido pela Equação 2.1, com o objetivo de melhorar a estabilidade do aprendizado e o calculo de RMSE.

Realizando a alteração de α para $0,4$ e de γ para $0,8$, resultou em uma melhoria significativa para o ativo PETR4, com redução do RMSE para 0,674%. Esse comportamento confirma a influência direta desses parâmetros sobre a capacidade de adaptação do modelo. Em contrapartida, para a VALE3 observou-se um aumento do RMSE para 4,758%, indicando que a mesma configuração não se mostra eficiente para um ativo de dinâmica mais estável. A Figura 14 apresenta os resultados obtidos em cada simulação após a modificação dos parâmetros.

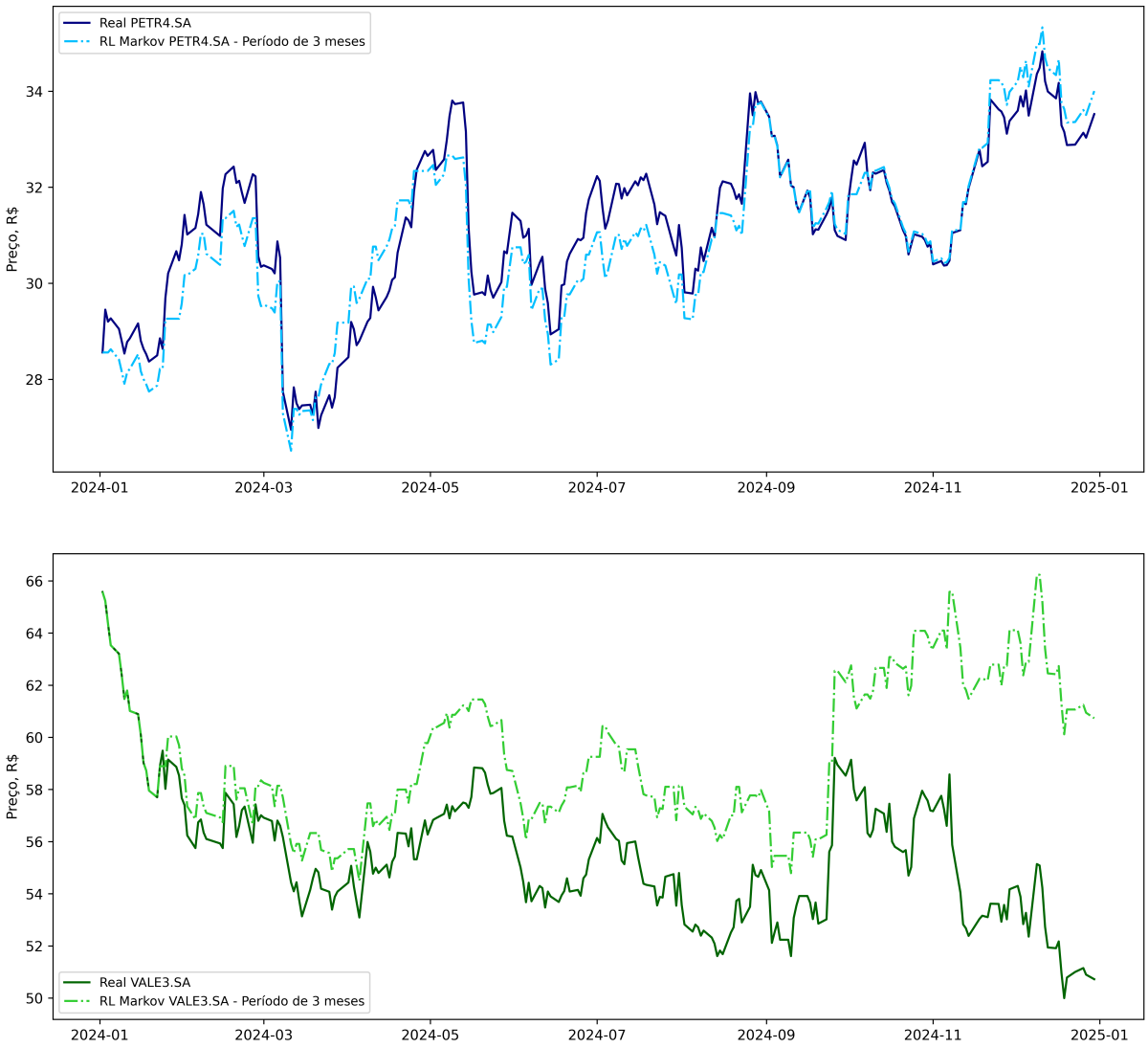


Figura 14 – Preço Real x Reinforcement Learning (RL) para os ativos PETR4 e VALE3 considerando 3 meses para treino e os parâmetros $\alpha = 0{,}4$ e $\gamma = 0{,}8$ (Autoria própria).

Considerando agora a VALE3, mesmo após o teste de diversas combinações de α e γ, não foi possível obter redução significativa no RMSE. Isso evidencia que, por possuir uma dinâmica mais estável e menos reativa, a VALE3 não se beneficia de ajustes voltados ao aprendizado rápido. Dessa forma, conclui-se que a eficácia do modelo de Q-Learning depende do perfil do ativo: enquanto ativos voláteis, como a PETR4, respondem melhor a parâmetros que favorecem respostas rápidas, ativos mais estáveis requerem políticas de aprendizado mais conservadoras e calibração específica.

4 Conclusão

Este trabalho teve como objetivo investigar a aplicação conjunta das Cadeias de Markov em tempo discreto e do Aprendizado por Reforço (Reinforcement Learning – RL) na precificação e previsão de ativos financeiros, utilizando como estudo de caso as ações da Petrobras (PETR4) e da Vale (VALE3). A proposta combinou a modelagem probabilística dos processos markovianos com a capacidade adaptativa do algoritmo Q-Learning, permitindo avaliar a dinâmica dos preços em ambientes financeiros voláteis e identificar como diferentes configurações de parâmetros influenciam o desempenho preditivo.

Os resultados obtidos a partir das matrizes de transição de Markov indicaram uma forte tendência de persistência dos estados, o que confirma que períodos de alta ou baixa tendem a se repetir no curto prazo. No entanto, o modelo puramente markoviano mostrou limitações na previsão de valores futuros, especialmente em períodos de grande instabilidade do mercado. A integração com o modelo de RL superou parcialmente essas limitações, uma vez que o agente aprendeu a ajustar suas decisões de compra e venda com base em recompensas, tornando o processo mais dinâmico e adaptativo.

A avaliação quantitativa, por meio da métrica Raiz Quadrada do Erro Médio (RMSE), mostrou que a acurácia do modelo depende do horizonte de treinamento. Para a PETR4, o menor erro foi obtido com 3 meses de treino (1,912%), enquanto para a VALE3 o melhor desempenho ocorreu com 6 meses (2,274%). Esses resultados indicam que períodos curtos capturam melhor tendências recentes, enquanto intervalos intermediários proporcionam maior estabilidade. Essa diferença reforça a natureza adaptativa do RL, que se ajusta às condições mais atuais do mercado, reagindo rapidamente a mudanças no comportamento dos preços e reduzindo o erro em comparação às simulações estáticas.

Em uma segunda etapa do estudo, buscou-se aprimorar o desempenho do modelo por meio da calibração dos parâmetros de aprendizado (α) e de desconto (γ) do algoritmo Q-Learning. Os testes mostraram que, para a PETR4, a configuração $\alpha = 0{,}4$ e $\gamma = 0{,}8$ proporcionou uma redução expressiva do RMSE para 0,674%, confirmando que ativos com maior volatilidade se beneficiam de políticas que priorizam recompensas de curto prazo e aprendizado mais rápido. Já para a VALE3, mesmo após a realização de múltiplas calibrações, não foi encontrado um conjunto de parâmetros que melhorasse o desempenho. Esse resultado reforça que a eficácia do modelo depende intrinsecamente das características

do ativo analisado: enquanto ativos mais sensíveis a choques conjunturais demandam maior responsividade do agente, ativos de comportamento mais estável requerem estratégias de aprendizado mais conservadoras e fatores de desconto mais elevados.

De modo geral, o modelo de RL apresentou desempenho superior ao de Markov puro, acompanhando mais de perto o comportamento real dos preços. Isso ocorre porque, ao incorporar o processo de aprendizado contínuo por recompensas, o RL é capaz de corrigir gradualmente suas decisões, identificando padrões de reversão e ajustando-se a novos regimes de mercado. Conclui-se que a integração dessas metodologias constitui uma abordagem promissora para previsão de séries financeiras, combinando a estrutura estatística das Cadeias de Markov com a adaptabilidade do Aprendizado por Reforço. Sugere-se, para trabalhos futuros, a ampliação do conjunto de dados, o uso de algoritmos mais avançados, como o Deep Q-Learning, e a inclusão de variáveis macroeconômicas, de modo a aprimorar a robustez e a aplicabilidade do modelo proposto.

Referências

ALVES, R.; DELGADO, C. *Processos estocásticos*. 1997.

(AWS), A. W. S. *O que é aprendizado por reforço?* 2024. Acesso em: 08 agosto 2025. Disponível em: <https://aws.amazon.com/pt/what-is/reinforcement-learning>.

BAI, Y. et al. A review of reinforcement learning in financial applications. *arXiv*, 2024. Disponível em: <https://arxiv.org/abs/2411.12746>.

BARBU, V.; D'AMICO, G.; BLASIS, R. D. Novel advancements in the markov chain stock model: analysis and inference. *Annals of Finance*, v. 13, 2017. Disponível em: <https://doi.org/10.1007/s10436-017-0297-9>.

BLASIS, R. D. *Markov Chain Modelling in Finance: Stock Valuation and Price Discovery*. Tese (Doctor of Philosophy Thesis) — University of Wollongong, 2019. Acesso em: Maio 2025. Disponível em: <https://ro.uow.edu.au/theses1/719>.

BUENO, R. de Losso da S. *Econometria de Séries Temporais*. São Paulo: Cengage Learning, 2012.

D'AMICO, G.; BLASIS, R. D. A multivariate markov chain stock model. *Scandinavian Actuarial Journal*, Taylor & Francis, v. 2020, n. 4, 2020. Disponível em: <https://doi.org/10.1080/03461238.2019.1661280>.

DOOB, J. L. *Stochastic Processes*. [S.l.]: Wiley, 1990. v. 1. (Wiley Classics Library, v. 1). ISBN 978-0-471-52369-7.

GHEZZI, L. L.; PICCARDI, C. Stock valuation along a markov chain. *Applied Mathematics and Computation*, v. 141, n. 2, 2003. ISSN 0096-3003. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0096300302002631>.

GU, Y.; WANG, H.; ZHANG, M. Implementation optimal control model of smart grid engineering cluster construction based on markov decision process. *Alexandria Engineering Journal*, Elsevier, v. 125, 2025. Disponível em: <https://doi.org/10.1016/j.aej.2025.04.046>.

GUSMãO, A. *5 princípios de aprendizagem por reforço explicados pelo especialista em IA, Hadelin de Ponteves*. 2020. Acessado em: 09/08/2025. Disponível em: <https://tempodeinovacao.com.br/5-principios-de-aprendizagem-por-reforco-explicados-pelo-especialista-em-ia-hadelin-de-ponteves>.

HASHIOKA, J. A. S. *Modelo de precificação de ativos por cadeias de Markov*. Dissertação (Dissertação de Mestrado) — Universidade Estadual Paulista (UNESP), São Paulo, 2018.

JAMES, G. et al. *An Introduction to Statistical Learning: with Applications in R*. 2nd. ed. [S.l.]: Springer, 2021.

MUREL, J.; KAVLAKOGLU, E. *O que é aprendizagem de reforço?* 2024. Acesso em: 08 agosto 2025. Disponível em: <https://www.ibm.com/br-pt/think/topics/reinforcement-learning>.

NIELSEN, A. *Análise prática de séries temporais: predição com estatística e aprendizado de máquina*. [S.l.]: Alta Books, 2021.

STERN, J. M. et al. Otimização e processos estocásticos aplicados à economia e finanças. *arXiv*, 2020. Disponível em: <https://arxiv.org/pdf/2005.13459>.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. 2nd. ed. [S.l.]: MIT Press, 2018.

APÊNDICE A – Código Python para construção das matrizes de transição

```python
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx

# --- 1. Baixar dados dos dois ativos ---
tickers = ['PETR4.SA', 'VALE3.SA']
data = yf.download(tickers, start="2024-01-01", end="2024-12-31", auto_adjust=True)['Close']

# --- 2. Calcular retornos ---
returns = data.pct_change().dropna()

# Função para criar matriz de transição e grafo
def markov_chain(returns, ticker_name):
    # --- 3. Discretizar retornos em 3 estados ---
    q1, q2 = returns.quantile([0.33, 0.66])
    states = returns.apply(lambda x: 0 if x < q1 else (2 if x > q2 else 1))

    # --- 4. Construir matriz de transição ---
    n_states = 3
    transition_matrix = np.zeros((n_states, n_states))

    for (i, j) in zip(states[:-1], states[1:]):
        transition_matrix[i][j] += 1

    # Normalizar por linha
    transition_matrix = transition_matrix / transition_matrix.sum(axis=1, keepdims=True)

    # DataFrame bonito
    df_transitions = pd.DataFrame(
        transition_matrix,
        columns=['Baixa (0)', 'Estável (1)', 'Alta (2)'],
        index=['Baixa (0)', 'Estável (1)', 'Alta (2)']
    )
    print(f"\nMatriz de Transição de Estados - {ticker_name}:\n")
    print(df_transitions.round(3))

    # --- 5. Grafo da cadeia de Markov ---
```

```
    G = nx.DiGraph()
    state_names = ['Baixa', 'Estável', 'Alta']
    for i in range(n_states):
        for j in range(n_states):
            prob = transition_matrix[i][j]
            if prob > 0:
                G.add_edge(state_names[i], state_names[j], weight=round(prob, 2))

    pos = nx.circular_layout(G)
    plt.figure(figsize=(7,5))
    nx.draw(G, pos, with_labels=True, node_size=2000, node_color="skyblue",
            font_size=12, arrowsize=20)
    labels = nx.get_edge_attributes(G, 'weight')
    nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
    plt.title(f"Cadeia de Markov - Estados de Retorno ({ticker_name})")
    plt.show()

# --- Aplicar para cada ativo ---
markov_chain(returns['PETR4.SA'], "PETR4")
markov_chain(returns['VALE3.SA'], "VALE3")
```

APÊNDICE B – Código Python para Distribuição dos Retornos Logarítmicos

```python
# ================================================================
# Distribuição dos Retornos Logarítmicos - PETR4 e VALE3
# ================================================================

import numpy as np
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# 1. Baixar dados de treino
tickers = ["PETR4.SA", "VALE3.SA"]
data_train = yf.download(
    tickers, start="2023-01-01", end="2023-12-31", auto_adjust=True
)['Close']

# 2. Calcular log-retornos
log_returns = np.log(data_train).diff().dropna()

# Definir cores personalizadas
colors = {
    "PETR4.SA": "navy",
    "VALE3.SA": "darkgreen"
}

# 3. Plotar distribuições
for ticker in tickers:
    x = log_returns[ticker].dropna().values
    mu, sigma = np.mean(x), np.std(x, ddof=1)

    xs = np.linspace(x.min(), x.max(), 200)
    pdf = stats.norm.pdf(xs, mu, sigma)

    plt.figure(figsize=(8,4))
    sns.histplot(
        x, bins=50, stat="density", color=colors[ticker], alpha=0.6,
        label="Barras: Distribuição empírica\n dos log-retornos"
    )
    plt.plot(
        xs, pdf, 'r-', lw=2,
```

```
        label=f"Curva: Normal ajustada\n = {mu:.4f},  = {sigma:.4f}"
    )
    plt.title(f"{ticker.replace('.SA','')} - Distribuição dos Retornos
    ↪ Logarítmicos")
    plt.xlabel("Log-retorno diário")
    plt.ylabel("Densidade de probabilidade")
    plt.legend(loc="upper right", fontsize=9, frameon=True)
    plt.tight_layout()
    plt.savefig(f"Distribuicao{ticker}.png", dpi=300, bbox_inches="tight")
    plt.show()
```

APÊNDICE C – Código Python para modelagem de Markov e RL considerando 1 ano de treino

```python
import numpy as np
import random
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt

# ================================================================
# Paleta de cores (azuis para PETR4, verdes para VALE3)
# ================================================================
colors = {
    "PETR4.SA": {
        "real": "navy",        # azul escuro (real)
        "markov": "royalblue", # azul médio (simulação)
        "RL": "deepskyblue"    # azul claro (RL)
    },
    "VALE3.SA": {
        "real": "darkgreen",   # verde escuro (real)
        "markov": "seagreen",  # verde médio (simulação)
        "RL": "limegreen"      # verde claro (RL)
    }
}

# ================================================================
# 1. Baixar dados de treino (2023) e teste (2024)
# ================================================================
tickers = ["PETR4.SA", "VALE3.SA"]
data_train = yf.download(tickers, start="2023-01-01", end="2023-12-31",
↪  auto_adjust=True)['Close']
data_matrix = yf.download(tickers, start="2023-01-01", end="2023-12-31",
↪  auto_adjust=True)['Close']
data_test  = yf.download(tickers, start="2024-01-01", end="2024-12-31",
↪  auto_adjust=True)['Close']

# ================================================================
# 2. Calcular retornos diários e discretizar estados
# ================================================================
returns_matrix = data_matrix.pct_change().dropna()
returns_train  = data_train.pct_change().dropna()
```

```
36
37  quantiles = {
38      ticker: returns_train[ticker].quantile([0.33, 0.66]).values
39      for ticker in tickers
40  }
41
42  def classify_return_value(r, q1, q2):
43      return 0 if r < q1 else (2 if r > q2 else 1)
44
45  states_train = pd.DataFrame({
46      ticker: returns_train[ticker].apply(lambda r: classify_return_value(r,
        ↪  *quantiles[ticker]))
47      for ticker in tickers
48  })
49
50  states_matrix = pd.DataFrame({
51      ticker: returns_matrix[ticker].apply(lambda r: classify_return_value(r,
        ↪  *quantiles[ticker]))
52      for ticker in tickers
53  })
54
55  # ================================================================
56  # 3. Estimar matriz de transição de Markov
57  # ================================================================
58  def estimate_transition_matrix(states, n_states=3):
59      P = np.zeros((n_states, n_states))
60      for (i, j) in zip(states[:-1], states[1:]):
61          P[i, j] += 1
62      row_sums = P.sum(axis=1, keepdims=True)
63      P = np.divide(P, row_sums, where=row_sums!=0)
64      return P
65
66  matrices = {ticker: estimate_transition_matrix(states_matrix[ticker].values) for
    ↪  ticker in tickers}
67
68  # ================================================================
69  # 4. Simular preços em 2024 com Cadeia de Markov
70  # ================================================================
71
72  returns_map = {
73      ticker: returns_train[ticker].groupby(states_train[ticker]).mean().to_dict()
74      for ticker in tickers
75  }
76
77  np.random.seed(0)
78  def simulate_markov_prices(P, initial_price, n_days, initial_state, ticker):
79      prices = [initial_price]
80      state = initial_state
81      for _ in range(n_days):
```

```
        next_state = np.random.choice(len(P), p=P[state])
        r = returns_map[ticker].get(next_state, 0)  # usa retorno médio do estado
        price = prices[-1] * (1 + r)
        prices.append(price)
        state = next_state
    return prices

n_days = len(data_test)
simulated_prices = {}
for ticker in tickers:
    last_price = data_matrix[ticker].iloc[-1]
    last_return = returns_matrix[ticker].iloc[-1]
    q1, q2 = quantiles[ticker]
    initial_state = classify_return_value(last_return, q1, q2)
    simulated_prices[ticker] = simulate_markov_prices(matrices[ticker], last_price,
    ↪  n_days, initial_state, ticker)

# --- Plot real x simulado ---
plt.figure(figsize=(14,6))
for ticker in tickers:
    plt.plot(data_test.index, data_test[ticker],
             label=f"Real {ticker}", color=colors[ticker]["real"], linestyle='-')
    plt.plot(data_test.index.insert(0, data_test.index[0] - pd.Timedelta(days=1)),
             simulated_prices[ticker], '--',
             label=f"Simulação Markov {ticker}", color=colors[ticker]["markov"])
plt.title("Preços Reais vs Simulação de Markov")
plt.ylabel("Preço, R$");
plt.legend();
plt.savefig("precos_reais_vs_markov_1ano.png", dpi=600, bbox_inches="tight")
plt.show()

# ============================================================
# 5. Aprendizagem por Reforço - Q-Learning
# ============================================================
n_states, n_actions = 3, 2  # estados e ações (0 = short, 1 = long)
Q = np.zeros((n_states, n_actions))
alpha, gamma, epsilon = 0.1, 0.9, 0.1
episodes = 5000
np.random.seed(0)

def choose_action(state):
    if np.random.rand() < epsilon:
        return np.random.choice(n_actions)
    return np.argmax(Q[state])

# --- Treinamento ---
for ep in range(episodes):
    for ticker in tickers:
        states = states_train[ticker].values
```

```
        rets = returns_train[ticker].values
        for t in range(len(states)-1):
            s = states[t]
            a = choose_action(s)
            r = rets[t+1] if a == 1 else -rets[t+1]
            s_next = states[t+1]
            Q[s,a] = Q[s,a] + alpha * (r + gamma*np.max(Q[s_next]) - Q[s,a])

# ================================================================
# 6. Backtest em 2024 usando política aprendida
# ================================================================
positions, portfolios = {}, {}
for ticker in tickers:
    prices = [data_train[ticker].iloc[-1]]
    last_return = returns_train[ticker].iloc[-1]
    q1, q2 = quantiles[ticker]
    state = classify_return_value(last_return, q1, q2)

    pos_list, portfolio = [], [prices[0]]
    real_prices = data_test[ticker].values

    for i, date in enumerate(data_test.index):
        action = np.argmax(Q[state])  # política aprendida
        pos_list.append(action)
        price = real_prices[i]
        prices.append(price)

        ret = (prices[-1]-prices[-2])/prices[-2] if len(prices) >= 2 else 0
        portfolio.append(portfolio[-1] * (1 + action*ret))
        state = classify_return_value(ret, q1, q2)

    positions[ticker] = pos_list
    portfolios[ticker] = portfolio[1:]  # remove inicial


# 2) Real x Simulação de Markov x RL (somente PETR4)
plt.figure(figsize=(14,6))
ticker = "PETR4.SA"
plt.plot(data_test.index, data_test[ticker],
         label=f"Real {ticker}", color=colors[ticker]["real"], linestyle='-')
plt.plot(data_test.index.insert(0, data_test.index[0] - pd.Timedelta(days=1)),
         simulated_prices[ticker], '--',
         label=f"Simulação Markov {ticker}", color=colors[ticker]["markov"])
plt.plot(data_test.index, portfolios[ticker], '-.',
         label=f"RL Markov {ticker}", color=colors[ticker]["RL"])
plt.title("PETR4: Preço Real x Simulação Markov x Reinforcement Learning - RL")
plt.ylabel("Preço, R$")
plt.legend(fontsize=9)
plt.savefig("petr4_real_markov_rl_1ano.png", dpi=600, bbox_inches="tight")
```

```
plt.show()

# 3) Real x Simulação de Markov x RL (somente VALE3)
plt.figure(figsize=(14,6))
ticker = "VALE3.SA"
plt.plot(data_test.index, data_test[ticker],
         label=f"Real {ticker}", color=colors[ticker]["real"], linestyle='-')
plt.plot(data_test.index.insert(0, data_test.index[0] - pd.Timedelta(days=1)),
         simulated_prices[ticker], '--',
         label=f"Simulação Markov {ticker}", color=colors[ticker]["markov"])
plt.plot(data_test.index, portfolios[ticker], '-.',
         label=f"RL Markov {ticker}", color=colors[ticker]["RL"])
plt.title("VALE3: Preço Real x Simulação Markov x Reinforcement Learning - RL")
plt.ylabel("Preço, R$")
plt.legend(fontsize=9)
plt.savefig("vale3_real_markov_rl_1ano.png", dpi=600, bbox_inches="tight")
plt.show()
```

APÊNDICE D – Código Python para modelagem RL considerando 2 anos de treino

```python
import numpy as np
import random
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt

# ================================================================
# Paleta de cores (azuis para PETR4, verdes para VALE3)
# ================================================================
colors = {
    "PETR4.SA": {
        "real": "navy",         # azul escuro (real)
        "markov": "royalblue", # azul médio (simulação)
        "RL": "deepskyblue"    # azul claro (RL)
    },
    "VALE3.SA": {
        "real": "darkgreen",    # verde escuro (real)
        "markov": "seagreen",  # verde médio (simulação)
        "RL": "limegreen"      # verde claro (RL)
    }
}

# ================================================================
# 1. Baixar dados de treino (2023) e teste (2024)
# ================================================================
tickers = ["PETR4.SA", "VALE3.SA"]
data_train = yf.download(tickers, start="2022-01-01", end="2023-12-31",
↪  auto_adjust=True)['Close']
data_matrix = yf.download(tickers, start="2023-01-01", end="2023-12-31",
↪  auto_adjust=True)['Close']
data_test  = yf.download(tickers, start="2024-01-01", end="2024-12-31",
↪  auto_adjust=True)['Close']

# ================================================================
# 2. Calcular retornos diários e discretizar estados
# ================================================================
returns_matrix = data_matrix.pct_change().dropna()
returns_train  = data_train.pct_change().dropna()

quantiles = {
    ticker: returns_train[ticker].quantile([0.33, 0.66]).values
```

```
    for ticker in tickers
}

def classify_return_value(r, q1, q2):
    return 0 if r < q1 else (2 if r > q2 else 1)

states_train = pd.DataFrame({
    ticker: returns_train[ticker].apply(lambda r: classify_return_value(r,
    ↪ *quantiles[ticker]))
    for ticker in tickers
})

states_matrix = pd.DataFrame({
    ticker: returns_matrix[ticker].apply(lambda r: classify_return_value(r,
    ↪ *quantiles[ticker]))
    for ticker in tickers
})

# ================================================================
# 3. Estimar matriz de transição de Markov
# ================================================================
def estimate_transition_matrix(states, n_states=3):
    P = np.zeros((n_states, n_states))
    for (i, j) in zip(states[:-1], states[1:]):
        P[i, j] += 1
    row_sums = P.sum(axis=1, keepdims=True)
    P = np.divide(P, row_sums, where=row_sums!=0)
    return P

matrices = {ticker: estimate_transition_matrix(states_matrix[ticker].values) for
↪ ticker in tickers}

# ================================================================
# 4. Simular preços em 2024 com Cadeia de Markov
# ================================================================

returns_map = {
    ticker: returns_train[ticker].groupby(states_train[ticker]).mean().to_dict()
    for ticker in tickers
}

np.random.seed(0)
def simulate_markov_prices(P, initial_price, n_days, initial_state, ticker):
    prices = [initial_price]
    state = initial_state
    for _ in range(n_days):
        next_state = np.random.choice(len(P), p=P[state])
        r = returns_map[ticker].get(next_state, 0)  # usa retorno médio do estado
        price = prices[-1] * (1 + r)
```

```
        prices.append(price)
        state = next_state
    return prices

n_days = len(data_test)
simulated_prices = {}
for ticker in tickers:
    last_price = data_matrix[ticker].iloc[-1]
    last_return = returns_matrix[ticker].iloc[-1]
    q1, q2 = quantiles[ticker]
    initial_state = classify_return_value(last_return, q1, q2)
    simulated_prices[ticker] = simulate_markov_prices(matrices[ticker], last_price,
    ↪  n_days, initial_state, ticker)

# ============================================================
# 5. Aprendizagem por Reforço - Q-Learning
# ============================================================
n_states, n_actions = 3, 2  # estados e ações (0 = short, 1 = long)
Q = np.zeros((n_states, n_actions))
alpha, gamma, epsilon = 0.1, 0.9, 0.1
episodes = 5000
np.random.seed(0)

def choose_action(state):
    if np.random.rand() < epsilon:
        return np.random.choice(n_actions)
    return np.argmax(Q[state])

# --- Treinamento ---
for ep in range(episodes):
    for ticker in tickers:
        states = states_train[ticker].values
        rets = returns_train[ticker].values
        for t in range(len(states)-1):
            s = states[t]
            a = choose_action(s)
            r = rets[t+1] if a == 1 else -rets[t+1]
            s_next = states[t+1]
            Q[s,a] = Q[s,a] + alpha * (r + gamma*np.max(Q[s_next]) - Q[s,a])

# ============================================================
# 6. Backtest em 2024 usando política aprendida
# ============================================================
positions, portfolios = {}, {}
for ticker in tickers:
    prices = [data_train[ticker].iloc[-1]]
    last_return = returns_train[ticker].iloc[-1]
    q1, q2 = quantiles[ticker]
    state = classify_return_value(last_return, q1, q2)
```

```
    pos_list, portfolio = [], [prices[0]]
    real_prices = data_test[ticker].values

    for i, date in enumerate(data_test.index):
        action = np.argmax(Q[state])  # política aprendida
        pos_list.append(action)
        price = real_prices[i]
        prices.append(price)

        ret = (prices[-1]-prices[-2])/prices[-2] if len(prices) >= 2 else 0
        portfolio.append(portfolio[-1] * (1 + action*ret))
        state = classify_return_value(ret, q1, q2)

    positions[ticker] = pos_list
    portfolios[ticker] = portfolio[1:]  # remove inicial


# 2) Real x Simulação de Markov x RL (somente PETR4)
plt.figure(figsize=(14,6))
ticker = "PETR4.SA"
plt.plot(data_test.index, data_test[ticker],
         label=f"Real {ticker}", color=colors[ticker]["real"], linestyle='-')
plt.plot(data_test.index, portfolios[ticker], '-.',
         label=f"RL Markov {ticker} - Período de 2 anos",
         ↪  color=colors[ticker]["RL"])
plt.ylabel("Preço, R$")
plt.legend(fontsize=9, loc="upper left")
plt.savefig("petr4_real_markov_rl_2ano.png", dpi=600, bbox_inches="tight")
plt.show()

# 3) Real x Simulação de Markov x RL (somente VALE3)
plt.figure(figsize=(14,6))
ticker = "VALE3.SA"
plt.plot(data_test.index, data_test[ticker],
         label=f"Real {ticker}", color=colors[ticker]["real"], linestyle='-')
plt.plot(data_test.index, portfolios[ticker], '-.',
         label=f"RL Markov {ticker} - Período de 2 anos",
         ↪  color=colors[ticker]["RL"])
plt.ylabel("Preço, R$")
plt.legend(fontsize=9)
plt.savefig("vale3_real_markov_rl_2ano.png", dpi=600, bbox_inches="tight")
plt.show()
```

APÊNDICE E – Código Python para modelagem RL considerando 6 meses e 3 meses de treino

Utilizado o mesmo código que o do Apêndice D, mas alterando o *data_train* no código:

```
data_train = yf.download(tickers, start="2023-07-01", end="2023-12-31",
↪  auto_adjust=True)['Close']
```

```
data_train = yf.download(tickers, start="2023-10-01", end="2023-12-31",
↪  auto_adjust=True)['Close']
```