PUC GOIÁS

DESAFIO IV JCPOLI 2025: INVESTE.AI — SUA IA, SEU BOT, SUA ESTRATÉGIA

"CRIATIVIDADE E INOVAÇÃO PARA UM FUTURO SUSTENTÁVEL"

IV JORNADA CIENTÍFICA DA ESCOLA POLITÉCNICA E DE ARTES

23 A 26 DE ABRIL DE 2025

Bolsa de Valores

Sistema de Trading
autônomo por IA.

Bem Vindos

Apresentado por:

Amaury Cordeiro Júnior - 20241013700145

Andrei Ferreira Inomata - 20241013700420

Sofia Souza Costa - 20241013700498

Guilherme Magalhães - 20241013700161

O que é Bolsa de Valores?

A bolsa de valores é uma instituição financeira onde são negociados diversos valores mobiliários como ações e títulos, conectando empresas que precisam de capital com investidores. Existem múltiplas bolsas ao redor do mundo, como a NYSE e Nasdaq nos EUA, a London Stock Exchange no Reino Unido e a B3 no Brasil. Cada bolsa opera com regulamentações próprias, mas todas compartilham o objetivo de facilitar negociações em ambiente regulamentado, proporcionando liquidez aos ativos financeiros, estabelecendo preços justos e permitindo que empresas captem recursos enquanto investidores participam do crescimento econômico.

História da

Bolsa de Valores

Das origens à modernidade

A primeira bolsa de valores formalmente estabelecida foi a **Bolsa de Amsterdã em 1602**, coincidindo com a criação da **Companhia Holandesa das Índias Orientais**, embora o conceito tenha raízes nas **feiras medievais europeias**. Esta instituição pioneira estabeleceu os princípios fundamentais que conectam diretamente o passado ao sistema financeiro global contemporâneo: emissão de ações negociáveis, formação de preços por oferta e demanda, e investimento sem participação direta na gestão. Ao longo dos séculos, essa estrutura evoluiu incorporando **tecnologias digitais** que transformaram negociações presenciais em **sistemas eletrônicos globais**, democratizando o acesso aos investimentos e desenvolvendo produtos financeiros sofisticados.

Como funciona

Bolsa de Valores

O Mecanismo por trás do Trading

A bolsa de valores opera através de um sistema centralizado que conecta compradores e vendedores via order book, onde são registradas todas as intenções de negociação com seus respectivos preços e volumes. O matching engine da bolsa funciona como um mediador automático, combinando ordens compatíveis quando um comprador está disposto a pagar o que um vendedor pede, permitindo diferentes estratégias como ordens a mercado ou ordens limitadas. Para garantir liquidez contínua, market makers mantêm ofertas constantes de compra e venda. Atualmente, este processo é predominantemente eletrônico e algorítmico, com sistemas capazes de executar milhares de operações por segundo, tudo sob rigorosa supervisão regulatória para assegurar transparência e equidade nas transações.

Tipos de

Bolsas de Valores

Mercado Primário e Secundário

- **Bolsas Tradicionais:** São físicas e têm um local específico onde ocorrem as negociações, como a B3 (Brasil), NYSE (EUA).
- **Bolsas Eletrônicas:** Operam totalmente online, sem presença física, como a NASDAQ.
- **Por tipo de ativo negociado:** Bolsa de ações, mercadorias e futuros, opções e criptomoedas
- **Por abrangência:** Regionais, Nacionais e Internacionais.

Primário

O mercado primário é o ambiente onde novos títulos e valores mobiliários são emitidos e vendidos pela primeira vez, permitindo que empresas e governos captem recursos diretamente dos investidores.

Neste mercado ocorrem eventos como IPOs (Ofertas Públicas Iniciais) e emissões de novas ações ou debêntures, sendo que o dinheiro investido vai diretamente para o emissor do título, geralmente com intermediação de instituições financeiras especializadas.

Secundário

O mercado secundário, por sua vez, é onde os títulos já emitidos são negociados entre investidores, sem que o dinheiro retorne para a empresa ou entidade emissora.

Este mercado proporciona liquidez aos títulos, permitindo que investidores possam vender seus ativos antes do vencimento.

As bolsas de valores são os principais exemplos de mercado secundário, onde diariamente milhões de transações ocorrem entre compradores e vendedores, estabelecendo o preço dos ativos conforme a oferta e demanda.

Por que investir

Bolsa de Valores

Benefícios

1 - **Potencial de valorização do capital** superior a investimentos conservadores no longo prazo;

2 - **Renda passiva regular** através de dividendos;

3 - **Alta liquidez** para conversão rápida em dinheiro quando necessário;

4 - **Diversificação** entre diferentes setores e mercados para redução de riscos;

5 - **Proteção contra inflação**, já que ações representam participação em ativos reais;

6 - **Democratização do acesso ao crescimento econômico**, permitindo que pequenos investidores participem do desenvolvimento de grandes empresas;

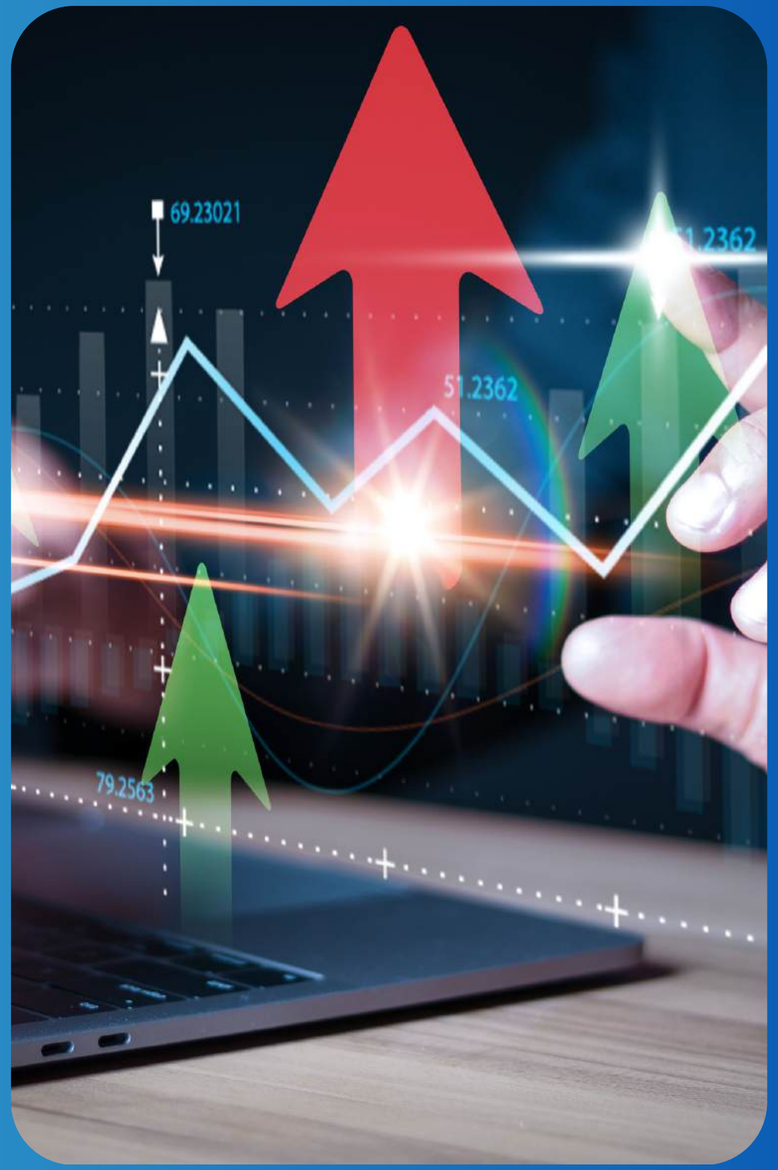
Riscos da Bolsa de Valores

Investimento

Entendendo potenciais armadilhas

Investir na bolsa de valores implica diversos riscos, incluindo a volatilidade de mercado, vieses comportamentais que levam a decisões emocionais, dificuldades de timing, assimetria informacional, problemas de liquidez e concentração excessiva em poucos ativos. Estes desafios podem comprometer significativamente os retornos e expor investidores a perdas substanciais, especialmente durante períodos de turbulência nos mercados.

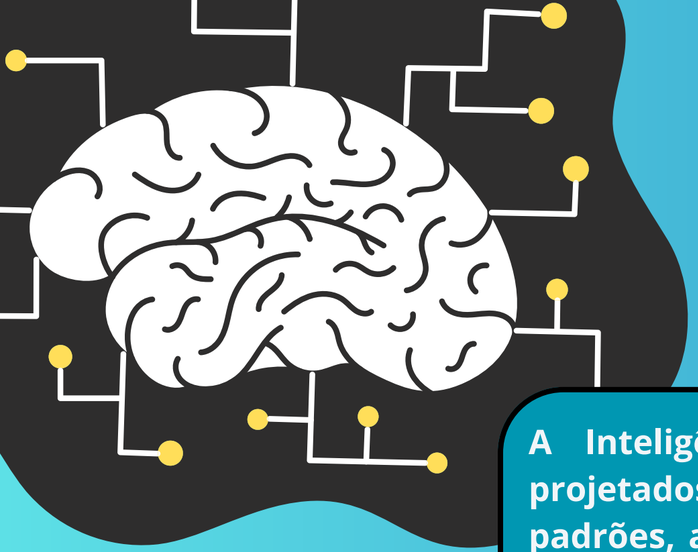
A tecnologia moderna oferece soluções poderosas para mitigar esses riscos através da inteligência artificial e robôs de trading. Algoritmos de IA podem processar enormes volumes de dados em tempo real, identificando padrões e tendências emergentes que escapam à percepção humana, enquanto sistemas automatizados de trading eliminam o componente emocional das decisões de investimento, implementando estratégias disciplinadas com recursos, como diversificação automática, stop-loss precisos e rebalanceamento dinâmico de carteiras. Embora não eliminem completamente os riscos, estas ferramentas tecnológicas permitem um gerenciamento mais eficiente e sistemático dos investimentos.



O uso de Inteligência Artificial e Robôs de Trading

Automatização dos processos

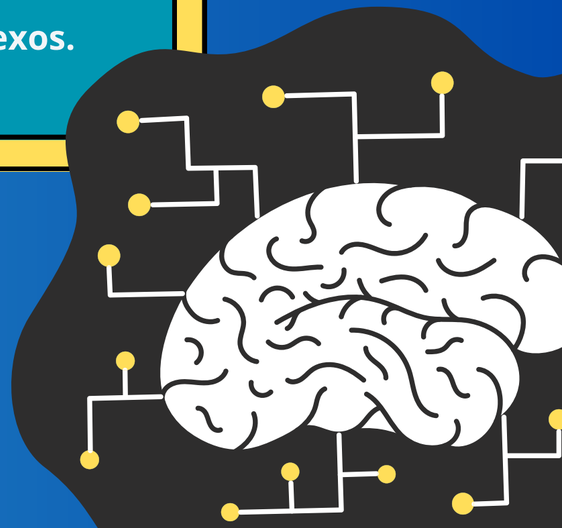
Um robô de trading com IA é um sistema automatizado que executa operações nos mercados financeiros utilizando algoritmos de inteligência artificial para tomar decisões de investimento. Diferente dos robôs tradicionais baseados em regras fixas, estes sistemas utilizam técnicas avançadas como redes neurais, aprendizado profundo e processamento de linguagem natural para analisar enormes volumes de dados – incluindo preços históricos, indicadores técnicos, notícias e sentimento de mercado – identificando padrões complexos que seriam imperceptíveis para humanos. Sua principal vantagem está na capacidade de aprendizado contínuo, adaptando-se às mudanças no mercado e refinando suas estratégias com base nos resultados anteriores.



COMO A IA FUNCIONA

A Inteligência Artificial (IA) fundamentalmente consiste em sistemas computacionais projetados para simular aspectos da inteligência humana, processando dados para identificar padrões, aprender com experiências e tomar decisões. O funcionamento básico da IA envolve algoritmos que analisam grandes volumes de dados, extraem características relevantes, constroem modelos matemáticos para representar relações encontradas, e depois utilizam esses modelos para fazer previsões ou tomar decisões quando expostos a novos dados

Seus principais tipos incluem:
Aprendizado Supervisionado - usa dados rotulados para fazer previsões;
Aprendizado Não-Supervisionado - descobre padrões em dados não rotulados;
Aprendizado por Reforço - aprende por tentativa e erro com sistema de recompensas;
Redes Neurais - inspiradas no cérebro humano para reconhecimento de padrões complexos.



Aprendizado por Reforço

O Aprendizado por Reforço (RL) é uma vertente da Inteligência Artificial que permite a um agente, como o nosso robô de trading, aprender a tomar decisões ótimas de forma autônoma. Isso ocorre através de um ciclo contínuo de interações com um ambiente: o agente observa o estado atual, escolhe uma ação, recebe uma recompensa (positiva por um bom resultado, negativa por um ruim) e transita para um novo estado. O objetivo do agente é desenvolver uma política, ou seja, uma estratégia, que maximize a soma das recompensas recebidas ao longo do tempo, aprendendo com base na exploração de novas possibilidades e no aproveitamento de táticas que se mostraram bem-sucedidas anteriormente.

No domínio do trading financeiro, o mercado se torna o ambiente, e os dados de mercado (como preços históricos, volumes e indicadores técnicos) constituem o estado que o agente analisa. As ações do robô são as operações de comprar, vender ou manter posições em ativos, e as recompensas são diretamente derivadas dos lucros ou prejuízos financeiros resultantes dessas operações. Para este projeto, implementamos o algoritmo A2C (Advantage Actor-Critic). Este é um método avançado que permite ao robô não apenas aprender quais ações são vantajosas em determinados cenários de mercado, mas também a avaliar a qualidade desses cenários, otimizando assim suas estratégias de negociação para diferentes condições e buscando resultados consistentes.

Separação dos Dados

A base de dados utilizada para treinar e validar nosso robô de trading é composta por séries temporais históricas de ações. Estes dados abrangem os períodos de 01/01/2017 a 31/12/2019 e de 01/01/2022 a 31/12/2024. Os dados foram fornecidos ao sistema através de um arquivo no formato CSV, um método comum para importar cotações que podem ser originadas de diversas fontes, incluindo APIs financeiras como a do yfinance.

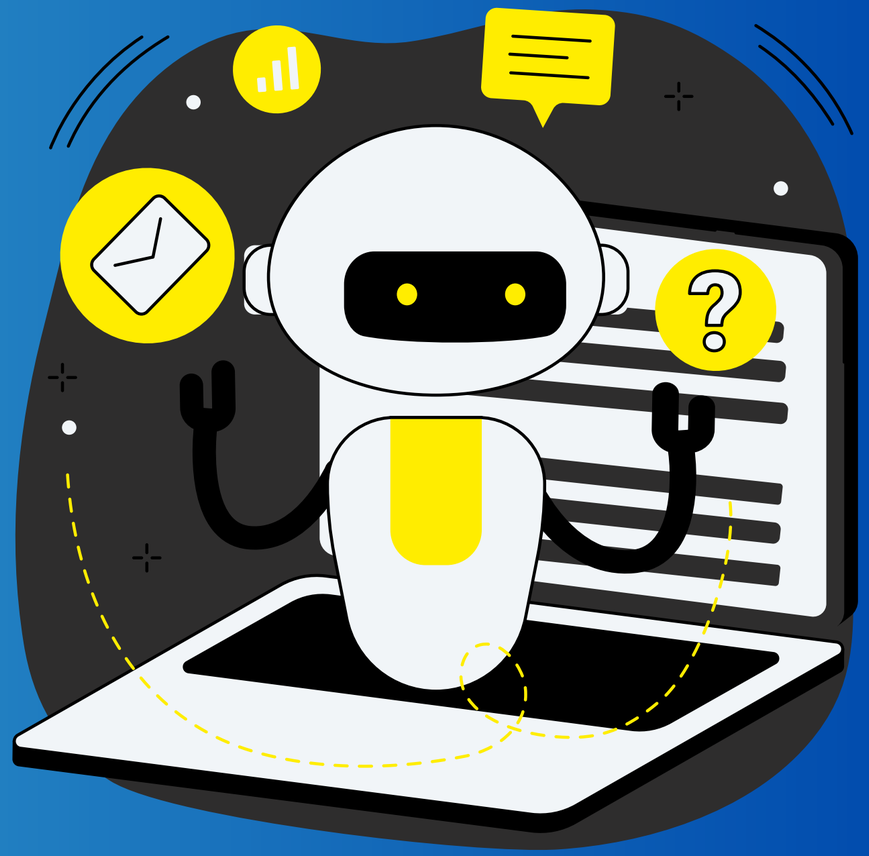
Cada registro no conjunto de dados contém informações cruciais para a análise e decisão do robô, estruturadas nas seguintes colunas principais: Date (a data da cotação, utilizada como referência temporal), Ticker (o símbolo identificador do ativo na bolsa), Open (preço de abertura do ativo no período), High (preço máximo atingido), Low (preço mínimo), Close (preço de fechamento) e Volume (quantidade de ativos negociados). Antes de serem efetivamente utilizados pelo modelo de inteligência artificial, esses dados brutos passam por uma etapa fundamental de pré-processamento, que inclui a limpeza de possíveis inconsistências, o tratamento de dados faltantes e a correta indexação e ordenação temporal.

Algoritmos e Estratégia

A estratégia de negociação empregada por este robô não é baseada em um conjunto fixo de regras pré-programadas. Ela é **aprendida dinamicamente** pelo agente de Inteligência Artificial, **o modelo A2C (Advantage Actor-Critic)**.

Durante a fase de treinamento, o agente **interage milhares de vezes com o ambiente de mercado simulado**, realizando operações de compra, venda ou manutenção de posição. A cada ação, ele **recebe um feedback na forma de uma recompensa** (geralmente relacionada ao lucro ou prejuízo da operação). Através deste processo intensivo de tentativa, erro e recompensa, o agente A2C **desenvolve autonomamente uma política de decisão**, aprendendo a identificar padrões e correlações nos dados de mercado que, historicamente, se mostraram preditivos de movimentos de preço favoráveis e lucrativos.

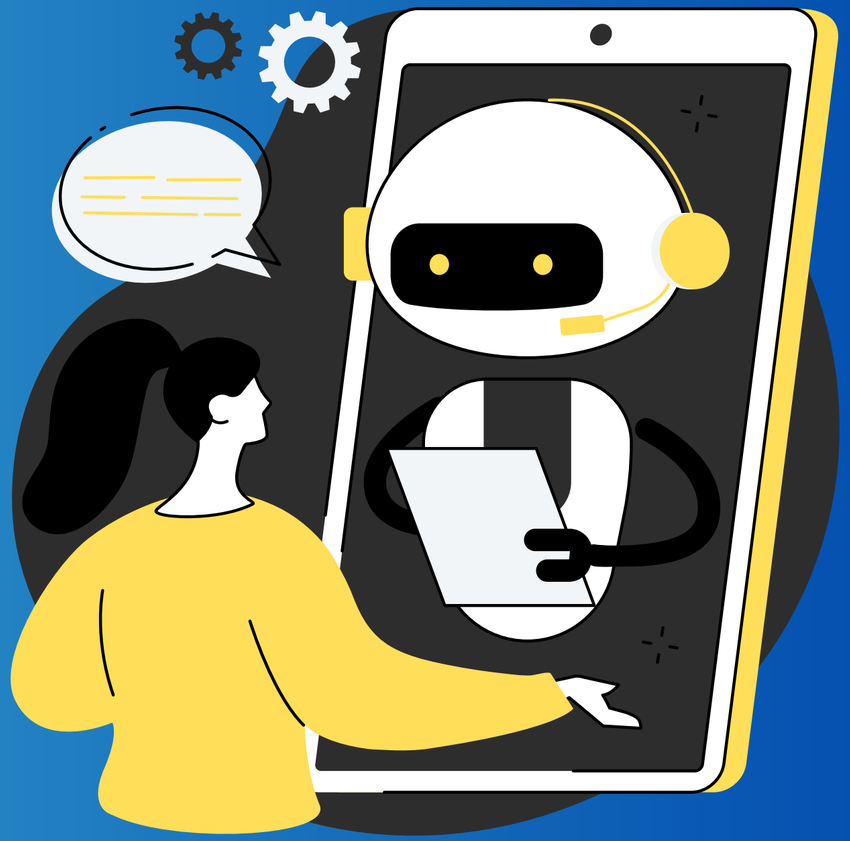
Para formular sua estratégia em cada ponto de decisão, o robô **analisa o estado atual do mercado**, que é composto por uma janela de observação dos dados históricos mais recentes do ativo. Este estado não inclui apenas os preços de abertura, máxima, mínima, fechamento e o volume de negociações, mas também os valores dos indicadores técnicos que calculamos: a **Média Móvel Simples** (SMA), o **Índice de Força Relativa** (RSI) e o **Saldo de Volume** (OBV). Com base nessa combinação de informações, a política (estratégia) aprendida pelo A2C determina a ação que tem a maior probabilidade de levar a uma recompensa positiva.



Visualização – Streamlit

Para facilitar a análise e a interpretação dos resultados do nosso robô de trading, desenvolvemos um **dashboard interativo** utilizando a biblioteca **Streamlit** em Python. Esta ferramenta nos permite criar interfaces web ricas e dinâmicas diretamente a partir de **scripts Python**, tornando a exploração dos dados de backtesting e do comportamento do modelo de Aprendizado por Reforço **muito mais acessível e intuitiva**. O objetivo principal do nosso dashboard é oferecer uma visão clara do **desempenho da estratégia**, permitindo que **o usuário carregue seus próprios dados de mercado em formato CSV**, **selecione ativos específicos** (tickers) para uma análise detalhada, e configure parâmetros chave da simulação, como o **valor do investimento inicial** para o cálculo do lucro em Reais e a **duração do treinamento do agente** (em timesteps).

O dashboard apresenta os resultados de forma organizada e visual. Para cada ticker analisado, são exibidos: o **lucro obtido** (tanto em termos percentuais quanto no valor monetário em Reais correspondente ao investimento inicial), a **recompensa total acumulada** pelo agente de RL, e um **gráfico detalhado** da simulação que mostra a evolução dos preços do ativo sobreposto com os pontos exatos onde o modelo decidiu realizar operações de compra (verde) e venda (vermelho). Adicionalmente, uma **tabela qualitativa** oferece um resumo das principais métricas de desempenho e os logs de processamento podem ser consultados para acompanhar as etapas da análise. A interface também oferece a funcionalidade de **analisar todos os tickers do arquivo de dados** em lote, com uma barra de progresso indicando o andamento, e ao final, apresenta um resumo comparativo da performance entre os diferentes ativos. Para demonstração e acesso facilitado, especialmente no ambiente **Google Colab**, esta aplicação Streamlit foi disponibilizada através de um **túnel ngrok**, tornando a interface acessível por meio de um **link público no navegador:**
https://b1ae-34-66-45-144.ngrok-free.app/

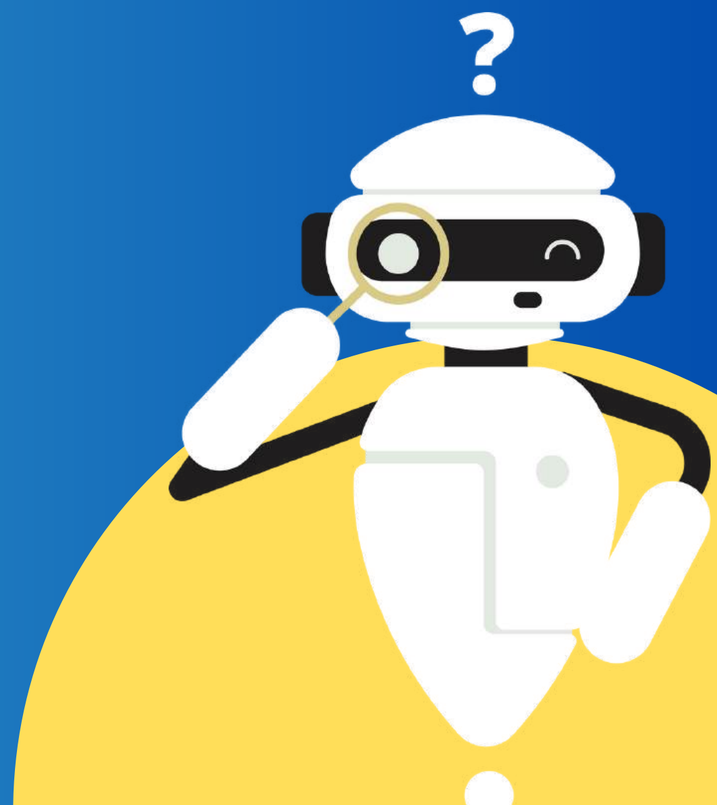


Modelo – Instalação

A fase inaugural do projeto consistiu na configuração do ambiente de desenvolvimento no Google Colaboratory, uma plataforma baseada em nuvem que facilita a execução de código Python e a colaboração. O primeiro bloco de comando é dedicado ao provisionamento das dependências de software. Sua execução garante que todas as bibliotecas Python e ferramentas especializadas, requeridas para as diversas etapas do projeto, sejam corretamente instaladas e estejam prontas para utilização, estabelecendo assim a infraestrutura computacional necessária.

A instalação dessas dependências é realizada através de um gerenciador de pacotes Python, que automaticamente busca, baixa e instala as versões especificadas ou as mais recentes disponíveis nos repositórios públicos. Este processo é crucial para assegurar a compatibilidade entre os diferentes componentes do sistema e para permitir o acesso às funcionalidades completas e atualizadas de cada biblioteca, desde a manipulação de dados até o treinamento de modelos de inteligência artificial e a criação da interface gráfica do usuário.

Os componentes de software instalados neste bloco são críticos e abrangem um espectro de funcionalidades: bibliotecas para a construção da interface gráfica interativa e para a sua exposição na web quando executada em ambientes como o Colab; pacotes especializados para a criação e simulação de ambientes de negociação financeira, adaptados para aprendizado por reforço; um framework robusto para a implementação de algoritmos de aprendizado por reforço, como o A2C; e, finalmente, bibliotecas fundamentais para a manipulação eficiente de estruturas de dados, para a computação numérica de alto desempenho e para a geração de visualizações gráficas detalhadas.

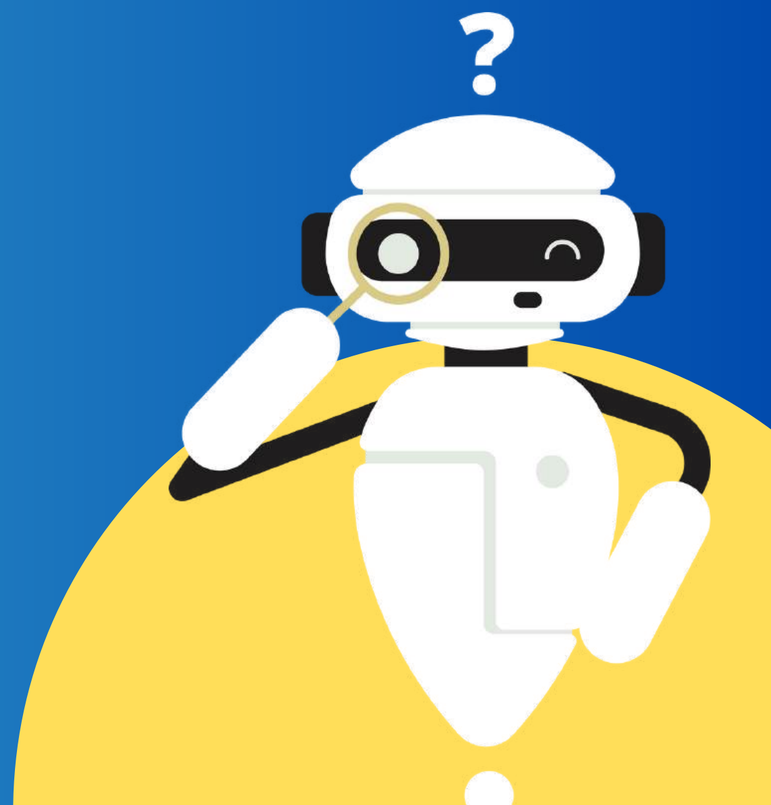


Modelo – Motor de Análise

Com o ambiente de desenvolvimento devidamente configurado, a etapa subsequente concentrou-se na construção do módulo central do sistema, o backtesting_engine.py. Este componente foi projetado para encapsular toda a lógica fundamental de processamento de dados, engenharia de features, simulação do ambiente de negociação e avaliação de estratégias de trading. Sua concepção como um módulo independente visa promover a organização, a reutilização de código e a facilidade de manutenção do sistema.

Internamente, este motor integra diversas funcionalidades críticas. Entre elas, destacam-se a definição de um ambiente de negociação customizado (MyCustomEnv), que herda de uma classe base de simulação de trading (StocksEnv) e é adaptado para incorporar os sinais derivados de indicadores técnicos através da função add_signals. O módulo também contém rotinas para o carregamento e validação de dados de mercado a partir de arquivos CSV (carregar_dados_csv_master), procedimentos para o pré-processamento e engenharia de features (preprocess_and_feature_engineer_for_ticker), incluindo o cálculo de indicadores de tendência, momento e volume, e utilitários para a formatação de resultados financeiros em moeda local (format_brl).

Uma das responsabilidades primordiais do backtesting_engine.py é a função testar_ticker_com_modelo_global. Esta função é designada para carregar um modelo de aprendizado por reforço previamente treinado (o modelo global) e avaliar seu desempenho quando aplicado a novos conjuntos de dados de teste para um ticker específico. A estrutura modular deste motor permite que tanto os scripts de treinamento quanto a interface de visualização interativa utilizem suas funcionalidades de forma consistente.



Modelo – Treinamento Global

Após a estruturação do motor de análise, o foco se voltou para a criação do script train_global_model.py, responsável por orquestrar o processo de treinamento de um único modelo de inteligência artificial generalizado. A concepção deste modelo global visa permitir que o agente aprenda estratégias de negociação a partir da análise de um conjunto consolidado de dados históricos provenientes de múltiplos ativos financeiros, em vez de se especializar em um único ativo. O objetivo é fomentar a capacidade do modelo de identificar padrões de mercado mais amplos e generalizáveis.

O funcionamento deste script inicia com o carregamento da base de dados de treinamento completa. Para cada ativo financeiro (ticker) presente no conjunto de dados, são aplicadas as rotinas de pré-processamento e engenharia de features definidas no backtesting_engine.py. Uma etapa crucial aqui é a adição de uma feature de identificação normalizada para cada ticker, permitindo que o modelo global distinga e potencialmente adapte sua estratégia para diferentes ativos. O algoritmo de aprendizado por reforço (A2C) é então treinado de forma iterativa, sendo exposto sequencialmente aos dados processados de cada ticker para acumular conhecimento e otimizar sua política de negociação.

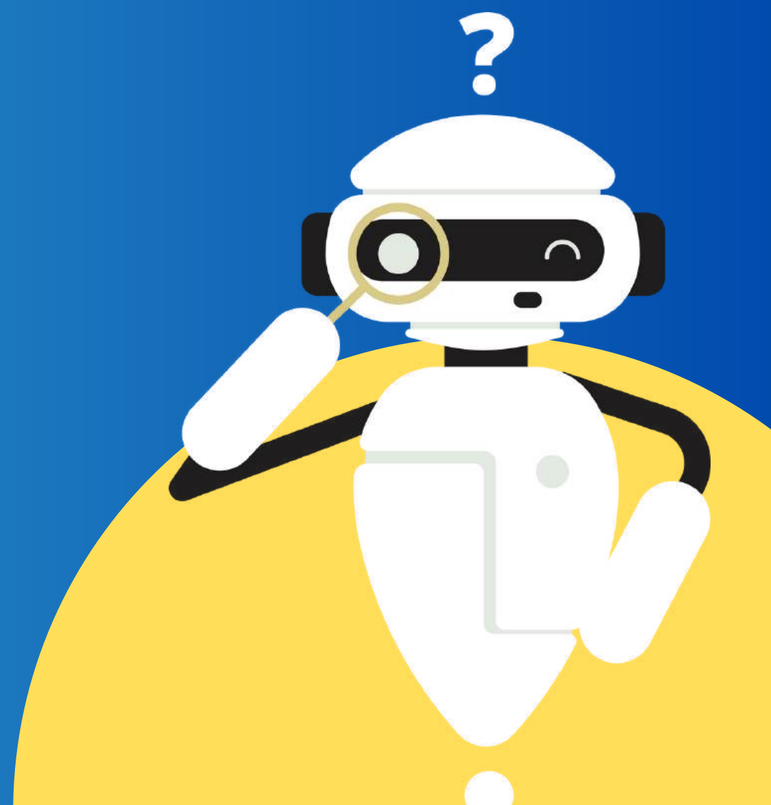
Ao concluir o ciclo de treinamento sobre todo o conjunto de ativos, o script train_global_model.py gera dois artefatos principais que são persistidos em disco. O primeiro é o arquivo contendo o próprio modelo de aprendizado por reforço global treinado. O segundo é um arquivo de mapeamento que associa cada identificador de ticker ao seu correspondente valor normalizado utilizado como feature durante o treinamento. Ambos os artefatos são indispensáveis para a etapa subsequente de avaliação do desempenho do modelo em dados não vistos

Modelo – Dados de Treino

Uma vez que os scripts para o motor de análise e para o treinamento do modelo global estão definidos, a próxima fase requer uma intervenção manual do usuário: o carregamento do arquivo CSV que contém os dados históricos de treinamento. Este conjunto de dados representa a fonte primária de informação a partir da qual o agente de aprendizado por reforço irá aprender e desenvolver suas estratégias de negociação.

Este arquivo CSV deve ser abrangente, contendo o histórico de cotações – incluindo Date, identificador do Ticker, preços de Abertura (Open), Máxima (High), Mínima (Low), Fechamento (Close) e Volume – para todos os ativos financeiros que serão utilizados para ensinar o modelo global. É fundamental que o nome do arquivo carregado no ambiente do Google Colab corresponda precisamente ao nome especificado na variável de configuração dentro do script train_global_model.py, para assegurar que o processo de treinamento utilize a base de dados correta.

A qualidade, a integridade e a representatividade dos dados neste arquivo são de suma importância, pois influenciam diretamente a capacidade do modelo de aprender padrões significativos e de generalizar seu conhecimento para novas situações de mercado. As etapas de pré-processamento definidas anteriormente visam tratar ruídos e inconsistências, mas a validade fundamental dos dados de entrada é um pré-requisito para o sucesso do treinamento.

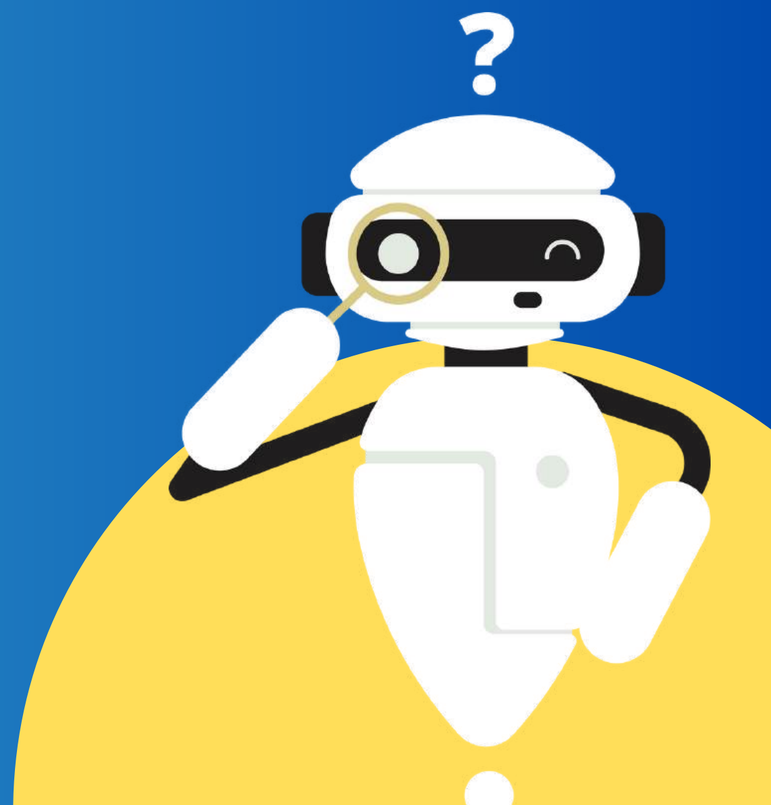


Modelo – Treino Offline

Com o conjunto de dados de treinamento devidamente carregado no ambiente, a próxima célula do notebook é responsável por iniciar o processo de treinamento offline do modelo global A2C. A execução do comando !python train_global_model.py aciona o script que orquestra todo o pipeline de aprendizado, desde o processamento dos dados de cada ticker até a otimização iterativa do agente de RL.

Durante esta fase, o sistema utiliza as funções do backtesting_engine.py para pré-processar os dados de cada ativo, incluindo o cálculo dos indicadores técnicos (SMA, RSI, OBV) e a adição da feature de identificação do ticker. O modelo A2C é então treinado sequencialmente com os dados de cada um desses ativos, permitindo que aprenda a partir de uma gama diversificada de cenários de mercado. Este é um processo que pode ser computacionalmente intensivo, com sua duração variando de acordo com o volume de dados, o número de tickers e os parâmetros de treinamento, como o total de passos de tempo (total_timesteps).

Ao final desta execução, que pode levar um tempo considerável, os principais resultados são persistidos no ambiente do Colab. Estes incluem o arquivo do modelo global A2C treinado e o arquivo JSON contendo o mapeamento dos identificadores de ticker. Estes artefatos representam a "inteligência" aprendida pelo robô e são essenciais para as etapas subsequentes de teste e avaliação de desempenho.

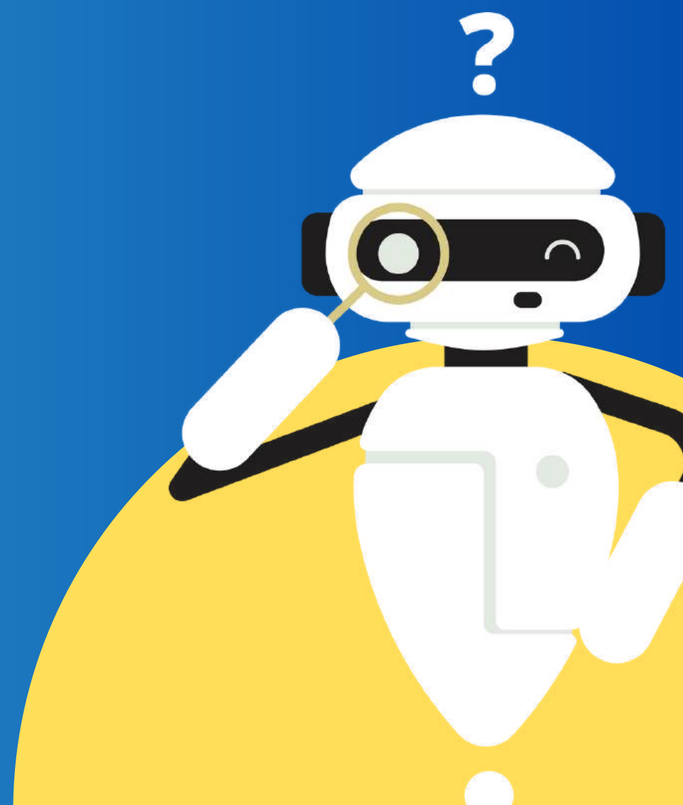


Modelo – Interface Gráfica

Após o treinamento e salvamento do modelo global, o foco se desloca para a criação de uma ferramenta que permita testar e visualizar seu desempenho de forma interativa. O próximo bloco de comando utiliza o comando %%writefile para gerar o arquivo app_streamlit_global_tester.py, que contém o código para uma aplicação web construída com a biblioteca Streamlit.

O propósito desta aplicação é servir como uma interface gráfica do usuário (GUI) dedicada ao teste do modelo global pré-treinado. Ela é projetada para permitir que o usuário carregue um novo conjunto de dados de mercado (os dados de teste), selecione tickers específicos deste conjunto para análise, e defina parâmetros para a simulação, como o valor de um investimento inicial hipotético. A interface então utiliza o backtesting_engine.py para carregar o modelo global e simular suas operações nos dados de teste fornecidos.

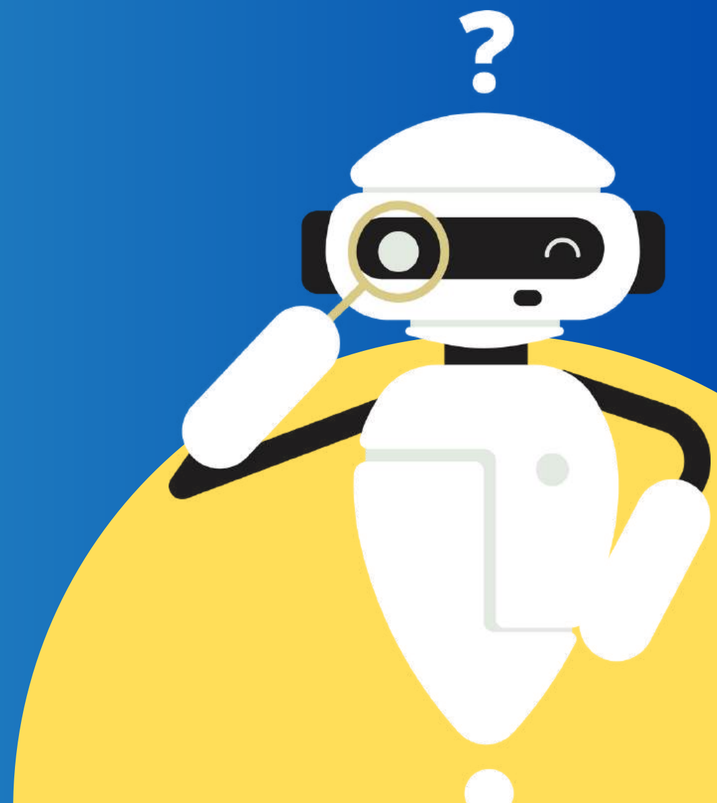
A grande vantagem desta abordagem é a facilidade de uso e a capacidade de visualização. Os resultados da simulação, incluindo gráficos de desempenho, métricas de lucro e prejuízo (tanto em termos percentuais quanto monetários), e outras informações relevantes, são apresentados de forma clara e organizada diretamente no navegador web, tornando a avaliação do modelo acessível mesmo para usuários sem profundo conhecimento técnico do processo de treinamento subjacente.



Modelo – Dados de Teste

Assim como na etapa de treinamento, esta fase envolve uma ação manual do usuário, crucial para a correta avaliação do modelo. O usuário deve realizar o upload de um arquivo CSV contendo o conjunto de dados que será utilizado exclusivamente para testar o desempenho do modelo global treinado. É altamente recomendável que este conjunto de dados seja distinto daquele usado no treinamento, cobrindo, por exemplo, um período de tempo diferente ou ativos não vistos anteriormente pelo modelo, para uma análise fidedigna de sua capacidade de generalização.

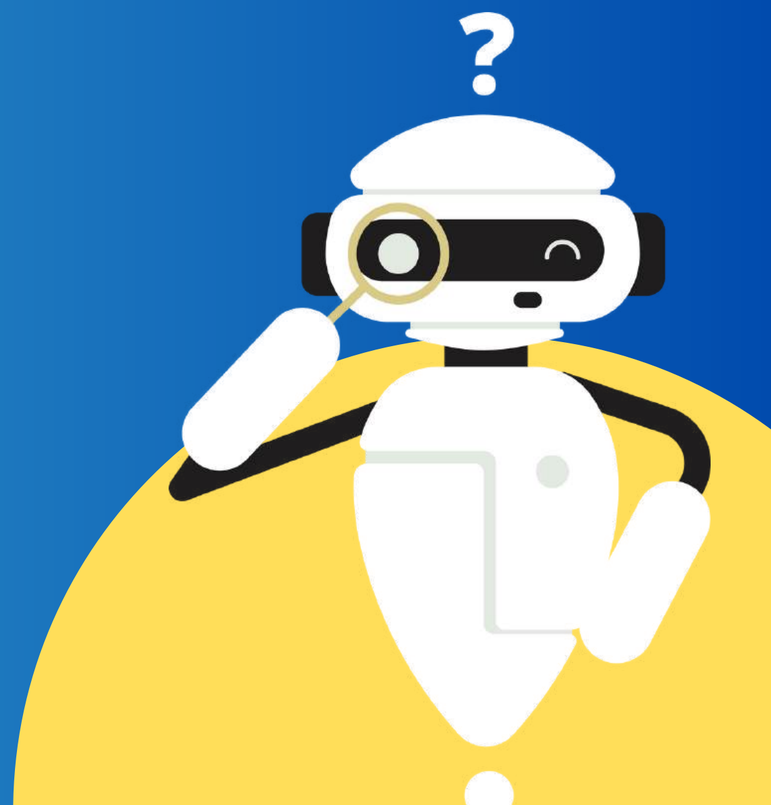
Adicionalmente, antes de iniciar o aplicativo Streamlit de teste, é fundamental assegurar que os artefatos gerados durante a fase de treinamento offline estejam presentes e acessíveis no ambiente do Colab. Isso inclui o arquivo do modelo A2C global salvo (ex: A2C_global_model.zip) e o arquivo de mapeamento de ticker para ID (ticker_to_id_map.json). O app_streamlit_global_tester.py dependerá desses arquivos para carregar a estratégia aprendida e para preparar corretamente os dados de teste com as features de identificação de ticker correspondentes.



Modelo – Acesso Web

A célula final deste processo é dedicada ao lançamento da aplicação **Streamlit de teste** (app_streamlit_global_tester.py) e à sua disponibilização para acesso através da internet. Inicialmente, o código configura o **ngrok** utilizando o **authtoken** pessoal do usuário, uma etapa importante para garantir a estabilidade e funcionalidade do túnel de conexão que será criado. Comandos são também executados para tentar encerrar quaisquer processos streamlit ou ngrok de sessões anteriores, prevenindo possíveis conflitos de porta.

Uma vez que o ambiente está preparado, o servidor da aplicação Streamlit é iniciado em segundo plano no ambiente do Colab. A ferramenta ngrok então estabelece uma conexão segura entre a porta local onde o Streamlit está rodando e um endereço URL público na internet. Este link é exibido na saída da célula, e ao acessá-lo em um navegador web, **o usuário pode interagir com o dashboard de teste**. Dentro do aplicativo, será possível carregar os dados de teste, selecionar os tickers de interesse e visualizar, após uma rápida simulação, o desempenho do modelo global treinado, analisando os gráficos, métricas financeiras e logs de processamento. A célula também inclui orientações para auxiliar na resolução de problemas comuns do ngrok, como o erro de limite de sessões.



Referências

Documentação:
Streamlit: https://docs.streamlit.io/
Pyngrok: https://pypi.org/project/pyngrok/
Gymnasium :https://gymnasium.farama.org/
Gym: https://www.gymlibrary.dev/index.html
Stable-Baselines3: https://stable-baselines3.readthedocs.io/en/master/
Matplotlib: https://matplotlib.org/stable/index.html
Numpy: https://numpy.org/doc/
Pandas: https://pandas.pydata.org/docs/

Máquina Virtual:
Google Colab: https://colab.google