RELATÓRIO TÉCNICO - VIII DESAFIO EM CIÊNCIAS DE DADOS

****PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS****

****Implementação de Agente A2C para Trading com Interface Interativa Streamlit****

Grupo: Nulos e Vazios

Amaury Cordeiro Junior

Andrei Ferreira Inomata

Guilherme Magalhães Lima

Sofia Souza Costa

****Resumo****

Este trabalho apresenta o desenvolvimento e a implementação de um agente de trading automatizado baseado no algoritmo de aprendizado por reforço A2C (Advantage Actor-Critic), integrado a uma interface gráfica interativa construída com Streamlit. O sistema é projetado para tomar decisões de compra, venda ou manutenção de posições em ativos financeiros, utilizando dados históricos de cotações fornecidos em formato CSV. A metodologia abrange o pré-processamento dos dados, a engenharia de features com indicadores técnicos (SMA, RSI, OBV) calculados manualmente, a criação de um ambiente de negociação personalizado herdando de `gym-anytrading`, o treinamento do agente A2C utilizando `Stable Baselines3`, e sua subsequente avaliação em dados não vistos. A interface Streamlit permite ao usuário carregar seus dados, selecionar tickers, definir parâmetros de simulação como investimento inicial e duração do treinamento, e visualizar os resultados através de gráficos de simulação, métricas de lucro (percentual e monetário) e tabelas qualitativas. O projeto visa oferecer uma ferramenta prática para experimentação e análise de estratégias de trading baseadas em aprendizado por reforço, no contexto do VIII Desafio em Ciência de Dados - Investe.AI.

****1. Introdução e Objetivos****

Este trabalho detalha o desenvolvimento, treinamento e avaliação de um agente de trading automatizado que utiliza o algoritmo de aprendizado por reforço A2C (Advantage Actor-Critic). Uma componente chave do projeto é a criação de uma interface gráfica interativa com Streamlit, que facilita a análise de desempenho e a experimentação com diferentes ativos. O sistema foi concebido para o VIII Desafio

em Ciência de Dados - Investe.AI, cujo tema é "Investe.AI Sua IA, Seu Bot, Sua Estratégia".

O objetivo principal consiste em criar um agente capaz de aprender estratégias de trading a partir de dados históricos de mercado e demonstrar sua eficácia em simulações. Especificamente, o agente deve ser capaz de processar dados de cotações, utilizar indicadores técnicos para embasar suas decisões, e aprender uma política de negociação que maximize uma função de recompensa relacionada ao lucro. A interface Streamlit visa tornar o processo de teste e avaliação transparente e acessível, permitindo a visualização do comportamento do agente e de suas métricas de desempenho para diferentes tickers. Utilizamos dados históricos fornecidos em formato CSV, com o sistema dividindo os dados de cada ticker em conjuntos de treinamento e avaliação para testar a generalização do modelo.

****2. Fundamentação Teórica****

****2.1 Aprendizado por Reforço (RL)****

O Aprendizado por Reforço é um paradigma de aprendizado de máquina onde um agente aprende a tomar uma sequência de decisões interagindo com um ambiente para atingir um objetivo. O agente recebe feedback na forma de recompensas (positivas ou negativas) por suas ações e aprende, por tentativa e erro, uma política (estratégia) que maximiza a recompensa total esperada a longo prazo. Componentes chave do RL incluem o agente, o ambiente, o estado (uma representação da situação atual do ambiente), as ações (decisões que o agente pode tomar) e a função de recompensa. No contexto de robôs de trading, o mercado financeiro é o ambiente, os dados de mercado (preços, volumes, indicadores) compõem o estado, as ações são comprar/vender/manter, e as recompensas são baseadas nos resultados financeiros.

****2.2 Advantage Actor-Critic (A2C)****

O A2C é um algoritmo popular de aprendizado por reforço do tipo "on-policy" e "actor-critic". Ele possui duas redes neurais principais: o ****Ator (Policy Network)****, que aprende a política (qual ação tomar em um determinado estado), e o ****Crítico (Value Network)****, que aprende a função de valor (estima quão bom é estar em um determinado estado ou tomar uma determinada ação). O A2C utiliza a "função de vantagem" – a diferença entre o retorno real obtido e o valor estimado pelo crítico – para guiar as atualizações de ambas as redes, promovendo um aprendizado mais

estável e eficiente. Para este projeto, utilizamos a implementação do A2C fornecida pela biblioteca `Stable Baselines3`.

****2.3 Gymnasium, gym-anytrading e Streamlit****

* ****Gymnasium:**** Sucessor do OpenAI Gym, é uma biblioteca padrão que fornece uma interface comum para o desenvolvimento e comparação de algoritmos de aprendizado por reforço, definindo a estrutura básica de interação entre agente e ambiente.

* ****gym-anytrading:**** É uma biblioteca que estende o Gymnasium para criar ambientes especificamente voltados para a simulação de trading de ações e outros instrumentos financeiros. Ela gerencia aspectos como o estado da carteira, cálculo de lucros e o processamento de dados de séries temporais financeiras.

* ****Streamlit:**** É um framework Python open-source que permite criar e compartilhar aplicativos web interativos para projetos de dados e aprendizado de máquina com poucas linhas de código, facilitando a visualização e a interação com os modelos.

****3. Metodologia****

****3.1 Estrutura do Código e Ambiente de Desenvolvimento****

A implementação foi realizada integralmente em Python, utilizando o Google Colab como ambiente de desenvolvimento e execução. O código foi modularizado em dois arquivos principais:

* `backtesting_engine.py`: Contém a lógica central de processamento de dados, cálculo de indicadores, definição do ambiente de RL, treinamento do modelo A2C e sua avaliação.

* `app_streamlit_online.py`: Responsável pela criação da interface gráfica do usuário (GUI) com Streamlit, que interage com o `backtesting_engine.py` para executar análises e exibir resultados.

As bibliotecas chave incluem `pandas` e `numpy` para manipulação de dados, `matplotlib` para gráficos, `gymnasium` e `gym-anytrading` para o ambiente de RL, `Stable Baselines3` para o algoritmo A2C, e `Streamlit` para a interface. `pyngrok` é utilizado para expor a aplicação Streamlit do Colab na web.

****3.2 Preparação e Análise dos Dados****

Os dados históricos de ativos financeiros são carregados a partir de um arquivo CSV fornecido pelo usuário. O processo para cada ticker selecionado envolve:

1. **Carregamento e Filtragem:** O DataFrame principal é carregado, e os dados são filtrados para o `Ticker` específico.

2. **Indexação e Limpeza:** A coluna `Date` é convertida para datetime e definida como índice. As colunas OHLCV (Open, High, Low, Close, Volume) são convertidas para tipo numérico, e valores ausentes são tratados por preenchimento (`ffill`, `bfill`).

3. **Engenharia de Features:** Os seguintes indicadores técnicos são calculados manualmente:

   ***SMA (Média Móvel Simples):** Período de 12 dias sobre o preço de fechamento.

   ***RSI (Índice de Força Relativa):** Período de 14 dias.

   ***OBV (On-Balance Volume):** Baseado no preço de fechamento e volume.

   Os valores NaN resultantes (no início da série) são preenchidos com 0.

3.3 Ambiente de Trading Personalizado

Uma classe `MyCustomEnv` foi criada, herdando de `StocksEnv` (`gym-anytrading`), para integrar nossos indicadores ao estado observado pelo agente.

* **Função `add_signals`:** Processa o DataFrame do ticker, selecionando `Low`, `Volume`, `SMA`, `RSI` e `OBV` para compor a observação do agente.

* **Estado:** Uma janela (`window_size`) dos dados de preço e indicadores mais recentes.

* **Ações:** Comprar, vender ou manter a posição.

* **Recompensa:** Baseada no lucro/prejuízo percentual das operações, conforme definido pelo `gym-anytrading`.

3.4 Arquitetura de Rede e Algoritmo

O agente de trading utiliza o modelo **A2C** de `Stable Baselines3` com a política padrão `MlpPolicy` (Perceptron Multicamadas). Esta rede neural mapeia o estado do ambiente (observações) para uma distribuição de probabilidade sobre as ações possíveis.

3.5 Processo de Treinamento

Para cada ticker:

1. **Divisão dos Dados:** Os dados são divididos em conjuntos de treinamento (ex: 70%) e avaliação (ex: 30%).

2. **Ambiente de Treino:** Uma instância de `MyCustomEnv` é criada com os dados de treinamento.

3. **Treinamento:** O modelo A2C é treinado (`model.learn()`) por um número de `total_timesteps` definido pelo usuário.

3.6 Teste e Avaliação

1. **Ambiente de Avaliação:** Uma instância de `MyCustomEnv` é criada com os dados de avaliação.

2. **Execução:** O agente treinado opera de forma determinística no ambiente de avaliação.

3. **Métricas:** `total_reward` e `total_profit` (lucro percentual) são coletados.

4. **Lucro em R$:** Calculado multiplicando o `total_profit` pelo "Valor de Investimento Inicial (R$)" fornecido pelo usuário.

5. **Visualização:** Um gráfico Matplotlib é gerado mostrando as operações.

---

4. Resultados e Análises

4.1 Métricas de Desempenho

O sistema reporta, para cada ticker analisado através da interface Streamlit:

* **Lucro/Prejuízo (%):** Resultado percentual no período de avaliação.

* **Lucro/Prejuízo (R$):** Estimativa monetária baseada no investimento inicial definido.

* **Recompensa Total do Agente:** Soma das recompensas do agente na avaliação.

* **Tabela Qualitativa:** Resumo incluindo ticker, investimento, recompensas, lucros e status do treinamento.

4.2 Visualização de Resultados com Streamlit (`app_streamlit_online.py`)

A interface Streamlit é o principal meio de interação e visualização:

* **Carregamento de Dados e Configuração:** Permite upload de CSV, seleção de ticker, definição de investimento inicial e timesteps de treinamento.

* **Modos de Análise:** Suporta análise de um ticker individual ou de todos os tickers do arquivo em lote, com barra de progresso para o último.

* **Exibição Detalhada por Ticker:**

  * Gráfico da simulação de avaliação com operações de compra (verde) e venda (vermelho).

  * Métricas financeiras destacadas (lucro %, lucro R$).

  * Tabela qualitativa com resumo.

  * Logs de processamento detalhados.

* **Resumo Geral:** Se múltiplos tickers são analisados, uma tabela comparativa de performance e métricas agregadas (lucro médio, lucro total R$) são apresentadas.

* **Acesso Remoto (Colab):** A aplicação é exposta via `ngrok` para acesso por navegador.

4.3 Análise Crítica dos Resultados Simulados

As simulações conduzidas através da interface Streamlit indicam que o modelo A2C, alimentado com os indicadores SMA, RSI e OBV, é capaz de aprender estratégias de negociação que podem resultar em lucratividade em dados de avaliação não vistos. O desempenho varia consideravelmente entre diferentes tickers, refletindo as dinâmicas únicas de cada ativo. Por exemplo, em testes anteriores, observamos tickers com lucros percentuais expressivos e alinhamento positivo com a recompensa do agente, enquanto outros apresentaram lucros financeiros com recompensas de agente menos otimizadas, sugerindo que a função de recompensa padrão do ambiente pode nem sempre capturar perfeitamente o objetivo de maximização de lucro monetário em todos os contextos. A flexibilidade da interface Streamlit permite a rápida iteração e teste desses cenários, facilitando a identificação de ativos onde a estratégia se mostra

mais promissora e a observação do comportamento do agente. A sensibilidade aos `total_timesteps` de treinamento e ao valor de investimento inicial (para cálculo do lucro em R$) também são aspectos observáveis através da ferramenta.

---

5. Pontos Fortes e Limitações

5.1 Pontos Fortes

* **Aprendizado Autônomo:** O A2C aprende estratégias sem regras explícitas, adaptando-se aos padrões dos dados.

* **Interface Interativa e Visual:** O Streamlit facilita o uso, configuração de simulações e interpretação dos resultados, mesmo para usuários com menos conhecimento técnico em RL.

* **Engenharia de Features Transparente:** O uso de indicadores técnicos conhecidos (SMA, RSI, OBV) torna a base de decisão do agente mais compreensível.

* **Modularidade:** A separação entre o motor de backtesting (`backtesting_engine.py`) e a interface (`app_streamlit_online.py`) promove organização e facilita futuras manutenções.

* **Flexibilidade:** Capacidade de analisar diferentes tickers de um mesmo arquivo de dados.

5.2 Limitações e Possíveis Melhorias

* **Realismo da Simulação:** O ambiente `gym-anytrading` padrão não inclui custos de transação (corretagem, taxas) ou *s*lippage*, o que pode levar a uma visão otimista dos lucros.

* **Risco de Overfitting:** A estratégia aprendida pode ser excessivamente ajustada aos dados de treinamento. Validações mais robustas (ex: walk-forward) são recomendadas.

* **Sensibilidade a Hiperparâmetros:** O desempenho de modelos de RL é sensível à configuração de hiperparâmetros (ex: `total_timesteps`, arquitetura da rede, taxa de aprendizado).

* **Função de Recompensa:** A recompensa padrão pode não ser ideal. Explorar funções que incorporem risco (ex: Sharpe Ratio) ou penalizem excesso de trades poderia ser benéfico.

* **Features Adicionais:** A inclusão de mais indicadores técnicos, dados fundamentalistas ou análise de sentimento poderia aprimorar o modelo.

* **Desempenho no Colab para Lotes Grandes:** O treinamento sequencial de muitos tickers via Streamlit no Colab pode ser demorado e sujeito a timeouts da sessão. Uma arquitetura de processamento offline para análises em larga escala seria mais eficiente.

---

6. Conclusão

Este projeto demonstrou com sucesso a viabilidade da implementação de um agente de trading baseado no algoritmo A2C, complementado por uma interface gráfica interativa desenvolvida com Streamlit. O sistema permite o carregamento de dados de mercado, o treinamento de modelos de aprendizado por reforço para tickers específicos e a visualização detalhada do desempenho simulado. Os resultados indicam que o agente pode aprender estratégias lucrativas para certos ativos, e a interface facilita a exploração e análise desses resultados.

Apesar das limitações inerentes a simulações e ao escopo do projeto, a combinação de `Stable Baselines3` para o RL e `Streamlit` para a interface provou ser uma abordagem poderosa para a prototipagem e experimentação no campo do trading algorítmico. As direções para trabalhos futuros incluem o refinamento do ambiente de simulação, a exploração de funções de recompensa mais sofisticadas e a validação extensiva da estratégia em diferentes condições de mercado e com um conjunto mais amplo de features. O sistema desenvolvido cumpre o objetivo de fornecer uma ferramenta prática para o estudo e aplicação de IA no domínio financeiro, alinhado com o espírito do VIII Desafio em Ciência de Dados - Investe.AI.

****7. Referências****

* Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In **International conference on machine learning** (pp. 1928-1937). PMLR. (Referência para A2C/A3C)

* Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. **Journal of Machine Learning Research, 22**(268), 1-8.

* Documentação do Gymnasium: https://gymnasium.farama.org/

* Repositório `gym-anytrading` (Adam King): https://github.com/AminHP/gym-anytrading (ou o fork/versão utilizada)

* Documentação do Streamlit: https://docs.streamlit.io/

* Documentação do Pandas: https://pandas.pydata.org/pandas-docs/stable/

* Documentação do NumPy: https://numpy.org/doc/

* Documentação do Matplotlib: https://matplotlib.org/stable/contents.html

---

****8. Apêndice: Configuração de Reprodutibilidade****

O desenvolvimento foi realizado em Python, primariamente utilizando o ambiente Google Colaboratory. As versões das bibliotecas são gerenciadas pelo comando `!pip install ... --upgrade` no Colab, que busca as versões estáveis mais recentes no momento da execução. Para garantir reprodutibilidade estrita em experimentos futuros, seria recomendado fixar as versões das bibliotecas em um arquivo `requirements.txt`.

A biblioteca `Stable Baselines3` permite a configuração de uma `seed` (semente aleatória) durante a inicialização do modelo A2C (ex: `A2C('MlpPolicy', ..., seed=42)`). Embora não tenhamos implementado uma `seed` global fixa em todas as iterações de desenvolvimento dos scripts para este guia, essa é uma prática crucial para garantir que os resultados do treinamento do modelo de RL sejam exatamente reprodutíveis. A preparação dos dados e a estrutura do ambiente, conforme descritas, contribuem para a reprodutibilidade conceitual da abordagem.