Politécnica UFRJ

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA POLITÉCNICA

Carteira do Tourinho: Simulador de Alocação e Otimização de Carteira de Investimentos

Projeto da disciplina de Computação II da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção dos créditos da disciplina.

Víctor Bernardes de Morais (Líder)	DRE: 124031375
Luiza Watanabe Dubauskas	DRE: 119178861
Gabriel Peixoto Costa Marques Barbosa	DRE: 124026590
Felipe Mariano Lessa Chaves	DRE: 124055298

Dezembro de 2024

Rio de Janeiro, Brasil

SUMÁRIO

1. INTRODUÇÃO..... 3
2. METODOLOGIA..... 3
3. DESCRIÇÃO DO SISTEMA..... 3
3.1. PRINCIPAIS FUNCIONALIDADES..... 3
3.1.1. Classificação do perfil do investidor:..... 3
3.1.2. Simulação de investimentos:..... 3
3.1.3. Previsão com Rede Neural (Narada):..... 3
3.1.4. Integração com API:..... 4
3.1.5. Gráficos de Pizza:..... 4
3.2. ESTRUTURA DE CLASSES E MÉTODOS..... 4
3.3. INTEGRAÇÃO COM A INTERFACE GRÁFICA..... 5
4. RESULTADOS E DISCUSSÃO..... 5
5. CONCLUSÃO..... 5
6. REFERÊNCIAS..... 5

1. INTRODUÇÃO

O graduado em Engenharia de Produção é capacitado para atuar no mercado financeiro, setor conhecido por sua complexidade e volatilidade. Nesse sentido, este trabalho descreve o desenvolvimento de um sistema que **simula investimentos** e oferece características educativas para o usuário. Assim, a proposta é oferecer **aprendizado sobre investimentos** de modo **acessível** e **interativo** com o auxílio de conceitos de computação.

2. METODOLOGIA

O desenvolvimento começou com brainstormings e discussões para definirmos quais funcionalidades iriam ser incluídas e organizamos o planejamento no Trello. Com as funcionalidades definidas, entendemos quais seriam as melhores ferramentas a serem utilizadas, priorizando aquelas ensinadas nas aulas de Computação II, escolhendo o **Python** como linguagem principal. Além disso, o **PyQt6** para a interface gráfica, o **Matplotlib** para a criação de gráficos, a **API do Yahoo Finance** para obter os dados financeiros e o **TensorFlow/Keras** para implementar o **modelo de rede neural com LSTM**. A partir disso, o sistema foi desenvolvido, como será descrito nos tópicos a seguir.

3. DESCRIÇÃO DO SISTEMA

3.1. PRINCIPAIS FUNCIONALIDADES

3.1.1. Classificação do perfil do investidor:

Logo no início, o sistema conduz o usuário por uma série de perguntas para identificar seu perfil de investidor (conservador, moderado ou arrojado). Essa classificação irá personalizar as sugestões de ativos e estratégias para que estejam alinhadas com os objetivos do usuário.

3.1.2. Simulação de investimentos:

A funcionalidade principal do sistema é permitir que o usuário simule a compra e venda de ativos - ações, títulos, commodities e criptomoedas. A cada operação, o sistema atualiza os valores com base nas cotações mais recentes, criando uma experiência realista que reflete as condições atuais do mercado.

3.1.3. Previsão com Rede Neural (Narada):

O método "usar_narada" é uma das partes mais avançadas do sistema, utilizando um modelo de rede neural recorrente com camadas LSTM para prever o comportamento de preços de ativos em curtos períodos [1]. Essa funcionalidade permite ao usuário analisar tendências futuras com base em padrões históricos, facilitando o aprendizado sobre como o mercado se comporta.

O processo de previsão começa com a coleta de dados históricos por meio da API do Yahoo Finance. Esses dados incluem preços de fechamento, que são processados para formar janelas de tempo utilizadas como entrada para a rede neural. O modelo foi configurado com múltiplas camadas LSTM, cada uma contendo 50 neurônios, para capturar tanto padrões de curto quanto de longo prazo. Além disso, a regularização é garantida pelo uso da técnica de dropout, que desativa aleatoriamente 20% dos neurônios durante o treinamento, garantindo maior generalização do modelo, pois evita o overfitting, que é quando um modelo aprende tanto os detalhes e ruídos dos dados de treinamento que perde a capacidade de generalizar para novos dados.

Após o treinamento, a rede neural é capaz de prever o próximo valor na sequência com base nos dados fornecidos. Por exemplo, se o usuário deseja prever o comportamento de um ativo como o Bitcoin (BTC), o modelo utiliza os preços históricos desse ativo dos últimos 60 dias para gerar uma previsão para o dia seguinte.

3.1.4. Integração com API:

O sistema utiliza a API do Yahoo Finance para obter dados financeiros, garantindo que as simulações sejam baseadas em informações atualizadas e precisas. Sempre que o usuário acessa sua carteira, realiza uma simulação ou utiliza o método "usar_narada", o sistema faz requisições à API, buscando cotações dos ativos. Os dados retornados incluem diversas informações, mas o sistema utiliza exclusivamente o preço de fechamento como base para todas as operações.

3.1.5. Gráficos de Pizza:

Para ajudar o usuário a visualizar e gerenciar sua carteira, o sistema gera gráficos de pizza que mostram a proporção dos investimentos realizados em diferentes ativos. Essa funcionalidade utiliza a biblioteca matplotlib para criar representações gráficas claras e interativas.

3.2. ESTRUTURA DE CLASSES E MÉTODOS

O sistema foi desenvolvido com base nos conceitos de Programação Orientada a Objetos e utiliza o conceito de **abstração** ao representar o usuário como uma classe chamada "Cliente". Essa classe **encapsula** o saldo com o objetivo de garantir que ele seja um número válido a partir de um setter. Também define métodos que realizam operações específicas, como mostrados a seguir:

- **Método "investir":** permite ao usuário aplicar valores em ativos disponíveis, atualizando os atributos do objeto correspondente;

- **Método "vender":** similar ao "investir", mas permite a retirada de valores investidos e simula ganhos ou perdas dependendo da variação da cotação;

- **Método "calcular_lucro_prejuizo":** atualiza o valor dos investimentos com base nas cotações mais recentes dos ativos;

- **Método "usar_narada":** aplica o modelo preditivo baseado em LSTM para estimar os preços futuros dos ativos;

- **Método "ver_pizza":** gera o gráfico de proporção de investimentos, utilizando bibliotecas como matplotlib para a visualização.

Além disso, a **herança** foi usada na estrutura da interface gráfica, onde um modelo das telas foi gerado como uma “classe mãe” para que então as “classes filhas” pudessem integrar as funções aos devidos botões das janelas e, também implementada na construção de “Narada” já que as previsões de cada ativo seriam as “filhas” de uma classe abstrata, sendo essa a classe “Narada”.

Nesse tópico, a **classe abstrata** “Narada” foi utilizada para facilitar a abstração das suas “classes filhas”, essas que devem conter certos métodos, sem exceção.

Já o **encapsulamento** é aplicado no sistema para proteger os dados sensíveis do usuário e garantir que apenas métodos específicos possam acessá-los ou modificá-los. Por exemplo, o atributo da classe "Cliente", saldo, é privado e só pode ser alterado por métodos como "investir" ou "vender".

Seguindo, o **polimorfismo** foi utilizado na implementação de métodos relacionados à **interface gráfica**, como, por exemplo, classes que herdam as janelas principais redefinem comportamentos para ajustar funcionalidades específicas, como a interação com botões.

Ainda, o **tratamento de exceções** foi utilizado para verificar entradas inválidas e a existência de modelos já treinados e a **persistência de dados** foi simplesmente aplicada para salvar e carregar dados quando o usuário fechar e abrir o aplicativo, assim como ensinado em aula.

Por fim, o **uso de bibliotecas diversas** foi muito explorado e, resumindo, bibliotecas como: abc (classe “Narada”), numpy (tratamento de dados para “Narada”), matplotlib (geração de gráficos como os de pizza e os de linhas), pandas (manipular arquivos diversos, principalmente relacionados

ao “Narada”), datetime (dados sobre datas atuais e passadas para obter informações sobre variações de investimentos), yfinance (informações sobre ativos), sklearn.preprocessing (normalizar dados), tensorflow.keras (utilizado na função de criar camadas de redes neurais), PyQt6 (biblioteca de interface gráfica). De forma bruta e simplificada, essas foram as funções das bibliotecas usadas.

3.3. INTEGRAÇÃO COM A INTERFACE GRÁFICA

A interface gráfica foi construída utilizando o framework PyQt6, que permite criar janelas interativas e intuitivas. Cada funcionalidade do sistema está associada a uma janela ou botão na interface, facilitando a navegação do usuário.

Para isso, a ferramenta QT Designer [2] foi utilizada para facilitar e agilizar o processo de criação da interface.

Primeiramente, foi criada uma classe modelo para cada uma das janelas principais da interface: a tela inicial, a tela para classificação do perfil do investidor e o menu principal. Para realizar a interação dos botões com a interface gerada por essas classes, novas classes foram criadas para que então herdassem o conteúdo desses modelos para assim ocorrer a integração com a interface gráfica.

4. RESULTADOS E DISCUSSÃO

Com base nos objetivos iniciais em relação a esse trabalho, chegamos à conclusão de que os resultados obtidos na execução do seu planejamento foram muito satisfatórios. Objetivos iniciais, como: criar simulações de investimentos, ajustar o aplicativo para torná-lo mais próximo de uma simulação real, desenvolver uma forma de “prever” o comportamento futuro dos ativos, representar graficamente as variações dos produtos de investimento e apresentar conceitos básicos de investimentos para iniciantes; foram todos articulados e desenvolvidos como imaginado.

Além disso, uma função a mais foi desenvolvida no decorrer da realização do trabalho, sendo essa a função de gerar o gráfico de pizza, que confere ao investidor versatilidade ao manipular seus investimentos.

5. CONCLUSÃO

No geral, o código foi concluído com êxito, superando até mesmo nossas próprias expectativas ao observarmos como seu funcionamento, de fato, mostra sua utilidade para um usuário. Suas simulações, taxas de variação e previsões realizadas por Narada nos incentivaram até mesmo a utilizarmos o código para uso pessoal, mostrando mais uma vez nossa satisfação com o resultado obtido.

Novamente, o conteúdo apresentado em sala de aula foi devidamente aplicado nesse trabalho, incluindo: abstração, herança, encapsulamento, polimorfismo, classe abstrata, interface, tratamento de exceções, persistência de dados, bibliotecas. Portanto, é um trabalho abrangente que compila as necessidades propostas de forma eficaz.

E sucintamente, o papel de cada um no trabalho foi: Víctor - Back-end; Gabriel - Front-end; Luiza e Felipe - Suporte e Educativo.

6. REFERÊNCIAS

[1] NeuralNine. Predicting Stock Prices in Python. Youtube, 02/02/21. Disponível em: https://www.youtube.com/watch?v=PuZY9q-aKLw. Acesso em: 08/12/24

[2] freeCodeCamp.org. Learn Python GUI Development for Desktop – PySide6 and Qt Tutorial. Youtube, 05/12/22. Disponível em: https://www.youtube.com/watch?v=Z1N9JzNax2k. Acesso em: 08/11/24