02 - Construção do Modelo

Relatório Explicativo do Código de Construção do Modelo

Este documento tem como objetivo explicar detalhadamente o funcionamento do código responsável pela construção do modelo de Rede Neural Artificial (RNA), que será utilizado para prever se o preço do Bitcoin (BTC-USD) subirá ou cairá no dia seguinte.

---

1. Visão Geral do Código

O código está estruturado em quatro etapas principais:

1. **Carregamento dos Dados**: Lê os dados processados a partir de um arquivo CSV.
2. **Preparação dos Dados**: Separa os dados de entrada (features) e o rótulo (target).
3. **Criação do Modelo de Rede Neural**: Define a arquitetura da RNA.
4. **Salvamento do Modelo**: Armazena o modelo treinável em disco para uso futuro.

---

2. Explicação por Etapas

2.1 Carregamento dos Dados

A função `carregar_dados()` é responsável por importar o conjunto de dados gerado na etapa anterior (`bitcoin_data.csv`):

- Utiliza o **Pandas** para ler o arquivo CSV.
- Define a coluna de índice como a coluna 0 (as datas).
- Imprime uma mensagem com o número de linhas e colunas carregadas.

2.2 Preparação dos Dados

A função `preparar_dados(btc)` separa os dados em:

- `X`: Variáveis de entrada (features), que incluem os seguintes indicadores:
    - `Close`: Preço de fechamento normalizado.
    - `SMA_14`: Média Móvel Simples de 14 dias.
    - `EMA_14`: Média Móvel Exponencial de 14 dias.
    - `RSI_14`: Índice de Força Relativa.
    - `Volatility`: Volatilidade (desvio padrão de 14 dias).
    - `Volume_MA_14`: Média Móvel do Volume.
- `y`: Rótulo (target), que indica se o preço do Bitcoin subiu (`1`) ou caiu (`0`) no dia seguinte.

Essa separação é essencial para o treinamento supervisionado de modelos de machine learning.

2.3 Criação do Modelo de Rede Neural Artificial

A função `criar_modelo()` define e compila a arquitetura da RNA utilizando o framework **TensorFlow (Keras)**:

Arquitetura:

- **Camada de Entrada (`Input`)**:
    - Define que o modelo receberá 6 entradas (as features listadas acima).
- **Camada Oculta 1 (`Dense`)**:
    - 64 neurônios com ativação ReLU.

    - Escolhida por sua capacidade de capturar padrões não-lineares.

- **Dropout (0.2)**:
    - Desativa 20% dos neurônios de forma aleatória durante o treinamento.
    - Ajuda a evitar o overfitting, promovendo uma melhor generalização do modelo.
- **Camada Oculta 2 (`Dense`)**:
    - 32 neurônios com ativação ReLU.
    - Fornece uma segunda camada de abstração para o aprendizado.
- **Camada de Saída (`Dense`)**:
    - 1 único neurônio com ativação sigmoide (`sigmoid`), ideal para problemas de classificação binária (0 ou 1).

Compilação do Modelo:

- **Otimizador:** `adam`, amplamente utilizado por sua eficiência e velocidade.
- **Função de perda:** `binary_crossentropy`, indicada para classificação binária.
- **Métrica de avaliação:** `accuracy`, que mede a porcentagem de acertos.

2.4 Salvamento do Modelo

A função `salvar_modelo(modelo)` armazena o modelo criado no formato `.keras`, que é o novo padrão recomendado pelo TensorFlow:

- O parâmetro `include_optimizer=False` é utilizado para salvar o modelo de forma mais leve, sem o estado interno do otimizador.
- O arquivo gerado (`modelo_rna.keras`) poderá ser carregado posteriormente para treinamento ou inferência.

3. Conclusão

Este código é responsável por construir a estrutura da Rede Neural Artificial que será treinada para prever a direção do preço do Bitcoin com base em indicadores técnicos.

A separação clara das etapas permite que o pipeline de desenvolvimento siga de forma modular, mantendo o modelo salvo para reutilização futura sem a necessidade de recriação. Isso proporciona maior eficiência e reprodutibilidade nos experimentos.