03 - Treinamento do Modelo

Relatório Explicativo do Código de Treinamento da Rede Neural

Este relatório apresenta uma explicação detalhada sobre o código responsável pelo treinamento do modelo de Rede Neural Artificial (RNA), que tem como objetivo prever se o preço do Bitcoin subirá ou cairá no dia seguinte com base em indicadores técnicos.

---

1. Visão Geral do Código

Este script é dividido em cinco etapas principais:

1. **Carregamento dos Dados**: Lê os dados normalizados e processados do arquivo CSV.
2. **Preparação dos Dados**: Separa os dados de entrada (features) e o rótulo (target).
3. **Carregamento do Modelo**: Importa a arquitetura da RNA previamente construída.
4. **Treinamento da Rede Neural**: Executa o treinamento do modelo com os dados.
5. **Salvamento do Modelo Treinado**: Salva o modelo treinado para uso posterior.

---

2. Explicação por Etapas

2.1 Carregamento dos Dados

A função `carregar_dados()` realiza o seguinte:

- Lê o arquivo `bitcoin_data.csv` usando o **Pandas**.
- Converte todas as colunas para tipo numérico, utilizando `pd.to_numeric` com o parâmetro `errors='coerce'`. Isso força a conversão dos dados e substitui

qualquer valor inválido por `NaN`.

- Remove todas as linhas com valores ausentes (`NaN`) usando `dropna()` para garantir que os dados estejam limpos e prontos para o treinamento.

Essa etapa é fundamental para evitar erros durante o processo de modelagem e garantir que os dados estejam formatados corretamente.

2.2 Preparação dos Dados

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

- `X`: As features utilizadas como entrada no modelo:
    - `Close`, `SMA_14`, `EMA_14`, `RSI_14`, `Volatility`, `Volume_MA_14`.
- `y`: A variável alvo (`Target`), que indica se o preço do Bitcoin subiu (`1`) ou caiu (`0`) no dia seguinte.

Além disso, a função assegura que:

- `X` esteja no formato `float` (números reais).
- `y` esteja no formato `int` (inteiros 0 ou 1).

2.3 Carregamento do Modelo

A função `carregar_modelo()` utiliza o método `load_model()` da biblioteca Keras para carregar o modelo salvo previamente no arquivo `modelo_rna.keras`, criado na etapa anterior de construção do modelo (02_modelo).

Com isso, o modelo pode ser treinado sem precisar ser recriado.

2.4 Treinamento da Rede Neural

A função `treinar_modelo()` é a etapa central do script. Ela realiza:

1. **Divisão dos Dados**:
    - Separa o conjunto de dados em treino (80%) e teste (20%) usando `train_test_split()`.
    - Define uma `random_state=42` para garantir reprodutibilidade dos resultados.
2. **Treinamento do Modelo**:
    - O método `fit()` é utilizado para treinar o modelo por **50 épocas** (`epochs=50`) e com **tamanho de lote igual a 32** (`batch_size=32`).
    - O modelo é validado durante o treinamento usando o conjunto de teste, com o parâmetro `validation_data=(X_teste, y_teste)`.
    - Os resultados (perda e acurácia) são armazenados no objeto `historico`, que poderá ser utilizado posteriormente para análise gráfica do desempenho do treinamento.

2.5 Salvamento do Modelo Treinado

A função `salvar_modelo()` salva o modelo já treinado em um novo arquivo chamado `modelo_rna_treinado.keras`. Esse modelo agora está ajustado aos dados históricos do Bitcoin e pode ser utilizado na fase de avaliação e previsão.

---

3. Conclusão

Este script é responsável por transformar a arquitetura inicial da Rede Neural Artificial em um modelo funcional e treinado, pronto para ser testado em dados reais e prever movimentos futuros do preço do Bitcoin.

O código é modular, bem estruturado e pode ser facilmente reutilizado para outros conjuntos de dados ou configurações de treinamento. O uso de boas práticas como limpeza

dos dados, separação treino/teste e salvamento de modelos garante maior robustez e reprodutibilidade.