Relatório Explicativo do Código de Coleta e Processamento de Dados

Este documento tem como objetivo explicar detalhadamente o funcionamento do código utilizado para a coleta e processamento dos dados do Bitcoin (BTC-USD).

1. Visão Geral do Código

O código está estruturado em quatro etapas principais:

1. **Coleta de Dados**: Baixa histórico de preços e volumes do Bitcoin.
2. **Cálculo de Indicadores**: Gera métricas técnicas como Médias Móveis, RSI e Volatilidade.
3. **Preparação e Limpeza dos Dados**: Remove valores nulos e normaliza os dados.
4. **Armazenamento**: Salva os dados tratados em um arquivo CSV.

2. Explicação por Etapas

2.1 Coleta de Dados

O código utiliza a biblioteca `yfinance` para baixar os dados históricos do Bitcoin:

```
btc = yf.download("BTC-USD", period="2y", interval="1d")
```

- **Símbolo 'BTC-USD'**: Indica que queremos dados do Bitcoin em relação ao dólar.
- ``: Obtém dados dos últimos 2 anos.
- ``: Define que os dados serão diários.

2.2 Cálculo de Indicadores

Foram implementados cinco indicadores técnicos:

1. **Média Móvel Simples (SMA - Simple Moving Average)**:

```
btc["SMA_14"] = btc["Close"].rolling(window=14).mean()
```

Calcula a média dos últimos 14 dias do preço de fechamento.

2. **Média Móvel Exponencial (EMA - Exponential Moving Average)**:

```
btc["EMA_14"] = btc["Close"].ewm(span=14, adjust=False).mean()
```

Dá mais peso aos preços recentes para reagir mais rápido a mudanças.

3. **Índice de Força Relativa (RSI - Relative Strength Index)**:

```
btc["RSI_14"] = 100 - (100 / (1 + btc["Close"].pct_change().rolling(14)
                              .apply(lambda x: (x[x > 0].sum() / -x[x < 0].sum()) if -x[x < 0].sum() != 0 else 1)))
```

Mede a velocidade e magnitude das variações de preço.

4. **Volatilidade**:

```
btc["Volatility"] = btc["Close"].rolling(window=14).std()
```

Calcula o desvio padrão dos preços de fechamento nos últimos 14 dias.

5. **Média Móvel do Volume**:

```
btc["Volume_MA_14"] = btc["Volume"].rolling(window=14).mean()
```

Indica a tendência de negociação.

2.3 Preparação e Limpeza dos Dados

- **Remoção de valores ausentes**:

```
btc.dropna(inplace=True)
```

Garante que o conjunto de dados esteja completo.

- **Normalização com Min-Max Scaling**:

```
scaler = MinMaxScaler()
colunas_para_normalizar = ["Close", "SMA_14", "EMA_14", "RSI_14", "Volatility", "Volume_MA_14"]
btc[colunas_para_normalizar] = scaler.fit_transform(btc[colunas_para_normalizar])
```

Traz os valores para a escala de 0 a 1, tornando o treinamento da rede neural mais eficiente.

2.4 Armazenamento dos Dados

Os dados são salvos em um arquivo CSV para uso futuro:

```
btc.to_csv("bitcoin_data.csv", index=True)
```

O `index=True` garante que as datas sejam mantidas no arquivo.

3. Conclusão

Esse código prepara os dados do Bitcoin para serem utilizados no modelo de redes neurais. Ele garante que os dados estejam limpos, organizados e prontos para a fase de treinamento do modelo.

Caso seja necessário coletar novos dados, o código pode ser reexecutado, mas para manter consistência nos testes, recomenda-se usar um conjunto de dados fixo.