# Documentação do Código de Previsão de Preços do Bitcoin com LSTM Bidirecional

Este código utiliza uma rede neural bidirecional LSTM para prever os preços do Bitcoin com base em dados históricos obtidos por meio da biblioteca yfinance. A arquitetura da rede inclui camadas LSTM, Dropout e Dense, permitindo previsão multivariada ajustável. O código está estruturado em uma classe BitcoinPredictor, que organiza o fluxo de trabalho desde a coleta de dados até a visualização dos resultados.

## Dependências

Para executar o código, as seguintes bibliotecas Python são necessárias:

* yfinance: para baixar dados históricos de mercado.
* numpy, pandas: para manipulação e processamento de dados.
* matplotlib: para plotar gráficos dos preços e previsões.
* sklearn.preprocessing.MinMaxScaler: para escalonamento dos dados.
* keras: para construção e treinamento da rede neural bidirecional LSTM.

## Estrutura da Classe BitcoinPredictor

### Inicialização (\_\_init\_\_)

Parâmetros:

* ticker: Ticker do ativo a ser previsto, padrão é 'BTC-USD'.
* start, end: Intervalo de datas para coleta dos dados históricos.
* time\_step: Número de passos de tempo usados como entrada para a previsão.
* prediction\_steps: Número de passos futuros a serem previstos.
* train\_ratio: Proporção dos dados usada para treinamento (o restante é usado para teste).
* scaler: Um objeto para escalonar os dados para o intervalo [0, 1].

### Métodos Principais

1. download\_data: Baixa os dados de preços de fechamento do ativo definido e armazena na variável data.
2. preprocess\_data: Escalona os dados para o intervalo [0, 1] e divide em conjuntos de treinamento e teste com base na proporção train\_ratio.
3. create\_dataset: Cria conjuntos de entrada (X) e saída (Y) com base em time\_step e prediction\_steps. Cada entrada X possui time\_step valores, enquanto Y tem os valores subsequentes para a previsão.
4. build\_bidirectional\_model: Constrói e compila um modelo LSTM bidirecional com:
   * Duas camadas LSTM bidirecionais de 100 unidades cada.
   * Camadas Dropout para evitar overfitting.
   * Camada Dense final para saída com prediction\_steps.
5. train\_model: Treina o modelo com early stopping baseado na métrica de erro de treinamento, evitando overfitting. Parâmetros incluem batch\_size e patience para o early stopping.
6. predict: Faz previsões sobre os dados de teste. Retorna as previsões escaladas de volta ao intervalo original.
7. predict\_new\_data: Preve o próximo preço do ativo com base nos dados recentes, permitindo previsões em tempo real para novos dados de mercado.
8. plot\_results: Plota os resultados das previsões juntamente com os valores reais de preços em um gráfico. Um preço futuro previsto é exibido como uma linha horizontal pontilhada.

### Execução Principal (\_\_main\_\_)

O código é executado da seguinte forma:

1. Instancia a classe BitcoinPredictor e baixa os dados históricos.
2. Pré-processa os dados e constrói o modelo bidirecional LSTM.
3. Treina o modelo e faz previsões sobre os dados de teste.
4. Realiza uma previsão sobre os dados mais recentes e plota os resultados em um gráfico.

Este é um guia inicial; cada método será expandido para incluir detalhes sobre o funcionamento interno, como pré-processamento e transformações de dados, além de um guia de interpretação dos gráficos gerados.