Estratégias High-Frequency

Por José Eduardo e Lorenzo Cavalcante

Introdução

Dentro da bolsa de valores e do mercado de capitais em geral, diversos investidores buscam formas claras de visualização dos dados apresentados, principalmente, porque um mesmo ativo pode ter inúmeras variáveis que impactam em suas tendências e devem ser levadas em conta para a tomada de decisão. Então, é comum que investidores façam uso de estratégias de visualização gráfica para conseguirem entender as tendências que o mercado tem de forma clara, utilizando isso na busca pela maximização dos ganhos. Em nosso caso, iremos utilizar uma estratégia chamada Ichimoku Cloud para fundamentar nossas posições de compra e venda ao longo do tempo.

Ichimoku Cloud

Originalmente conhecida por Ichimoku Kyo Hyo, que traduzindo significa “um olhar sobre o equilibrio gráfico”, essa estratégia busca por meio de suas linhas de tendência e “nuvens” evidenciar se o ativo será valorizado ou não, por consequência, se é um momento propício para se entrar na posição.

O **Ichimoku Cloud**, foi desenvolvido no final dos anos 1930 por um jornalista japonês chamado Goichi Hosoda, que escrevia sob o pseudônimo Ichimoku Sanjin, que significa “homem da montanha que vê tudo de relance”. Insatisfeito com os métodos tradicionais de análise técnica, Hosoda buscou criar um sistema que proporcionasse uma visão rápida e abrangente das condições do mercado. Ele dedicou cerca de 30 anos ao aperfeiçoamento deste indicador, contando com a ajuda

de mais de 1000 estudantes para testar e refinar os cálculos. O Ichimoku Cloud foi finalmente publicado em 1969, e, apesar de antigo, ainda se mostra extremamente eficaz para avaliar posições de entrada e saída no mercado.

O gráfico Ichimoku é, assim como na maioria dos gráficos utilizados no mercado de capitais, uma relação entre o tempo e o valor negociado de um ativo, sendo o eixo x a data de negociação e o eixo y o valor negociado, porém possui algumas linhas adicionais para captar tendências.

Partindo para as linhas particulares do gráfico Ichimoku, temos 5 novas componentes, sendo que duas delas compõem o contorno da tal "nuvem", duas são médias móveis e uma será o próprio valor do ativo.

- **Tenkam Sem**, ou linha de conversão, usada como média móvel dos pontos médios dos últimos 9 períodos.
- **Kjun Sem**, ou linha de base, plotada como uma média móvel dos pontos médios dos últimos 26 períodos de tempo.
- **Senkou Span A**, plotado como o ponto médio de Tenkan Sen e Kijun Sen, com a linha projetada 26 períodos no futuro.
- **Senkou Span B**, plotado como o ponto médio de entre as 52 últimas altas e 52 últimas baixas
- **Chikou Span**, plota o valor de 26 períodos no passado.

Dessa forma, cada linha representa uma ideia sobre a evolução da ação a ser analisada ao longo do tempo e nos dão ferramentas para tentar prever com mais precisão qual melhor atitude deve ser tomada quanto à posição no ativo analisado.

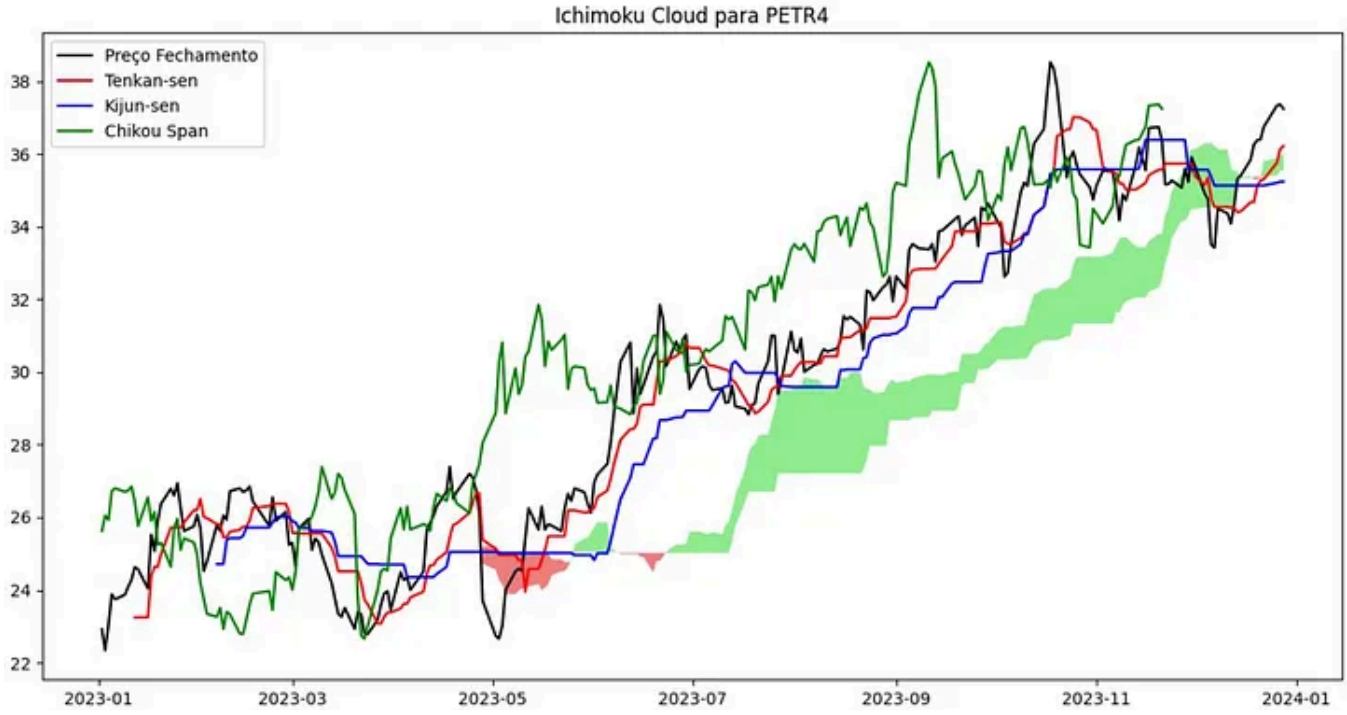
- **Tenkam Sem**, revela a tendência de curto prazo do ativo.
- **Kjun Sem**, indica a tendência de médio prazo do ativo.
- **Senkou Span A** e **Senkou Span B**, juntas formam a "nuvem" que dá nome à estratégia, nuvem esta que nos indica um potencial suporte e resistência para o gráfico.
- **Chikou Span**, é utilizado para confirmar tendências expostas pelas outras linhas.

Plotando o gráfico

Partiremos então para a parte computacional do nosso projeto, iniciaremos com a plotagem do gráfico Ichimoku Cloud, onde definiremos suas linhas particulares e a ação a ser trabalhada, bem como as bibliotecas importantes para todo o resto do código.

```
# Importando bibliotecas necessárias
import pandas as pd                # Biblioteca para manipulação de dados em es
import numpy as np                 # Biblioteca para operações matemáticas e nu
import yfinance as yf              # Biblioteca para obtenção de dados financei

# Utilizando dados da Petrobras
dados = yf.download('PETR4.SA', start='2023-01-01', end='2024-01-01', interval=

## Criação das linhas do Ichimoku Cloud ##

# Tenkan-sen (Linha de Conversão)
# Calcula a média entre o maior preço máximo e o menor preço mínimo nos últimos
periodo_tenkan = 9
dados['Tenkan_sen'] = (
    dados['High'].rolling(window=periodo_tenkan).max() +   # Preço máximo nos ú
    dados['Low'].rolling(window=periodo_tenkan).min()      # Preço mínimo nos ú
) / 2                                                      # Média dos valores

# Kijun-sen (Linha de Base)
# Calcula a média entre o maior preço máximo e o menor preço mínimo nos últimos
periodo_kijun = 26
dados['Kijun_sen'] = (
    dados['High'].rolling(window=periodo_kijun).max() +    # Preço máximo nos ú
    dados['Low'].rolling(window=periodo_kijun).min()       # Preço mínimo nos ú
) / 2                                                      # Média dos valores

# Senkou Span A (Primeira linha da nuvem)
# Média de Tenkan-sen e Kijun-sen, projetada 26 períodos no futuro
dados['Senkou_span_A'] = (
    (dados['Tenkan_sen'] + dados['Kijun_sen']) / 2          # Média das linhas
).shift(periodo_kijun)                                     # Deslocamento de 26

# Senkou Span B (Segunda linha da nuvem)
# Calcula a média entre o maior preço máximo e o menor preço mínimo nos últimos
períodos no futuro
periodo_senkou_b = 52
dados['Senkou_span_B'] = (
    dados['High'].rolling(window=periodo_senkou_b).max() + # Preço máximo nos ú
    dados['Low'].rolling(window=periodo_senkou_b).min()    # Preço mínimo nos ú
) / 2                                                      # Média dos valores
dados['Senkou_span_B'] = dados['Senkou_span_B'].shift(periodo_kijun)  # Desloca
períodos à frente
```

```
# Chikou Span (Linha de Atraso)
# Preço de fechamento atual deslocado 26 períodos para trás
dados['Chikou_span'] = dados['Close'].shift(-periodo_kijun)

# Plotagem do Gráfico
import matplotlib.pyplot as plt     # Biblioteca para criação de gráficos

plt.figure(figsize=(14, 7))  # Define o tamanho da figura do gráfico

# Plotando o preço de fechamento da ação
plt.plot(dados.index, dados['Close'], label='Preço Fechamento', color='black')

# Plotando a linha Tenkan-sen
plt.plot(dados.index, dados['Tenkan_sen'], label='Tenkan-sen', color='red')

# Plotando a linha Kijun-sen
plt.plot(dados.index, dados['Kijun_sen'], label='Kijun-sen', color='blue')

# Preenchendo a área entre Senkou Span A e Senkou Span B para formar a nuvem (K
# Quando Senkou Span A está acima de Senkou Span B, preenche com verde claro
plt.fill_between(
    dados.index,
    dados['Senkou_span_A'],
    dados['Senkou_span_B'],
    where=dados['Senkou_span_A'] >= dados['Senkou_span_B'],
    facecolor='lightgreen',
    interpolate=True
)

# Quando Senkou Span A está abaixo de Senkou Span B, preenche com vermelho clar
plt.fill_between(
    dados.index,
    dados['Senkou_span_A'],
    dados['Senkou_span_B'],
    where=dados['Senkou_span_A'] < dados['Senkou_span_B'],
    facecolor='lightcoral',
    interpolate=True
)

# Plotando a linha Chikou Span
plt.plot(dados.index, dados['Chikou_span'], label='Chikou Span', color='green')

# Adicionando legenda, título e exibindo o gráfico
plt.legend(loc='best')
plt.title('Ichimoku Cloud para PETR4')
plt.show()
```



Aqui temos o resultado do nosso código para a ação da PETR4.SA.

Back-testing

Back-testing é uma técnica amplamente utilizada em finanças quantitativas para avaliar a eficácia de uma estratégia de investimento ou modelo financeiro, aplicando-o a dados históricos.

O objetivo é verificar como o modelo ou estratégia teria se comportado no passado, fornecendo uma estimativa de seu desempenho futuro. Esse processo é essencial porque permite testar as suposições e hipóteses antes de colocar dinheiro real em risco, além de identificar possíveis problemas ou ajustes necessários na estratégia.

Após a tarefa de garantir a confiabilidade dos dados, haveria, ainda, um novo desafio para quantificar os resultados da aplicação do ichimoku no período escolhido, mas por sorte já existem bibliotecas que facilitam nosso trabalho. Para isso, a biblioteca que iremos usar será a backtrader.

O Backtrader é uma biblioteca Python de código aberto projetada para o desenvolvimento de estratégias de negociação e backtesting de dados financeiros. Ele fornece uma estrutura robusta que permite a pesquisadores, analistas e traders simular o desempenho de estratégias de negociação ao longo de dados históricos. Ao abstrair as complexidades do gerenciamento de dados, execução de ordens e análise de desempenho, o Backtrader permite que os usuários se concentrem nos aspectos essenciais do desenvolvimento e avaliação de estratégias.

No núcleo do Backtrader está uma arquitetura modular e orientada a objetos composta por vários componentes-chave:

1. **Cerebro Engine:** É o motor central que orquestra o processo de backtesting. Ele gerencia feeds de dados, estratégias, corretoras simuladas (brokers), analisadores e observadores. Os usuários instanciam um objeto Cerebro para configurar e executar seus backtests.

2. **Classe de Estratégia:** Os usuários criam estratégias de negociação subclasseando a classe bt.Strategy. Essa classe fornece métodos de ciclo de vida como __init__, next, start e stop, que são substituídos para implementar lógica personalizada. O método next é crucial, pois define as operações a serem executadas em cada passo temporal.

3. **Feeds de Dados:** O Backtrader suporta várias fontes de dados, incluindo arquivos CSV, DataFrames do pandas e feeds de dados em tempo real de corretoras. Os feeds de dados são adicionados ao motor Cerebro e fornecem as séries temporais sequenciais necessárias para o backtesting.

4. **Indicadores e Observadores:**
   a. Indicadores: Possui indicadores técnicos pré-definidos como Médias Móveis, RSI, MACD e Bandas de Bollinger. Os usuários também podem criar indicadores personalizados subclasseando bt.Indicator.
   b. Observadores: Componentes que monitoram e registram eventos durante o backtest, como drawdowns, lucros e perdas, e execuções de ordens.

5. **Simulação de Broker:** Um broker simulado que lida com a execução de ordens, gerenciamento de caixa, cálculo de comissões e acompanhamento de posições. Ele imita condições reais de mercado, permitindo uma simulação realista da estratégia.

6. **Analisadores:** Ferramentas que processam os resultados do backtest para produzir métricas de desempenho. Analisadores integrados calculam retornos, índices de Sharpe, drawdowns e outros. Analisadores personalizados podem ser criados para métricas específicas.

7. **Framework de Otimização:** Suporta a otimização de parâmetros, permitindo que estratégias sejam executadas com diferentes combinações de parâmetros. Isso é facilitado através do método optstrategy, permitindo uma exploração eficiente do espaço de parâmetros da estratégia.

Por fim, combinamos o indicador Ichimoku Cloud com o Índice de Força Relativa (RSI). Isso porque O RSI aponta as condições de sobrecompra e sobrevenda no mercado, noções fundamentais para aumentar a lucratividade do nosso modelo, dado que são fatores não contemplados pelo Ichimoku Cloud. Agora, iremos combinar esses dois indicadores na estratégia proposta para encontrar os pontos ótimos de entrada e saída no mercado, com o objetivo de aumentar a eficácia da estratégia.

```
import backtrader as bt
import yfinance as yf
import pandas as pd
import itertools



class IchimokuStrategy(bt.Strategy):
    params = (
        ('tenkan_period', 9),
        ('kijun_period', 26),
        ('senkou_b_period', 52),
        ('stop_loss', 0.05),  # Stop-loss em 5%
        ('rsi_period', 14),
        ('rsi_upper', 75),    # Ajustando os limites do RSI para tomar mais ope
        ('rsi_lower', 25),
        ('leverage', 2),      # Usando alavancagem de 2:1
    )

    def __init__(self):
        ichimoku = bt.indicators.Ichimoku(
            self.data,
            tenkan=self.params.tenkan_period,
            kijun=self.params.kijun_period,
            senkou=self.params.senkou_b_period
        )

        self.tenkan_sen = ichimoku.tenkan_sen
        self.kijun_sen = ichimoku.kijun_sen
        self.senkou_span_a = ichimoku.senkou_span_a
        self.senkou_span_b = ichimoku.senkou_span_b
        self.chikou_span = ichimoku.chikou_span
        self.order = None

        # Adicionando RSI para evitar sobrecompra/sobrevenda
        self.rsi = bt.indicators.RelativeStrengthIndex(period=self.params.rsi_p

    def next(self):
        cash_available = self.broker.get_cash()
```

```
            # Aumentando o trade_size com base na alavancagem definida
            trade_size = (cash_available * self.params.leverage) // self.data.close
alavancado

            if not self.position:
                # Condições para uma compra
                if (self.data.close > self.senkou_span_a[0] and self.data.close > s
                    self.rsi < self.params.rsi_upper):
                    self.order = self.buy(size=trade_size)
                # Condições para uma venda
                elif (self.data.close < self.senkou_span_a[0] and self.data.close <
                      self.rsi > self.params.rsi_lower):
                    self.order = self.sell(size=trade_size)
            else:
                # Aplicando stop-loss (5%)
                if self.position.size > 0:  # Posição Long
                    if (self.data.close < self.senkou_span_a[0] or self.data.close
                        self.data.close < (self.position.price * (1 - self.params.s
                        self.close()
                elif self.position.size < 0:  # Posição Short
                    if (self.data.close > self.senkou_span_a[0] or self.data.close
                        self.data.close > (self.position.price * (1 + self.params.s
                        self.close()

    def run_backtest(params):
        tenkan_period, kijun_period, senkou_b_period, stop_loss, rsi_period, rsi_up
        cerebro = bt.Cerebro()
        cerebro.addstrategy(IchimokuStrategy, tenkan_period=tenkan_period, kijun_pe
                            senkou_b_period=senkou_b_period, stop_loss=stop_loss,
                            rsi_period=rsi_period, rsi_upper=rsi_upper, rsi_lower=r

        # Baixar dados históricos da ação a ser utilizada(IMPORTANTE)
        dados = bt.feeds.PandasData(dataname=yf.download('PETR4.SA', start='2023-01
01', interval='1d'))
        cerebro.adddata(dados)

        # Configurar alavancagem e comissão
        cerebro.broker.set_cash(100000.0)

        cerebro.broker.setcommission(commission=0.001, leverage=2)  # Alavancagem c

        cerebro.run()
        return (tenkan_period, kijun_period, senkou_b_period, stop_loss, rsi_perioc
cerebro.broker.getvalue())

    def main():
        # Gerar todas as combinações de parâmetros
        param_combinations = list(itertools.product(
            range(7, 11),        # Tenkan-sen
            range(26, 32),       # Kijun-sen
            range(52, 61),       # Senkou B
            [0.01, 0.02, 0.03],  # Stop-loss
            [10, 14, 20],        # RSI período
```

```
            [65, 70, 75],        # RSI sobrecompra
            [25, 30, 35]         # RSI sobrevenda
        ))

        # Executar os backtests sequencialmente
        best_result = None
        best_capital = 0

        for params in param_combinations:
            result = run_backtest(params)
            #print(f'Tenkan: {result[0]}, Kijun: {result[1]}, Senkou B: {result[2]}
            #      f'RSI Período: {result[4]}, RSI Sobrecompra: {result[5]}, RSI Sc
            #      f'Capital Final: {result[7]}')
            if result[7] > best_capital:
                best_capital = result[7]
                best_result = result

        # Mostrar os melhores parâmetros e o melhor capital final
        if best_result:
            print(f'\nMelhores Parâmetros: Tenkan: {best_result[0]}, Kijun: {best_r
    {best_result[2]}, Stop Loss: {best_result[3]}')
            print(f'RSI Período: {best_result[4]}, RSI Sobrecompra: {best_result[5]
    {best_result[6]}')
            print(f'Melhor Capital Final: {best_result[7]}')

    main()
```

Além das bibliotecas tradicionais para o tratamento de dados financeiros, utilizamos o itertools para fazer a exploração combinatória de parâmetros, o que será explicado mais a frente. Seguindo a documentação da bilioteca Backtrader, definimos a classe IchimokuStrategy, que derivará suas propriedades da classe bt.Strategy e é a classe base para a estratégia do Backtrader. Nesta classe, definimos parâmetros para nossa estratégia, como períodos dos componentes do Ichimoku (Tenkan-sen, Kijun-sen, Senkou Span B), um valor para o stop loss, os parâmetros do RSI e o nível de alavancagem.

Dentro do método __init__, iremos inicializar os indicadores usados na estratégia. Estabelecemos o Ichimoku com os períodos desejados e provisionamos as linhas-chave — Tenkan-sen, Kijun-sen, Senkou Span A, Senkou Span B e Chikou Span. Também inicializaremos o indicador RSI, com o período tomado dos parâmetros. No bloco de lógica para sinais de compra e venda, usamos esses dados dos indicadores. A lógica de negociação é implementada dentro do método next, que é

chamado pelo Backtrader a cada novo ponto de dados (ou seja, a cada novo dia, no caso de nossos dados serem diários).

Primeiro, fazemos cálculos para o capital disponível e, em seguida, para o tamanho da posição, de acordo com o valor alavancado. Se uma posição não estiver aberta, verificamos os critérios para entrar no mercado. Entramos comprados quando o preço de fechamento atual está acima das linhas Senkou Span A e B, sinalizando uma tendência de alta, e se o RSI estiver abaixo do limite de sobrecompra para evitar condições de sobrecompra. Entramos vendidos se o preço de fechamento estiver abaixo das linhas Senkou Span A e B e o RSI estiver acima do limite inferior; assim, evitamos uma posição de sobrevenda. Se já estamos no mercado, então temos que continuar observando as condições de saída. Para isso, colocamos um stop loss: um limite para qualquer perda potencial. Se o preço for contra nossa posição por mais de um determinado percentual de limiar, fechamos a posição. Também sairemos da posição caso o preço cruze contra nossa posição atual as linhas Senkou Span A ou B, o que pode ser considerado uma indicação de reversão de tendência.

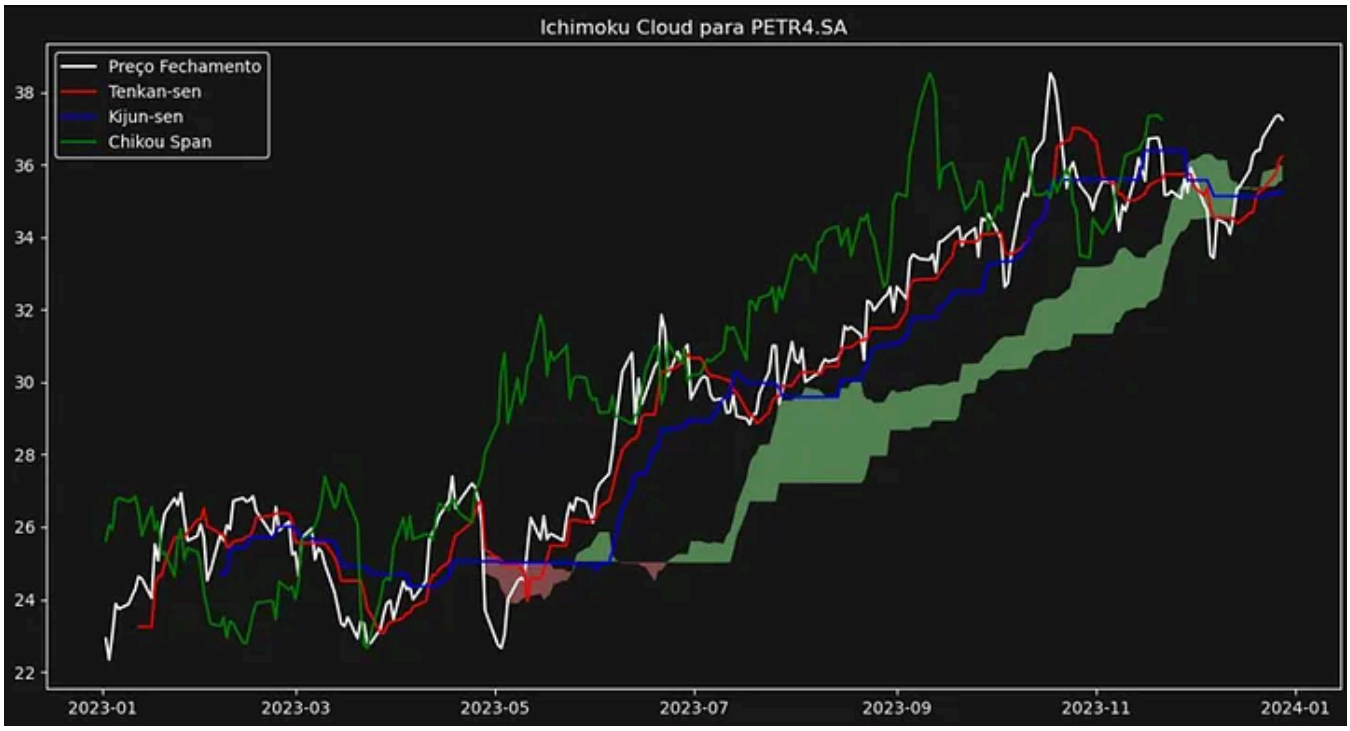
A função run_backtest(params) inicializa o ambiente de backtesting para uma dada combinação de parâmetros. Ela inicializa uma instância do cerebro e carrega nossa estratégia com os parâmetros fornecidos. Baixa dados históricos para PETR4.SA usando o yfinance e os adiciona à instância do cerebro; também define o capital inicial, a comissão para simular custos de transações e a alavancagem. cerebro.run() retorna os parâmetros usados e o valor final do portfólio após a execução da estratégia, fornecendo uma forma de revisar o desempenho da estratégia sob essa configuração.

Na função principal, há a geração de todas as possíveis combinações de parâmetros a serem testados com itertools.product. Isso nos ajudará a buscar com uma estratégia sistemática de exploração através de várias configurações em um histórico de desempenho muito alto. Agora, para cada combinação de parâmetros, simplesmente mantemos o registro disso. No final, procuramos o maior patrimônio final e imprimimos os parâmetros para os quais esse patrimônio ocorreu. Isso é útil ao otimizar a estratégia: ajustar períodos dos indicadores, limites do RSI, valor do stop loss ou outros parâmetros que possam ajudar na melhoria do desempenho. A executamos um backtest chamando run_backtest(params) e técnica de gerenciamento de risco de stop loss e alavancagem é muito útil, mas deve ser usada com cuidado, tendo em mente que aumenta potenciais ganhos e perdas potenciais.

Resultados Finais

Contextualizando, utilizamos uma carteira de 500.000 Reais, e distribuição igual entre cinco ações, PETR4, MGLU3, ITUB4, AZUL4 e NORD3. Nota-se que todas as empresas são de setores diferentes e possuem características diversas acerca da sua atuação no mercado, possibilitando uma análise de resultados mais abrangente.

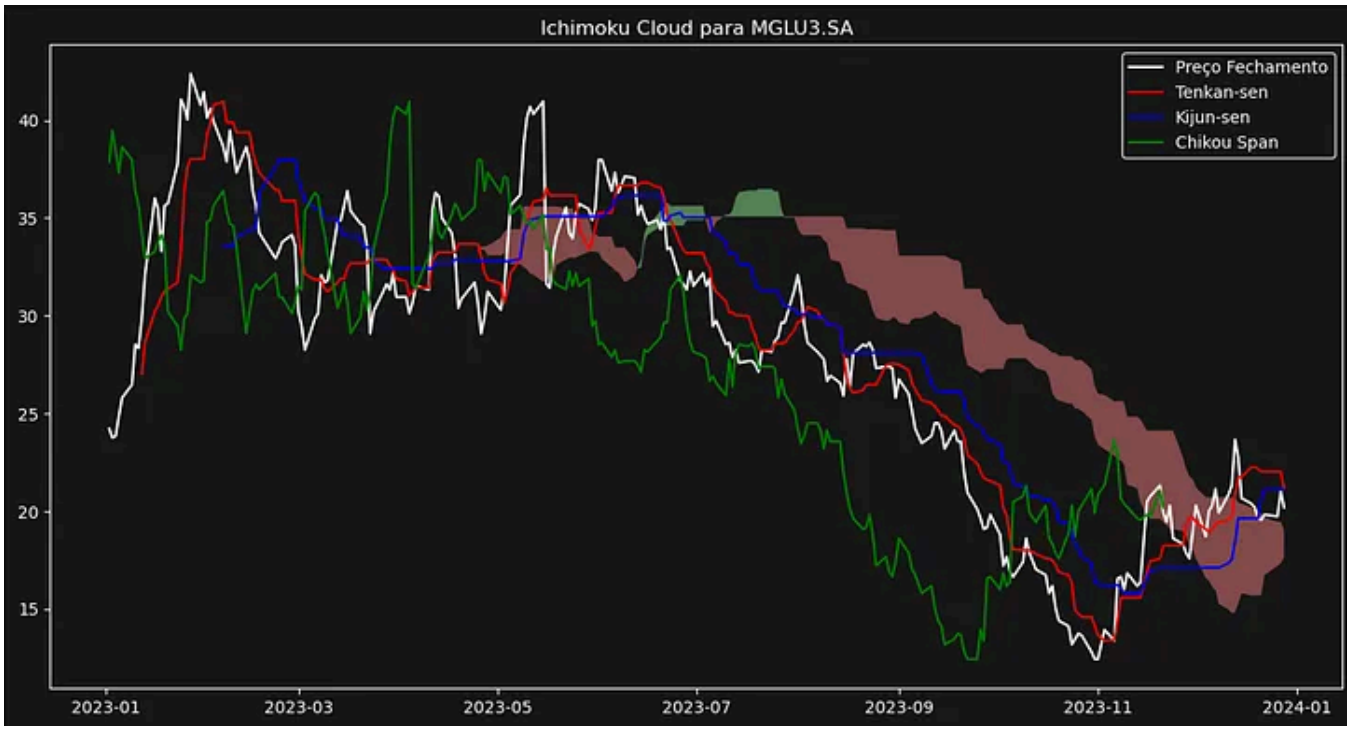
Petrobrás (PETR4)



- Melhores Parâmetros: Tenkan: 7, Kijun: 26, Senkou B: 57, Stop Loss: 0.01
- RSI Período: 10, RSI Sobrecompra: 65, RSI Sobrevenda: 35
- Melhor Capital Final: 96976.2145641327

Nota-se que perdemos 4% do nosso capital inicial, o que é curioso, pois a PETR4 teve uma valorização de 70% ao longo do período registrado, ou seja, perdemos em uma ação teoricamente muito lucrativa. Ao analisar os tradings efetuados diariamente, notamos que, por conta do crescimento constante do ativo, não há muitos sinais de compra devido à condição utilizada em nosso código, quando as linhas Senkou A e B estão acima da curva de crescimento do preço do ativo. Logo, quando entramos comprados o único sinal de venda existente é o stop-loss, onde, por definição perdemos capital.

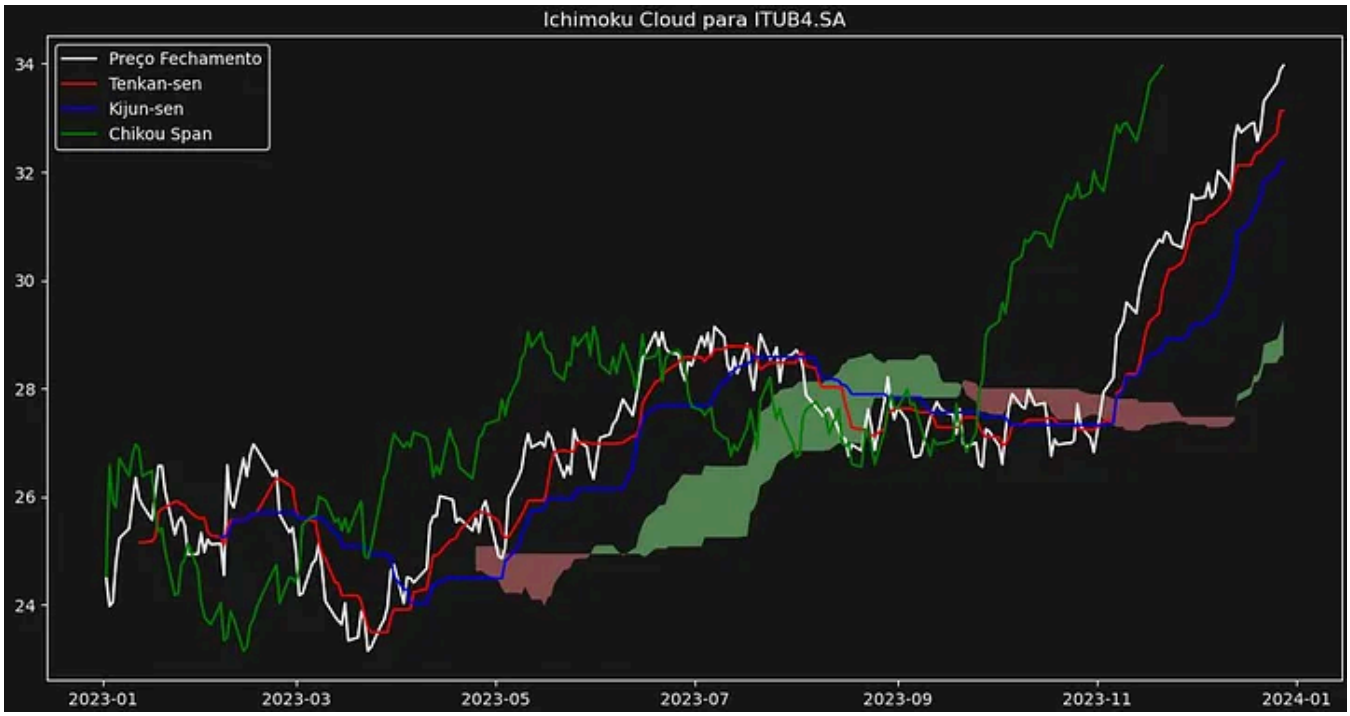
Magazine Luiza (MGLU3)



- Melhores Parâmetros: Tenkan: 7, Kijun: 26, Senkou B: 58, Stop Loss: 0.01
- RSI Período: 10, RSI Sobrecompra: 65, RSI Sobrevenda: 25
- Melhor Capital Final: 163892.8999345207

No caso da MGLU3, temos o efeito contrário da PETR4, onde uma ação em desvalorização de 60%, aproximadamente, lucramos 63% em relação ao capital inicial de 100.000 reais. Fato esse que produz insights importantes sobre nosso algoritmo de trading e nos leva a algumas hipóteses, uma delas é a de que ele tem eficiência aumentada em periodos de Alta Volatilidade, pois ativa com mais frequência os gatilhos de compra e venda, e a outra hipótese é a de que ele funciona melhor em ativos em desvalorização, como observado no exemplo da MGLU3. Para checar isso, escolhemos ITUB4, uma ação bastante valorizada no ano de 2023, e AZUL4, uma ação que despencou em 2023.

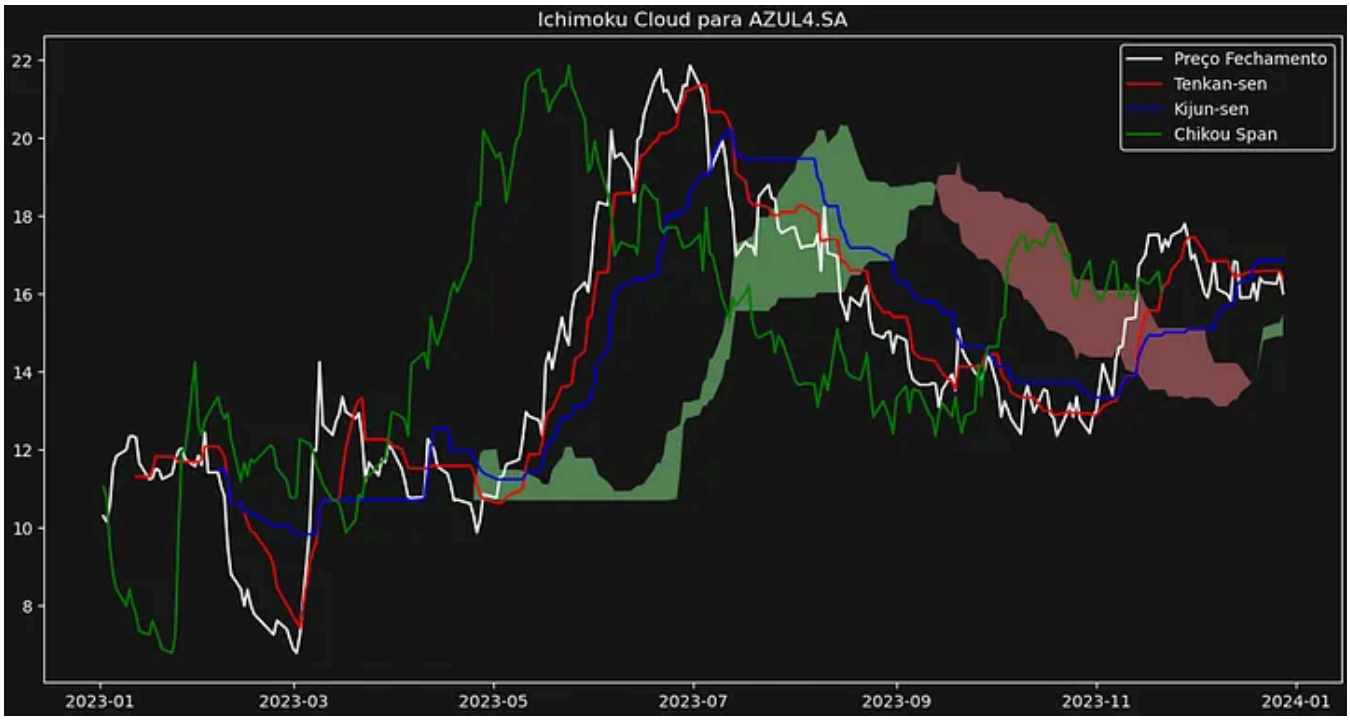
Itaú (ITUB4)



- Melhores Parâmetros: Tenkan: 7, Kijun: 26, Senkou B: 52, Stop Loss: 0.01
- RSI Período: 10, RSI Sobrecompra: 65, RSI Sobrevenda: 35
- Melhor Capital Final: 83146.63952479736

Observando nossos resultados, perdemos 17% do capital inicial, em uma ação que valorizou cerca de 45% no ano de 2023. Indo para uma análise mais profunda, isto é, olhando quais trades foram realizadas, observamos que nosso algoritmo foi, novamente, incapaz de detectar a tendência de alta do ativo, perdendo boa parte do capital com o Stop-Loss em curto prazo e, principalmente, no mês 3, onde ele detectou o movimento de alta, mas, na prática, os preços despencaram logo em seguida.

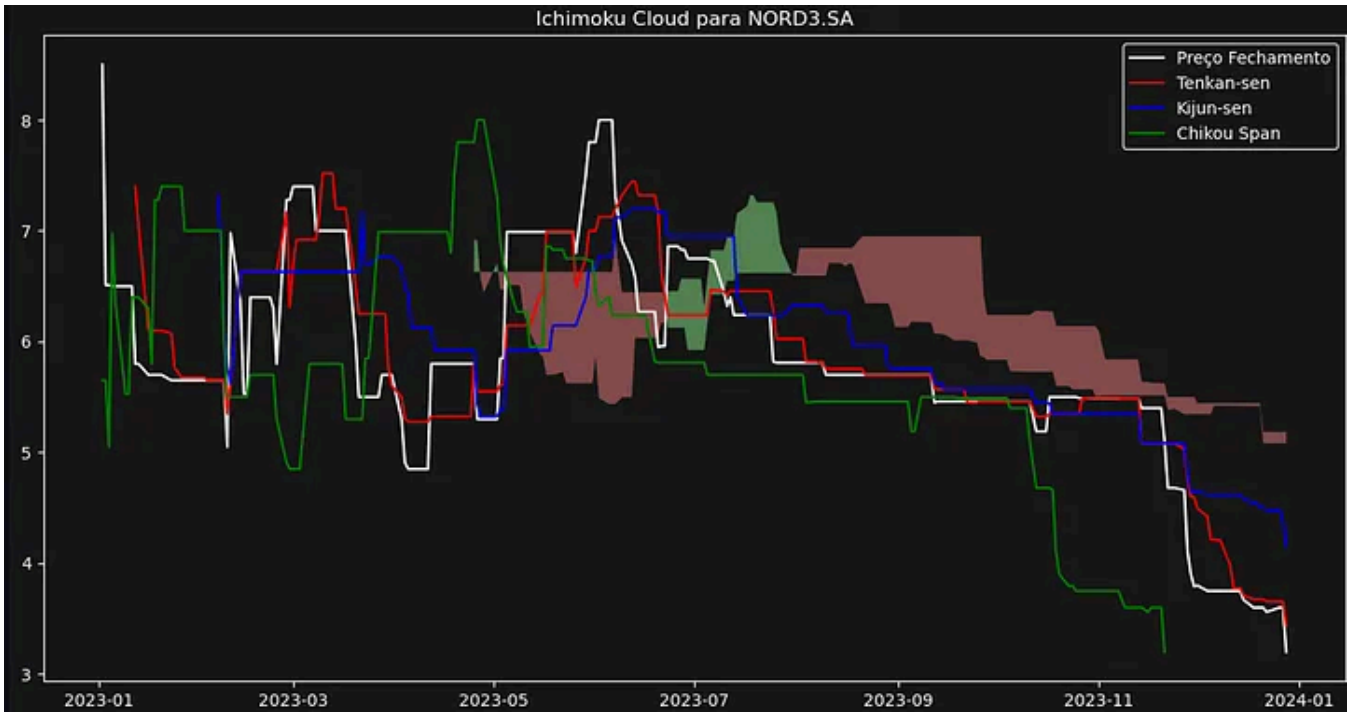
Azul (AZUL4)



- Melhores Parâmetros: Tenkan: 7, Kijun: 26, Senkou B: 55, Stop Loss: 0.01
- RSI Período: 10, RSI Sobrecompra: 65, RSI Sobrevenda: 35
- Melhor Capital Final: 113502.07690647691

A fim de testar nossa hipótese sobre o algoritmo ser mais eficiente em momentos de alta volatilidade, escolhemos a AZUL4 para fazermos o HFT (High-Frequency Trading). Como se pode observar no gráfico, esse ativo foi altamente volátil no período registrado, e tivemos um capital 13% maior que o inicial, isso porque tivemos diversos gatilhos de entrada no mercado.

Nordon (NORD3)



- Melhores Parâmetros: Tenkan: 7, Kijun: 26, Senkou B: 59, Stop Loss: 0.01
- RSI Período: 10, RSI Sobrecompra: 65, RSI Sobrevenda: 35
- Melhor Capital Final: 210410.83310475538

Por fim, escolhemos uma ação com pouquíssima liquidez e altamente volátil, onde obtivemos os nossos melhores resultados, cerca de 110% de lucro em relação ao capital inicial. Apesar de serem todas as empresas de setores diferentes é possível se observar que nossa teoria sobre a alta volatilidade aparenta estar correta, dado que houveram vários gatilhos de entrada e saída no mercado, aproveitando as altas e as baixas de maneira favorável ao aumento do capital.

Conclusão

A análise gráfica Ichimoku atinge seu objetivo principal de ajudar a visualizar as tendências que um ativo tem no mercado, entretanto possui limitações, que se resumem ao tipo de ativo e sua volatilidade — evidenciado nos resultados finais.

Além disso, como qualquer ferramenta técnica, o Ichimoku deve ser utilizado em conjunto com outras formas de análise para que o investidor tenha uma visão mais completa. Também é importante considerar o contexto econômico e os eventos macroeconômicos que podem impactar o mercado de maneira súbita, já que o Ichimoku, por ser uma ferramenta preditiva baseada em dados históricos recentes, pode não capturar movimentos inesperados no longo prazo.

Por fim, o uso adequado desta técnica necessita de prática e familiaridade, pois a interpretação dos seus componentes pode variar de acordo com o mercado e as condições específicas do ativo analisado.

Recomendações de extensões do VS Code para otimizar a sua produtividade

Apr 13 73

In FEA.dev by FEA.dev

Preço dos Carros no Brasil: Análise de Dados com Python

Por Guilherme Freitas, Henrique Lindoso e Milena Ramos

Nov 6, 2023 78

In FEA.dev by FEA.dev

Python Dentro do Excel: Uma Revolução para a Análise de Dados

Por Felippe Bertollo e Rodrigo Souza

Aug 11 3

FEA.dev

Roadmap de Finanças Quantitativas: Profissional em “I”

Por Gabriel Braz e Lorenzo Filippin

Sep 13 1

See all from FEA.dev

Recommended from Medium

In Mac O'Clock by Andrew Zuo

The M4 MacBook Pro Makes Me Want To Buy A Windows Laptop

Oct 31 1K 74

Jon Krakauer

Embrace the Misery

The world is going to hell. Here's how to deal with it.

4d ago 7.2K 165

Lists

Staff Picks

762 stories · 1427 saves

Stories to Help You Level-Up at Work

19 stories · 860 saves

Self-Improvement 101

20 stories · 2985 saves

Productivity 101

20 stories · 2538 saves

Harendra

How I Am Using a Lifetime 100% Free Server

Get a server with 24 GB RAM + 4 CPU + 200 GB Storage + Always Free

Oct 26 3.6K 41

In Stackademic by Abdur Rahman

Python is No More The King of Data Science

5 Reasons Why Python is Losing Its Crown

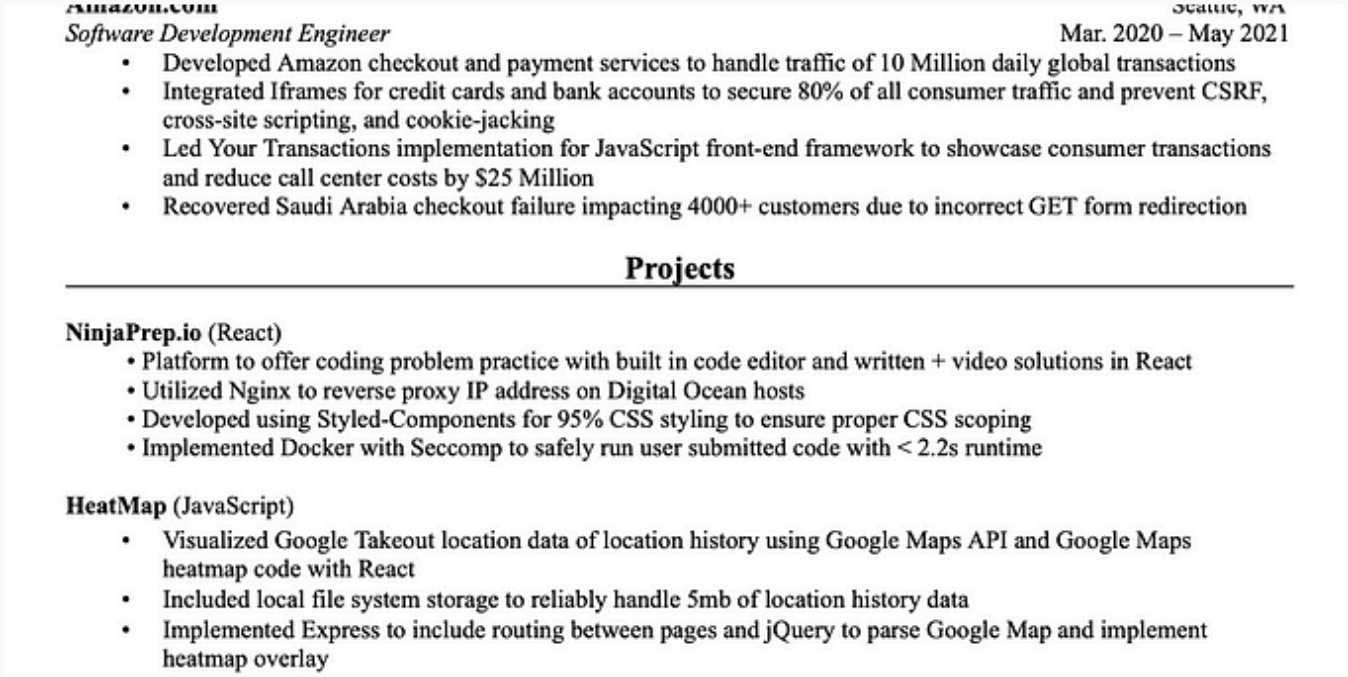
Oct 23 4.3K 23

In Code Like A Girl by Aleena

Java Annotations vs. Python Decorators

Which One Is Better? Which One Is Overused?

5d ago 593 19



Amazon.com Seattle, WA
Software Development Engineer Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by $25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)
- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)
- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay

In Level Up Coding by Alexander Nguyen

The resume that got a software engineer a $300,000 job at Google.

1-page. Well-formatted.

 Jun 1  25K  501 

See more recommendations