GIẢI THÍCH CHIẾN LƯỢC

1. Bộ lọc dữ liệu

Khai báo thư viện: - vnstock: Thư viện chuyên phân tích dữ liệu chứng khoán Việt Nam. - pandas as pd: Khai báo thư viện pandas với tên gọi tắt pd. Pandas là thư viện chuyên dùng để thao tác và phân tích dữ liệu dạng bảng (DataFrame). - numpy as np: Khai báo thư viện numpy với tên gọi tắt np. NumPy là thư viện nền tảng cho các tính toán khoa học trong Python, cung cấp các mảng đa chiều hiệu suất cao. - matplotlib.pyplot as plt: Khai báo thư viện matplotlib.pyplot với tên gọi tắt plt. Thư viện này dùng để tạo các biểu đồ (plot) để trực quan hóa dữ liệu. - ffn: Thư viện chuyên dùng cho việc tính toán các chỉ số tài chính. - bt: Thư viện dùng để xây dựng các chiến lược giao dịch (backtesting). - %matplotlib inline giúp hiển thị các biểu đồ được tạo bằng plt trực tiếp trong Jupyter Notebook hoặc môi trường tương tự.	from vnstock import * import pandas as pd import numpy as np import matplotlib.pyplot as plt import ffn import bt %matplotlib inline

Lấy và lọc dữ liệu Lấy dữ liệu của các mã cổ phiếu trên sàn HOSE, HNX vad UPCOM	params = {"exchangeName": "HOSE, HNX, UPCOM"}
Gọi hàm stock_screening_insights với các tham số: - params: Dictionary chứa tên các sàn giao dịch. - size=1700: Số lượng cổ phiếu tối đa được lấy. - drop_lang='vi': Loại bỏ nội dung tiếng Việt (nếu có).	df = stock_screening_insights(params, size=1700, drop_lang='vi')
Hợp nhất với dữ liệu công ty niêm yết để lấy thêm thông tin	listing_df = listing_companies() df = df.merge(listing_df, how='left', on='ticker')
Lọc cổ phiếu trong chỉ số VN30 Ưu điểm của các cổ phiếu trong VN30: - Thanh khoản cao, dễ mua bán. - Vốn hóa lớn, tiềm năng tăng trưởng ổn định. - Rủi ro thấp, an toàn cho nhà đầu tư. - Dẫn dắt thị trường, xu hướng tăng/giảm rõ ràng. - Dễ theo dõi thông tin, cập nhật tình hình doanh nghiệp.	df = df[df['VN30']==True]
Sử dụng hai yếu tố để chọn cổ phiếu:	Price above SMA(50), Price above SMA(100), rsi14 between 30 and 75 df = df[(df['priceVsSma50.en'] == 'Price

1. priceVsSma50.en: - Biểu thị giá cổ phiếu so với đường trung bình động (SMA) 50 ngày. - Giá trị "Price above SMA(50)" cho biết giá cổ phiếu hiện tại đang cao hơn SMA 50 ngày. 2. priceVsSma100.en: - Tương tự như priceVsSma50.en, nhưng so sánh giá cổ phiếu với SMA 100 ngày. - Giá trị "Price above SMA(100)" cho biết giá cổ phiếu hiện tại đang cao hơn SMA 100 ngày. Kết hợp hai yếu tố: - Lọc ra các cổ phiếu có giá hiện tại cao hơn cả SMA 50 ngày và SMA 100 ngày. - Điều này cho thấy xu hướng tăng giá tiềm năng của cổ phiếu trong trung hạn (50 ngày) và dài hạn (100 ngày).	above SMA(50)') & (df['priceVsSMA100.en'] == 'Price above SMA(100)')]
Điền giá trị NaN vào cột 'evEbitda' bằng 0. Lý do là vì trong dữ liệu có nhiều công ty không có dữ liệu về evEbitda và thêm vào nếu lọc evEbitda sẽ không vô tình loại bỏ những cổ phiếu này. Lọc ra các mã cổ phiếu thoả mãn các điều kiện sau:	df['evEbitda'].fillna(0, inplace=True) df = df[(df['epsGrowth5Year'] > 0) & (df['revenueGrowth5Year'] > 0) & (df['pe'] < 20) & (df['marketCap'] < 300000) & (df['rsi14'] >= 30) & (df['rsi14'] <= 75) & (df['evEbitda'] >= 0) & (df['financialHealth'] > 3.0)]

- Tăng trưởng lợi nhuận trên mỗi cổ phiếu (EPS) 5 năm qua > 0: cho thấy doanh nghiệp có hiệu quả kinh doanh tốt. - Tăng trưởng doanh thu 5 năm qua > 0: cho thấy doanh nghiệp đang phát triển. - Tỷ lệ P/E (PE) < 20: cho thấy giá cổ phiếu đang được định giá hợp lý. - Vốn hóa thị trường (Market Cap) < 300.000 tỷ đồng: tập trung vào các công ty vừa và nhỏ, có tiềm năng tăng trưởng cao. - Chỉ báo RSI (Relative Strength Index) 14 ngày nằm trong khoảng 30 - 75: cho thấy cổ phiếu không quá mua hay quá bán. - Giá trị doanh nghiệp trên lợi nhuận trước lãi vay, thuế, khấu hao và khấu hao (EV/EBITDA) >= 0: cho thấy doanh nghiệp có khả năng tạo ra lợi nhuận. - Chỉ số sức khỏe tài chính (Financial Health) > 3.0: cho thấy doanh nghiệp có tình hình tài chính lành mạnh. - Đánh giá của cổ phiếu (stockRating) > 3.0: cho thấy doanh nghiệp có hoạt động kinh doanh ổn định.	
Phân tích từng cổ phiếu - Các cổ đều là ngân hàng do VN30 đa số cổ ngân hàng và tăng trưởng nó cũng tốt hơn những cổ phiếu khác.	stock_ls_analysis("ACB, CTG, MBB, TCB ")

Lấy dữ liệu từ file của BTC và làm sạch	file_names = ["ACB.csv", "CTG.csv", "MBB.csv"]
Tạo danh sách lưu trữ các DataFrame riêng lẻ	data = []
Đọc từng tệp CSV Chuyển đổi cột 'Date' thành kiểu datetime, hỗ trợ các thao tác thời gian trong tương lai.	for file_name in file_names: df = pd.read_csv(file_name) df['Date'] = pd.to_datetime(df['Date'])
Đặt 'Date' làm index, giúp việc truy cập và phân tích dữ liệu theo thời gian trở nên thuận tiện hơn.	df.set_index('Date', inplace=True)
Dòng này đổi tên cột 'Close' thành tên của file (loại bỏ 4 ký tự '.csv' cuối cùng), giúp nhận biết dữ liệu từ các file khác nhau dễ dàng hơn trong DataFrame tổng hợp.	df.rename(columns={'Close': file_name[:-4]}, inplace=True)
Thêm DataFrame đã xử lý (df) vào danh sách data. Sau khi vòng lặp kết thúc, danh sách data sẽ chứa các DataFrame từ tất cả các file CSV được đọc và xử lý.	data.append(df)
Hợp nhất tất cả các DataFrames trên chỉ mục của chúng - Hàm concat của thư viện Pandas dùng để kết hợp nhiều DataFrame. - data: Tham số này là một list chứa các DataFrame cần kết hợp (được tạo ra từ vòng lặp xử lý các file CSV ở đoạn trước). - axis=1: Tham số này xác định trục để thực hiện phép kết hợp. Trong này, axis=1 nghĩa là kết hợp các	merged_df = pd.concat(data, axis=1)

DataFrame theo cột (dọc theo các cột). Chọn các hàng bắt đầu từ ngày 1-1-2018, chỉ giữ lại cột “Close” và loại bỏ các dòng có giá trị thiếu (NaN):	data = merged_df.loc['2020-04-01':, ['ACB', 'CTG', 'MBB']] data = data.dropna()

2. Chiến lược SMA

Tính toán SMA (Đường trung bình động) & Trực quan hoá	sma = data.rolling(50).mean() plot = bt.merge(data, sma).plot(figsize=(15, 5))

Thuật toán xác định chọn:
 	
- Lớp SelectWhere dùng để chọn lọc các chứng khoán dựa trên một tín hiệu.
+ bt.Algo**:** Đây là lớp nền tảng để xây dựng các thuật toán giao dịch trong thư viện bt.
+ signal (tham số đầu vào): Là một bảng dữ liệu (DataFrame) chứa tín hiệu giao dịch. Tín hiệu này thường là giá trị True hoặc False.
 	
Phương thức hoạt động:
 	
- Khi tạo lớp SelectWhere, ta cung cấp bảng dữ liệu tín hiệu signal làm tham số.
	class SelectWhere(bt.Algo):
	  def __init__(self, signal):
	  self.signal = signal
	  def __call__(self, target):
	  if target.now in self.signal.index:
	  sig = self.signal.loc[target.now]
	  selected = list(sig.index[sig])
	  target.temp['selected'] = selected
	  return True

- Lọc theo tín hiệu: Khi được gọi (__call__), lớp này kiểm tra xem ngày hiện tại (target.now) có nằm trong bảng tín hiệu self.signal không. Nếu có: + Lấy ra tín hiệu cho ngày hiện tại (sig). + Tạo danh sách các chứng khoán được chọn lọc dựa trên tín hiệu True (selected). + Lưu danh sách này vào biến tạm 'selected' của đối tượng target. Nếu không: + Không thực hiện lọc. - Trả về True**:** Cuối cùng, phương thức trả về True để báo hiệu việc thực hiện thành công.	
Tạo ra tín hiệu giao dịch bằng cách so sánh dữ liệu giá cổ phiếu (data) với đường trung bình động 50 ngày (sma). Mục đích là xác định các cổ phiếu có giá cao hơn SMA 50 ngày tại mỗi thời điểm.	signal = data > sma
Khởi tạo một chiến lược backtest có tên "above50sma" và sử dụng các thuật toán sau: - Sử dụng tín hiệu giao dịch (data > sma) để chọn các cổ phiếu có giá cao hơn SMA 50 ngày tại mỗi thời điểm. - Phân bổ trọng số bằng nhau cho các cổ phiếu được lựa chọn, nghĩa là đầu tư đồng đều vào các cổ phiếu đó.	s=bt.Strategy('above50sma', [SelectWhere(data > sma), bt.algos.WeighEqually(), bt.algos.Rebalance()])

- Thực hiện cân bằng lại danh mục đầu tư định kỳ (thường xuyên), đảm bảo duy trì tỷ lệ phân bổ ban đầu giữa các cổ phiếu.	
Khởi tạo và chạy backtest (t) sử dụng chiến lược đã định nghĩa (s) và dữ liệu giá cổ phiếu (data) để thực hiện kiểm tra.	t = bt.Backtest(s, data) res_1 = bt.run(t1) res_1.plot(); res_1.display()

3. Chiến lược giao thoa đường trung bình động (SMA Crossover)

Lớp WeighTarget cho phép thiết lập tỷ trọng mục tiêu cho danh mục đầu tư theo từng ngày trong backtest. Nó lấy tỷ trọng từ DataFrame target_weights và lưu trữ vào biến tạm trong đối tượng backtest target.	class WeighTarget(bt.Algo):
	def __init__(self, target_weights):
	self.tw = target_weights
	def __call__(self, target):
	if target.now in self.tw.index:
	w = self.tw.loc[target.now]
	target.temp['weights'] = w.dropna()
	return True

Tính toán đường trung bình động (SMA) 10 ngày và 200 ngày cho dữ liệu giá cổ phiếu (data).	sma10= data.rolling(10).mean() sma200 = data.rolling(200).mean()
Tạo DataFrame tỷ trọng mục tiêu: - Sao chép DataFrame của sma200 để tạo DataFrame tw lưu trữ tỷ trọng mục tiêu. Cấu trúc của hai DataFrame này sẽ giống nhau. - Gán giá trị 1.0 cho các tỷ trọng mục tiêu tương ứng với các cổ phiếu có giá vượt lên trên SMA 10 ngày. Điều này thể hiện chiến lược muốn mua (long) những cổ phiếu có xu hướng tăng giá.	tw = sma200.copy() tw[sma10 > sma200] = 1.0 tw[sma10 <= sma200] = -1.0 tw[sma200.isnull()] = 0.0

- Gán giá trị -1.0 cho các tỷ trọng mục tiêu tương ứng với các cổ phiếu có giá bằng hoặc thấp hơn SMA 10 ngày. Điều này có thể thể hiện chiến lược muốn bán (short) những cổ phiếu đang đi ngang hoặc giảm giá (tùy thuộc vào cách hiểu của chiến lược cụ thể). - Gán giá trị 0.0 cho các tỷ trọng mục tiêu tại các thời điểm mà SMA 200 chưa được tính toán xong (giá trị NaN). Điều này đảm bảo không có tác động đến danh mục đầu tư trong giai đoạn khởi đầu backtest khi dữ liệu chưa đủ 200 ngày.	
Khởi tạo chiến lược backtest "ma_cross" sử dụng hai thuật toán: - Sử dụng đối tượng WeighTarget đã tạo trước đó để thiết lập tỷ trọng mục tiêu dựa trên DataFrame tw. - Thực hiện cân bằng lại danh mục đầu tư theo tỷ trọng mục tiêu đã tính toán, đảm bảo danh mục luôn tuân theo chiến lược.	ma_cross = bt.Strategy('ma_cross', [WeighTarget(tw),
Khởi tạo đối tượng backtest t sử dụng chiến lược "ma_cross" và dữ liệu giá cổ phiếu (data). - Chạy backtest và lưu trữ kết quả trong biến res. - Hiển thị kết quả backtest dưới dạng biểu đồ và bảng chi tiết. Biểu đồ có thể cho thấy đường lợi nhuận của chiến lược, còn bảng sẽ cung cấp các chỉ số hiệu suất như lợi nhuận	bt.algos.Rebalance()]) t = bt.Backtest(ma_cross, data) res = bt.run(t) res.plot(); res.display()

trung bình, tỷ lệ rút vốn tối đa, Sharpe ratio, v.v.	

4. Chiến lược mua và giữ

Khởi tạo thuật toán runMonthlyAlgo đảm bảo các hành động trong chiến lược chỉ được thực hiện vào đầu mỗi tháng.	runMonthlyAlgo= = bt.algos.RunMonthly(run_on_first_date=True)
Định nghĩa Series weights với tỷ trọng mục tiêu cho mỗi tài sản (cột trong data). Ví dụ, giá trị (0.3, 0.4, 0.3) thể hiện tỷ lệ phân bổ mong muốn cho từng tài sản trong danh mục đầu tư. Khởi tạo thuật toán weighSpecifiedAlgo sử dụng Series weights. Thuật toán này chỉ được kích hoạt khi runMonthlyAlgo trả về True (vào đầu mỗi tháng). Khi được kích hoạt, nó thiết lập tỷ trọng mục tiêu cho danh mục đầu tư dựa trên giá trị trong weights. Khởi tạo thuật toán rebalAlgo đơn giản để điều chỉnh danh mục đầu tư phù hợp với tỷ trọng mục tiêu do weighSpecifiedAlgo thiết lập. Thuật toán này chạy bất cứ khi nào weighSpecifiedAlgo thiết lập tỷ trọng mới, đảm bảo danh mục đầu tư luôn cân bằng theo chiến lược.	weights = pd.Series([0.3, 0.4, 0.3],index = data.columns) weighSpecifiedAlgo = bt.algos.WeighSpecified(**weights) rebalAlgo = bt.algos.Rebalance()
Định nghĩa chiến lược backtest "static" sử dụng ba thuật toán: - Đảm bảo thực thi hàng tháng. - Thiết lập tỷ trọng mục tiêu vào đầu mỗi tháng.	strat = bt.Strategy('static', [runMonthlyAlgo, weighSpecifiedAlgo, rebalAlgo]

- Cân bằng lại danh mục đầu tư để phù hợp với tỷ trọng mục tiêu. Khởi tạo và chạy đối tượng backtest với chiến lược strat, dữ liệu giá lịch sử data và đặt integer_positions=False. Tham số này cho phép chia nhỏ cổ phiếu, có thể cần thiết tùy thuộc vào loại tài sản và chiến lược của bạn.	) t3 = bt.Backtest(strat, data, integer_positions=False) res_3 = bt.run(t3) res_3.display() res_3.plot()

5. Chiến lược kết hợp

Tạo và backtest các chiến lược riêng biệt cho trọng số bằng nhau và trọng số nghịch đảo biến động. Sau đó, kết hợp các chiến lược này, cân bằng chúng đều đặn trong một danh mục đầu tư kết hợp.	strategy_names = np.array(['Equal Weight', 'Inv Vol']) runMonthlyAlgo = bt.algos.RunMonthly(run_on_first_date=True, run_on_end_of_period=True) selectAllAlgo = bt.algos.SelectAll() rebalanceAlgo = bt.algos.Rebalance() strats = [] tests = []

Đoạn code này lặp qua từng tên chiến lược, chọn thuật toán cân bằng danh mục phù hợp, sau đó tạo và lưu trữ chiến lược cùng với phiên backtest tương ứng. Cuối cùng, ta có danh sách các chiến lược strats	for i,s in enumerate(strategy_names):
	if s == "Equal Weight":
	wAlgo = bt.algos.WeighEqually()
	elif s == "Inv Vol":
	wAlgo = bt.algos.WeighInvVol()
	
	strat = bt.Strategy(
	str(s),
	[

và danh sách các phiên backtest tests để có thể chạy và phân tích kết quả.	runMonthlyAlgo, selectAllAlgo, wAlgo, rebalanceAlgo]) strats.append(strat) t = bt.Backtest(strat, data, integer_positions = False, progress_bar=False) tests.append(t)
Tạo một chiến lược mới có tên "Combined". - Sử dụng đối số children để bao gồm các chiến lược riêng lẻ như các thành phần. - Áp dụng WeighEqually để phân bổ vốn đều đặn giữa các chiến lược con. Backtest chiến lược kết hợp: - Tạo đối tượng backtest sử dụng chiến lược kết hợp và dữ liệu. - Chạy backtest. - Vẽ biểu đồ tỷ trọng tài sản theo thời gian để trực quan hóa thay đổi phân bổ.	combined_strategy = bt.Strategy('Combined', algos = [runMonthlyAlgo, selectAllAlgo, bt.algos.WeighEqually(), rebalanceAlgo], children = [x.strategy for x in tests]) t4 = bt.Backtest(combined_strategy, data, integer_positions = False, progress_bar = False) res_4 = bt.run(t4) res_4.display() res_4.plot()

- Về tổng quan bài làm, chúng tôi sử dụng thời gian bắt đầu sau Covid-19 vì đó là dữ liệu outlier không phù hợp đưa vào backtest và xây dựng đa dạng các chiến lược với từng mục tiêu như là chiến lược SMA sẽ nhắm vào trung và dài hạn, Chiến lược Buy-and-Hold thì sẽ nhắm vào dài hạn và cuối cùng sẽ là chiến lược kết hợp tạo ra chiến lược gốc (kết hợp) với 2 chiến lược con (Trọng lượng bằng nhau, Số lượng đầu tư). Hiện tại các chiến lược vẫn chưa có chỉnh số liệu để phù hợp cho thị trường chứng khoán cơ sở Việt Nam, tuy nhiên thì kết quả sẽ vẫn không thay đổi nhiều chẳng hạn như T+1,5 do là các chiến lược đều nhắm vào các mục tiêu trung và dài hạn nên sẽ ít có các giao dịch ngắn hạn hoặc là thuế phí 0.35% thì cũng do giao dịch trung và dài hạn nên là khi trừ đi phí total return vẫn không thay đổi nhiều và nếu tính theo như vậy thuật toán vẫn có lợi nhuận cao hơn các quỹ trong nước.