# INPUT: Lista de tickers y fuente de datos

tickers = ["NVDA", "AAPL", "MSFT", "META", "GOOGL", "TSLA", "AMZN", "GE", "HD", "CAT", "WMT"]

datos\_historicos = cargar\_datos\_OHLCV(tickers, periodo="5 años") # Ej: desde API como Yahoo Finance

# BLOQUE A: Preparación de Datos

FOR cada ticker EN tickers:

datos = datos\_historicos[ticker] # DataFrame con columnas: Open, High, Low, Close, Volume

# Limpieza: Eliminar NaN, outliers (ej: valores > 3 desv. est.)

datos = limpiar\_datos(datos)

# Normalización: Calcular retornos diarios y semanales

datos["Retorno\_Diario"] = (datos["Close"] - datos["Close"].shift(1)) / datos["Close"].shift(1)

datos["Retorno\_Semanal"] = (datos["Close"] - datos["Close"].shift(5)) / datos["Close"].shift(5) # Asumiendo 5 días hábiles

# Resample para datos semanales (para indicadores de largo plazo)

datos\_semanales = resample\_a\_semanal(datos) # Agregar OHLCV semanal

# BLOQUE B: Cálculo de Indicadores Técnicos

# Indicadores para corto plazo (diario, períodos cortos)

macd\_corto = calcular\_MACD(datos["Close"], rapido=12, lento=26, señal=9)

rsi\_corto = calcular\_RSI(datos["Close"], periodo=14)

bollinger\_corto = calcular\_Bollinger(datos["Close"], periodo=20, desv=2)

ma\_corto = calcular\_MA(datos["Close"], sma\_periodo=50, ema\_periodo=20) # SMA y EMA cortas

patrones\_corto = detectar\_patrones(datos["Close"], ventana=30) # Ej: Hombro-Cabeza-Hombro en ventana corta

tendencias\_corto = identificar\_tendencias(datos["Close"], soporte\_resistencia\_ventana=20)

# Indicadores para largo plazo (semanal, períodos largos)

macd\_largo = calcular\_MACD(datos\_semanales["Close"], rapido=26, lento=52, señal=18)

rsi\_largo = calcular\_RSI(datos\_semanales["Close"], periodo=28)

bollinger\_largo = calcular\_Bollinger(datos\_semanales["Close"], periodo=40, desv=2.5)

ma\_largo = calcular\_MA(datos\_semanales["Close"], sma\_periodo=100, ema\_periodo=50) # SMA y EMA largas

patrones\_largo = detectar\_patrones(datos\_semanales["Close"], ventana=60) # Ventana más amplia

tendencias\_largo = identificar\_tendencias(datos\_semanales["Close"], soporte\_resistencia\_ventana=40)

# BLOQUE C: Generador de Recomendaciones

# Generar señales individuales para corto plazo

señales\_corto = []

señales\_corto.append(generar\_señal\_MACD(macd\_corto)) # +1 si cruce alcista, -1 bajista, 0 neutro

señales\_corto.append(generar\_señal\_RSI(rsi\_corto)) # +1 si <30 (sobreventa), -1 si >70 (sobrecompra), 0 otherwise

señales\_corto.append(generar\_señal\_Bollinger(bollinger\_corto, datos["Close"].iloc[-1])) # +1 si por debajo banda baja, -1 por arriba alta

señales\_corto.append(generar\_señal\_MA(ma\_corto, datos["Close"].iloc[-1])) # +1 si EMA > SMA (alcista), -1 inverso

señales\_corto.append(generar\_señal\_Patrones(patrones\_corto)) # +1 si patrón alcista (ej: triángulo ascendente), -1 bajista

señales\_corto.append(generar\_señal\_Tendencias(tendencias\_corto, datos["Close"].iloc[-1])) # +1 si por encima soporte, -1 por debajo resistencia

# Generar señales individuales para largo plazo (similar lógica, pero con datos semanales)

señales\_largo = []

señales\_largo.append(generar\_señal\_MACD(macd\_largo))

señales\_largo.append(generar\_señal\_RSI(rsi\_largo))

señales\_largo.append(generar\_señal\_Bollinger(bollinger\_largo, datos\_semanales["Close"].iloc[-1]))

señales\_largo.append(generar\_señal\_MA(ma\_largo, datos\_semanales["Close"].iloc[-1]))

señales\_largo.append(generar\_señal\_Patrones(patrones\_largo))

señales\_largo.append(generar\_señal\_Tendencias(tendencias\_largo, datos\_semanales["Close"].iloc[-1]))

# Regla de consenso: Mayoría simple (al menos 3 de 6 señales coinciden en +1 o -1)

recomendacion\_dia = calcular\_mayoria(señales\_corto) # "Compra" si sum(señales) >= 3, "Venta" si <= -3, "Neutro" otherwise

recomendacion\_semana = calcular\_mayoria(señales\_largo) # Similar

# OUTPUT: Diccionario o tabla con resultados

resultados[ticker] = {"Recomendacion\_Dia": recomendacion\_dia, "Recomendacion\_Semana": recomendacion\_semana}

# Funciones auxiliares (definidas en pseudocódigo)

FUNCTION calcular\_MACD(precios, rapido, lento, señal):

ema\_rapido = EMA(precios, rapido)

ema\_lento = EMA(precios, lento)

macd\_line = ema\_rapido - ema\_lento

signal\_line = EMA(macd\_line, señal)

RETURN {"macd": macd\_line, "signal": signal\_line}

FUNCTION generar\_señal\_MACD(macd\_data):

IF macd\_data["macd"].iloc[-1] > macd\_data["signal"].iloc[-1] AND macd\_data["macd"].iloc[-2] <= macd\_data["signal"].iloc[-2]:

RETURN +1 # Cruce alcista

ELSE IF macd\_data["macd"].iloc[-1] < macd\_data["signal"].iloc[-1] AND macd\_data["macd"].iloc[-2] >= macd\_data["signal"].iloc[-2]:

RETURN -1 # Cruce bajista

ELSE:

RETURN 0

# Similar para otras funciones: RSI (fórmula estándar), Bollinger (media ± desv), etc.

# Función de mayoría

FUNCTION calcular\_mayoria(señales):

suma = sum(señales)

IF suma >= 3:

RETURN "Compra"

ELSE IF suma <= -3:

RETURN "Venta"

ELSE:

RETURN "Neutro"

# forecasting\_model.py

"""

Modelo de Forecasting Financiero basado en indicadores técnicos.

Genera recomendaciones de compra/venta para tickers específicos en horizontes diario y semanal.

Autor: [Tu Nombre]

Fecha: Septiembre 2025

Requisitos: pip install yfinance pandas numpy

"""

import pandas as pd

import numpy as np

import yfinance as yf

from datetime import datetime, timedelta

# Lista de tickers del documento

TICKERS = ["NVDA", "AAPL", "MSFT", "META", "GOOGL", "TSLA", "AMZN", "GE", "HD", "CAT", "WMT"]

def cargar\_datos\_OHLCV(tickers, periodo="5y"):

"""

Carga datos OHLCV desde Yahoo Finance.

"""

datos = {}

for ticker in tickers:

try:

df = yf.download(ticker, period=periodo)

datos[ticker] = df[['Open', 'High', 'Low', 'Close', 'Volume']]

except Exception as e:

print(f"Error al descargar {ticker}: {e}")

return datos

def limpiar\_datos(df):

"""

Limpia datos: elimina NaN y outliers (más de 3 desv. est. en Close).

"""

df = df.dropna()

mean = df['Close'].mean()

std = df['Close'].std()

df = df[(df['Close'] > mean - 3\*std) & (df['Close'] < mean + 3\*std)]

return df

def resample\_a\_semanal(df):

"""

Resamplea datos diarios a semanales.

"""

return df.resample('W').agg({'Open': 'first', 'High': 'max', 'Low': 'min', 'Close': 'last', 'Volume': 'sum'})

def calcular\_EMA(series, periodo):

"""

Calcula Exponential Moving Average.

"""

return series.ewm(span=periodo, adjust=False).mean()

def calcular\_MACD(close, rapido=12, lento=26, señal=9):

"""

Calcula MACD.

"""

ema\_rapido = calcular\_EMA(close, rapido)

ema\_lento = calcular\_EMA(close, lento)

macd\_line = ema\_rapido - ema\_lento

signal\_line = calcular\_EMA(macd\_line, señal)

return {'macd': macd\_line, 'signal': signal\_line}

def generar\_señal\_MACD(macd\_data):

"""

Genera señal para MACD: +1 cruce alcista, -1 bajista, 0 neutro.

"""

if len(macd\_data['macd']) < 2:

return 0

if macd\_data['macd'].iloc[-1] > macd\_data['signal'].iloc[-1] and macd\_data['macd'].iloc[-2] <= macd\_data['signal'].iloc[-2]:

return 1

elif macd\_data['macd'].iloc[-1] < macd\_data['signal'].iloc[-1] and macd\_data['macd'].iloc[-2] >= macd\_data['signal'].iloc[-2]:

return -1

return 0

def calcular\_RSI(close, periodo=14):

"""

Calcula RSI.

"""

delta = close.diff()

gain = (delta.where(delta > 0, 0)).rolling(window=periodo).mean()

loss = (-delta.where(delta < 0, 0)).rolling(window=periodo).mean()

rs = gain / loss

return 100 - (100 / (1 + rs))

def generar\_señal\_RSI(rsi):

"""

Genera señal para RSI: +1 sobreventa (<30), -1 sobrecompra (>70).

"""

if rsi.iloc[-1] < 30:

return 1

elif rsi.iloc[-1] > 70:

return -1

return 0

def calcular\_Bollinger(close, periodo=20, desv=2):

"""

Calcula Bollinger Bands.

"""

sma = close.rolling(window=periodo).mean()

std = close.rolling(window=periodo).std()

upper = sma + (std \* desv)

lower = sma - (std \* desv)

return {'upper': upper, 'lower': lower, 'middle': sma}

def generar\_señal\_Bollinger(bollinger, current\_price):

"""

Genera señal para Bollinger: +1 por debajo lower, -1 por arriba upper.

"""

if current\_price < bollinger['lower'].iloc[-1]:

return 1

elif current\_price > bollinger['upper'].iloc[-1]:

return -1

return 0

def calcular\_MA(close, sma\_periodo=50, ema\_periodo=20):

"""

Calcula SMA y EMA.

"""

return {'sma': close.rolling(window=sma\_periodo).mean(), 'ema': calcular\_EMA(close, ema\_periodo)}

def generar\_señal\_MA(ma\_data, current\_price):

"""

Genera señal para MA: +1 si EMA > SMA, -1 inverso.

"""

if ma\_data['ema'].iloc[-1] > ma\_data['sma'].iloc[-1]:

return 1

elif ma\_data['ema'].iloc[-1] < ma\_data['sma'].iloc[-1]:

return -1

return 0

def detectar\_patrones(close, ventana=30):

"""

Detecta patrones simples (placeholder: usa reglas básicas para HCH o triángulos).

Retorna 'alcista', 'bajista' o 'neutro'.

"""

# Implementación simplificada: si max en medio y mins a lados -> HCH bajista

recent = close.iloc[-ventana:]

mid = ventana // 2

if recent.iloc[mid] == recent.max() and recent.iloc[0] > recent.iloc[-1] and recent.iloc[-1] < recent.mean():

return 'bajista'

elif recent.iloc[mid] == recent.min() and recent.iloc[0] < recent.iloc[-1] and recent.iloc[-1] > recent.mean():

return 'alcista'

return 'neutro'

def generar\_señal\_Patrones(patrones):

"""

Genera señal para patrones.

"""

if patrones == 'alcista':

return 1

elif patrones == 'bajista':

return -1

return 0

def identificar\_tendencias(close, ventana=20):

"""

Identifica soportes y resistencias (min/max en ventana).

"""

recent = close.iloc[-ventana:]

return {'soporte': recent.min(), 'resistencia': recent.max()}

def generar\_señal\_Tendencias(tendencias, current\_price):

"""

Genera señal para tendencias: +1 por encima soporte, -1 por debajo resistencia (simplificado).

"""

if current\_price > tendencias['soporte']:

return 1

elif current\_price < tendencias['resistencia']:

return -1

return 0

def calcular\_mayoria(señales):

"""

Calcula mayoría: Compra si >=3, Venta si <=-3, Neutro otherwise.

"""

suma = sum(señales)

if suma >= 3:

return "Compra"

elif suma <= -3:

return "Venta"

return "Neutro"

def main():

"""

Función principal: Procesa todos los tickers y genera recomendaciones.

"""

datos\_historicos = cargar\_datos\_OHLCV(TICKERS)

resultados = {}

for ticker, datos in datos\_historicos.items():

if datos.empty:

continue

datos = limpiar\_datos(datos)

datos\_semanales = resample\_a\_semanal(datos)

# Indicadores corto plazo

macd\_corto = calcular\_MACD(datos['Close'], 12, 26, 9)

rsi\_corto = calcular\_RSI(datos['Close'], 14)

bollinger\_corto = calcular\_Bollinger(datos['Close'], 20, 2)

ma\_corto = calcular\_MA(datos['Close'], 50, 20)

patrones\_corto = detectar\_patrones(datos['Close'], 30)

tendencias\_corto = identificar\_tendencias(datos['Close'], 20)

señales\_corto = [

generar\_señal\_MACD(macd\_corto),

generar\_señal\_RSI(rsi\_corto),

generar\_señal\_Bollinger(bollinger\_corto, datos['Close'].iloc[-1]),

generar\_señal\_MA(ma\_corto, datos['Close'].iloc[-1]),

generar\_señal\_Patrones(patrones\_corto),

generar\_señal\_Tendencias(tendencias\_corto, datos['Close'].iloc[-1])

]

# Indicadores largo plazo

macd\_largo = calcular\_MACD(datos\_semanales['Close'], 26, 52, 18)

rsi\_largo = calcular\_RSI(datos\_semanales['Close'], 28)

bollinger\_largo = calcular\_Bollinger(datos\_semanales['Close'], 40, 2.5)

ma\_largo = calcular\_MA(datos\_semanales['Close'], 100, 50)

patrones\_largo = detectar\_patrones(datos\_semanales['Close'], 60)

tendencias\_largo = identificar\_tendencias(datos\_semanales['Close'], 40)

señales\_largo = [

generar\_señal\_MACD(macd\_largo),

generar\_señal\_RSI(rsi\_largo),

generar\_señal\_Bollinger(bollinger\_largo, datos\_semanales['Close'].iloc[-1]),

generar\_señal\_MA(ma\_largo, datos\_semanales['Close'].iloc[-1]),

generar\_señal\_Patrones(patrones\_largo),

generar\_señal\_Tendencias(tendencias\_largo, datos\_semanales['Close'].iloc[-1])

]

recomendacion\_dia = calcular\_mayoria(señales\_corto)

recomendacion\_semana = calcular\_mayoria(señales\_largo)

resultados[ticker] = {

"Recomendacion\_Dia": recomendacion\_dia,

"Recomendacion\_Semana": recomendacion\_semana

}

# Mostrar resultados

print(pd.DataFrame.from\_dict(resultados, orient='index'))

if \_\_name\_\_ == "\_\_main\_\_":

main()  
  
# Modelo de Forecasting Financiero

## Descripción

Implementación en Python de un modelo de forecasting basado en indicadores técnicos para recomendaciones de inversión.

## Instalación

pip install yfinance pandas numpy

## Uso

python forecasting\_model.py

## Notas

- Basado en el diseño del documento "15 de Sep\_JARR.pdf".

- No es consejo financiero; solo educativo.