Tutorial: Creación de un Bot de Trading Automatizado para Bitcoin

1. Configuración del Entorno

Primero, importamos las bibliotecas necesarias para trabajar con los datos y realizar análisis del mercado en tiempo real:

- *yfinance* para obtener los datos históricos de Bitcoin.
- *BeautifulSoup* y *requests* para extraer datos de páginas web.
- matplotlib y seaborn para la visualización.
- pandas y numpy para la manipulación de datos.

```
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', None)  # Mostrar todas las columnas
pd.set_option('display.expand_frame_repr', False)  # Evitar que el DataFr

import time
from datetime import datetime, timedelta

import requests
from bs4 import BeautifulSoup

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
plt.rc('figure', figsize = (5, 5))  # Establecer el tamaño predeterminado

from IPython.display import clear_output

import yfinance as yf
```

2. Carga y Preprocesamiento de Datos

2.1 Cargando los Datos

Primero, cargamos los datos históricos del precio de Bitcoin usando la API de yFinance. También se obtiene la información de precios en tiempo real desde CoinMarketCap para comparar los datos.

IMPORTANTE: Los selectores HTML pueden cambiar con las actualizaciones del sitio web. Si el scraping deja de funcionar, revisa la estructura actualizada del HTML.

```
def importar_base_bitcoin():
    global df_bitcoin, precio_actual, tendencia, media_bitcoin, algoritmo_decision, color

    end_date = datetime.now()
    start_date = end_date - timedelta(days=7)

    # Descargar datos de Bitcoin usando yfinance
    df_bitcoin = yf.download("BTC-USD", start=start_date, end=end_date, interval="5m")
```

2.2 Tratamiento de Datos

El siguiente paso es procesar los datos históricos de Bitcoin y eliminar registros con valores faltantes o duplicados, además de asegurarnos de que los datos sean consistentes y estén listos para el análisis. A continuación, se describen los pasos realizados:

- Definición de la función limpieza_datos():
  Se declara la función que realiza el preprocesamiento de los datos de Bitcoin.
- Uso de variable global:
  Se declara que df_bitcoin, df_bitcoin_limpio y media_bitcoin son variables globales que se utilizarán dentro de la función.
- Número inicial de datos:
  Se guarda el número inicial de registros antes de comenzar el proceso de limpieza.
- Copia del DataFrame original:
  Se crea una copia del DataFrame original para evitar modificarlo directamente durante el preprocesamiento.

- Eliminación de índices duplicados:
  Se eliminan los registros que tienen índices duplicados, manteniendo solo la primera aparición.
- Relleno de valores nulos:
  En la columna Close, se rellenan los valores nulos con el último valor no nulo disponible (método ffill).
- Algunos ejemplos de cómo iniciar el código:

```
def limpieza_datos():
    global df_bitcoin, df_bitcoin_limpio, media_bitcoin

    # Número inicial de datos
    num_datos_inicial = df_bitcoin.shape[0]
    print(f"Número inicial de datos: {num_datos_inicial}")

    # Copiar el DataFrame original para no modificarlo directamente
    df_bitcoin_limpio = df_bitcoin.copy()

    # Paso 1: Eliminar índices duplicados
    df_bitcoin_limpio = df_bitcoin_limpio[~df_bitcoin_limpio.index.duplicated(keep='first
    print(f"Duplicados eliminados: {num_datos_inicial - df_bitcoin_limpio.shape[0]}")

    # Paso 2: Rellenar valores nulos en la columna 'Close' con el último valor válido
    df_bitcoin_limpio['Close'] = df_bitcoin_limpio['Close'].ffill()
```

- Eliminación de filas con volumen cero:
  Se eliminan las filas donde el valor de la columna Volume es menor o igual a cero, lo cual indica que no hubo transacciones en ese momento.
- Cálculo del rango intercuartílico (IQR):
  Se calculan los cuartiles Q1 y Q3 para la columna Close, y se utiliza el rango intercuartílico para identificar posibles valores atípicos (outliers).
- Eliminación de outliers:
  Se eliminan los registros cuyo valor en la columna Close se encuentre fuera de los límites definidos por el IQR, es decir, aquellos valores por debajo del límite inferior o por encima del límite superior.
- Cálculo del precio promedio de Bitcoin después de la limpieza:
  Tras eliminar los outliers, se calcula el precio promedio de Bitcoin utilizando la columna Close.

- Número final de datos:
  Se guarda el número de registros después de completar el proceso de limpieza, y se calcula la cantidad de datos eliminados durante el proceso.

3. Cálculo de Indicadores Técnicos

3.1 Medias Móviles Simples (SMA)

Se calculan dos medias móviles simples (SMA) que servirán como indicadores clave para la toma de decisiones:

- **SMA corto (10 periodos)**: Representa la tendencia a corto plazo.
- **SMA largo (50 periodos)**: Representa la tendencia a largo plazo.

```
def calcular_sma():
    global df_bitcoin_limpio

    # Calcular la SMA de corto plazo (por ejemplo, 10 periodos)
    df_bitcoin_limpio['SMA_corto'] = df_bitcoin_limpio['Close'].rolling(window=10).mean()

    # Calcular la SMA de largo plazo (por ejemplo, 50 periodos)
    df_bitcoin_limpio['SMA_largo'] = df_bitcoin_limpio['Close'].rolling(window=50).mean()
```

4. Toma de Decisiones

El siguiente paso es implementar un algoritmo que utilice las medias móviles simples (SMA) calculadas previamente junto con la tendencia actual del mercado para decidir si comprar, vender o mantener. A continuación, se describen los pasos realizados:

- Definición de la función tomar_decisiones():
  Se declara la función que implementa el algoritmo de decisión del bot de trading.
- Uso de variables globales:
  Se declara que df_bitcoin, precio_actual, tendencia, media_bitcoin, algoritmo_decision y color son variables globales que se utilizarán dentro de la función para almacenar los resultados de las decisiones.
- Obtención de las medias móviles:
  Se obtienen los valores más recientes de la SMA corta y la SMA larga del DataFrame df_bitcoin_limpio, calculados en pasos anteriores. Estos valores son clave para determinar las decisiones de trading.

- Condición para Comprar:
  Si la SMA corta (que representa la tendencia a corto plazo) es mayor que la SMA larga (que representa la tendencia a largo plazo), y la tendencia del mercado es alcista (tendencia == 'alta'), entonces el algoritmo decide comprar.
- Algunos ejemplos de cómo iniciar el código:

```
def tomar_decisiones():
    global df_bitcoin, precio_actual, tendencia, media_bitcoin, algoritmo_decision, color

    # Obtener las últimas SMA calculadas en el DataFrame df_bitcoin_limpio
    sma_corto_actual = df_bitcoin_limpio['SMA_corto'].iloc[-1]
    sma_largo_actual = df_bitcoin_limpio['SMA_largo'].iloc[-1]

    # Algoritmo de decisión basado en SMA y tendencia actuales
    if (sma_corto_actual > sma_largo_actual) and (tendencia == 'alta'):
```

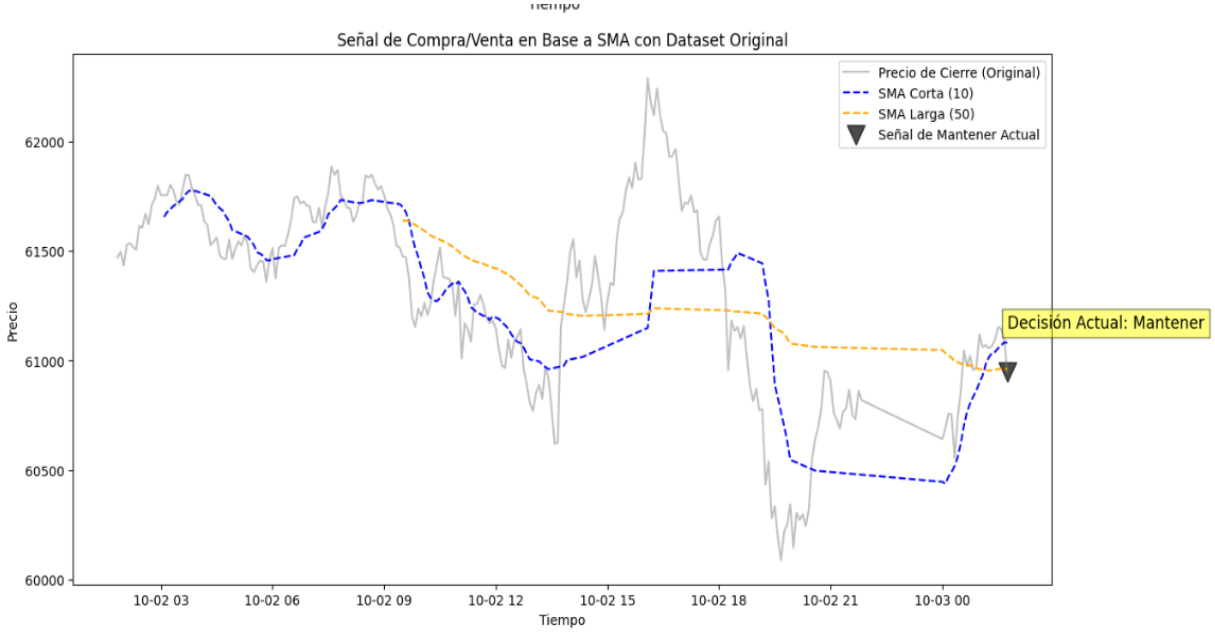
- Condición para Vender:
  Si la SMA corta es menor que la SMA larga y la tendencia del mercado es bajista (tendencia == 'baja'), entonces el algoritmo decide vender.
  El color de la señal se establece en rojo (#dc143c) para indicar una señal de venta.
- Condición para Mantener:
  Si no se cumple ninguna de las condiciones anteriores, es decir, no hay una señal clara, el algoritmo decide mantener la posición.
  El color de la señal se establece en negro (#000000) para indicar una señal de "mantener".
- Impresión del resultado:
  El resultado de la decisión se imprime para cada ciclo de toma de decisiones, mostrando si la recomendación es comprar, vender o mantener.

Recuerden que la tendencia de una hora puede cambiar rápidamente. Siempre revisen las señales de color en el gráfico antes de tomar decisiones.

5. Visualización

Se pueden generar gráficos para visualizar las decisiones tomadas por el bot, junto con la evolución del precio de Bitcoin y las medias móviles:

- Gráficos de precios con medias móviles superpuestas.
- Con la Señales de compra o venta destacada actual.



6. Ciclo Automatizado

El bot está diseñado para ejecutarse de forma continua, tomando decisiones cada 5 minutos. Cada ciclo implica:

- Descargar los datos más recientes.
- Limpiar los datos para asegurar la precisión.
- Calcular las medias móviles.
- Evaluar las condiciones de compra o venta según las reglas establecidas.

```
while True:
    print("Iniciando ciclo de análisis...")

    # Paso 1: Descargar los datos y extraer el precio actual
    importar_base_bitcoin()
    extraer_tendencias()

    # Paso 2: Limpiar los datos y calcular las medias móviles
    limpieza_datos()
    calcular_sma()

    # Paso 3: Tomar la decisión de compra/venta
    tomar_decisiones()

    # Paso 4: Graficar las tendencias y señales
    graficar_senal_actual()

    # Esperar 5 minutos antes de la siguiente decisión
    print("Esperando 5 minutos antes de la siguiente decisión...")
    time.sleep(300)
```