



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Desarrollo de un Bot de
Trading**



Presentado por Sergio Rebollo Ortega
en Universidad de Burgos — 10 de junio
de 2024

Tutor: Jose Manuel Galán Ordax



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. José Manuel Galán Ordax y Dña. Virginia Ahedo García, profesores del departamento de Ingeniería Civil, área de Organización de Empresas.

Expone:

Que el alumno D. Sergio Rebollo Ortega, con DNI 71362051C, ha realizado el Trabajo final de Grado en Ingeniería Informática Desarrollo de un Bot de Trading.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 10 de junio de 2024

Vº. Bº. del Tutor:

D. nombre tutor

Vº. Bº. del co-tutor:

D. nombre co-tutor

Resumen

El presente Trabajo de Fin de Grado (TFG) desarrolla un bot de trading destinado a la automatización de la gestión de carteras de inversión, específicamente en los mercados de divisas y materias primas. El bot diseñado se apoya en estrategias de análisis técnico chartista, incluyendo el uso de indicadores como el RSI (Relative Strength Index) y herramientas analíticas como las líneas de tendencia, para realizar operaciones de compra y venta de activos financieros de manera autónoma.

El proyecto se inicia con una revisión bibliográfica exhaustiva para establecer las bases teóricas del trading y el uso de herramientas analíticas específicas. Posteriormente, se procede a la programación del bot en Python, utilizando librerías especializadas en análisis de datos y algoritmos de machine learning para identificar patrones de mercado y tomar decisiones de inversión.

El proceso de desarrollo ha incluido pruebas rigurosas para verificar la precisión y la eficacia del bot bajo diversas condiciones de mercado. Los resultados indican que el bot puede generar operaciones rentables dependiendo de la temporalidad, con un nivel adecuado de riesgo, demostrando su potencial como herramienta de apoyo para inversores y traders en el ámbito financiero.

Descriptores

Bot de trading, trading automatizado, mercado de divisas, análisis chartista, indicadores RSI, líneas de tendencia, hilos, python...

Abstract

This Graduation Project develops a trading bot aimed at automating the management of investment portfolios, specifically in the forex and commodities markets. The designed bot relies on chartist technical analysis strategies, including the use of indicators such as the RSI (Relative Strength Index) and analytical tools like trend lines, to autonomously perform buy and sell operations of financial assets.

The project begins with an exhaustive bibliographic review to establish the theoretical foundations of trading and the use of specific analytical tools. Subsequently, the bot is programmed in Python, using libraries specialized in data analysis and machine learning algorithms to identify market patterns and make investment decisions.

The development process included rigorous testing to verify the accuracy and effectiveness of the bot under various market conditions. The results indicate that the bot can generate profitable operations depending on the timeframe, with an appropriate level of risk, demonstrating its potential as a support tool for investors and traders in the financial field.

Keywords

Trading bot ,automated trading, forex, chartist analysis, RSI indicators, trend lines, threads, python.

Índice general

Índice general	iii
Índice de figuras	v
Índice de tablas	vi
1. Introducción	1
2. Objetivos del proyecto	5
2.1. Introducción	5
2.2. Objetivos Generales	5
2.3. Objetivos Técnicos	5
2.4. Objetivos Personales	6
3. Conceptos teóricos	7
3.1. ¿Qué son los mercados financieros?	7
3.2. Tipos de mercados financieros	8
3.3. Invertir	12
3.4. Trading	13
3.5. Análisis fundamental	14
3.6. Análisis técnico	15
3.7. Estrategia del bot	16
3.8. Aplicación de la Estrategia	19
3.9. Programación Threading	21
3.10. Programación orientada a objetos (POO)	23
3.11. Estructura utilizadas	24

4. Técnicas y herramientas	27
5. Aspectos relevantes del desarrollo del proyecto	39
5.1. Programación	39
5.2. Análisis de resultados	67
6. Trabajos relacionados	75
7. Conclusiones y Líneas de trabajo futuras	81
Bibliografía	83

Índice de figuras

3.1. Negociación divisas	10
5.1. EURUSD_DAILY	68
5.2. GBPUSD_H4 - 1	69
5.3. GBPUSD_H4 - 2	70
5.4. USDJPY_H1-1	71
5.5. USDJPY_H1-2	72

Índice de tablas

3.1. Monedas con mayor negociación en el mercado FOREX	9
3.2. Criptomonedas con mayor reconocimiento	12

1. Introducción

El trading financiero se ha convertido en una actividad cada vez más popular en los últimos años, gracias a la aparición de las nuevas tecnologías y la facilidad de acceso a los mercados financieros. Los brokers se han expandido muy rápido debido a la facilidad de crear publicidad en las redes sociales, por eso hay que tener mucho cuidado con aquellos que no estén regulados correctamente. En este contexto, el desarrollo de un bot de trading financiero se ha convertido en una herramienta de gran utilidad para los inversores y las grandes instituciones como pueden ser los bancos, ya que permite automatizar procesos y tomar decisiones de inversión de una forma automática, rápida, y eficiente.

El objetivo de este Trabajo de Fin de Grado (TFG) es el diseño y desarrollo de un bot de trading para la gestión automatizada de carteras de inversión destinado sobre todo a la negociación de divisas y materias primas. Este bot permitirá la compra y venta de activos financieros con la utilización de estrategias de análisis chartista junto con la utilización de otras herramientas como el indicador RSI o las líneas de tendencia. Para ello el bot recogerá y analizara datos del mercado, identificara patrones y realizará operaciones de compra o venta según lo considere de forma automática. Utilizaremos diferentes herramientas y lenguajes de programación, así como diversas técnicas de análisis de datos y algoritmos.

En primer lugar, se llevará a cabo una revisión bibliográfica de estudios relacionados, de conceptos básicos del trading y de herramientas utilizadas en el análisis técnico y fundamental de los mercados financieros. Posteriormente, se definirán las estrategias de inversión a desarrollar en el bot, teniendo en cuenta la volatilidad de los mercados, riesgo asumido y los objetivos de rentabilidad esperados. Una vez definidas las estrategias, se procederá al desarrollo del bot de trading en Python.

Primero desarrollaremos un bot de trading para probar la estrategia programada a través de datos históricos obtenidos en formato csv. Una vez programado realizaremos pruebas con diferentes activos financieros para comprobar la rentabilidad de nuestro bot de trading, realizando estadísticas y observando en qué parámetros podremos mejorar para poder ser rentables o mejorar la rentabilidad. Tras este paso pasaremos a realizar a programar el bot para que realice las operaciones cuando el mercado financiero esté abierto.

El último paso será explicar las librerías, herramientas y softwares utilizados para realizar los diferentes pasos que requiere la programación de un bot de trading. También realizaremos una explicación del código que hemos programado.

Estructura de la memoria

La estructura que se ha seguido para documentar la memoria es la siguiente:

1. **Introducción** Se explicará el desarrollo que seguirá el proyecto, seguido de un desglose de la estructura de la memoria, explicando los puntos de los que va a constar y los enlaces correspondientes al repositorio donde se aloja el proyecto en Github.
2. **Objetivos del proyecto** Explicación de los objetivos que se quieren conseguir durante el desarrollo del proyecto.
3. **Conceptos teóricos** Contextualización de la parte teórica en la que se basa el proyecto para lograr entender su desarrollo, junto con la explicación de conceptos clave.
4. **Técnicas y herramientas** Explicación de las herramientas y técnicas utilizadas durante el desarrollo del proyecto.
5. **Aspectos relevantes del desarrollo** Descripción de los eventos sucedidos durante el desarrollo del proyecto, explicación del código programado y contratiempos o problemas que se han experimentado.
6. **Trabajos relacionados** Estudio de trabajos o proyectos similares que pueden servir como referencia.
7. **Conclusiones y líneas de trabajo futuras** Conclusiones a las que se ha llegado durante el desarrollo del proyecto y mejoras que se pueden realizar en futuras versiones.

También se explicará la estructura seguida en los anexos del proyecto.

1. **Planificación del proyecto** Se realizará un estudio del desarrollo que se ha seguido durante el proyecto junto con la viabilidad tanto económica como legal del proyecto.
2. **Especificación de requisitos** Explicación de los requisitos marcados según los objetivos que se han establecido.
3. **Especificación de diseño** Estrutura de los datos y librerías que se han utilizado en el proyecto.
4. **Manual del programador** Manual de como utilizar o manipular el proyecto.
5. **Manual de usuario** Manual para la ejecución del proyecto.

Repositorio Github

[Repositorio GitHub del Bot de Trading](#)

2. Objetivos del proyecto

2.1. Introducción

En este apartado se expondrán los diferentes objetivos que se han establecido para el desarrollo del Bot de Trading. Primero, se explicarán los objetivos generales del proyecto; a continuación, se detallarán los objetivos técnicos; y por último, se presentarán los objetivos personales.

2.2. Objetivos Generales

En este apartado se describen los objetivos generales del proyecto, es decir, aquellos que se han establecido al inicio del proyecto como implementaciones para crear el Bot de trading ya que es un proyecto realizado desde el principio y no una continuación.

- Capacidad del bot de trading para operar mediante la estrategia programada.
- Crear una interfaz para el funcionamiento del bot.
- Documentar las operaciones realizadas.

2.3. Objetivos Técnicos

Objetivos relacionados con las herramientas y las metodologías.

- Desarrollar el proyecto conforme a la metodología ágil, específicamente utilizando Scrum para la gestión de sprints y la iteración continua.s.

- Emplear GitHub para el control de versiones.
- Emplear Zenhub para la gestión de proyectos.
- Conocer y documentar la programación con hilos.
- Utilizar Pycharm como entorno de desarrollo integrado (IDE) para gestionar el repositorio local.
- Programar el bot utilizando Python como lenguaje de programación principal. Las librerías de Python que se utilizarán incluyen: pandas, NumPy, Pandas, TA-Lib, Tkinter.

2.4. Objetivos Personales

Objetivos relacionados en el desarrollo de mis habilidades y conocimientos.

- Profundizar mis conocimientos en análisis técnico y estrategias de trading, entendiendo cómo se aplican en un entorno automatizado.
- Fortalecer mis competencias en la programación con Python, aprendiendo a utilizar nuevas librerías y frameworks que sean útiles en el ámbito del trading automatizado.
- Adquirir experiencia práctica en la gestión de proyectos utilizando metodologías ágil y herramientas como Zenhub y GitHub.
- Desarrollar la capacidad de crear documentación técnica clara y precisa, que sea útil tanto para usuarios como para otros desarrolladores.
- Mejorar mis habilidades en el desarrollo y manejo de hilos en Python, permitiéndome gestionar múltiples tareas concurrentes de manera eficiente dentro del bot de trading.

3. Conceptos teóricos

3.1. ¿Qué son los mercados financieros?

Empezaremos poniendo en contexto y explicando diferentes conceptos y estrategias para poder entender cómo se ha realizado el bot de trading. En primer lugar, comenzaremos con la explicación de qué son los mercados financieros y sus tipos.

Los mercados financieros son el lugar donde se realiza la compra y la venta de los diferentes activos financieros que existen. Puede ser físico o encontrarse en internet; en la actualidad, la mayoría de los mercados se encuentran en esta última. Si en el mercado financiero hay una oferta y una demanda con un precio establecido, cantidad y fecha de liquidación, se forma una operación. Estos mercados funcionan gracias a los intermediarios que ayudan a satisfacer las diferentes necesidades de los compradores y vendedores. [34]

La utilidad de los mercados financieros es la obtención de ahorros provenientes de diferentes fuentes y, además, el financiamiento, que nos ayuda a la canalización de capitales a largo plazo, cambiar deudas por el capital propio y proporciona un puente entre el mercado de dinero y los fondos de capital. Este también tiene una repercusión en el ámbito económico y social, permitiendo determinar los precios, obtener previsiones para el crecimiento económico y observar la situación en la que se encuentran las empresas en los diferentes países. En el ámbito social, permite el ahorro, la ampliación del conocimiento del mercado financiero y una transparencia a la hora de realizar el tráfico jurídico mercantil.ite [43]

En el mercado financiero hay tres roles: los compradores, vendedores e intermediarios:

- **Compradores:** Se dedican a comprar acciones, bonos, letras y otros instrumentos a los vendedores por un precio establecido.
- **Vendedores:** Se dedican a vender acciones, bonos, letras y otros instrumentos a los compradores por un precio establecido.
- **Intermediarios:** Se encargan de ayudar a realizar la transferencia de los activos, manejar el riesgo de las operaciones, administrar el pago de las operaciones y proporcionar herramientas para acceder a los mercados financieros.

Hay diferentes tipos de intermediarios, como pueden ser bancos comerciales, agentes de bolsa expertos en la inversión, asociaciones de crédito, fondos de pensiones o empresas de capital de riesgo.

También hay que explicar el precio por el que se transfieren estos activos financieros o tasa de interés. Esta tasa permite controlar el precio de la venta y la compra, y se determina en el mercado. [34]

3.2. Tipos de mercados financieros

Existen muchos tipos de mercados financieros que se pueden clasificar según su funcionamiento, el objetivo de inversión, la regulación del mercado, la negociación de los activos financieros y el tipo de activos financieros que se desean comprar. Este último criterio es el que explicaremos con más detalle, ya que es el más relevante.

Mercado de divisas

Este mercado, también conocido como FOREX, es el que mayor volumen de capital mueve diariamente y utiliza como activo las divisas, que son el medio de pago de los países. Por ejemplo, cuando viajas a otros países y realizas un intercambio entre la moneda utilizada en tu país y la moneda local, estás realizando una operación de Forex.

El mercado de divisas es aquel donde se compran y venden diferentes monedas cuyo objetivo original era ayudar al flujo monetario generado por el comercio a nivel internacional. A diferencia de la bolsa de valores, es un mercado descentralizado.

El FOREX opera las 24 horas del día y cinco días a la semana, cerrando solo los sábados y los domingos. Las horas más importantes dentro de este

mercado, o donde hay más volatilidad, son a las 9:00 de la mañana cuando abre el mercado europeo y a las 16:00 cuando abre el mercado americano.

El FOREX se caracteriza por ser el mercado que más transacciones realiza al día, por tener una liquidez extrema, ser muy difícil de manipular y muy volátil, lo que implica un mayor riesgo al realizar inversiones. [8]

Posición	Moneda	Código ISO 4217	Símbolo
1	Dólar estadounidense	USD	\$
2	Euro	EUR	€
3	Yen japonés	JEN	¥
4	Libra esterlina	GBP	£
5	Franco suizo	CHF	-
6	Dólar australiano	AUD	\$

Tabla 3.1: Monedas con mayor negociación en el mercado FOREX

En la Figura 3.1 se observa una tabla con las principales divisas mundiales.

El código ISO 4217 es un código de tres letras que nos permite identificar las divisas de diferentes países.

Las monedas más negociadas representan los siguientes porcentajes:

- **USD:** tiene una negociación del 60 %
- **EUR:** representa un 20 % del total del FOREX
- **JPY:** tiene una participación igual o inferior al 10 %
- **GBP:** tiene una participación del 5 %
- **CHF:** representa un porcentaje inferior al 5 %
- **AUD:** al igual que el CHF, su porcentaje es inferior al 5 %

En la Figura 3.1 se observa un gráfico con la participación dentro del mundo de las divisas.

Este mercado es completamente diferente a los demás, ya que en él participan una gran variedad de inversores, desde bancos centrales como el BCE (Banco Central Europeo) y la FED (La Reserva Federal de EE.UU.)

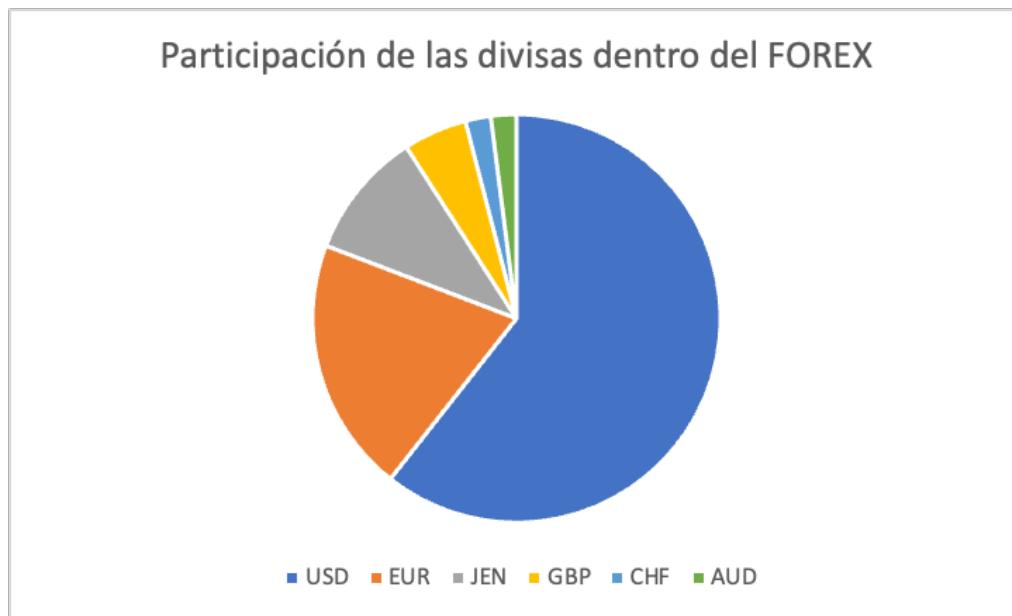


Figura 3.1: Negociación divisas

hasta inversores particulares. El activo más negociado de este mercado es el dólar estadounidense, seguido del euro.

Algunos de los activos más negociados son, por ejemplo:

1. EUR/USD – Euro/Dólar
2. GBP/USD – Libra/Dólar
3. EUR/GBP – Euro/Libra
4. USD/JPY – Dólar/Yen

Mercados de acciones

Este mercado es el más conocido mundialmente. El activo que se negocia son las acciones, las cuales representan una pequeña parte de una empresa. Cuando se adquieren, convierten al poseedor de la acción en accionista de la empresa y propietario de una fracción de la compañía a la que pertenece la acción.

Las acciones son producto de un movimiento por parte de la empresa para conseguir financiación saliendo a bolsa. Es decir, cuando una compañía

sale a bolsa y compramos una acción de esta compañía, estamos prestando nuestro dinero para que la empresa pueda realizar mejoras e innovaciones para obtener más beneficios.

Una vez que se adquieren las acciones, existen unos derechos y obligaciones que se deben cumplir. Por ejemplo, los accionistas tienen derecho a asistir a las juntas de accionistas de la compañía, derecho a los dividendos y derecho a obtener información de la compañía. [28]

Para concluir con este mercado, expondremos algunos ejemplos de acciones muy conocidas. La compañía Google tiene una capitalización bursátil de 1,67 billones de dólares y las acciones tienen un valor de 132,52 dólares en el mercado, por lo que Google tiene 12.601.871 acciones en circulación en el mercado bursátil. [39]

Mercado de bonos

Los bonos son emitidos por las empresas, el estado o algún organismo público a largo plazo. Son de renta fija, lo que significa que su rentabilidad se conoce antes de la fecha en la que vence el bono. [40]

Mercado de commodities

Las commodities son bienes producidos por el ser humano o disponibles en la naturaleza, por lo que podemos decir que es el mercado de materias primas. Por ejemplo, algunas commodities son el maíz, la soja, el petróleo, el oro y la plata.

Las commodities se dividen en diferentes grupos: los granos (la soja, el trigo, el maíz), las softs (el algodón, el cacao, el café), las energías (petróleo, gas natural, etanol), los metales (el oro, la plata, el cobre) y las carnes y sus derivados. [41]

Mercado de criptomonedas

Es un mercado en auge, especialmente con el aumento del valor del bitcoin a lo largo de la década del 2010-2020. Se trata de un mercado donde se negocia con criptomonedas, que son activos digitales que utilizan cifrados criptográficos para realizar sus transacciones. No existen en formato físico, sino que están dentro de una cartera digital. Estas criptomonedas tienen un valor muy volátil ya que los inversionistas especulan mucho con su precio.

La principal diferencia con el mercado de divisas es que es totalmente descentralizado. Por ejemplo, el dólar está regulado por la FED o el euro

por el Banco Central Europeo, mientras que el bitcoin no está regulado por ninguna institución financiera.

Las criptomonedas utilizan la tecnología blockchain para realizar las transacciones. Además, a diferencia de las divisas tradicionales, su precio dependerá de las perspectivas del proyecto de la empresa para la criptomoneda, en lugar de las decisiones tomadas por los gobiernos para hacer fluctuar las divisas. [42]

Algunos ejemplos de criptomonedas:

Criptomoneda	Símbolo
Bitcoin	BTC
Ethereum	ETH
Cardano	ADA
Tether	USDT
Ripple	XRP
Dogecoin	DOGE
Binance Coin	BNB
USDCoin	USDC
Polkadot	DOT
Solana	SOL

Tabla 3.2: Criptomonedas con mayor reconocimiento

En la Figura 3.2 se refleja un gráfico con las principales criptomonedas.

3.3. Invertir

Una vez que hemos introducido y explicado lo que son los mercados financieros y qué tipos existen, realizaré una breve explicación de las inversiones antes de comenzar a explicar el trading y el trading algorítmico.

A la hora de invertir en los diferentes mercados financieros que hemos explicado, hay dos formas de invertir:

- **Entrar en largo:** este término significa comprar un activo esperando que el activo financiero que hemos comprado aumente de valor y nos dé la rentabilidad que esperábamos.
- **Entrar en corto:** se refiere a vender un activo que pensamos que va a perder valor en el plazo que hayamos estimado, ya sea a corto plazo o a largo plazo.

3.4. Trading

¿Qué es el trading?

Este concepto es muy importante definirlo, ya que este trabajo se basará en realizar una técnica de trading algorítmica para poder operar en los mercados financieros en tiempo real.

El trading es un tipo de inversión a través del cual se compran o venden activos financieros con el objetivo de obtener una rentabilidad según las pretensiones del trader mediante bonos, acciones, ETF, divisas y otros activos financieros.

Este tipo de inversión implica riesgos y es necesario tener conocimientos y destrezas bien definidas para poder obtener una rentabilidad en los mercados financieros y no sufrir pérdidas significativas que afecten nuestro capital de inversión. [35]

A la hora de realizar trading, los inversores que practican este tipo de inversión suelen tener una estrategia muy específica que rara vez se saltan para realizar una compra de un activo en el mercado financiero.

En el momento que hemos adquirido o validado los conocimientos sobre el trading y los mercados financieros, podremos desarrollar nuestra propia estrategia de trading para obtener rentabilidad. Esta estrategia se puede aplicar a numerosos activos financieros como acciones, divisas, ETF o futuros. Cada producto en el que queramos operar tendrá diferentes riesgos y características.

Para realizar trading, es aconsejable arriesgar del 1% al 5% de nuestra cuenta global para que las pérdidas no sean suficientemente grandes como para cerrar nuestra cuenta o impedirnos seguir operando. [44]

Tipos de Trading

A la hora de realizar trading, se pueden utilizar diferentes enfoques. Nosotros operaremos en diferentes temporalidades para comparar qué tipo de trading se adapta mejor a nuestro bot programado:

- **Trading a largo plazo:** Estrategia que realiza operaciones a largo plazo, por lo que el análisis de la estrategia se realiza en temporalidades muy altas. Esta estrategia permite obtener rentabilidades altas sin tener que estar pendiente de las operaciones, ya que se trata de aprovechar

los movimientos tendenciales de un activo durante varios años, un año o varios meses. Esta técnica utiliza más el análisis fundamental del activo que el análisis técnico.

- **Swing trading:** Estrategia de corto plazo que se enfoca en obtener rentabilidad aprovechando los impulsos del mercado en periodos de unos días o semanas. En esta técnica, los traders se basan más en el análisis técnico que en el análisis fundamental a la hora de realizar las operaciones, apoyándose en herramientas como el RSI.
- **Trading intradía:** Este tipo de trading se realiza durante el día, es decir, se compran y venden activos en el mercado financiero con una duración de un día, buscando obtener ganancias rápidas en un periodo muy corto. Se necesita tener conocimientos y habilidades muy avanzadas para realizar correctamente este tipo de operaciones.
- **Scalping:** Estrategia que consiste en comprar o vender activos financieros constantemente en un periodo de tiempo muy corto, es decir, las compras y ventas de los activos se realizan en cuestión de minutos, buscando impulsos rápidos del mercado. Es necesario un alto grado de autocontrol y disciplina para ejecutar correctamente este tipo de trading, ya que si se realiza incorrectamente puede llevar a pérdidas rápidas y significativas de capital.

Existen más técnicas de trading, pero estas son las más importantes. También existe el trading automático, que se basa en aplicar alguna de estas estrategias de manera automática sin necesidad de estar pendiente de la pantalla.

Otra modalidad es el copy trading, que se basa en copiar la estrategia de inversores que han demostrado rentabilidad a lo largo de un periodo aplicando su estrategia. [24]

El trading algorítmico se trata de utilizar patrones matemáticos y estadísticos para identificar oportunidades de compra y venta en los diferentes mercados financieros.

3.5. Análisis fundamental

El análisis fundamental es un tipo de análisis bursátil que utiliza el análisis de datos del balance de una economía o de una empresa, los resultados publicados y los datos de la bolsa. Es decir, se trata de analizar una empresa,

una divisa u otro activo financiero mediante el resultado de noticias y el análisis de datos de los activos para comparar los resultados extraídos con los valores reales en bolsa. Por ejemplo, si realizamos el análisis de la empresa Google y obtenemos un valor por debajo de los valores que se muestran en la bolsa, significa que la empresa en realidad no vale lo que se está cotizando. [11]

Hay dos tipos de análisis fundamental:

- **Análisis Top-Down:** Este tipo de análisis fundamental se basa en ir estrechando el análisis cada vez más, es decir, empezar analizando la economía global para avanzar hasta el análisis de los datos de la empresa que queremos investigar para obtener rentabilidad. De esta manera, se identifican los sectores más prometedores de la economía.
- **Análisis Bottom-Up:** Este análisis se enfoca únicamente en los datos de la empresa. Se refiere a los niveles de endeudamiento de la empresa, la capitalización de la empresa o sus inversiones.

3.6. Análisis técnico

El análisis técnico es la disciplina que estudia los impulsos de los mercados financieros a través de gráficos para analizar y pronosticar futuros movimientos de los diferentes activos financieros. Este análisis abarca el precio del activo en cada momento y el volumen que se está negociando por parte de los inversores.

Para entender este concepto, es fundamental comprender que los precios de los activos se mueven a través de tendencias. Por eso es muy importante la representación gráfica del precio de los activos, ya que nos proporcionará la información suficiente para saber y comprender hacia dónde se está moviendo el mercado.

Además de graficar los precios, es crucial el uso de diferentes herramientas dentro de este análisis para pronosticar los futuros movimientos de los mercados financieros. Por ejemplo, las líneas de tendencia se utilizan para conocer la dirección del precio, dibujando una línea que une los últimos máximos si es una tendencia bajista o uniendo los últimos mínimos si es un movimiento alcista. En la actualidad, hay herramientas más modernas que automatizan estos procesos, como las SMA (Simple Moving Averages) que son líneas de tendencia de diferentes períodos que se pueden ajustar para

obtener la dirección del mercado sin tener que dibujar líneas manualmente, facilitando así el análisis. [32]

3.7. Estrategia del bot

Nuestro bot de trading funcionará a través de un análisis técnico que programaremos y explicaremos más adelante, incluyendo los métodos y variables utilizados para desarrollar la estrategia.

Esta estrategia se basará en la utilización de las herramientas RSI, SMA y el retroceso de Fibonacci para buscar oportunidades de compra o venta en los activos financieros con los que operemos en el mercado.

A continuación, explicaremos cada una de estas herramientas antes de describir el funcionamiento de la estrategia.

RSI

El Índice de Fuerza Relativa (RSI) es una herramienta muy popular y efectiva entre los inversores. Creado por J. Welles Wilder en la década de 1970, el RSI se ha convertido en un índice fundamental para comprobar la fuerza y la tendencia de un movimiento en los mercados financieros, así como para identificar situaciones de sobreventa o sobrecompra. [29]

Es un indicador oscilante que calcula la fuerza de los cambios recientes en los precios en correlación con los beneficios y pérdidas promedios. El RSI se mide con la siguiente fórmula:

$$RSI = 100 - \frac{100}{1 + RS} \quad (3.1)$$

El valor RS se calcula dividiendo el promedio de los beneficios a lo largo de un período específico entre el promedio de las pérdidas durante el mismo período.

$$RS = \frac{\text{Promedio de Beneficios}}{\text{Promedio de Pérdidas}} \quad (3.2)$$

Para interpretar el RSI en el análisis técnico, hay que entender que esta herramienta oscila entre los valores 0 y 100:

- Si el valor está por encima de 70, se considera que hay sobrecompra, lo que indica que el activo está por encima del valor esperado y que hay probabilidades altas de que el precio baje.
- Si el valor está por debajo de 30, se considera que hay sobreventa, lo que indica que el activo está por debajo del valor esperado y que hay probabilidades altas de que el precio suba.

El RSI se calcula normalmente en un período de 14 sesiones, pero este período puede ajustarse según las necesidades o preferencias de los inversores y las condiciones del mercado. Un período bajo implica más sensibilidad de la herramienta, generando señales más frecuentes, mientras que un período alto suaviza los valores del indicador.

Las divergencias son señales importantes que se producen cuando el precio del activo y los valores del RSI no concuerdan. Una divergencia alcista se produce cuando los mínimos del precio de un activo son más bajos pero los valores del RSI son más altos, y una divergencia bajista se produce al revés, cuando los máximos del precio son más altos pero los valores del RSI son más bajos.

A pesar de ser una herramienta útil, el RSI tiene sus limitaciones. En mercados con fuertes tendencias, el RSI puede mantenerse indicando sobrecompra o sobreventa por períodos largos de tiempo. Además, es fundamental usar el RSI junto con otros análisis e indicadores para tener una comprensión más completa de la situación del mercado. [32]

SMA

La media móvil simple (SMA) es una herramienta fundamental para comprender y anticipar las tendencias en los mercados financieros. Aunque aparentemente sencilla, la SMA ofrece a los inversionistas y operadores un claro panorama sobre la dirección y fortaleza de una tendencia, proporcionando así un valioso marco para la toma de decisiones informadas.

La SMA es un indicador que suaviza las fluctuaciones de precios al calcular el promedio de un conjunto específico de datos durante un período determinado. La fórmula básica para calcular la SMA consiste en sumar los precios de cierre durante cierto número de períodos y dividir esa suma entre la cantidad de períodos. [30]

$$SMA = \frac{\text{Suma de precios de cierre de } N \text{ sesiones}}{N} \quad (3.3)$$

La SMA es principalmente una herramienta para detectar tendencias. Cuando el precio de un activo está por encima de la línea móvil, se considera que la tendencia es alcista. Si la línea móvil está por encima del precio del activo, se considera que la tendencia es bajista. Además, ofrece señales de compra o venta; si el precio cruza la línea móvil de arriba hacia abajo, es una posible señal de venta, y si el precio cruza la línea móvil de abajo hacia arriba, es una señal de compra.

Esta herramienta se puede adaptar a todo tipo de estrategias. Se ajusta a períodos cortos o largos simplemente modificando los períodos de la línea de tendencia. Una media móvil de pocas sesiones es más sensible a la dirección del precio, mientras que una media móvil de más sesiones suaviza la dirección del precio.

A pesar de ser muy poderosa, esta herramienta tiene limitaciones. Puede reaccionar tarde o demasiado lentamente a los cambios bruscos del precio. Lo ideal es combinarla con otros indicadores para ser más precisos a la hora de ejecutar las operaciones y no llegar tarde. [32]

Fibonacci

El retroceso de Fibonacci es una herramienta poderosa en el mundo de las inversiones. Se basa en los principios matemáticos de la secuencia de Fibonacci.

La secuencia de Fibonacci es una serie matemática que comienza con 0 y 1, donde cada número siguiente es la suma de los dos números anteriores en la secuencia. Por ejemplo, la serie de números es 0, 1, 1, 2, 3, 5, 8, 13, etc.

Esto establece la base para el retroceso de Fibonacci. Los niveles clave para operar que se derivan de la secuencia numérica de Fibonacci son los porcentajes de 23.6 %, 38.2 %, 50 %, 61.8 % y 78.6 %.

Los niveles de retroceso de Fibonacci se utilizan para identificar posibles niveles de soporte y resistencia en el precio de un activo durante una tendencia. Para obtener los niveles de Fibonacci, hay que identificar el punto inicial del movimiento; si es alcista, se toma el último mínimo importante, y si es bajista, el último máximo importante. Luego, se identifica el punto final, que es el último máximo importante si es una tendencia alcista y el último mínimo importante si es una tendencia bajista.

Una vez que se han identificado el punto inicial y el final, hay que interpretar los niveles que se crean. Un retroceso del 61.8 % a menudo se considera significativo, ya que sugiere que la tendencia principal está

perdiendo fuerza. Por otro lado, un retroceso del 38.2 % indica una corrección más leve. Estos niveles se utilizan como principales puntos de entrada y salida en una operación. [31]

Además del retroceso de Fibonacci, existen las extensiones de Fibonacci, que también son muy utilizadas por los analistas técnicos. Estas se calculan proyectando los niveles más allá del punto inicial y pueden ayudar a identificar posibles objetivos de precios en una tendencia en desarrollo.

Esta herramienta no es infalible, ya que el precio en movimientos tendenciales no siempre realiza el retroceso dentro de los porcentajes que establece. Como las demás herramientas, esta también sería mucho más eficaz si se combina con otros indicadores. [32]

3.8. Aplicación de la Estrategia

La aplicación de la estrategia de trading se basará en la combinación de tres indicadores. Esta combinación puede utilizarse para desarrollar una estrategia de trading robusta.

Media Móvil de 100 Periodos

La media móvil de 100 periodos ayuda a los traders a identificar la tendencia general del mercado. Una media móvil ascendente indica una tendencia alcista, mientras que una descendente señala una tendencia bajista. Esta información es crucial para establecer el contexto en el que se aplicarán los otros dos indicadores.

Niveles de Fibonacci

Los niveles de retroceso de Fibonacci, especialmente el 38.2 % y el 61.8 %, son esenciales en esta estrategia. Estos niveles son considerados puntos de reversión potenciales en el mercado, proporcionando objetivos para entrar en operaciones.

Índice de Fuerza Relativa (RSI)

El RSI es un indicador de momento que mide la velocidad y el cambio de los movimientos de precios. Un RSI por debajo de 40 sugiere una posible reversión en una tendencia alcista, mientras que un RSI por encima de 60 indica una potencial reversión en una tendencia bajista.

Implementación de la Estrategia

La estrategia implica entrar en una operación cuando la media móvil de 100 períodos confirma la tendencia general, el precio rebota en el nivel de Fibonacci del 38.2 %, y el RSI proporciona una confirmación adicional (por debajo de 40 en tendencias alcistas y por encima de 60 en tendencias bajistas). El *stop loss* se establece por debajo del nivel de 61.8 % de Fibonacci para limitar las pérdidas.

Riesgos y Consideraciones

Aunque esta estrategia puede ser efectiva, no está exenta de riesgos. La interpretación incorrecta de los indicadores o cambios imprevistos en el mercado pueden llevar a pérdidas. Por lo tanto, es crucial una gestión de riesgo adecuada y un análisis constante del mercado.

Pruebas y Optimización

Por último, esta estrategia se probará en diferentes activos y en diferentes temporalidades para evaluar sus rentabilidades y determinar cuál se adapta mejor. Además, se expondrán los defectos encontrados durante las pruebas para poder realizar ajustes y optimizaciones.

3.9. Programación Threading

Para la programación de nuestro Bot hemos utilizado la programación en hilos (*threads*). Esta programación es un concepto fundamental en el ámbito de la informática, en concreto en el ámbito de sistemas operativos y aplicaciones que requieren realizar varias tareas a la vez.

Concepto de Hilo

Un hilo es una secuencia de control que se encuentra en el programa y que se puede ejecutar de forma individual en la CPU. Es muy parecido a un programa secuencial ya que tiene un inicio, una secuencia de ejecución y un final. A pesar de esto, un hilo no es un programa completo y se compila dentro de un programa más grande, de modo que estos hilos comparten recursos y entornos.

Diferencias entre Hilos y Procesos

Los hilos son diferentes de los procesos tradicionales en varios aspectos. Mientras que los procesos son unidades individuales con su espacio de direcciones y sus recursos en el sistema, los hilos se establecen como subconjuntos de los procesos y, por lo tanto, comparten el estado y los diversos recursos que ofrece el proceso, como por ejemplo la memoria. Esto es una ventaja, ya que permite un intercambio de información entre los hilos de una forma más sencilla, pero a su vez, un hilo que presente errores puede afectar al funcionamiento de todo el proceso. [26]

Multitarea en Sistemas Operativos

La multitarea de los sistemas operativos puede ser de dos tipos: multitarea preventiva o cooperativa. En la multitarea preventiva, el sistema operativo decide cuándo debe pausar el hilo y cambiar a otro diferente. Sin embargo, en la multitarea cooperativa, son los hilos los que deben ceder el control. Esto afecta a cómo se programan y gestionan los hilos.

Sistemas de un Solo Núcleo

En sistemas con un solo núcleo de CPU, los hilos aportan una forma de multitarea, dando la impresión de que se están ejecutando en paralelo, aunque en realidad se ejecutan en ciclos, es decir, cuando se ejecuta uno y termina, pasa a ejecutarse otro.

Sistemas Multinúcleo

En los sistemas multinúcleo, sí se puede obtener paralelismo, consiguiendo que se ejecuten múltiples hilos o procesos a la vez. Esta programación es muy útil también en máquinas que solo tienen un único núcleo para aplicaciones que pueden beneficiarse de ella, como mantener la interfaz de usuario activa mientras se realizan tareas lentas o bloqueantes entre otros hilos del proceso. [37]

Desafíos de la Programación Multihilo

La programación multihilo viene con sus propios desafíos. Los hilos comparten la misma memoria, lo que permite la interconexión entre ellos, pero a su vez también pueden concluir en condiciones de carrera y problemas de sincronización.

Condiciones de Carrera

Una condición de carrera se produce cuando el proceso de ejecución de los hilos afecta al resultado final del programa. Esto puede ser difícil de detectar y corregir debido a la naturaleza no determinista de la programación de hilos.

Técnicas de Sincronización

Para solucionar estos problemas, se utilizan técnicas como la sincronización para asegurar que solo un hilo pueda acceder a un recurso y así no se produzca un colapso a la hora de acceder a ese recurso si múltiples hilos quieren trabajar con él. También se utilizan operaciones atómicas para garantizar la integridad de las operaciones sobre la información compartida, y el uso de información inalterable para que no se produzcan modificaciones concurrentes. [48]

Ventajas y Consideraciones

La programación en hilos es una herramienta muy poderosa y útil para mejorar el rendimiento y la capacidad de respuesta de las aplicaciones, pero es necesario una cuidadosa gestión de los problemas de sincronización y de seguridad al acceder a los diversos recursos de los que dispone el proceso que se está ejecutando.

3.10. Programación orientada a objetos (POO)

La programación orientada a objetos (POO) es un paradigma de programación que organiza el diseño del software en torno a objetos, en lugar de acciones y datos en lugar de lógica. En la POO, los objetos pueden considerarse como instancias de clases, que son plantillas para crear objetos con atributos y métodos específicos. Los atributos son variables que pertenecen a la clase y definen las características del objeto, mientras que los métodos son funciones que definen los comportamientos del objeto.

Encapsulamiento

Uno de los principios fundamentales de la POO es el encapsulamiento, que implica ocultar los detalles internos de un objeto y exponer solo lo necesario. Esto se logra mediante modificadores de acceso como *private*, *protected* y *public*.

Herencia

Otro principio clave es la herencia, que permite a una clase derivar de otra y heredar sus atributos y métodos, facilitando la reutilización del código y la creación de jerarquías de clases.

Polimorfismo

El polimorfismo permite que una misma operación se comporte de diferentes maneras en distintos objetos, lo que añade flexibilidad y escalabilidad al diseño del software.

Abstracción

La abstracción es otro concepto crucial en la POO, ya que permite simplificar la complejidad ocultando los detalles innecesarios y mostrando solo las características esenciales del objeto. Este enfoque facilita el diseño y mantenimiento del software, ya que los desarrolladores pueden centrarse en los aspectos más importantes de los objetos que están creando. [3]

3.11. Estructura utilizadas

Barrera

En programación concurrente, una barrera es un mecanismo de sincronización que permite a múltiples hilos o procesos esperar hasta que todos hayan alcanzado un cierto punto de ejecución. Es útil cuando se necesita que un conjunto de tareas se complete antes de que otro conjunto de tareas pueda comenzar.

Eventos de Threading

Un evento es un mecanismo de sincronización utilizado para indicar que un evento ha ocurrido. Los eventos son útiles en la programación multihilo para la comunicación entre hilos. Los hilos pueden esperar que ocurra un evento y, una vez que el evento es "set"(establecido), todos los hilos esperando en ese evento son notificados y pueden proceder.

Bucles

Un bucle es una estructura de control que permite repetir una secuencia de instrucciones un número determinado de veces o hasta que se cumpla una condición específica. Los tipos más comunes de bucles son `for` y `while`. En un bucle `for`, se itera sobre una secuencia de elementos, mientras que un bucle `while` continúa ejecutándose mientras una condición sea verdadera. [5]

Diccionarios

En programación, un diccionario es una colección de pares clave-valor. Cada clave en un diccionario es única y se utiliza para acceder a su valor correspondiente. Los diccionarios son muy útiles para almacenar datos asociados de manera que se pueda acceder a los valores mediante sus claves. [2]

Listas

Una lista es una colección ordenada de elementos que puede contener elementos de diferentes tipos. Las listas permiten agregar, eliminar y modificar elementos, y son una estructura de datos muy utilizada en muchos lenguajes de programación debido a su flexibilidad y facilidad de uso. [4]

Dataframe

Un DataFrame es una estructura de datos bidimensional ampliamente utilizada en el análisis de datos y en bibliotecas como `pandas` en Python. Su diseño es similar al de una tabla en una base de datos, lo que permite organizar los datos en filas y columnas. Cada columna en un DataFrame puede contener un tipo de dato diferente, lo que lo hace extremadamente flexible y útil para una variedad de tareas analíticas.

Una de las características más destacadas de un DataFrame es su estructura tabular, que facilita el acceso y la manipulación de datos. Cada fila y columna tiene etiquetas, lo que permite un acceso rápido y significativo a los datos. Esta capacidad para manejar tipos de datos mixtos en diferentes columnas es una ventaja significativa sobre otras estructuras de datos como los arrays de NumPy, que requieren que todos los elementos sean del mismo tipo.

Los DataFrames también son conocidos por permitir operaciones rápidas y eficientes. La biblioteca `pandas` proporciona una amplia gama de funciones optimizadas para el rendimiento en la manipulación y el análisis de datos. Estas operaciones incluyen la filtración, agrupación, agregación, y la fusión de datos, entre otras. Además, los DataFrames se integran bien con otras bibliotecas de Python, como NumPy para cálculos numéricos, Matplotlib para visualización de datos y scikit-learn para machine learning, lo que los hace una herramienta versátil y poderosa en el ecosistema de Python.
[1]

4. Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el desarrollo del proyecto:

Numpy

NumPy es una biblioteca de Python que proporciona una estructura de datos poderosa y eficiente conocida como el *array* n-dimensional o *ndarray*. Este objeto central de NumPy es la clave para realizar operaciones rápidas en *arrays* de datos homogéneos, es decir, elementos de un mismo tipo. A diferencia de las secuencias estándar de Python, que pueden crecer dinámicamente, los *arrays* de NumPy tienen un tamaño fijo en su creación, lo cual contribuye a la eficiencia en el cálculo numérico.

Uno de los motivos de la rapidez de NumPy es su vectorización, que permite realizar operaciones elementales sin necesidad de bucles explícitos, lo que lleva a menudo a una ejecución más rápida y a un código más legible. Además, NumPy se integra y es la base para una gran cantidad de paquetes científicos y matemáticos en Python, lo que lo hace esencial en el ecosistema de la ciencia de datos.

La biblioteca proporciona rutinas diversas para operaciones que incluyen matemáticas, lógica, manipulación de formas, clasificación, selección, transformadas de Fourier discretas, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más. NumPy destaca también por su capacidad para trabajar con *arrays* de manera eficiente, lo que permite realizar operaciones complejas de álgebra lineal, estadística y otras matemáticas aplicadas.

NumPy facilita la indexación y el *slicing* de *arrays*, permitiendo acceder y manipular subconjuntos de datos de manera sencilla. Además, se pueden realizar operaciones matemáticas básicas como adición, sustracción, multiplicación y división de *arrays* de manera directa y eficiente.

Además de su funcionalidad básica, NumPy ofrece un conjunto de herramientas avanzadas para trabajar con datos, como funciones para encontrar valores máximos y mínimos, operaciones matriciales y funciones estadísticas.

Para los que están comenzando con NumPy, hay guías para principiantes disponibles que introducen los conceptos principales de NumPy y enlazan a tutoriales adicionales. La documentación de NumPy está bien estructurada y ofrece una guía del usuario, una referencia de la API y una guía para contribuyentes. [13]

TA-Lib

La biblioteca TA-Lib es una herramienta de análisis técnico ampliamente utilizada por desarrolladores de software de *trading* que buscan realizar cálculos técnicos complejos sobre datos financieros y construir estrategias de *trading*. Escrita originalmente en lenguaje C, TA-Lib proporciona más de 150 indicadores técnicos y funciones de *trading*, como ADX, MACD, RSI, Estocástico y Bandas de Bollinger, además de patrones de reconocimiento de velas japonesas. Su núcleo en C/C++ permite un alto rendimiento, aunque también está disponible para Python a través de una API y puede ser integrada libremente tanto en aplicaciones de código abierto como comerciales.

Por otro lado, la biblioteca *Technical Analysis Library in Python* es una biblioteca de análisis técnico que permite realizar ingeniería de características desde conjuntos de datos de series temporales financieras, como precio de apertura, cierre, máximo, mínimo y volumen. Construida sobre las bibliotecas de Python Pandas y NumPy, es útil para el análisis de datos financieros y está diseñada para trabajar con Python 3.6 o superior. Esta biblioteca permite agregar de manera eficiente un conjunto completo de características de análisis técnico a un *DataFrame* de Pandas.

La biblioteca TA, que se encuentra en GitHub, también ofrece una variedad de indicadores y se puede instalar fácilmente con *pip*. Proporciona indicadores como el Índice de Flujo de Dinero (MFI), Bandas de Bollinger (BB), Índice de Fuerza Relativa (RSI), entre otros. Es una herramienta potente para aquellos que trabajan con datos financieros y buscan realizar análisis técnicos y preparar los datos para el modelado de *machine learning*.

Para comenzar a usar la biblioteca TA, necesitarás tener un conjunto de datos financieros que incluya columnas como 'Timestamp', 'Open', 'High', 'Low', 'Close' y 'Volume'. Es importante limpiar o llenar los valores *NaN* antes de agregar las características de análisis técnico. La biblioteca proporciona ejemplos de código y la capacidad de visualizar las características, lo que puede ser útil para comprender mejor los datos y las señales proporcionadas por los indicadores técnicos. [14]

Time

La biblioteca `datetime` en Python es un módulo versátil que proporciona clases para manipular fechas y horas. A continuación, se ofrece un resumen detallado y técnico de sus características principales, basado en la documentación oficial y recursos adicionales.

La gestión de fechas y horas es esencial en muchas aplicaciones, desde el registro de eventos hasta la planificación de actividades futuras. La manipulación de fechas y horas puede ser compleja debido a factores como los años bisiestos, los segundos intercalares y los cambios de horario de verano. La biblioteca `datetime` facilita estas operaciones, aunque requiere comprensión y cuidado en su uso, especialmente en aplicaciones que dependen de precisión temporal o manejan zonas horarias. [16]

Pandas

Pandas es una biblioteca de código abierto, con licencia BSD, que ofrece estructuras de datos y herramientas de análisis de alto rendimiento y fáciles de usar para el lenguaje de programación Python. Su diseño está orientado a trabajar de manera intuitiva con datos relacionales o etiquetados, lo que la convierte en una herramienta esencial para el análisis de datos en Python. [46]

Las características de Pandas son las siguientes:

- **Estructuras de Datos:** Pandas introduce dos nuevas estructuras de datos a Python: `DataFrame` y `Series`.
 - **DataFrame:** Es una estructura de datos bidimensional, mutable en tamaño y potencialmente heterogénea, similar a una tabla de base de datos o una hoja de cálculo de Excel. Cada columna puede tener un tipo de dato diferente (numérico, cadena, booleano, etc.). Los `DataFrames` tienen filas y columnas etiquetadas y permiten

operaciones aritméticas y de manipulación de datos de manera eficiente.

- **Series:** Es una estructura de datos unidimensional que puede contener cualquier tipo de datos (enteros, cadenas, floats, objetos de Python, etc.). Es similar a una columna en una hoja de cálculo o a una columna en una base de datos.
- **Lectura y Escritura de Datos:** Pandas soporta múltiples formatos para leer y escribir datos, incluyendo CSV, Excel, bases de datos SQL, y muchos otros.
- **Manipulación de Datos:** Ofrece funciones para reorganizar, girar, dividir y combinar conjuntos de datos. Permite filtrar, limpiar y preparar datos para el análisis.
- **Análisis de Datos:** Pandas facilita el análisis de datos con funciones para computar estadísticas descriptivas, correlaciones, manejo de datos faltantes, y más.
- **Trabajo con Fechas y Horas:** Proporciona funcionalidades robustas para trabajar con fechas y horas, incluyendo manipulación de rangos de fechas, conversión de zonas horarias, entre otros. [1]

Threading

Esta es la librería más importante, ya que es el esqueleto de nuestro Bot de Trading. Sin esta librería no podríamos entender el funcionamiento del programa.

La biblioteca `threading` en Python es un módulo que facilita la concurrencia a través de hilos. Un hilo es la unidad más pequeña de procesamiento que puede ser gestionada por un sistema operativo. El módulo `threading` permite a Python ejecutar múltiples operaciones de manera concurrente, lo que es especialmente útil en aplicaciones *I/O-bound* o para mejorar el rendimiento de aplicaciones *CPU-bound* mediante la distribución de carga en múltiples hilos.

Para la implementación de `threading` en nuestro programa es muy importante entender los conceptos básicos de cómo programar con hilos. [20]

Representa un hilo de ejecución individual.

1. **Creación:** Se crea instanciando `Thread` con un objeto *callable* (función) y argumentos.
2. **Inicio:** Se inicia con el método `start()`.
3. **Espera:** `join()` permite esperar a que un hilo termine su ejecución.

CSV

1. Lectura de Datos

- `csv.reader(csvfile, dialect='excel', **fmtparams)`: Retorna un objeto lector que iterará sobre las líneas en el archivo CSV dado. `csvfile` puede ser cualquier objeto que soporte el protocolo de iterador y devuelva una cadena cada vez que su método `__next__()` es llamado.
- **Manejo de Dialectos:** El módulo `csv` define dialectos (`excel`, `unix`, etc.) que describen características comunes de diferentes tipos de archivos CSV [15].

2. Escritura de Datos

- `csv.writer(csvfile, dialect='excel', **fmtparams)`: Retorna un objeto escritor que convierte los datos del usuario en cadenas delimitadas por comas y los almacena en un archivo CSV.
- Métodos como `writerow` y `writerows`: Permiten escribir filas individuales o múltiples filas a la vez.

3. Manejo de Diccionarios

- **csv.DictReader:** Lee el archivo CSV en un diccionario, donde las claves son proporcionadas por la fila de encabezados.
- **csv.DictWriter:** Escribe diccionarios en un archivo CSV, manejando opcionalmente los encabezados.

Os

La biblioteca `os` en Python proporciona una forma de interactuar con el sistema operativo. Permite realizar operaciones como [18]:

- Navegación y manipulación del sistema de archivos.
- Manejo de procesos.
- Recuperación de información del entorno del sistema.
- Manipulación de permisos de archivos.

Logging

La biblioteca `logging` en Python proporciona un sistema flexible para generar mensajes de registro (logs) de la ejecución de programas. Permite configurar diferentes niveles de severidad, formatos de mensajes, y destinos de salida. [17]

Traceback

La biblioteca `traceback` en Python proporciona utilidades para extraer, formatear y generar información de rastreo de excepciones. Es útil para obtener detalles sobre los errores ocurridos durante la ejecución de un programa. [21]

Subprocess

La biblioteca `subprocess` en Python permite la ejecución de nuevos programas y comandos del sistema desde un script de Python. Proporciona más control sobre la creación y comunicación con los procesos que el módulo `os`. Esta biblioteca es utilizada en el proyecto para exponer las operaciones realizadas por el Bot. [19]

Tkinter

La biblioteca `tkinter` es un módulo en Python para la creación de interfaces gráficas de usuario (GUI). Es un envoltorio estándar para Tk, que es un toolkit gráfico multiplataforma que se usa bajo el entorno de escritorio X Window System en sistemas Unix, Windows y macOS. `tkinter` se destaca por su simplicidad y está incluido por defecto en la mayoría de las distribuciones de Python, lo que lo hace accesible y ampliamente utilizado para la creación rápida de prototipos de GUI y aplicaciones ligeras. [?]

1. **Ventana Principal:** En `tkinter`, todo programa debe crear una ventana principal usando `Tk()`. Esta ventana sirve como el contenedor principal para todos los demás elementos de la GUI.
2. **Widgets:** Son los elementos de construcción para crear una GUI.
3. **Gestión de Diseño:** `tkinter` ofrece varias opciones para la gestión del diseño.
4. **Eventos y Bindings:** La interactividad en `tkinter` se maneja a través de eventos (como clics del mouse o pulsaciones de teclas) y bindings (asociar eventos con funciones de manejo).
5. **Variables de Control:** Especialmente diseñadas para manejar cambios en los widgets, como `StringVar`, `IntVar`, etc.

MetaTrader

MetaTrader es una plataforma completa que ofrece una amplia variedad de herramientas de análisis técnico, gráficos, indicadores y osciladores, lo que la hace una herramienta popular entre traders de todo el mundo. MetaTrader también ofrece una interfaz de programación de aplicaciones (API) para que los desarrolladores puedan crear programas personalizados, como robots de trading automatizados.

La plataforma MetaTrader está disponible en dos versiones principales: MetaTrader 4 (MT4) y MetaTrader 5 (MT5). La versión MT4 es la más popular, aunque MT5 también se está volviendo cada vez más popular en algunos mercados. Ambas versiones ofrecen funcionalidades avanzadas y una amplia variedad de opciones de personalización.

MetaTrader es compatible con varios corredores y proveedores de datos, lo que permite a los usuarios acceder a una amplia gama de instrumentos financieros y mercados. La plataforma también es conocida por su alta seguridad, estabilidad y velocidad de ejecución.

Esta plataforma permite extraer los datos históricos de los activos con los que se va a operar durante el proyecto. [27]

Pycharm

PyCharm es un entorno de programación integrado (IDE) que es específico para Python y que ofrece una gran cantidad de herramientas esenciales para que los programadores de Python se sientan cómodos y sea intuitivo

para ellos. Contiene una estrecha interacción para recrear un entorno que sea conveniente para poder desarrollar de forma productiva aplicaciones en Python, web y más.

PyCharm está disponible en dos versiones. La primera es una versión gratuita que se llama Community y la segunda se llama Professional, que es una versión de pago y ofrece a los clientes herramientas adicionales como el soporte a frameworks web populares como Django, Flask, Pyramid o web2py, así como una integración de sistemas para el control de versiones como Git, Mercurial, Perforce y Subversion.

Una de las características que hacen a PyCharm especial es su editor de código inteligente con asistencia para métodos y clases. Viene con un depurador de código incorporado y herramientas que permiten refactorizar el código. Proporciona una interfaz unificada que permite trabajar con diversos sistemas de control de versiones y permite la ejecución, depuración y pruebas de aplicaciones en hosts remotos o en máquinas virtuales.

PyCharm permite mantener el código limpio y depurado, identificando errores y problemas para después dar soluciones rápidas y eficaces. Tiene a su disposición plantillas de código y sugerencias para la creación de código a partir del uso. También ofrece opciones que permiten sobrescribir o implementar más funciones.

Esta plataforma incluye además recursos educativos para poder aprender a programar en Python. Tiene una interfaz educativa más sencilla e interactiva, ideal para los que acaban de empezar a programar en el lenguaje Python.

Para poder instalar PyCharm se necesitan unos recursos mínimos y recomendados, que varían desde 4 GB de memoria y 2.5 GB de espacio en el disco para la mínima versión, hasta 8 GB de memoria para la versión recomendada para todos los sistemas. Su compatibilidad va desde sistemas operativos como macOS de 64 bits 10.11, Microsoft Windows 7 SP1 y distribuciones Linux que puedan dar soporte a Gnome, KDE o Unity DE.

PyCharm también tiene desventajas como el elevado precio de la versión de pago. Además, puede consumir muchos recursos a la hora de ejecutar un programa y eso puede causar que se vuelva lento.

La instalación de PyCharm es sencilla y se puede realizar desde la página oficial de JetBrains. Para empezar a programar en un nuevo proyecto es muy intuitiva, permitiendo seleccionar una plantilla o entorno que mejor se adapte a nuestras necesidades como Django o Virtualenv. [25]

Programación en Python

Para la elaboración de nuestro programa hemos elegido el lenguaje de programación de Python.

Este lenguaje es muy popular por sus prestaciones y sencillez a la hora de programar. Es un lenguaje interpretado, de alto nivel, con tipado dinámico y una fuerte orientación a objetos. Su sintaxis, como hemos citado, es clara y legible, lo que reduce los costos de mantenimiento del código y hace que la implementación de los diferentes programas sea muy sencilla de realizar. Además, Python soporta módulos y paquetes, lo que hace más flexible al programa y la reutilización del código.

Es un lenguaje multiplataforma, esto significa que el mismo programa puede ejecutarse en diversos sistemas operativos sin realizar modificaciones adicionales. Se considera multi-paradigma, ya que permite que la programación sea estructurada, orientada a objetos y funcional. Esto le otorga una gran flexibilidad al momento de resolver problemas a la hora de programar, ya que podemos elegir el paradigma que mejor se ajusta a nuestras necesidades.

Python ofrece una gran variedad de bibliotecas estándar que permiten realizar y adaptarse a una gran variedad de tareas, desde el manejo de bases de datos hasta el procesamiento de datos y la programación con servicios web. Además, permite modificar objetos y variables en tiempo de ejecución, lo que facilita la depuración del código. [47]

Es muy sencillo de aprender, ya que ofrece una curva de aprendizaje suave para aquellos programadores que son novatos en Python. Por otro lado, es lo suficientemente potente como para ser utilizado por desarrolladores profesionales en la programación de software complejo en diferentes sectores como el aprendizaje automático, ciencia de datos, bases de datos y más.

La programación orientada a objetos que ofrece Python incluye herencia, polimorfismo y encapsulación. La herencia permite crear clases derivadas que reutilizan código de las clases base. El polimorfismo posibilita que una interfaz se implemente de múltiples formas, y la encapsulación oculta los detalles de la implementación de un objeto y muestra solo los detalles que son seguros para el uso de otros objetos.

Si se desea profundizar en el aprendizaje de Python, existen suficientes recursos avanzados que abordan trucos, prácticas, patrones, diseños y técnicas que mejoran la programación de los sistemas en Python.

Python es un lenguaje que nos permite realizar diferentes tareas y es muy potente, lo cual hace que se adapte a una amplia gama de aplicaciones. Además, posee una comunidad activa y sorprendentemente amplia que brinda soporte continuo.

Para el desarrollo de nuestro bot de trading, nosotros utilizaremos Python 3.11 y programaremos en PyCharm CE. [22]

SCRUM - Metodología ágil

La metodología ágil es un enfoque de gestión de proyectos que divide los proyectos en fases dinámicas conocidas como sprints, permitiendo a los equipos planificar, ejecutar y evaluar continuamente el progreso. Se basa en cuatro valores fundamentales del Manifiesto Ágil:

- 1. Individuos e interacciones sobre procesos y herramientas.**
- 2. Software funcionando sobre documentación extensiva.**
- 3. Colaboración con el cliente sobre negociación de contratos.**
- 4. Respuesta ante el cambio sobre seguir un plan.**

Estos valores fomentan la flexibilidad, la colaboración continua y la entrega rápida de software funcional.

Entre los principios clave de la metodología ágil destacan la entrega frecuente de software funcional, la aceptación de cambios en los requisitos incluso en etapas tardías, el mantenimiento de un ritmo de trabajo sostenible y la colaboración constante entre el equipo de desarrollo y los clientes. Metodologías ágiles populares como Scrum y Kanban proporcionan estructuras específicas para implementar estos principios, mejorando la eficiencia y la adaptabilidad del desarrollo de software. Los beneficios incluyen una mayor satisfacción del cliente, una entrega más rápida y una valorización de las ideas de los empleados, reduciendo así el retrabajo. [7]

Herramientas de gestión de proyectos

Github

GitHub es una plataforma robusta para el control de versiones y la colaboración, que también incluye herramientas para la gestión de proyectos. Usando GitHub Projects, los equipos pueden planificar y rastrear el trabajo

a través de tableros y hojas de cálculo adaptables que se integran perfectamente con issues y pull requests. Esto permite actualizaciones en tiempo real y visibilidad del progreso de las tareas. GitHub soporta la creación de campos y vistas personalizados para ajustarse a las necesidades específicas de un proyecto, haciéndolo una herramienta flexible para diversos flujos de trabajo. Además, GitHub se integra con otras herramientas para mejorar las capacidades de gestión de proyectos, como ZenHub, que añade características como gráficos de burndown y planificación de sprints a la interfaz de GitHub. [23]

Zenhub

ZenHub es una solución de gestión de proyectos diseñada para funcionar directamente dentro de GitHub. Proporciona características avanzadas como tableros Kanban y Scrum, seguimiento de velocidad y pronósticos de lanzamientos. La integración de ZenHub permite a los usuarios gestionar sus proyectos sin salir de GitHub, añadiendo funcionalidades como la planificación de sprints y gráficos de burndown. Esta herramienta es particularmente útil para equipos ágiles, proporcionando una manera fluida de organizar y priorizar el trabajo, rastrear el progreso y colaborar de manera efectiva. ZenHub ofrece planes tanto gratuitos como de pago, adaptándose a diferentes tamaños de equipos y necesidades. [49]

Herramientas de documentación

Word

Durante el proyecto, se utiliza Word para realizar anotaciones preliminares y estructurar la base del proyecto, lo que ayudó a mantener una visión clara y organizada desde el inicio. Word es un procesador de texto altamente reconocido que permite estructurar el contenido de manera que quede claro y conciso. [33]

Latex

LaTeX es una herramienta potente para la preparación de documentos técnicos y científicos de alta calidad, utilizada principalmente para la elaboración de informes, tesis y artículos académicos. Ofrece un control preciso sobre la tipografía y la estructura del documento, lo que resulta ideal para proyectos que requieren una presentación profesional y consistente. En el proyecto, se utiliza LaTeX para la fase final y la presentación del proyecto, asegurando un formato limpio y una apariencia profesional. La capacidad

de LaTeX para manejar referencias bibliográficas, fórmulas matemáticas y gráficos complejos lo hace especialmente adecuado para documentos técnicos avanzados. [38]

Gestión de referencias

Mendeley es una herramienta de gestión de referencias y colaboración que facilita la organización, el almacenamiento y la citación de artículos académicos y otros documentos de investigación. Permite a los usuarios crear bibliotecas personales, anotar PDF, y generar citas y bibliografías en diversos estilos. Además, Mendeley ofrece capacidades de colaboración mediante grupos privados y públicos, permitiendo compartir referencias y notas con colegas, lo cual es particularmente útil en proyectos colaborativos de investigación. Esta herramienta fue especialmente útil para organizar y gestionar las referencias bibliográficas durante el desarrollo de nuestro proyecto. [12]

5. Aspectos relevantes del desarrollo del proyecto

5.1. Programación

En este apartado se explicará como se ha programado el Bot de Trading explicando una por una las clases que lo componen.

VELAS_1

La clase `Candle` es una abstracción de la vela japonesa que permite una manipulación y almacenamiento eficiente de los datos relacionados con cada vela en el contexto de trading y análisis de mercados. Permite a los usuarios del código representar de forma precisa cada vela dentro de un marco temporal específico, facilitando el análisis detallado de tendencias y patrones en los gráficos de precios.

Además del manejo individual de las velas a través de la clase `Candle`, el código también proporciona una funcionalidad para trabajar con conjuntos de velas. Esto se logra mediante la conversión de una lista de objetos `Candle` en un `DataFrame` de `pandas`, una estructura de datos que es ampliamente utilizada para el análisis de datos en Python debido a su flexibilidad y potencia. Este enfoque permite aplicar técnicas de análisis de datos avanzadas a conjuntos de velas, lo cual es fundamental en muchas estrategias de trading y en la toma de decisiones basada en datos.

Importaciones

- **import pandas as pd:** Biblioteca de Python ampliamente utilizada para el análisis y la manipulación de datos. En este contexto, se utiliza para crear y manejar **DataFrames**, que son estructuras de datos tabulares.
- **import os:** El módulo **os** es fundamental para el trato con las estructuras del proyecto.
- **import logging:** El módulo **logging** es fundamental para manejar la salida de errores.

Funcion - Constructoras

Cuando se crea un nuevo objeto **Candle**, el constructor se activa automáticamente y configura los atributos de la vela a sus valores por defecto.

Estos atributos incluyen **time_last_candle**, que representa el tiempo de la última vela en formato de **timestamp**, y **open**, **close**, **high**, **low**, **volume**, que representan respectivamente el precio de apertura, cierre, el precio máximo, el precio mínimo y el volumen de trading de la vela. Inicialmente, estos atributos se establecen en cero (0), indicando que no tienen valores asignados aún. Esta configuración inicial es importante porque proporciona un punto de partida claro y consistente para cada objeto **Candle**, asegurando que todos los atributos estén correctamente definidos antes de su uso posterior en análisis o cálculos.

Funcion - set

Cuando se llama al método **set**, se le pasan varios parámetros: **timestamp**, **open**, **high**, **low**, **close** y **tick_volume**. Estos parámetros representan respectivamente el momento en que se formó la vela, el precio de apertura, el precio más alto alcanzado, el precio más bajo, el precio de cierre y el volumen de transacciones durante el período de la vela. El método actualiza los atributos de la instancia de **Candle** con estos valores, lo que permite que la vela refleje con precisión un período específico de trading en el mercado. Este proceso es fundamental para garantizar que los datos de la vela estén completos y sean precisos, lo que es crucial para cualquier análisis técnico posterior basado en estos datos.

Funcion - get

Al ejecutar `get`, el método no recibe parámetros y procede a acceder directamente a los atributos internos del objeto `Candle`. Recolecta los valores de `open`, `close`, `high`, `low`, `volume` y `time_last_candle`, y los organiza en una lista en el orden mencionado. Esta lista es entonces devuelta por el método, proporcionando una representación compacta y completa de la información de la vela.

Funciones - obtener_datos

Este método facilita la manipulación y el análisis más avanzado de los datos de las velas japonesas, aprovechando las potentes capacidades de `pandas`.

Cuando se llama a `obtener_datos`, recibe como parámetro una lista de objetos `Candle`. Cada objeto `Candle` contiene información detallada sobre una vela japonesa, incluyendo precios de apertura, cierre, máximo, mínimo, volumen y una marca de tiempo. El método inicia creando un `DataFrame` vacío con columnas nombradas para representar estos atributos. Luego, itera sobre cada objeto `Candle` en la lista proporcionada, utilizando el método `get` de cada objeto `Candle` para recuperar sus datos. Estos datos se añaden al `DataFrame`, fila por fila, construyendo así una representación tabular completa de todos los objetos `Candle`.

VELAS

El código de la clase realiza la lectura y actualización de los datos de las velas japonesas para su posterior utilización en otros apartados. A continuación, se describen las importaciones necesarias, las constantes y variables globales, y las funciones principales del módulo.

Importaciones

- **import csv:** Esta importación es esencial para manejar archivos en formato CSV (valores separados por comas), que es un formato común en el almacenamiento y transferencia de datos financieros.
- **from src.VELAS_1 import Candle:** Aquí se importa la clase `Candle` desde un módulo específico. Esta clase encapsula la lógica y los atributos de una vela japonesa, como fecha, precio de apertura, cierre, máximo y mínimo.

- **import datetime:** El módulo `datetime` es fundamental para manejar fechas y horas, lo que es crucial en análisis financieros donde el tiempo es un factor clave.
- **import os:** El módulo `os` es fundamental para el trato con las estructuras del proyecto.
- **import logging:** El módulo `logging` es fundamental para manejar la salida de errores.

Constantes y Variables Globales

- **WINDOW_CANDLES = 200:** Esta constante define el número de velas que se mantendrán en memoria. El límite de 200 sugiere un equilibrio entre tener suficientes datos para el análisis y gestionar eficientemente la memoria.

Función update_candles

El proceso comienza abriendo el archivo CSV especificado en el parámetro `fichero`. El método utiliza la biblioteca `csv` de Python para leer los datos del archivo. Al leer el archivo, salta la primera línea, que generalmente es un encabezado, y luego itera sobre cada fila del CSV.

Para cada fila, el método realiza varias operaciones para transformar los datos crudos en una representación útil. Primero, convierte los campos de fecha y hora de la fila en un objeto de fecha y hora de Python y luego en una marca de tiempo Unix (`timestamp`), lo que facilita su manejo y cálculo en operaciones posteriores.

Después de procesar la fecha y hora, el método crea un objeto `Candle`, que representa una vela japonesa individual. Este objeto se inicializa con los datos de la fila, que incluyen el precio de apertura, cierre, máximo, mínimo y volumen, todos valores esenciales en el análisis de velas japonesas. Una vez creado, el objeto `Candle` se agrega a una lista de velas (`candles`).

El método también implementa un manejo de errores para capturar y manejar excepciones que puedan surgir durante la lectura del archivo o la conversión de datos, como errores de formato en las fechas, horas o valores numéricos. Si ocurre un error al procesar una fila, el método imprime un mensaje de error detallado, lo que ayuda en la depuración y el mantenimiento del código.

Función `thread_candles`

Esta función se ejecuta como un hilo independiente, lo que permite realizar la carga y actualización de las velas de manera concurrente con otras tareas del sistema, asegurando así una operación eficiente y sin interrupciones.

La función comienza inicializando los datos de las velas japonesas utilizando la función `update_candles`, que carga los datos desde un archivo CSV especificado en el parámetro `fichero`. Este proceso de carga implica leer el archivo CSV, procesar cada fila para convertirla en un objeto `Candle` y agregarlo a una lista. Cada objeto `Candle` representa una vela japonesa individual, conteniendo información detallada como fecha, hora, precios de apertura, cierre, máximo, mínimo y volumen. Una vez que los datos iniciales de las velas se cargan y se almacenan en el diccionario `data` bajo la clave `candles`, la función establece el estado `candles_ready` en `True` para indicar que los datos iniciales están listos para su uso.

Posteriormente, el método entra en un bucle controlado por el evento `pill2kill`, que permite que el hilo se detenga de manera controlada cuando se activa este evento. Dentro del bucle, el método continúa leyendo y procesando las nuevas líneas del archivo CSV, si las hay, actualizando continuamente la lista de velas en `data`. Este proceso implica convertir las fechas y horas de las filas del CSV a objetos de fecha/hora y luego a marcas de tiempo, y luego crear y agregar nuevos objetos `Candle` a la lista. El método también se asegura de mantener el tamaño de la lista de velas dentro de un límite predefinido (`WINDOW_CANDLES`), eliminando las velas más antiguas según sea necesario para dar cabida a las nuevas.

Finalmente, cada vez que se procesa una fila nueva, el hilo se sincroniza con otros hilos a través de una barrera (`barrera.wait()`), asegurando así que la actualización de los datos se realice de manera ordenada y coherente con otras operaciones que se ejecutan en paralelo en el sistema.

En resumen, `thread_candles` es una función esencial para mantener actualizados los datos de las velas japonesas en el sistema de trading. Su diseño aprovecha la programación concurrente para asegurar que los datos se procesen de forma continua y eficiente, lo que es vital para el análisis técnico y la toma de decisiones en tiempo real en el trading.

RSI

Esta clase se centra en calcular y actualizar en tiempo real el Índice de Fuerza Relativa (RSI), un indicador técnico utilizado en el análisis de mercados financieros. La clase opera mediante hilos en Python, lo que permite realizar estas tareas de manera concurrente sin interrumpir otras operaciones.

El proceso comienza esperando a que los datos necesarios para el cálculo estén listos. Una vez que los datos están disponibles, la clase procede a calcular el RSI utilizando estos datos. Este cálculo se realiza de forma periódica y continua mientras el hilo está activo.

Para asegurar una ejecución fluida y coordinada, la clase utiliza mecanismos de sincronización entre hilos. Esto permite que el hilo de cálculo del RSI trabaje en armonía con otros hilos que puedan estar ejecutándose simultáneamente, evitando conflictos y garantizando la consistencia de los datos.

En caso de errores o excepciones durante el cálculo o la sincronización, la clase está diseñada para capturar estos problemas y reportarlos adecuadamente. Esto es crucial para la estabilidad y fiabilidad del sistema en un entorno de trading en tiempo real, donde la precisión y la rapidez son esenciales.

Importaciones

- **import pandas as pd:** Pandas es una biblioteca de Python ampliamente utilizada para la manipulación y análisis de datos. Proporciona estructuras de datos potentes y flexibles, como los **DataFrames**, que son ideales para trabajar con datos tabulares.
- **from src.VELAS_1 import obtener_datos, Candle:** Este import indica que se están utilizando funciones o clases específicas de un módulo personalizado llamado **VELAS_1**, ubicado en un paquete o directorio **src**. **obtener_datos** probablemente sea una función diseñada para obtener y posiblemente procesar datos de velas, que son representaciones gráficas comunes en el análisis técnico de mercados financieros. **Candle** podría ser una clase que define la estructura o características de una vela individual, incluyendo aspectos como el precio de apertura, cierre, máximo y mínimo.
- **from ta.momentum import RSIIIndicator:** Esta línea importa **RSIIIndicator** de la biblioteca **ta.momentum**. **ta** es una biblioteca de

análisis técnico para Python, y `RSIIndicator` es una clase o función diseñada específicamente para calcular el Índice de Fuerza Relativa (RSI).

- **import time:** Es útil para manejar operaciones que dependen del tiempo, como temporizadores, retrasos o la medición de la duración de ciertas tareas.
- **import os:** El módulo `os` es fundamental para el trato con las estructuras del proyecto.
- **import logging:** El módulo `logging` es fundamental para manejar la salida de errores.

Función `thread_rsi`

La función `thread_rsi` está diseñada para ser el núcleo de un hilo en Python que se encarga de calcular y actualizar el indicador de Análisis Técnico conocido como Índice de Fuerza Relativa (RSI). Esta función opera dentro de un entorno de programación concurrente, donde múltiples hilos pueden ejecutarse simultáneamente para realizar tareas diferentes de manera eficiente.

Al inicio, la función entra en un bucle de espera, indicado por el mensaje '`Waiting for candles...`'. Este bucle verifica continuamente si los datos de las velas (`candles`), necesarios para calcular el RSI, están listos. Durante esta espera, hay una pausa de un segundo en cada iteración para evitar un consumo excesivo de recursos.

Una vez que los datos están listos, la función procede a calcular el RSI llamando a `parameters_RSI`, que es otra función encargada de realizar el cálculo específico del RSI con los datos proporcionados. Tras el cálculo inicial, la función marca los datos del RSI como listos (`data[rsi_ready] = True`), indicando que el indicador está listo para su uso.

Posteriormente, la función entra en otro bucle, esta vez controlado por el objeto `pill2kill`, que es un evento de threading. Este bucle permite que la función continúe recalculando el RSI de manera periódica hasta que se señale la finalización del hilo a través de `pill2kill`. En cada iteración de este bucle, se recalcula el RSI y se sincroniza con otros hilos mediante una barrera (`barrera.wait()`). Esta sincronización asegura que las operaciones entre diferentes hilos se realicen de manera ordenada y coherente.

La función está envuelta en un bloque `try-except` para manejar cualquier excepción que pueda surgir durante su ejecución. En caso de un error,

imprime un mensaje indicando el problema, lo que ayuda en la depuración y mantenimiento del código.

Función parameters_RSI

La función comienza extrayendo los datos de las velas (`candlesticks`) del mercado financiero, que son esenciales para el cálculo del RSI. Estos datos están contenidos en un diccionario llamado `data`, bajo la clave `candles`". Utiliza la función `obtener_datos` para transformar estos datos en un `DataFrame` de `pandas`, una estructura de datos ideal para el manejo y análisis de series temporales como las que se encuentran en los mercados financieros.

Con el `DataFrame` preparado, la función procede a instanciar un objeto de `RSIIndicator`, una herramienta de la biblioteca de análisis técnico `ta`. Este objeto se configura para calcular el RSI usando la columna `CLOSE`" del `DataFrame`, que representa los precios de cierre de las velas. El cálculo del RSI se realiza utilizando un periodo de ventana de 14, que es un estándar común en el análisis técnico.

Una vez obtenido el RSI, la función actualiza el diccionario `data` con los valores recientes del RSI. Específicamente, almacena los dos últimos valores calculados del RSI, que son relevantes para análisis de tendencias a corto plazo en el trading. Este paso es crucial porque permite que otras partes del sistema tengan acceso a los valores más recientes del RSI para tomar decisiones de inversión o análisis más informadas.

La función está diseñada para manejar errores y excepciones que podrían surgir durante el proceso de cálculo. Si ocurre un error, se captura la excepción y se imprime un mensaje de error detallado. Esto es importante para la depuración y el mantenimiento continuo del código, asegurando que cualquier problema pueda ser rápidamente identificado y resuelto.

SMA

Esta clase está diseñada para gestionar el cálculo de la Media Móvil Simple (SMA) en un entorno de trading automatizado. El enfoque principal es monitorear y actualizar continuamente la SMA, un indicador técnico clave utilizado en el análisis de los mercados financieros. Comienza con un proceso de espera, asegurándose de que los datos necesarios, como los precios de cierre de las velas, estén disponibles. Una vez que los datos están listos, procede a calcular la SMA.

El cálculo implica determinar la tendencia del mercado basándose en la relación del precio de cierre más reciente con la SMA. Según esta relación, el mercado se clasifica en una de tres categorías: alcista, bajista o en rango. Este análisis ayuda a entender la dirección general del mercado.

La clase también incorpora mecanismos de sincronización para operar eficientemente junto con otros procesos paralelos. Además, cuenta con una gestión de errores para manejar situaciones excepcionales y asegurar la fiabilidad y estabilidad del proceso de cálculo. En general, esta clase es un componente vital para automatizar el análisis técnico en el trading, proporcionando actualizaciones constantes sobre la tendencia del mercado basadas en la SMA.

Importaciones

- **from src.VELAS_1 import obtener_datos, Candle:** Aunque la clase específica `Candle` no se utiliza directamente, `obtener_datos` parece ser una función crucial en este código. Se usa para obtener y posiblemente procesar datos de velas financieras, convirtiéndolos en un formato adecuado (probablemente un `DataFrame` de Pandas) para el análisis posterior.
- **import time:** Esta biblioteca se utiliza para manejar operaciones relacionadas con el tiempo, como pausas en la ejecución del programa.
- **import os:** El módulo `os` es fundamental para el trato con las estructuras del proyecto.
- **import logging:** El módulo `logging` es fundamental para manejar la salida de errores.

Función `thread_SMA`

Esta función opera dentro de un hilo de ejecución, un concepto en programación que permite realizar múltiples tareas de manera simultánea y eficiente. Su rol principal es monitorear la disponibilidad de los datos necesarios para el cálculo de la SMA, como son los datos de las velas financieras, y ejecutar el proceso de cálculo una vez que estos datos estén listos.

Al inicio, `thread_SMA` entra en un estado de espera activa, comprobando repetidamente si los datos requeridos han sido preparados. Durante este tiempo, hace pausas cortas para no consumir recursos de manera innecesaria. Una vez que los datos están disponibles, procede a calcular la SMA utilizando otra función específica para este fin. Este cálculo inicial es seguido por la actualización de un indicador en el sistema que señala que la SMA está lista para ser utilizada.

La función no termina su tarea con un solo cálculo. En cambio, entra en un ciclo continuo donde regularmente recalcula la SMA. Este ciclo se mantiene hasta que se recibe una señal para detener el hilo, lo que se gestiona a través de un objeto denominado `pillkill`. Este mecanismo es crucial para permitir que el hilo se detenga de manera ordenada y segura cuando sea necesario.

Además, `thread_SMA` está diseñada para trabajar en conjunto con otros hilos que pueden estar ejecutándose en paralelo. Utiliza una barrera de sincronización para coordinar sus operaciones con estos otros hilos, asegurando que los cálculos y las actualizaciones se realicen de manera coherente y sin conflictos.

La función también incluye un manejo de excepciones robusto, capturando y reportando cualquier error que pueda surgir durante su ejecución. Esto es vital para la detección de problemas y la estabilidad general del sistema, especialmente importante en aplicaciones de trading donde la precisión y la fiabilidad son fundamentales.

Función `parameters_SMA`

Inicialmente, la función extrae los datos relevantes para el análisis a partir de un diccionario proporcionado como parámetro. Utiliza estos datos para construir un `DataFrame` de `pandas`, que es una estructura de datos eficiente y conveniente para manejar series temporales como las de los mercados financieros. En este `DataFrame`, se calcula la SMA de los precios de cierre utilizando un periodo de ventana específico, en este caso, 200 periodos. Esto

significa que la SMA se calcula como el promedio móvil de los últimos 200 precios de cierre.

Además de calcular la SMA, la función evalúa la posición del último precio de cierre en relación con esta media móvil. Aplica un margen de tolerancia para determinar si el precio actual está significativamente por encima o por debajo de la SMA. Si el último precio de cierre está por encima de la SMA ajustada por un factor de tolerancia, la tendencia del mercado se clasifica como alcista ('Bull'). Si está por debajo, se considera bajista ('Bear'). Si no se cumple ninguna de estas condiciones, se considera que el mercado está en un rango ('Range').

Esta evaluación es crucial porque proporciona una indicación rápida de la tendencia del mercado, lo que es valioso para los traders y analistas en la toma de decisiones. La clasificación resultante se almacena de nuevo en el diccionario de datos, haciendo que la información esté disponible para otras partes del sistema.

La función está diseñada con un manejo de excepciones para capturar y reportar errores que puedan surgir durante el proceso de cálculo. Esto es importante para garantizar la estabilidad y fiabilidad del proceso de análisis técnico, especialmente en entornos de trading donde decisiones rápidas y precisas son esenciales.

FIBO

Esta clase está diseñada para calcular y actualizar los niveles de Fibonacci, que son herramientas clave en el análisis técnico de los mercados financieros. Los niveles de Fibonacci se utilizan para identificar posibles zonas de soporte y resistencia basadas en máximos y mínimos históricos de precios. La clase opera dentro de un hilo de ejecución en un entorno de programación concurrente, lo que permite realizar el análisis y la actualización de estos niveles en tiempo real sin interrumpir otras tareas.

La lógica central de la clase se centra en monitorear la disponibilidad de los datos del mercado, esperando hasta que estén listos para ser procesados. Una vez que los datos están disponibles, la clase calcula los niveles de Fibonacci, utilizando máximos y mínimos de los precios de las velas financieras para determinar estos niveles. La clase se ajusta dinámicamente a los cambios en el mercado, recalculando los niveles de Fibonacci cuando se detectan nuevas condiciones de mercado, como una tendencia alcista, bajista o un rango de mercado.

Un aspecto importante de la clase es su capacidad para sincronizarse con otros hilos de ejecución, lo que asegura que los cálculos de Fibonacci se realicen de manera coherente y en armonía con otras operaciones analíticas. Además, la clase está equipada con un manejo robusto de excepciones para garantizar la estabilidad y fiabilidad del proceso, capturando y reportando cualquier error que pueda surgir durante el cálculo de los niveles de Fibonacci.

Importaciones

- **import src.VELAS_1:** Esta importación sugiere la inclusión de un módulo o paquete personalizado llamado `VELAS_1`, que probablemente contiene funciones y clases específicas relacionadas con el procesamiento o análisis de datos de velas financieras.
- **import numpy as np:** Numpy es una biblioteca fundamental en Python para cálculos numéricos y científicos.
- **import time:** Esta biblioteca se utiliza para manejar operaciones relacionadas con el tiempo, como pausas en la ejecución del programa.
- **import os:** El módulo `os` es fundamental para el trato con las estructuras del proyecto.
- **import logging:** El módulo `logging` es fundamental para manejar la salida de errores.

Función thread_fibo

La función `thread_fibonacci` desempeña un rol crucial en el cálculo de los niveles de Fibonacci en el contexto de un sistema de trading automatizado o una herramienta de análisis técnico. Esta función está diseñada para ejecutarse como un hilo separado, lo que permite realizar los cálculos de Fibonacci sin interferir con otras operaciones del sistema.

Al principio, la función entra en un modo de espera, indicado por el mensaje '`[Hilo Niveles de Fibonacci] - Esperando velas...`'. Esta espera activa consiste en verificar repetidamente si los datos necesarios para el cálculo de Fibonacci, en este caso, las velas financieras, están listos. Durante esta fase, la función utiliza pausas (con `time.sleep(0.001)`) para evitar el consumo excesivo de recursos del sistema.

Una vez que los datos están listos, la función procede a inicializar una estructura para almacenar los máximos y mínimos históricos, que son esenciales para calcular los niveles de Fibonacci. A continuación, se utiliza la

función `maximos_minimos` para identificar estos puntos clave en los datos de las velas, actualizando el diccionario `max_min` con los valores encontrados.

El cálculo de los niveles de Fibonacci se realiza considerando tanto las condiciones actuales del mercado como las operaciones recientes. Si el mercado se encuentra en un rango, la función ajusta los máximos y mínimos a los valores recientes para reflejar la situación actual. En cambio, si se han realizado operaciones recientes de compra o venta, la función actualiza los máximos y mínimos en consecuencia para capturar los cambios en el mercado provocados por estas operaciones.

Los niveles de Fibonacci se calculan utilizando los máximos y mínimos actualizados, aplicando los ratios de Fibonacci para determinar puntos de soporte y resistencia potenciales. Estos niveles son luego almacenados en el diccionario `data` para su uso en el análisis técnico y la toma de decisiones de trading.

La función se mantiene en un bucle continuo, recalculando los niveles de Fibonacci cada vez que se actualizan los datos de las velas. Este bucle solo se interrumpe cuando se recibe una señal para terminar el hilo a través del objeto `pill2kill`. Además, la función utiliza una barrera de sincronización (`barrera.wait()`) para asegurar que sus operaciones estén coordinadas con otros hilos que se ejecutan en paralelo.

El manejo de excepciones en la función garantiza que cualquier error durante el proceso de cálculo sea capturado y reportado adecuadamente, lo que es crucial para la estabilidad y fiabilidad del sistema de trading.

Función `maximos_minimos`

La función `maximos_minimos` dentro de la clase relacionada con Fibonacci es una función auxiliar diseñada para identificar los puntos máximos y mínimos en un conjunto de datos de velas financieras, que son críticos para calcular los niveles de Fibonacci.

El proceso comienza con la función recibiendo un `DataFrame` de `pandas` (`candle_df`) que contiene datos de velas, y un diccionario (`max_min`) que se utiliza para almacenar los valores máximos y mínimos identificados. Además, se pasa un diccionario adicional (`data`) que contiene datos relacionados con el trading y el análisis técnico.

Dentro de la función, primero se extraen las series de los máximos y mínimos de las velas del `DataFrame`. Estos datos se convierten en arrays de

Numpy para facilitar los cálculos numéricos. La función luego identifica los valores máximos y mínimos actuales dentro de estos arrays.

Para encontrar el máximo y mínimo más significativos, la función parece utilizar una técnica para observar los valores dentro de un rango específico, excluyendo los últimos 'n' elementos. Esto podría ser para evitar la volatilidad reciente o los datos de ruido al final de la serie. Luego, identifica los índices donde se encuentran estos máximos y mínimos.

Una vez identificados los índices, la función accede al `DataFrame` original para obtener las marcas de tiempo correspondientes a estos máximos y mínimos. Esto es importante porque en el análisis técnico, no solo el valor de los máximos y mínimos es relevante, sino también el momento en que ocurrieron.

Finalmente, la función actualiza el diccionario `max_min` con los valores máximos y mínimos encontrados, junto con sus respectivas marcas de tiempo. Además, actualiza el diccionario `data` con esta información, posiblemente para su uso en otros cálculos o para el seguimiento en el sistema de trading.

ORDER

La clase presentada es parte de un sistema de trading automatizado, centrada en la gestión de órdenes de compra y venta en el mercado financiero. La funcionalidad principal de esta clase implica analizar datos de mercado en tiempo real y tomar decisiones de trading basadas en criterios predefinidos de análisis técnico. Esto incluye la evaluación de indicadores como RSI, SMA, y niveles de Fibonacci para identificar oportunidades de trading, ya sea para abrir posiciones de compra o venta. La clase utiliza estos análisis para establecer cuándo entrar en el mercado y cuándo cerrar posiciones, buscando maximizar los beneficios y minimizar las pérdidas.

Además, la clase incluye mecanismos para el seguimiento y la gestión de las operaciones de trading una vez que están abiertas. Esto implica calcular el beneficio o la pérdida potencial de cada operación y tomar decisiones sobre si mantener o cerrar la posición basándose en la evolución del mercado. Por ejemplo, se pueden establecer niveles de *take profit* y *stop loss*, que son umbrales críticos para cerrar una operación con beneficio o limitar las pérdidas, respectivamente. Estas funcionalidades son esenciales en el trading automatizado, ya que permiten al sistema operar de manera eficiente y efectiva, reaccionando dinámicamente a las condiciones cambiantes del mercado.

Importaciones

- **import src.VELAS_1:** Esta importación sugiere la inclusión de un módulo personalizado, posiblemente denominado `VELAS_1`, que se encuentra en el paquete o directorio `src`. Este módulo probablemente contiene funcionalidades relacionadas con el manejo y análisis de velas japonesas, que son esenciales en el análisis técnico de los mercados financieros.
- **import src.BotCSV:** Similar a `src.VELAS_1`, esta importación indica el uso de otro módulo personalizado dentro del paquete `src`, llamado `BotCSV`. Este módulo podría estar diseñado para manejar operaciones específicas relacionadas con archivos CSV, como la lectura y escritura de datos de trading en este formato.
- **import time:** Esta biblioteca se utiliza para manejar operaciones relacionadas con el tiempo, como pausas en la ejecución del programa.
- **import os:** El módulo `os` es fundamental para el trato con las estructuras del proyecto.
- **import logging:** El módulo `logging` es fundamental para manejar la salida de errores.
- **import traceback:** El módulo `traceback` es fundamental para la ayuda en la custión del tratamiento de excepciones.

Variables Globales

- **SIGNAL_RSI_BUY:** Señal para comprar de la herramienta RSI.
- **SIGNAL_RSI_SELL:** Señal para vender de la herramienta RSI.
- **CANDLES_BETWEEN_OPERATIONS:** Velas que se dejan para poder realizar una operación de nuevo.

Función `thread_order`

Inicialmente, el método prepara el entorno para registrar las operaciones. Esto se hace creando un archivo CSV en una ruta especificada, donde se almacenarán los detalles de las operaciones de trading. Utiliza la clase `BotCSV.BotDBCSV` para manejar este archivo CSV, lo que implica crearlo si no existe y prepararlo para registrar operaciones.

El método luego entra en un bucle que se mantiene activo hasta que se recibe una señal para detenerlo (indicada por el `stop_event`). Dentro de este bucle, la función realiza varias tareas clave. Primero, espera a que los indicadores técnicos, como SMA, Fibonacci y RSI, estén listos para su uso. Esto asegura que las decisiones de trading se basen en datos completos y actualizados.

Una vez que los indicadores están listos, el método procede a revisar regularmente las condiciones del mercado y decide si es el momento adecuado para abrir una operación de compra o venta. Esto se hace analizando los datos actuales de las velas y comparándolos con los criterios establecidos, como los niveles de RSI y las condiciones específicas relacionadas con SMA y Fibonacci. Si se cumplen las condiciones para una operación de compra o venta, se registra esta decisión y se añade a una lista de operaciones.

Además, el método gestiona las operaciones en curso, evaluando si han alcanzado su objetivo de beneficio (*take profit*) o han cruzado el límite de pérdida aceptable (*stop loss*). Esto se hace calculando los niveles de *take profit* y *stop loss* con base en el precio actual de las velas y el apalancamiento aplicado. Si una operación alcanza cualquiera de estos umbrales, se cierra y se registra el resultado en el archivo CSV. Este seguimiento continuo es crucial para asegurar que las operaciones sean rentables o para minimizar las pérdidas en caso de cambios adversos en el mercado.

Finalmente, el método `thread_orders` utiliza una barrera de sincronización para coordinar su operación con otros hilos. Esto asegura que las decisiones se tomen basadas en la información más actual y en armonía con el resto del sistema de trading.

Función `condiciones_operacion`

El método `condiciones_operacion` es una función clave en el sistema de trading automatizado, destinada a evaluar las condiciones del mercado y decidir si es apropiado abrir operaciones de compra o venta. Su lógica se basa en un análisis técnico que incluye la evaluación de varios indicadores y condiciones del mercado.

En primer lugar, el método verifica si el mercado se encuentra en una tendencia alcista o bajista, utilizando el indicador SMA (Media Móvil Simple). Esto se hace mediante la comprobación del valor de `data['SMA']`. Si el SMA indica una tendencia alcista ('Bull') o bajista ('Bear'), el método procede a evaluar condiciones más específicas para abrir una operación. En el caso de una tendencia alcista, el método busca una oportunidad de compra

(‘Comprar’), mientras que en una tendencia bajista busca una oportunidad de venta (‘Vender’).

Para una operación de compra, el método comprueba si el precio más bajo de la última vela está por debajo de un nivel específico de Fibonacci (`data[‘FIBO_38’][‘level’]`) y si el mercado no está en un rango. También evalúa si el precio de apertura de la última vela es mayor que el de cierre (lo que podría indicar una posible reversión de tendencia) y si el último valor de RSI es menor que el umbral establecido para una señal de compra (`SIGNAL_RSI_BUY`). Si se cumplen todas estas condiciones, el método registra una operación de compra, almacenando los detalles relevantes en `data[‘OPERATION’]`.

Para una operación de venta, se aplica una lógica similar pero invertida, adecuada para identificar oportunidades de venta en un mercado bajista. Se verifican condiciones como que el precio más alto de la última vela supere el nivel de Fibonacci correspondiente, la tendencia del mercado no esté en un rango, el precio de apertura sea menor que el de cierre, y el valor de RSI sea mayor que el umbral de venta (`SIGNAL_RSI_SELL`).

Función beneficio_perdida

El método `beneficio_perdida` verifica si hay operaciones activas en la lista `operation`. Si hay operaciones en curso, el método itera a través de ellas para evaluar su rendimiento. Para cada operación, se realiza una serie de cálculos para determinar los niveles de *stop loss* y *take profit*. Estos cálculos se basan en factores como la distancia entre el precio de entrada y los niveles de Fibonacci, y se ajustan por el apalancamiento utilizado en la operación. El objetivo es establecer un límite superior (*take profit*) donde la operación se cerrará para asegurar las ganancias, y un límite inferior (*stop loss*) para minimizar las pérdidas en caso de que el mercado se mueva en contra de la posición.

El método luego compara el precio actual del mercado, obtenido del DataFrame `candle_df`, con estos niveles de *take profit* y *stop loss*. Si el precio del mercado cruza el nivel de *take profit*, se asume que la operación ha alcanzado su objetivo de beneficio y se registra una ganancia. El beneficio se calcula y se suma a la variable `total`, que lleva un registro del rendimiento total del sistema de trading. La operación se registra en el archivo CSV a través de `db_csv`, una instancia de `BotCSV.BotDBCSV`, y luego se elimina de la lista de operaciones activas.

Por otro lado, si el precio del mercado alcanza o cruza el nivel de *stop loss*, se asume que la operación ha incurrido en una pérdida. La cantidad de pérdida se calcula y se resta del **total**. Al igual que con las operaciones rentables, las operaciones con pérdidas se registran en el archivo CSV y se eliminan de la lista de operaciones activas.

BOT

La clase Bot en el código es un componente central de un sistema de trading automatizado. Está diseñada para coordinar múltiples hilos de ejecución, cada uno encargado de una tarea específica relacionada con el trading, como el análisis de datos de mercado, el cálculo de indicadores técnicos y la gestión de órdenes de trading. La clase maneja la inicialización, ejecución y el cierre controlado de estos hilos, permitiendo así operar de forma concurrente y sincronizada. Con estructuras para almacenar datos de trading y herramientas de control de flujo como eventos y cerrojos, Bot funciona como el núcleo organizativo de un sistema de trading complejo, integrando diferentes aspectos del análisis y la ejecución de operaciones en un entorno automatizado.

Importaciones

- **import src.FIBO:** Este importe sugiere la inclusión de un módulo relacionado con el análisis de Fibonacci, probablemente proporcionando funciones o clases para calcular los niveles de Fibonacci, que son utilizados en el análisis técnico de los mercados financieros.
- **import src.SMA:** Importa un módulo relacionado con la Media Móvil Simple (SMA). Este módulo probablemente contiene funcionalidades para calcular la SMA, un indicador técnico común utilizado para analizar tendencias de mercado.
- **import src.VELAS:** Este importe indica la inclusión de un módulo que se ocupa de las velas japonesas, una herramienta esencial en el análisis técnico para visualizar y analizar movimientos de precios en los mercados financieros.
- **import src.RSI:** Importa un módulo que probablemente se ocupa del cálculo del Índice de Fuerza Relativa (RSI), otro indicador técnico clave utilizado para identificar condiciones de sobrecompra o sobreventa en un activo.
- **import threading:** La importación de `threading` sugiere que la clase Bot utiliza la programación concurrente. `threading` es una biblioteca de Python que permite la ejecución de múltiples hilos, lo cual es esencial en un sistema de trading automatizado para realizar varias tareas de análisis y trading simultáneamente.

- **import src.ORDER:** Este importe probablemente se refiere a un módulo que gestiona las órdenes de trading. Este módulo puede contener funcionalidades para ejecutar, monitorear y gestionar órdenes de compra o venta en el mercado.
- **import os:** El módulo **os** es fundamental para el trato con las estructuras del proyecto.
- **import logging:** El módulo **logging** es fundamental para manejar la salida de errores.

Función Constructora

La función constructora de la clase **Bot** se encarga de inicializar la instancia de esta clase, que es central en un sistema de trading automatizado. Esta función configura y prepara varios elementos clave que son esenciales para el funcionamiento del bot.

Al inicio, la función constructora crea una lista vacía denominada **threads**. Esta lista está destinada a almacenar los objetos de hilo que se crearán y ejecutarán posteriormente. Estos hilos son fundamentales para realizar tareas de análisis y operaciones de trading de manera concurrente, lo que es una característica crucial en un sistema de trading automatizado.

A continuación, se inicializa un diccionario llamado **data**, que actúa como un contenedor central para almacenar una variedad de datos relacionados con el trading y el análisis técnico. Este diccionario incluye listas vacías y valores booleanos para diversos indicadores y datos, como velas, RSI, SMA, soporte, resistencia y niveles de Fibonacci. La estructura y organización de **data** sugieren que el bot realizará un seguimiento y análisis continuo de estos elementos para tomar decisiones de trading.

La función también establece una lista para operaciones de trading (**operation**), un evento de **threading** llamado **pill2kill** que se utiliza para señalizar a los hilos cuando deben detener su ejecución, y un cerrojo (**mutex**). El **mutex** es importante para manejar el acceso concurrente a recursos compartidos, asegurando que no haya conflictos o condiciones de carrera cuando múltiples hilos intentan acceder o modificar los mismos datos.

Además, se crea una barrera de sincronización para coordinar la operación de los diferentes hilos, asegurando que ciertas tareas se realicen en un orden controlado y sincronizado. Por último, se inicializa un contador (**total**) para realizar un seguimiento de algún valor total, cuyo propósito específico no se detalla en el fragmento de código.

Función `thread_rsi`

Al ejecutarse, `thread_RSI` crea un objeto de hilo (`threading.Thread`) y le asigna una función objetivo, que en este caso es `src.RSI.thread_rsi`, importada desde un módulo externo. La función objetivo es probablemente la que lleva a cabo el cálculo real del RSI. Además, `thread_RSI` pasa varios argumentos a esta función objetivo, incluyendo el evento `pill2kill` para controlar la finalización del hilo, el diccionario `data` para almacenar y acceder a datos relevantes, y la barrera `barrera` para la sincronización con otros hilos.

Una vez configurado, el hilo se inicia con el método `start()`. Esto lanza la ejecución de la función de cálculo del RSI en un flujo de control independiente. La adición del hilo a la lista `threads` de la clase `Bot` permite un seguimiento y manejo centralizado de todos los hilos activos en el sistema.

Función `thread_candle`

Crea un hilo utilizando la clase `threading.Thread` de Python. Este hilo está configurado para ejecutar una función específica, posiblemente definida en el módulo `src.VELAS`, que maneja la recolección y el procesamiento de datos de las velas japonesas. Esta función de procesamiento de velas es responsable de recopilar datos del mercado en tiempo real y transformarlos en un formato que sea útil para el análisis técnico.

La función `thread_candle` también pasa argumentos importantes a la función del hilo, incluyendo el evento `pill2kill` para controlar la finalización del hilo, el diccionario `data` que almacena los datos recopilados, una barrera `barrera` para la sincronización con otros hilos, y un argumento adicional `fichero`, que podría ser utilizado para especificar una fuente de datos o configuraciones relacionadas.

Una vez que el hilo está configurado adecuadamente, se inicia con el método `start()`, lanzando así la tarea de procesamiento de datos de velas en un flujo de control separado. Esto permite que el sistema realice análisis en tiempo real y tome decisiones de trading basadas en los últimos datos del mercado sin interrumpir otras operaciones importantes.

Función `thread_sma`

Al ejecutarse, `thread_sma` crea un objeto de hilo utilizando la clase `threading.Thread` de Python. Este hilo se configura para ejecutar una función específica para el cálculo de la SMA, probablemente definida en

el módulo `src.SMA`. La función asignada como objetivo del hilo llevaría a cabo el proceso de cálculo de la SMA basándose en los datos del mercado disponibles.

Además, la función `thread_sma` pasa argumentos cruciales a la función del hilo, incluyendo el evento `pill2kill`, que sirve para señalizar el momento de terminar la ejecución del hilo, el diccionario `data`, que contiene los datos del mercado necesarios para calcular la SMA, y una barrera de sincronización (`barrera`). Esta barrera es esencial para coordinar la operación de este hilo con otros hilos que se ejecutan en paralelo, asegurando que todas las tareas se realicen de manera coherente y ordenada.

Tras configurar el hilo con los parámetros adecuados, `thread_sma` lo inicia con el método `start()`. Esto comienza la ejecución de la tarea de cálculo de la SMA en un flujo de control independiente. El hilo se agrega a la lista de `threads` de la clase `Bot` para un seguimiento y manejo centralizado.

Función `thread_fibonacci`

Cuando se invoca, `thread_fibonacci` crea un objeto de hilo usando `threading.Thread`, asignándole la tarea de ejecutar una función específica que probablemente se encargue del cálculo y análisis de Fibonacci. Esta función, posiblemente definida en el módulo `src.FIBO`, se enfocaría en determinar los puntos de soporte y resistencia significativos utilizando los principios de Fibonacci, que son cruciales para formular estrategias de trading.

La función pasa varios argumentos a este hilo, incluyendo el evento `pill2kill` para controlar cuándo debe detenerse el hilo, el diccionario `data` para almacenar y acceder a los datos relevantes, y una barrera `barrera` para coordinar las operaciones de este hilo con otros hilos que se ejecutan simultáneamente.

Después de configurar el hilo con los parámetros necesarios, `thread_fibonacci` lo inicia con el método `start()`, permitiendo que la tarea de análisis de Fibonacci se realice de manera independiente. Esto es importante para mantener el flujo continuo y actualizado del análisis técnico en el sistema sin interferir con otras funciones críticas.

Función `thread_order`

Cuando se invoca, `thread_orders` crea un hilo utilizando la clase `threading.Thread` de Python, asignándole la tarea de manejar las órdenes de trading. Este

hilo ejecuta una función específica, probablemente definida en el módulo `src.ORDER`, enfocada en la gestión efectiva de las órdenes de compra o venta basadas en las estrategias de trading y señales identificadas.

La función transfiere varios argumentos importantes al hilo, incluyendo el evento `pill2kill` para señalizar la terminación del hilo, el diccionario `data` que contiene información relevante para la toma de decisiones de trading, la barrera `barrera` para sincronizar este hilo con otros hilos concurrentes, y parámetros adicionales como `operation`, `total`, `lotaje` y `fichero`, que son necesarios para la ejecución y gestión de las órdenes.

Una vez que el hilo está configurado, se inicia con el método `start()`.

Función `kill_threads`

Cuando se llama a `kill_threads`, la función primero emite una señal a través del evento `pill2kill`, que es compartido por todos los hilos en ejecución. Este evento actúa como un interruptor para informar a los hilos que deben concluir sus operaciones y prepararse para terminar. Al activar `pill2kill.set()`, se indica a todos los hilos que deben detener su ejecución.

Posteriormente, `kill_threads` recorre la lista de hilos almacenada en la instancia de `Bot` y espera a que cada uno de ellos finalice su ejecución. Esto se realiza a través del método `join()`, que bloquea el hilo actual hasta que el hilo al que se ha unido (`join`) haya terminado. Esto garantiza que todos los hilos finalicen correctamente antes de que el programa principal continúe o el bot se cierre por completo.

Función `stop`

La función `wait` en la clase `Bot` está diseñada para proporcionar un mecanismo de control manual para detener la ejecución del bot de trading. Esta función juega un papel crucial en la interacción entre el usuario y el sistema, permitiendo al usuario decidir cuándo finalizar todas las operaciones y procesos del bot.

Cuando se invoca, `wait` muestra un mensaje en la consola que invita al usuario a presionar la tecla `ENTER` para detener el bot. Al hacerlo, la función captura esta entrada del usuario y, en respuesta, llama a la función `kill_threads`. Esto inicia el proceso de detener todos los hilos activos de manera segura y ordenada, como se describe en la función `kill_threads`. De esta manera, `wait` ofrece una interfaz simple pero efectiva para que los usuarios puedan finalizar la ejecución del bot cuando lo consideren necesario, garantizando que todos los procesos se cierren adecuadamente.

BOTCSV

La clase `BotDBCSV` está diseñada para facilitar la interacción con archivos CSV en el contexto de un sistema de trading automatizado. Su propósito principal es manejar el almacenamiento y registro de datos relacionados con las operaciones de trading. La clase actúa como una interfaz entre el sistema de trading y un archivo CSV, permitiendo que las operaciones de trading, como compras y ventas, junto con sus detalles relevantes, se registren de manera estructurada y accesible. Al mantener un registro de estas operaciones, la clase contribuye a la capacidad de análisis y seguimiento del desempeño del sistema de trading a lo largo del tiempo.

La clase `BotDBCSV` ofrece funcionalidades clave para la creación y actualización de un archivo CSV. Esto incluye la capacidad de crear un nuevo archivo CSV si no existe, con encabezados apropiados para organizar la información de trading. Además, proporciona una funcionalidad para añadir registros de operaciones individuales al archivo CSV. Cada entrada incluye detalles importantes como el tipo de operación, fechas de apertura y cierre, precios de apertura y cierre, y la cantidad de dinero ganado o perdido en la operación. Estas capacidades hacen de `BotDBCSV` una herramienta valiosa para mantener un registro detallado y organizado de las actividades de trading, facilitando la revisión y el análisis del rendimiento del sistema de trading.

Importaciones

- **import csv:** Importa el módulo `csv` de Python, que proporciona funcionalidades para leer y escribir archivos en formato CSV.
- **import os:** El módulo `os` es fundamental para el trato con las estructuras del proyecto.
- **import logging:** El módulo `logging` es fundamental para manejar la salida de errores.

Constructor

Cuando se crea una nueva instancia de `BotDBCSV`, el constructor recibe un parámetro: `csv_file`. Este parámetro es una cadena de texto (`str`) que especifica la ruta al archivo CSV que la clase va a manejar. El constructor utiliza este parámetro para establecer el atributo `csv_file` de la instancia, almacenando así la ruta del archivo CSV. Esta ruta se usará posteriormente para operaciones de lectura y escritura en el archivo CSV correspondiente.

Método `create_csv`

Cuando se invoca el método `create_csv`, el código intenta abrir el archivo CSV especificado por la ruta almacenada en el atributo `csv_file` de la instancia. Utiliza el modo 'w' (escritura) para abrir el archivo, lo que significa que si el archivo ya existe, su contenido será sobrescrito. Sin embargo, el principal uso de este método es crear el archivo si no existe.

Dentro del contexto de apertura del archivo, el método utiliza la función `writer` del módulo `csv` para crear un objeto `writer`, que se utiliza para escribir en el archivo CSV. A continuación, el método escribe una fila de encabezados en el archivo CSV, definiendo así la estructura de los datos que se almacenarán. Estos encabezados incluyen campos como '`tipo_operacion`', '`fecha_apertura`', '`fecha_cierre`', '`precio_apertura`', '`precio_cierre`' y '`dinero_ganado_perdido`', que son esenciales para registrar de manera clara y ordenada las operaciones de trading.

Método `insert_operation`

Cuando se invoca el método `insert_operation`, recibe varios parámetros que describen los detalles de una operación de trading. Estos incluyen el tipo de operación (como 'Compra' o 'Venta'), las fechas de apertura y cierre de la operación, los precios de apertura y cierre, y la cantidad de dinero ganado o perdido en la operación. Estos parámetros son esenciales para proporcionar una visión completa de cada operación realizada por el sistema de trading.

Dentro del método, se abre el archivo CSV especificado por la ruta almacenada en `csv_file`, usando el modo 'a' (append). Esto permite agregar nuevas entradas al archivo sin sobrescribir el contenido existente. El método entonces utiliza el objeto `writer` de la biblioteca `csv` para escribir una nueva fila en el archivo CSV. Esta fila contiene los detalles de la operación de trading proporcionados en los parámetros, organizados en el mismo orden en que se especificaron los encabezados en el método `create_csv`.

Este proceso de registro es fundamental para mantener un historial de todas las operaciones realizadas por el sistema de trading.

Interfaz

Este script proporciona una interfaz de usuario para configurar y ejecutar un bot de trading. El usuario puede seleccionar el lotaje y el archivo CSV histórico a utilizar para el bot. El bot se inicia cuando se presiona el botón *Iniciar Bot*, siempre que se haya seleccionado un lotaje válido y un archivo CSV histórico.

El bot realiza varias operaciones basadas en el análisis técnico de los datos históricos, como el cálculo de velas, el índice de fuerza relativa (RSI), medias móviles simples (SMA) y niveles de Fibonacci. El resultado de las operaciones se guarda en un archivo de registro y se muestra al usuario.

Importaciones

- **import subprocess:** Importa el módulo `subprocess` de Python para ejecutar nuevos procesos.
- **import tkinter as tk:** Importa la biblioteca `tkinter` para crear interfaces gráficas de usuario.
- **from tkinter import ttk, messagebox:** Importa módulos específicos de `tkinter` para el uso de `ttk` (widgets temáticos) y `messagebox` (cuadros de mensaje).
- **import os:** Importa el módulo `os` para interactuar con el sistema operativo.
- **import traceback:** Importa el módulo `traceback` para el manejo de excepciones y trazado de errores.
- **import logging:** Importa el módulo `logging` para el registro de eventos y errores.
- **from src import BOT:** Importa el módulo personalizado `BOT` desde el paquete `src`, que contiene la lógica principal del bot de trading.

Constructor

La función constructora de la clase `TradingBotUI` tiene como propósito inicializar la interfaz de usuario del bot de trading. Para empezar, se recibe un parámetro llamado `root`, que representa la ventana principal de la interfaz de usuario, y se guarda esta referencia en el atributo `self.root` de

la instancia. El título de la ventana se establece en *Configuración del Bot de Trading*" para que el usuario sepa de inmediato el propósito de la aplicación.

A continuación, se crea una etiqueta (**Label**) de **tkinter** con el texto "*Lotaje:*" y se posiciona en la primera fila, primera columna de un grid layout. Junto a esta etiqueta, se crea un cuadro de entrada (**Entry**) de **tkinter** para que el usuario ingrese el valor del lotaje, posicionándolo en la primera fila, segunda columna del grid layout.

Además, se añade una segunda etiqueta (**Label**) con el texto "*Seleccionar Historico:*", ubicada en la segunda fila, primera columna del grid layout. Para permitir la selección del archivo histórico, se crea un menú desplegable (**Combobox**) de **ttk**, configurado en estado de solo lectura y con un ancho de 30 caracteres. Este menú desplegable se coloca en la segunda fila, segunda columna del grid layout.

Posteriormente, se llama al método **select_historico_files** para poblar el **Combobox** con los archivos CSV históricos disponibles. Este método busca en una carpeta específica todos los archivos con extensión **.csv** y los añade al menú desplegable. Finalmente, se crea un botón (**Button**) de **tkinter** con el texto *Íniciar Bot*" que, al ser presionado, llama al método **start_bot**. Este botón se posiciona en la tercera fila, ocupando ambas columnas y centrado horizontalmente con un margen vertical.

Método **select_historico_files**

El método **select_historico_files** es responsable de buscar y listar los archivos históricos disponibles para que el usuario pueda seleccionar uno. Primero, se construye la ruta a la carpeta *HistóricosDivisas*, que se encuentra dos niveles por encima del directorio actual del script. Luego, utiliza **os.listdir** para listar todos los archivos en esta carpeta y filtra esta lista para incluir solo aquellos con la extensión **.csv**. La lista de archivos CSV filtrada se asigna al atributo **values** del **Combobox historico_combobox**, y se establece el texto del **Combobox** a "*Seleccionar archivo*" para indicar al usuario que debe hacer una selección. En caso de que ocurra algún error durante este proceso, se muestra un mensaje de error al usuario mediante **messagebox.showerror** y se imprime la traza del error con **traceback.print_exc()**.

Método start_bot

El método valida la configuración proporcionada por el usuario y, si es correcta, inicia el bot de trading. Primero, se leen los valores ingresados en lotaje y histórico. Si no se ha ingresado un lotaje o no se ha seleccionado un archivo histórico, se muestra una advertencia al usuario solicitando que complete estos campos. Si ambos campos están completos, se intenta convertir el lotaje a un número decimal y se construye la ruta completa al archivo histórico seleccionado. Si esta ruta es válida, se llama al método `run_bot` con el lotaje, la ruta del archivo histórico y el archivo seleccionado. Si la conversión del lotaje falla, se muestra una advertencia indicando que el lotaje debe ser un número entero o decimal. Cualquier otro error se muestra mediante un mensaje de error y se imprime la traza del error.

Método run_bot

Finalmente, el método `run_bot` ejecuta el bot de trading con la configuración especificada. Imprime un mensaje indicando que el bot se está iniciando con el lotaje y los datos históricos seleccionados. Luego, crea una instancia del bot (`Bot1`) y lanza los hilos correspondientes para el procesamiento de velas, RSI, SMA, Fibonacci y órdenes de trading. Una vez que el bot ha sido configurado y lanzado, se intenta abrir el archivo de operaciones resultante en *Notepad* para que el usuario pueda revisar las operaciones realizadas por el bot. Si ocurre algún error durante la ejecución del bot, se muestra un mensaje de error y se imprime la traza del error.

Main

Importaciones

- **import sys:** Importa el módulo `subprocess` de Python para ejecutar nuevos procesos.
- **import os:** Importa el módulo `Interfaz`, que contiene la definición de la interfaz de usuario del bot de trading.
- **import tkinter as tk:** Importa la biblioteca `tkinter` para crear interfaces gráficas de usuario.
- **Interfaz:** Importa el módulo personalizado `BOT` desde el paquete `src`, que contiene la lógica principal del bot de trading.

Configuración del Path

El código añade el directorio principal del proyecto al `sys.path` para asegurar que los módulos personalizados pueden ser importados correctamente. Esto se logra construyendo la ruta absoluta al directorio padre del directorio actual del script y añadiéndola a `sys.path`. Esto permite que los módulos en el directorio `src` y otros directorios de nivel superior sean accesibles para importar en el script actual.

Punto de Entrada de la Aplicación

El script define el punto de entrada principal de la aplicación utilizando una condición que verifica si el módulo se está ejecutando como el programa principal. Esto se hace mediante la condición `if __name__ == "__main__"`.

5.2. Análisis de resultados

En este apartado se expondrán los resultados obtenidos durante la ejecución del Bot de Trading con tres ejemplos, en diferentes temporalidades para observar en qué entorno tiene una mejor rentabilidad. El periodo de prueba abarca de 2004 al 2024:

- EURUSD_Daily: Observaremos operaciones a largo plazo, las cuales pueden durar meses. En este entorno, el Bot puede funcionar mejor debido a que las tendencias se hacen más visibles y el mercado está más limpio.
- GBPUSD_H4: Observaremos el Bot en un entorno de medio plazo.
- USDJPY_H1: Observaremos el Bot en un entorno de corto plazo donde las operaciones pueden durar un día.

EURUSD_Daily

tipo_operacion,fecha_apertura,fecha_cierre,precio_apertura,precio_cierre,dinero_ganado_perdido
BUY,2005-01-19 00:00:00,2005-05-13 00:00:00,1.3026712,1.2616658712000002,--20.50266439999993
BUY,2005-01-20 00:00:00,2005-05-13 00:00:00,1.3026712,1.2616658712000002,--20.50266439999993
BUY,2006-10-10 00:00:00,2007-04-05 00:00:00,1.2534464,1.3430928000000002,44.82320000000064
BUY,2006-10-13 00:00:00,2007-04-06 00:00:00,1.2534464,1.3430928000000002,44.82320000000064
BUY,2007-01-16 00:00:00,2007-04-13 00:00:00,1.3027458,1.3519916,24.62290000000005
BUY,2007-08-20 00:00:00,2007-09-20 00:00:00,1.3485808000000001,1.4076616000000002,29.540400000000023
BUY,2007-08-21 00:00:00,2007-09-20 00:00:00,1.3485808000000001,1.4076616000000002,29.540400000000023
BUY,2007-12-19 00:00:00,2008-03-06 00:00:00,1.441973999999999,1.5303479999999998,44.18699999999976
BUY,2007-12-20 00:00:00,2008-03-07 00:00:00,1.441973999999999,1.5303479999999998,44.18699999999976
BUY,2008-05-07 00:00:00,2008-08-12 00:00:00,1.5368982,1.4953049982,--20.79660090000002
BUY,2011-05-13 00:00:00,2011-09-09 00:00:00,1.41497082,1.36487294082,--25.048939589999986
BUY,2013-02-28 00:00:00,2013-03-01 00:00:00,1.33097292,1.3049008729199998,--13.036023540000109
BUY,2013-03-01 00:00:00,2013-03-01 00:00:00,1.33097292,1.3049008729199998,--13.036023540000109
BUY,2014-01-22 00:00:00,2014-01-22 00:00:00,1.36890502,1.35494867502,--6.9781724899999
BUY,2014-05-20 00:00:00,2014-05-21 00:00:00,1.37960116,1.36605142116,--6.774869420000051
BUY,2014-05-21 00:00:00,2014-05-22 00:00:00,1.37960116,1.36605142116,--6.774869420000051
BUY,2013-11-08 00:00:00,2014-08-20 00:00:00,1.3507267,1.329772196999998,--10.477251650000131
BUY,2016-05-27 00:00:00,2016-06-03 00:00:00,1.12954402,1.10863623402,--10.453892989999991

Figura 5.1: EURUSD_DAILY

En la figura 5.1 se observa una tabla en formato CSV, la cual crea el Bot nada más acabar la ejecución, mostrando las operaciones realizadas durante el periodo de tiempo elegido.

Análisis de los resultados de las operaciones:

- Total Ganado: 261.72
- Total Perdido: -154.38
- Beneficio Neto (Rentabilidad): 107.34
- Número de Operaciones Ganadoras: 7
- Número de Operaciones Perdedoras: 11
- Ganancia Media por Operación Ganadora: 37.39
- Pérdida Media por Operación Perdedora: -14.03

Análisis:

- A pesar de que hubo más operaciones perdedoras (11) que ganadoras (7), el total ganado superó al total perdido, resultando en un beneficio neto positivo de 107.34.

- La ganancia media por operación ganadora es significativamente más alta (37.39) comparada con la pérdida media por operación perdedora (-14.03). Esto indica que, aunque las pérdidas fueron más frecuentes, las ganancias de las operaciones exitosas fueron suficientemente altas para compensar y resultar en un beneficio neto global positivo.
- La rentabilidad total del conjunto de operaciones es positiva, lo que sugiere que la estrategia, aunque arriesgada (más pérdidas que ganancias en términos de cantidad de operaciones), puede considerarse exitosa en términos financieros hasta la fecha de los datos proporcionados.

La estrategia en este entorno se observa que funciona, a pesar de que a veces el bot realiza varias operaciones en un mismo punto lo cuál no es un fallo pero habría que corregirlo en futuras versiones.

GBPUSD_H4

tipo_operacion,fecha_apertura,fecha_cierre,precio_apertura,precio_cierre,dinero_ganado_perdido
BUY,2004-02-23 00:00:00,2004-03-02 16:00:00,1.8728148,1.8453380148,-13.738392599999983
BUY,2004-06-14 00:00:00,2004-08-24 16:00:00,1.8140178,1.7904899178,-11.76394110000001
BUY,2004-06-14 08:00:00,2004-08-24 16:00:00,1.8140178,1.7904899178,-11.76394110000001
BUY,2004-06-14 04:00:00,2004-08-24 20:00:00,1.8140178,1.7904899178,-11.76394110000001
BUY,2005-04-27 16:00:00,2005-05-09 00:00:00,1.9018686,1.8875419686,-7.163315700000061
BUY,2005-09-12 20:00:00,2005-09-15 12:00:00,1.8241712,1.8062207712,-8.975214400000043
BUY,2005-08-19 00:00:00,2005-09-26 04:00:00,1.7916038,1.7735209038,-9.041448100000071
BUY,2005-12-21 08:00:00,2005-12-21 12:00:00,1.7595934,1.7445602934,-7.516553300000007
BUY,2006-01-30 00:00:00,2006-01-30 08:00:00,1.778038,1.766593638,-5.722180999999993
BUY,2006-11-15 00:00:00,2006-11-15 12:00:00,1.898027,1.8839871270000002,-7.01993649999988
BUY,2006-12-18 12:00:00,2007-03-05 12:00:00,1.9459798,1.9201980798,-12.890860099999957
SELL,2007-03-16 08:00:00,2007-03-16 08:00:00,1.9485584,1.9389786416,-4.7898791999998
BUY,2007-04-06 20:00:00,2007-04-09 12:00:00,1.9716422,1.9630927422,-4.274728900000024
BUY,2007-04-27 04:00:00,2007-05-15 12:00:00,1.9967384,1.9748847384,-7.926830800000007
BUY,2007-07-27 16:00:00,2007-08-14 12:00:00,2.0271146,2.0015818146,-12.766392699999907
BUY,2007-11-12 08:00:00,2007-11-13 00:00:00,2.0815672,2.0581725672,-11.697316399999824
BUY,2008-03-17 20:00:00,2008-04-01 16:00:00,1.9997539999999998,1.9734705539999997,-13.14172300000005
SELL,2009-03-16 16:00:00,2009-03-16 16:00:00,1.41043512,1.3946381248800002,-7.89849755999994
SELL,2009-03-17 00:00:00,2009-03-17 00:00:00,1.41043512,1.3946381248800002,-7.89849755999994
BUY,2009-04-20 16:00:00,2009-04-22 12:00:00,1.46823624,1.4449173962400002,-11.65942187999991
BUY,2009-08-10 04:00:00,2009-08-10 12:00:00,1.6759668,1.657174366799999,-9.396216600000118
SELL,2009-09-07 08:00:00,2009-09-07 08:00:00,1.6428798,1.6324510202,-5.214389900000005
BUY,2009-11-20 00:00:00,2009-11-27 00:00:00,1.6621857599999998,1.64488770576,-8.64902711999993
BUY,2011-01-26 00:00:00,2011-02-03 08:00:00,1.5808636,1.6272672000000001,23.20180000000005
BUY,2011-03-08 00:00:00,2011-03-08 12:00:00,1.6223625000000002,1.6133725125,-4.494993750000065

Figura 5.2: GBPUSD_H4 - 1

En la figura 5.2 se observan las operaciones realizadas hasta el año 2011.

En la figura 5.3 se observan las operaciones realizadas hasta el año 2016.

```

BUY,2011-02-11 20:00:00,2011-03-22 08:00:00,1.6082796799999999,1.6396493599999997,15.684839999999923
BUY,2011-05-03 20:00:00,2011-05-05 20:00:00,1.6518978,1.6367837778,--7.557011099999955
BUY,2011-05-04 00:00:00,2011-05-06 08:00:00,1.6518978,1.6367837778,--7.557011099999955
BUY,2011-11-10 04:00:00,2011-11-16 00:00:00,1.59289026,1.57678138026,--8.054439869999985
BUY,2012-03-05 08:00:00,2012-03-06 12:00:00,1.58598504,1.57631706504,--4.833987479999968
BUY,2012-02-14 12:00:00,2012-04-24 08:00:00,1.56957114,1.6155622800000002,22.9955700000000075
BUY,2012-08-31 04:00:00,2012-09-12 08:00:00,1.57834446,1.6108789200000002,16.267230000000008
BUY,2012-12-26 16:00:00,2012-12-27 16:00:00,1.6189780999999999,1.6101801081,--4.398995949999929
BUY,2013-04-17 20:00:00,2013-04-29 00:00:00,1.52668658,1.55005316,11.683290000000014
BUY,2013-04-18 00:00:00,2013-04-29 00:00:00,1.52668658,1.55005316,11.683290000000014
BUY,2013-05-10 04:00:00,2013-05-10 16:00:00,1.54466518,1.53356971518,--5.54773240999995
BUY,2013-05-10 08:00:00,2013-05-10 16:00:00,1.54466518,1.53356971518,--5.54773240999995
SELL,2013-07-18 20:00:00,2013-07-18 20:00:00,1.52235284,1.50819384716,-7.079496419999942
BUY,2013-08-30 08:00:00,2013-09-13 12:00:00,1.54822228,1.58620456,18.99114000000002
BUY,2015-03-03 08:00:00,2015-03-04 16:00:00,1.53936812,1.5280822681200001,--5.642925939999933
BUY,2015-05-04 08:00:00,2015-05-13 16:00:00,1.51558864,1.5746772800000002,29.544320000000007
BUY,2015-05-04 16:00:00,2015-05-13 16:00:00,1.51558864,1.5746772800000002,29.544320000000007
BUY,2015-05-05 04:00:00,2015-05-13 16:00:00,1.51558864,1.5746772800000002,29.544320000000007
BUY,2015-05-04 12:00:00,2015-05-13 20:00:00,1.51558864,1.5746772800000002,29.544320000000007
BUY,2015-05-04 20:00:00,2015-05-14 00:00:00,1.51558864,1.5746772800000002,29.544320000000007
BUY,2015-05-20 04:00:00,2015-05-27 12:00:00,1.5537332000000001,1.5351101532,--9.311523400000077
BUY,2016-05-06 12:00:00,2016-06-10 20:00:00,1.44622644,1.4290830464400002,--8.571696779999916

```

Figura 5.3: GBPUSD_H4 - 2

Análisis de los resultados de las operaciones:

- Total Ganado: 268.23
- Total Perdido: -291.27
- Beneficio Neto (Rentabilidad): -23.04
- Número de Operaciones Ganadoras: 12
- Número de Operaciones Perdedoras: 35
- Ganancia Media por Operación Ganadora: 22.35
- Pérdida Media por Operación Perdedora: -8.32

Análisis:

- Aunque el total ganado por las operaciones ganadoras es sustancial, el total perdido debido a las operaciones perdedoras es mayor, lo que resulta en una rentabilidad negativa de -23.04.

- Hay una mayor cantidad de operaciones perdedoras (35) en comparación con las ganadoras (12), lo cual es una señal de que la estrategia puede necesitar revisión o ajuste.
- La ganancia media por operación ganadora es significativamente más alta (22.35) comparada con la pérdida media por operación perdedora (-8.32). Esto indica que las operaciones ganadoras son bastante rentables en comparación con las pérdidas de las operaciones que no tienen éxito.

Este análisis sugiere que, aunque las operaciones exitosas son bastante rentables, la alta frecuencia de operaciones perdedoras está afectando negativamente la rentabilidad general.

USDJPY_H4

tipo_operacion	fecha_apertura	fecha_cierre	precio_apertura	precio_cierre	dinero_ganado_perdido
BUY	2008-06-02 13:00:00	2008-06-02 18:00:00	104.87298	104.15276298	--360.10850999999633
BUY	2008-06-02 15:00:00	2008-06-02 18:00:00	104.87298	104.15276298	--360.10850999999633
BUY	2008-06-02 14:00:00	2008-06-02 19:00:00	104.87298	104.15276298	--360.10850999999633
BUY	2008-07-24 21:00:00	2008-07-25 08:00:00	107.23656	106.67665656	--279.9517199999997
BUY	2008-07-28 21:00:00	2008-07-29 11:00:00	107.44756	107.7151199999998	133.779999999946
BUY	2008-08-12 21:00:00	2008-08-13 04:00:00	109.35714	108.6041471400001	--376.496429999996
BUY	2008-11-05 13:00:00	2008-11-05 22:00:00	98.9470600000001	97.86497706	--541.0414700000033
SELL	2008-11-14 03:00:00	2008-11-14 03:00:00	96.83076	96.0251692400001	-402.7953799999935
BUY	2008-12-30 15:00:00	2008-12-30 17:00:00	90.50542	90.1343554199999	--185.53229000000474
BUY	2008-12-29 14:00:00	2009-01-05 13:00:00	90.31914	93.5082800000001	1594.5700000000045
BUY	2009-01-07 16:00:00	2009-01-08 08:00:00	92.8566199999999	91.68160662	--587.5066899999979
BUY	2009-01-07 18:00:00	2009-01-08 08:00:00	92.8566199999999	91.68160662	--587.5066899999979
BUY	2009-01-07 17:00:00	2009-01-08 09:00:00	92.8566199999999	91.68160662	--587.5066899999979
BUY	2009-03-27 11:00:00	2009-03-27 12:00:00	98.14166	97.60263966	--269.51017000000377
BUY	2009-03-27 14:00:00	2009-03-30 06:00:00	98.14166	97.60263966	--269.51017000000377
BUY	2009-05-06 11:00:00	2009-05-11 15:00:00	98.33241	97.46902341	--431.6932949999952
BUY	2009-05-29 14:00:00	2009-05-29 16:00:00	96.20388	95.48054388	--361.668059999995
BUY	2009-06-09 18:00:00	2009-06-16 05:00:00	97.51772	96.5855977200001	--466.06113999999366
BUY	2009-06-09 20:00:00	2009-06-16 06:00:00	97.51772	96.5855977200001	--466.06113999999366
BUY	2009-07-21 08:00:00	2009-07-21 17:00:00	94.20554	93.7506155400001	--227.46222999999333
BUY	2009-08-11 15:00:00	2009-08-11 16:00:00	96.64018	95.8338901800001	--403.14490999999464
BUY	2009-10-20 06:00:00	2009-11-02 00:00:00	90.36264	89.68758264	--337.5286799999973
BUY	2009-12-08 04:00:00	2009-12-09 08:00:00	89.133264	88.042605264	--545.3293679999973
BUY	2009-12-18 04:00:00	2009-12-21 18:00:00	89.578428	90.997856	709.7139999999982
BUY	2010-02-22 14:00:00	2010-02-23 16:00:00	91.21083	90.5455338300001	--332.6480849999953

Figura 5.4: USDJPY_H1-1

En la figura 5.4 se observan las operaciones realizadas hasta el año 2010.

SELL,2010-05-10 15:00:00,2010-05-10 15:00:00,92.409432,90.913390568,-748.0207159999992
SELL,2010-05-10 16:00:00,2010-05-10 16:00:00,92.409432,90.913390568,-748.0207159999992
SELL,2010-05-10 20:00:00,2010-05-10 20:00:00,92.409432,90.913390568,-748.0207159999992
BUY,2010-06-04 16:00:00,2010-06-07 02:00:00,91.966142,91.327439142,--319.351429000001
BUY,2010-06-04 18:00:00,2010-06-07 02:00:00,91.966142,91.327439142,--319.351429000001
BUY,2010-06-04 17:00:00,2010-06-07 03:00:00,91.966142,91.327439142,--319.351429000001
SELL,2010-08-05 01:00:00,2010-08-05 01:00:00,86.340228,86.03772377199999,-151.25211400000182
SELL,2010-08-05 01:00:00,2010-08-05 01:00:00,86.340228,86.03772377199999,-151.25211400000182
BUY,2011-01-10 21:00:00,2011-01-19 02:00:00,82.884788,82.314814788,--284.9866059999968
BUY,2011-01-10 22:00:00,2011-01-19 03:00:00,82.884788,82.314814788,--284.9866059999968
SELL,2011-03-18 15:00:00,2011-03-18 15:00:00,80.657694,79.112339306,-772.6773470000055
BUY,2011-08-08 02:00:00,2011-08-08 11:00:00,78.724448,77.717756448,--503.34577599999625
BUY,2011-09-06 10:00:00,2011-09-12 07:00:00,77.245144,76.953825144,--145.65942799999476
BUY,2011-11-02 15:00:00,2011-11-14 09:00:00,78.012954,77.00296595399999,--504.9940230000019
BUY,2011-11-29 13:00:00,2011-11-30 14:00:00,77.787006,77.410506006,--188.2499969999998
BUY,2012-01-26 10:00:00,2012-01-26 16:00:00,77.734942,77.322657942,--206.14202899999867
BUY,2012-03-05 12:00:00,2012-03-06 17:00:00,81.1563,80.6389803,--258.6598500000008
BUY,2012-06-25 11:00:00,2012-06-26 10:00:00,79.91593999999999,79.40457594,--255.68202999999556
BUY,2012-10-04 21:00:00,2012-10-08 11:00:00,78.419444,78.162315444,--128.56427799999892
SELL,2014-01-30 17:00:00,2014-01-30 17:00:00,102.8301820000001,102.555270818,-137.45559100000548
BUY,2014-03-11 22:00:00,2014-03-13 09:00:00,103.009134,102.443320134,--282.9069329999996
BUY,2014-03-11 23:00:00,2014-03-13 15:00:00,103.009134,102.443320134,--282.9069329999996
BUY,2014-08-01 22:00:00,2014-08-06 18:00:00,102.537478,102.135284478,--201.09676099999518
BUY,2014-10-27 17:00:00,2014-10-30 06:00:00,107.684994,109.124988,719.9969999999993
BUY,2014-10-27 19:00:00,2014-10-30 06:00:00,107.684994,109.124988,719.9969999999993
BUY,2014-11-21 04:00:00,2014-12-16 10:00:00,117.627832,116.67937183200002,--474.23008399999134

Figura 5.5: USDJPY_H1-2

En la figura 5.5 se observan las operaciones realizadas hasta el año 2016.

Análisis de los resultados de las operaciones:

- Total Ganado: 3,878.06
- Total Perdido: -17,156.45
- Beneficio Neto (Rentabilidad): -13,278.39
- Número de Operaciones Ganadoras: 5
- Número de Operaciones Perdedoras: 46
- Ganancia Media por Operación Ganadora: 775.61
- Pérdida Media por Operación Perdedora: -372.97

Ánalisis:

- El total perdido supera significativamente el total ganado, resultando en una rentabilidad neta negativa bastante alta de -13,278.39.

- A pesar de que las operaciones ganadoras tienen una ganancia media bastante alta, el número de operaciones perdedoras es mucho mayor y sus pérdidas medias son considerables.
- Este patrón sugiere que la estrategia de trading utilizada es altamente riesgosa o ineficaz, dado que las pérdidas dominan los resultados generales.

Este análisis resalta la necesidad de revisar y posiblemente ajustar la estrategia de trading para mejorar la rentabilidad general y reducir las pérdidas.

Conclusión

Los análisis detallados en las secciones anteriores revelan resultados variados para el bot de trading en las diferentes temporalidades, es decir, diaria (EURUSD_Daily), de cuatro horas (GBPUSD_H4) y de una hora (USDJPY_H1). En cada temporalidad, el bot ha demostrado diferentes grados de éxito y desafíos, que reflejan tanto la efectividad de la estrategia de trading como las áreas potenciales para mejorar.

En la temporalidad diaria (EURUSD_Daily), aunque el número de operaciones perdedoras superó al de las ganadoras, el bot logró un beneficio neto positivo gracias a que las ganancias de las operaciones exitosas compensaron las pérdidas. Esto sugiere que, a pesar de los riesgos, la estrategia puede ser viable a largo plazo si se realizan ajustes para mejorar la frecuencia de operaciones exitosas.

Por otro lado, en la temporalidad de cuatro horas (GBPUSD_H4), el bot mostró una pérdida neta debido a un número significativamente mayor de operaciones perdedoras en comparación con las ganadoras. Aunque las ganancias por operación ganadora fueron razonables, no fueron suficientes para cubrir las frecuentes y considerables pérdidas. Esto indica que la estrategia en esta temporalidad requiere una revisión sustancial para mejorar su efectividad y rentabilidad.

Finalmente, en la temporalidad de una hora (USDJPY_H1), el rendimiento del bot fue claramente insatisfactorio, con pérdidas netas significativas y un alto número de operaciones perdedoras. La estrategia en esta temporalidad resultó ser altamente ineficaz y arriesgada, dominada por pérdidas que no solo eran frecuentes sino también cuantiosas.

6. Trabajos relacionados

Trading automático

El trading automático es la técnica que utilizaremos en nuestro trabajo y se basa en programar estrategias de trading para que se ejecuten automáticamente cuando se hayan confirmado los criterios de la estrategia.

El programa realiza las operaciones automáticamente, sin necesidad de estar presentes en la compra o venta de acciones durante las 24 horas del mercado financiero. Esta herramienta es utilizada sobre todo por los traders institucionales para manejar un gran número de operaciones a la vez sin ninguna dificultad.

Esta herramienta también puede ser muy valiosa para los inversores particulares, ya que les permite enfocarse en diversas actividades sin necesidad de estar pendientes, y a la vez generar rentabilidad a través de este sistema automático. Esto evita que los inversores particulares tengan que ajustar manualmente su estrategia conforme fluctúa el mercado, ya que el programa lo hará por ellos. [44]

Quants

Para desarrollar sistemas de trading automáticos es necesario entender el trading algorítmico, que es la disciplina encargada de crear estrategias mediante modelos matemáticos y estadísticos para identificar oportunidades de inversión en los mercados financieros. Los profesionales que realizan este tipo de estrategias se llaman *quants*, y utilizan modelos matemáticos, estadísticos y técnicas de programación para elaborar oportunidades de trading, gestionando adecuadamente el riesgo y el beneficio.

Actualidad de los bots de trading

La situación actual de los bots de trading es muy importante para comprender cómo se va a desarrollar el nuestro. En la actualidad, los bots de trading han aumentado en popularidad debido a los avances tecnológicos en el campo de la inteligencia artificial y el aprendizaje automático. Estos bots son programas informáticos que utilizan algoritmos y datos históricos para realizar operaciones financieras automáticamente.

Los bots de trading se utilizan principalmente en el mercado de divisas, aunque también se pueden utilizar en otros mercados financieros, como el mercado de valores y el mercado de criptomonedas. Los bots de trading pueden ser programados para realizar diversas estrategias de trading, desde operaciones simples de compra y venta hasta estrategias más complejas basadas en el análisis técnico y fundamental.

Sin embargo, los bots de trading también tienen algunas desventajas y riesgos asociados. Por ejemplo, los bots de trading pueden ser programados incorrectamente, lo que puede llevar a pérdidas significativas. Además, los bots no pueden reaccionar a noticias imprevistas o eventos del mercado que pueden tener un impacto en las operaciones financieras.

En general, la situación actual de los bots de trading es que se están utilizando cada vez más en los mercados financieros debido a su capacidad para realizar operaciones automáticamente y con mayor eficiencia. A medida que avanza la tecnología, es probable que los bots de trading se vuelvan aún más sofisticados y precisos, pero es importante tener en cuenta que siempre existen riesgos y desafíos asociados con el uso de estas herramientas.

En la actualidad, los bots de trading están evolucionando y mejorando constantemente gracias a los avances en la tecnología y al aumento de la demanda de herramientas automatizadas en los mercados financieros. Algunas de las mejoras más importantes que se están implementando en los bots de trading incluyen:

- **Inteligencia artificial y aprendizaje automático:** Los bots de trading están utilizando cada vez más técnicas de inteligencia artificial y aprendizaje automático para mejorar su capacidad de análisis y toma de decisiones. Esto les permite adaptarse a las condiciones cambiantes del mercado y mejorar su precisión en la predicción de tendencias y oportunidades de trading.
- **Personalización y configuración avanzada:** Los bots están ofreciendo cada vez más opciones de personalización y configuración avan-

zada para adaptarse a las necesidades y estrategias específicas de los usuarios. Esto permite a los usuarios ajustar el bot a sus necesidades y mejorar su efectividad en el trading.

- **Integración con múltiples plataformas:** Se están integrando cada vez más con múltiples plataformas de trading, lo que les permite operar en varios mercados y aumentar las oportunidades de trading.
- **Mejoras en la interfaz de usuario:** Se están mejorando las interfaces de usuario de los bots de trading para hacerlas más fáciles de usar y más accesibles para los traders, independientemente de su nivel de experiencia.

Como modelo de lenguaje, no se puede afirmar que un bot en particular sea el "mejor." en el mercado, ya que hay muchos factores que pueden influir en la efectividad de un bot de trading, como el tipo de mercado, la estrategia de trading utilizada, el nivel de personalización y configuración del bot, entre otros. [36]

3commas

3Commas es un bot de trading de criptomonedas basado en la nube que permite a los usuarios automatizar y personalizar sus operaciones en los mercados de criptomonedas. Fundado en 2017, 3Commas es una plataforma popular entre los traders de criptomonedas debido a su amplia gama de herramientas de trading automatizado, así como su interfaz fácil de usar.

Los usuarios de 3Commas pueden conectar sus cuentas de intercambio de criptomonedas a la plataforma y configurar estrategias de trading automatizadas utilizando herramientas como stop-loss, take-profit y trailing stop. Además, 3Commas ofrece herramientas avanzadas de análisis técnico y de tendencia, así como una función de backtesting que permite a los usuarios probar sus estrategias de trading en datos históricos.

Además de su plataforma de trading automatizado, 3Commas también ofrece una amplia gama de recursos educativos y de soporte al cliente, incluyendo tutoriales en video, webinars y un equipo de soporte disponible las 24 horas del día, los 7 días de la semana.

La plataforma es compatible con varios intercambios de criptomonedas, incluyendo Binance, Bitfinex, Coinbase Pro, entre otros. [6]

Gunbot

Gunbot es un bot de trading de criptomonedas altamente configurable que permite a los usuarios automatizar sus estrategias de inversión. La plataforma proporciona una amplia gama de herramientas y técnicas de trading, incluyendo indicadores técnicos, análisis de sentimiento y arbitraje.

Gunbot es reconocido por su versatilidad y nivel de personalización, ya que permite a los usuarios adaptar sus estrategias de trading para satisfacer sus necesidades y preferencias individuales. Los usuarios tienen la posibilidad de ajustar diversos parámetros, como la cantidad a invertir, el nivel de riesgo, y los niveles de stop-loss y take-profit.

La plataforma es compatible con varios intercambios de criptomonedas, tales como Binance, Bitfinex y Poloniex. Además, Gunbot cuenta con una comunidad activa de usuarios y desarrolladores que comparten información, estrategias y mejoras en la plataforma. [9]

Haasbot

Haasbot es una herramienta de trading automatizado para criptomonedas que permite a los usuarios ejecutar sus estrategias de trading en varios intercambios de criptomonedas de manera automática. La plataforma proporciona una amplia gama de herramientas y métodos de trading, incluyendo indicadores técnicos, análisis de sentimiento y arbitraje.

Haasbot destaca por su alto grado de personalización, permitiendo a los usuarios ajustar sus estrategias de trading según sus necesidades y preferencias. Los usuarios pueden modificar diversos parámetros, como la cantidad a invertir, el nivel de riesgo, y los niveles de stop-loss y take-profit.

La plataforma es compatible con múltiples intercambios de criptomonedas, como Binance, Bitfinex, Poloniex, entre otros. Además, Haasbot ofrece una amplia variedad de herramientas de análisis y seguimiento de carteras para ayudar a los usuarios a tomar decisiones informadas sobre sus inversiones.

Haasbot también cuenta con una interfaz de usuario intuitiva y una comunidad activa de usuarios y desarrolladores que comparten información, estrategias y mejoras en la plataforma. [10]

CryptoHopper

CryptoHopper es un bot de trading automatizado de criptomonedas que permite a los usuarios realizar de forma automática las estrategias de trading que ellos consideren oportunas en diversas criptomonedas. La plataforma ofrece una amplia gama de herramientas y estrategias de trading.

CryptoHopper es conocido por su facilidad de uso y accesibilidad, ya que la plataforma ofrece una interfaz de usuario intuitiva y fácil de usar. Los usuarios pueden ajustar varios parámetros, como la cantidad de inversión, el nivel de riesgo y los niveles de stop-loss y take-profit.

La plataforma es compatible con varios intercambios de criptomonedas, como Binance, Bitfinex, Coinbase Pro.

CryptoHopper también ofrece una comunidad activa de usuarios y desarrolladores que comparten información, estrategias y mejoras en la plataforma. La plataforma también cuenta con una versión de prueba gratuita de siete días para que los usuarios puedan probar la plataforma antes de utilizarla con dinero real. [45]

Zenbot

Zenbot es un bot de trading de código abierto y gratuito diseñado para operar con criptomonedas en varios intercambios. Fue desarrollado por la comunidad de código abierto y se ejecuta en la plataforma Node.js. Zenbot utiliza algoritmos de inteligencia artificial y aprendizaje automático para analizar los datos del mercado y ejecutar operaciones automatizadas en tiempo real.

Zenbot permite a los usuarios personalizar sus estrategias de trading, incluyendo la capacidad de ajustar los parámetros de entrada y salida, indicadores técnicos y niveles de stop-loss y take-profit. La plataforma también ofrece la capacidad de backtesting para ayudar a los usuarios a evaluar el rendimiento de sus estrategias de trading en diferentes escenarios de mercado.

Zenbot es compatible con varios intercambios de criptomonedas, como Binance, Kraken, GDAX y Bitfinex, entre otros. La plataforma también ofrece una API que permite a los usuarios conectarse a otros servicios de terceros, como servicios de notificación y gestión de cartera.

Es importante destacar que Zenbot es un proyecto de código abierto en constante evolución, y los usuarios deben estar dispuestos a dedicar tiempo

y esfuerzo para aprender y ajustar la plataforma a sus necesidades y preferencias. Además, como con cualquier herramienta de trading automatizado, el uso de Zenbot conlleva riesgos y es fundamental realizar una investigación exhaustiva antes de utilizar cualquier herramienta de trading automatizado en los mercados financieros.

7. Conclusiones y Líneas de trabajo futuras

Conclusiones

El desarrollo del Bot de Trading Automatizado ha sido un proyecto integral que abarca desde la conceptualización y diseño hasta la implementación y prueba de un sistema capaz de operar en los mercados financieros de manera autónoma. Este proyecto no solo ha permitido la creación de una herramienta valiosa para el trading, sino que también ha proporcionado una plataforma de aprendizaje significativa para el desarrollo de habilidades técnicas y personales en el ámbito del análisis financiero y la programación avanzada.

A lo largo del desarrollo del proyecto, se ha logrado construir desde cero un bot de trading automatizado con capacidades avanzadas. Esto incluye la implementación de estrategias de trading y la posibilidad de operar en diferentes mercados financieros. Se ha diseñado una interfaz de usuario intuitiva y atractiva que facilita la configuración y el monitoreo del bot, asegurando que los resultados de las operaciones puedan ser exportados en formatos compatibles con herramientas de análisis de datos. Esta creación integral del bot proporciona una base sólida para futuras mejoras y expansiones.

En cuanto a los objetivos técnicos, el uso de metodologías ágiles, herramientas de gestión de proyectos como Zebhub, y sistemas de control de versiones como GitHub, ha permitido un desarrollo estructurado y eficiente del proyecto. La elección de Python como lenguaje de programación principal, junto con el uso de librerías específicas para el análisis técnico y el desarrollo web, ha facilitado la creación de un sistema robusto y escalable.

Líneas de trabajo futuras

A pesar de los avances logrados, existen varias áreas en las que el proyecto puede ser mejorado aún más:

- **Mejorar la interfaz de usuario:** La interfaz de usuario puede ser mejorada para hacerla más intuitiva y visualmente atractiva. Esto incluye la incorporación de gráficos interactivos y paneles de control más detallados para el seguimiento de las operaciones en tiempo real.
- **Implementar el bot de trading en tiempo real:** Una mejora significativa sería la implementación del bot de trading para operar en tiempo real. Esto implicaría la integración con APIs de brokers en vivo y el manejo de datos de mercado en tiempo real, permitiendo que el bot tome decisiones de trading basadas en datos actualizados al segundo.
- **Ajustar parámetros de la estrategia para alcanzar la rentabilidad:** Es crucial ajustar y optimizar los parámetros de las estrategias de trading implementadas para mejorar la rentabilidad. Esto puede incluir la utilización de técnicas de machine learning para la optimización continua de los parámetros.
- **Implementar más estrategias de trading:** La inclusión de más estrategias de trading diversificará las capacidades del bot, permitiendo que opere en diferentes condiciones de mercado. Estrategias como el trading basado en noticias, estrategias de arbitraje y algoritmos de aprendizaje profundo pueden ser consideradas.

Bibliografía

- [1] Pandas documentation: Dataframe. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>, 2024.
- [2] Python dictionaries - w3schools. https://www.w3schools.com/python/python_dictionaries.asp, 2024.
- [3] Python documentation: Tutorial - classes. <https://docs.python.org/3/tutorial/classes.html>, 2024.
- [4] Python lists - w3schools. https://www.w3schools.com/python/python_lists.asp, 2024.
- [5] Python loops - w3schools. https://www.w3schools.com/python/python_loops.asp, 2024.
- [6] 3Commas. 3commas - automate your crypto trading. <https://3commas.io/es>, 2024.
- [7] Asana. Metodología Ágil, 2024.
- [8] BBVA. Mercado de divisas: qué es y cómo funciona. <https://www.bbva.com/es/mercado-divisas-que-es-como-funciona/>, 2024.
- [9] CriptoNoticias. Gumbot: una herramienta para automatizar el trading de criptomonedas, 2024.
- [10] DappGaml. Haasbot review 2024: The ultimate crypto trading solution, 2024.

- [11] Estrategias de Inversión. Análisis fundamental. <https://www.estrategiasdeinversion.com/herramientas/diccionario/analisis-fundamental/analisis-fundamental-t-1037>, 2024.
- [12] Universidad Pablo de Olavide. Mendeley: Gestores de referencias bibliográficas, 2024.
- [13] NumPy Developers. What is numpy?, 2023.
- [14] TA-Lib Developers. Ta-lib: Technical analysis library, 2024.
- [15] Python Software Foundation. csv — csv file reading and writing, 2024.
- [16] Python Software Foundation. datetime — basic date and time types, 2024.
- [17] Python Software Foundation. logging — logging facility for python, 2024.
- [18] Python Software Foundation. os — miscellaneous operating system interfaces, 2024.
- [19] Python Software Foundation. subprocess — subprocess management, 2024.
- [20] Python Software Foundation. threading — paralelismo basado en hilos, 2024.
- [21] Python Software Foundation. traceback — print or retrieve a stack traceback, 2024.
- [22] Python Software Foundation. What is python? executive summary, 2024.
- [23] GitHub. Planning and tracking work for your team or project, 2024.
- [24] Benjamin Graham. *The Intelligent Investor: The Definitive Book on Value Investing*. HarperCollins, New York, 2006.
- [25] JetBrains. Pycharm quick start guide, 2024.
- [26] IIT Kanpur. Programación en hilos. <https://www.iitk.ac.in/esc101/05Aug/tutorial/essential/threads/definition.html>, 2024.
- [27] MetaQuotes Ltd. Descargar metatrader 5, 2024.

- [28] Burton G. Malkiel. *A Random Walk Down Wall Street: The Time-Tested Strategy for Successful Investing*. W. W. Norton Company, New York, 2003.
- [29] Admiral Markets. Cómo sacar beneficio del indicador de trading rsi. <https://admiralmarkets.com/es/education/articles/forex-indicators/como-sacar-beneficio-del-indicador-de-trading-rsi>, 2024.
- [30] Admiral Markets. Media móvil simple. <https://admiralmarkets.com/es/education/articles/forex-indicators/media-movil-simple>, 2024.
- [31] Admiral Markets. Retrocesos de fibonacci. <https://admiralmarkets.com/es/education/articles/forex-indicators/retrocesos-de-fibonacci>, 2024.
- [32] Fred McAllen. *Charting and Technical Analysis*. CreateSpace Independent Publishing Platform, Scotts Valley, California, 2013.
- [33] Microsoft. Microsoft word, 2024.
- [34] John J. Murphy. *Technical Analysis of the Financial Markets*. New York Institute of Finance, New York, 1999.
- [35] Steve Nison. *Japanese Candlestick Charting Techniques*. Prentice Hall Press, New York, 1991.
- [36] José Oramas. What are crypto trading bots and how to use them?, 2024.
- [37] Internal Pointers. A gentle introduction to multithreading. <https://www.internalpointers.com/post/gentle-introduction-multithreading>, 2024.
- [38] LaTeX Project. Latex: A document preparation system, 2024.
- [39] Rankia. Acción - diccionario de bolsa. <https://www.rankia.com/diccionario/bolsa/accion>, 2024.
- [40] Rankia. Bonos - diccionario de bolsa. <https://www.rankia.com/diccionario/bolsa/bonos>, 2024.
- [41] Rankia. Commodity - diccionario de economía. <https://www.rankia.com/diccionario/economia/commodity>, 2024.

- [42] Rankia. Criptomoneda - diccionario de criptomonedas. <https://www.rankia.com/diccionario/criptomonedas/criptomoneda>, 2024.
- [43] Rankia. Mercado financiero - diccionario de bolsa. <https://www.rankia.com/diccionario/bolsa/mercado-financiero>, 2024.
- [44] Rankia. Trading: ¿qué es y cómo empezar? ¿cómo funciona? <https://www.rankia.com/blog/bolsa-desde-cero/5816904-trading-que-como-empezar-funciona>, 2024.
- [45] Wallet Reviewer. Haasbot vs cryptohopper: Which trading bot is better?, 2024.
- [46] Alfonso Serrano. La librería pandas, 2024.
- [47] Sharvin Shah. The ultimate guide to python: How to go from beginner to pro, 2020.
- [48] Wikipedia. Thread (computing). [https://en.wikipedia.org/wiki/Thread_\(computing\)](https://en.wikipedia.org/wiki/Thread_(computing)), 2024.
- [49] ZenHub. Zenhub: Github project management, 2024.