**Reporte de Estrategias de Trading utilizando análisis técnico**

-Estrategia y Razonamiento:

La estrategia construye un sistema de trading robusto y hecho con mucho esfuerzo para que no falle tan fácilmente (debido a muchos errores que tuve en el proceso), comenzando con tener los datos limpios (fechas organizadas, sin duplicados, etc.), mi estrategia combina varias señales técnicas que son el RSI, MACD, ADX con +DI/−DI, Bandas de Bollinger y Estocástico mediante un voto ponderado dándole más peso a la tendencia/fuerza, también consideré un umbral mínimo de volatilidad (ATR) para evitar ruido, también se utilizó un tamaño de posición por fracción de capital, Stop Loss y Take Profit a prueba y error con valores pequeños, y salvaguardas como breakeven, trailing por ATR y time-stop que los llamé como gestión avanzada; Los cortos se simulan en modo no leverage como fue solicitado y se incluyen comisiones. Los parámetros se optimizan con Optuna en un esquema walk-forward, evaluando sobre ventanas de validación con reglas duras (ganancia neta, límites de drawdown, suficientes operaciones) y un objetivo que premia estabilidad (Sharpe/Calmar) y penaliza exceso de riesgo, sobre-operación y señales inestables entre ventanas, en pocas palabras le puse restricciones penalizando y premiando lo que buscaba. Finalmente, se valida con métricas estándar que fueron Sharpe, Sortino, Calmar, CAGR, Volatilidad, MaxDD, WinRate, se muestran unas tablas de retornos y gráficos de equity y drawdown, buscando un balance simple y objetivo entre retorno y control del riesgo, y minimizando el sobreajuste.

-Análisis y preprocesamiento de datos:

El análisis y preprocesamiento fue una de las partes más complicadas para mí, en esta parte se intentó dejar una serie horaria limpia, ordenada y reproducible antes de calcular indicadores, ya que había detalles que me generaban problemas y errores. Primero se leyó el CSV de forma robusta (normalizando nombres de columnas y detectando cabeceras), se parseó la columna Date a datetime, se eliminó cualquier zona horaria si es que hubiera existido y se estableció como índice ordenado. Luego se convirtieron las columnas Open, High, Low, Close, Volume a numérico por cualquier cosa y se descartaron solo las filas sin OHLC válidos para evitar imputaciones arbitrarias. Para timestamps repetidos se aplicó un colapso OHLC lógico (Open=primero, High=máximo, Low=mínimo, Close=último; Volume=suma) y se garantizó un índice único. No se rellenaron huecos ni se re-muestreó la serie (se preservaron los gaps reales del mercado para no introducir sesgos). Finalmente, se verificó la integridad con conteos, rango temporal y un diagnóstico de duplicados y huecos por hora; con ello, la entrada al modelo se supone que queda estandarizada y libre de factores que distorsionen señales o métricas.

-Metodología e Implementación:

Todo el proyecto está dentro de la carpeta 002\_Introduction\_to\_trading, en la que podemos ver divididos todos los documentos .py, instrucciones y requerimientos del VENV. Dentro de la carpeta Data se aloja la serie histórica horaria de BTC/USDT que alimenta todo el pipeline. El CSV trae columnas estándar de mercado (al menos Date, Open, High, Low, Close y opcionalmente Volume) y puede incluir peculiaridades típicas de exportaciones (cabeceras re-embebidas, codificación BOM o filas no válidas). El proyecto asume que este archivo es la fuente de verdad para generar señales, ejecutar el backtest y calcular métricas, por lo que la limpieza posterior (parseo de fechas, coerción numérica, manejo de duplicados) se hace pensando en este formato.

\_\_init\_\_.py (atajos y submódulos).

Este archivo sirve como punto de entrada del paquete local: expone atajos a funciones clave (por ejemplo run\_all, backtest, generadores de señales, métricas y utilidades) para que puedan importarse de forma directa y consistente desde otros scripts. Su objetivo es simplificar los imports y dejar claro qué componentes forman parte de la “API pública” del proyecto, evitando que los consumidores tengan que conocer la ruta interna de cada módulo.

backtest.py.

Esta función simula la ejecución de la estrategia sobre datos ya señalizados (data\_sig) y devuelve la curva de capital (equity), el detalle de operaciones (trades) y el efectivo final. Internamente, primero normaliza la entrada: ordena por tiempo, asegura unicidad del índice, valida la presencia de Close y obtiene la señal discreta (signal ∈ {-1,0,1}) ya sea desde signal o de buy\_signal/sell\_signal. Si se activa la gestión opcional como trailing por ATR, calcula un \_atr por barra; y si se fija un objetivo de volatilidad (target\_vol), ajusta dinámicamente el fraccionamiento de riesgo por barra en función de una vol anualizada estimada, acotándolo con max\_rf\_scale. En el bucle principal recorre barra a barra: si hay posición abierta, gestiona breakeven, trailing stop por ATR y time stop; y cierra por stop-loss o take-profit con comisiones. Soporta largos y cortos en modo no\_leverage así nombrado por simplicidad y para dejarlo en claro. En short inmoviliza un colateral igual a la inversión y al cerrar libera ese colateral y descuenta el coste de recompra. Si no hay posición, evalúa la señal: con signal=1 abre largo comprando un número de “shares” proporcional a cash \* \_rf\_t; con signal=-1 abre corto vendiendo en descubierto bajo la misma lógica de tamaño (y reservando colateral). Tras cada barra marca el valor de equity incluyendo mark-to-market y, al final, liquida cualquier posición remanente. El diseño separa mecánicas de entrada/salida, aplica comisiones en cada operación, respeta límites sin apalancamiento en short, y permite gestión opcional (breakeven, trailing, time stop, target-vol) para modular el perfil riesgo/retorno sin cambiar las reglas de señal, esta función fue una de las más problemáticas y complejas, pero al final cumple con el objetivo.

config.py (parámetros centralizados del proyecto)

Este módulo define, de forma tipada y ordenada, la configuración base del proyecto. Con ROOT fija la ruta del directorio del paquete para poder construir rutas relativas de manera robusta. Las @dataclass agrupan parámetros por dominio: DataCfg concentra la ruta por defecto del CSV y los nombres de columnas clave (facilitando cambiar la fuente de datos sin tocar lógica); StratCfg reúne hiperparámetros de indicadores técnicos (ventanas y umbrales de RSI, MACD y ADX), de modo que la generación de señales lea todo desde un único sitio; y BtCfg encapsula las opciones del backtest (comisión, capital inicial, activación de cortos “no\_leverage”, SL/TP y un n\_shares de referencia). Este diseño separa configuración de código, evita “valores mágicos” esparcidos, y permite versionar/ajustar escenarios con menor riesgo de errores.

io\_data.py (limpieza conservadora y robusta)

En este módulo tenemos la función de limpiar series de tiempo, esta función toma el DataFrame crudo y lo deja listo para análisis y backtest, priorizando no perder información válida. Primero normaliza nombres de columnas (eliminando BOM y espacios) y descarta Unix si aparece. Luego aplica una doble defensa contra cabeceras re-embebidas (líneas de encabezado que a veces se “cuelan” en medio del CSV): filtra filas cuya columna Date no tenga pinta de fecha/epoch, con patrones flexibles. Después parsea fechas con errors='coerce', elimina solo filas cuyo Date no pudo convertirse y define el índice temporal (sin zona horaria). Convierte OHLC a numérico con coerción evitando que strings residuales rompan el flujo y al final descarta únicamente las filas que tengan algún NaN en Open/High/Low/Close. Por último, resuelve timestamps duplicados conservando el último (keep='last'), que suele corresponder a la fila más “reciente/correcta” cuando hay colisiones. Se devuelve la serie ordenada y limpia, sin modificar más de lo imprescindible. También tenemos otra función de diagnóstico rápido del índice temporal, esta es una función auxiliar que fue hecha para verificar la salud del índice de fechas: reporta cuántos duplicados existen y cuántos “gaps” horarios no uniformes se detectan (comparando las diferencias sucesivas respecto a 1 hora). Esta función fue útil para auditar importaciones, detectar problemas de muestreo y entender si las caídas de filas vienen de duplicados o de huecos reales en la data.

main.py (Orquestador de todo el pipeline).

run\_all es la función se dría que principal, esta función lo que hace es que recibe el DataFrame ya limpio y con índice temporal, lanza la optimización de hiperparámetros con Optuna y valida el mejor conjunto en un split 60/20/20, centrando la evaluación en Validation (20%). Primero asegura que el índice esté ordenado y sin duplicados. Configura un TPESampler (con seed con el fin de no perder nuestros resultados, multivariate, group y constant\_liar para estabilidad en la búsqueda) y un MedianPruner (descarta trials flojos tras un arranque) para acelerar la búsqueda. Si se proveen seed\_trials, los coloca en la cola para guiar la exploración. Objective(trial, data, n\_chunks, short\_mode) es nuestro optimizador el cuál es llamado, hace walk-forward y devuelve un score medio penalizado por dispersión, diseñado para favorecer configuraciones que rindan en validación sin sobre-ajustar. Al terminar, toma best\_params, normaliza SL/TP, arma los kwargs de gestión con \_extract\_mgmt\_kwargs (breakeven, trailing ATR, time-stop, etc.) si es que hubiera y se habilitaran y ejecuta un backtest solo en Validation aplicando el filtro de régimen (SMA-200) únicamente aquí para evitar leakage en la fase de búsqueda. Calcula métricas (Sharpe, Sortino, Calmar, MaxDD, CAGR, volatilidad anual, WinRate) que fueron solicitadas, genera las tablas de retornos (mensual, trimestral, anual) y, opcionalmente, grafica la curva de Validation. Imprime un resumen claro (mejor valor, hiperparámetros óptimos, métricas y tablas) y devuelve un diccionario completo con el estudio de Optuna, los mejores parámetros, la equity/operaciones de Validation, métricas y los flags usados (fees, capital, modo de cortos, gestión). Esa salida compacta permite reutilizar el mejor set tanto para un backtest, como para reporting.

Metrics.py

Nuestra función principal aquí compute\_metrics calcula las métricas clave de desempeño a partir de la equity y el registro de trades. Primero limpia NaNs y obtiene los rendimientos por barra; con el índice temporal infiere “periodos por año” (ppy) para anualizar retorno y volatilidad de forma consistente sin depender de la frecuencia fija. El CAGR se calcula con la duración real en años (robusto ante huecos). Con retorno y volatilidad anualizados construye Sharpe; para Sortino usa solo la desviación de rendimientos negativos. El Max Drawdown se obtiene como el mínimo de la serie equity/rolling-max – 1. El Calmar es CAGR dividido por el drawdown en valor absoluto (si existe drawdown). Finalmente, WinRate proviene de la proporción de operaciones con pnl>0. Se devuelve un diccionario compacto con Sharpe, Sortino, Calmar, MaxDD, WinRate, AnnualReturn, AnnualVol y el capital final. \_safe\_calmar(eq, dd\_floor=0.05, calmar\_cap=10.0) Entrega un Calmar “seguro” para usar en scoring evitando valores extremos, ya que antes de hacerlo me daba un primedio arriba de 20. Calcula CAGR con años reales, y el drawdown con un piso (dd\_floor) para que drawdowns casi nulos no inflen Calmar. Además aplica un cap superior (calmar\_cap) para limitar outliers. Se usa en la optimización como métrica estable y menos manipulable. La fución de print\_equity\_debug Imprime un resumen rápido y legible para depurar: valor inicial/final/máximo de la equity, x-factor (multiplicador del capital), MaxDD y número de operaciones. Está pensado para ver de un vistazo si la simulación tiene sentido antes de entrar en reportes extensos. returns\_tables(equity) de igual manera como su nombre lo dice nos genera las tablas de retornos mensual, trimestral y anual. Re-muestrea la equity a diaria (last) para evitar sesgos intradía, calcula rendimientos compuestos por periodo y entrega: una tabla Year × Month(1…12), otra Year × Quarter(1…4) y una serie anual. Esto facilita el análisis de estacionalidad y consistencia del sistema.

Optimization.py

suggest\_hyperparams Define el espacio de búsqueda de Optuna: rangos para risk\_fraction, SL/TP, ventanas y umbrales de RSI/MACD/ADX/Bollinger/Stochastic, y parámetros de gestión avanzada (breakeven, trailing ATR, time-stop, y opcional vol targeting) en esta parte si estuve mucho tiempo a prueba y error hasta que me diera algún resultado coherente, ya que en Validation era casi seguro que me diera pérdida, en esta parte decidí utilizar límites conservadores para favorecer configuraciones realistas (riesgo fraccional moderado, RR razonable, etc.). El resultado es un diccionario con todos los hiperparámetros listos para probar.

\_extract\_mgmt\_kwargs(p) Filtra del diccionario de hiperparámetros solo los kwargs que entiende el backtest en materia de gestión (breakeven, trailing, time-stop, ATR window, target\_vol, lookback, cap de escala). Separa responsabilidades: la búsqueda puede definir muchos parámetros, pero el backtest solo recibe lo que realmente usa.

objective Es la función objetivo del estudio Optuna con walk-forward conservador. Para cada conjunto de parámetros sugeridos:

1. Crea chunks de entrenamiento/validación (y en tu implementación también se evalúa en test y validation por bloque).

2. Genera señales, aplica (cuando corresponde) un filtro de régimen SMA-200 solo en evaluación para evitar leakage.

3. Corre backtest en TEST y VALIDATION, obtiene métricas y Calmar seguro.

4. Aplica reglas duras en VALIDATION (mínimos de Calmar/Sharpe, drawdown no extremo, capital final > inicial y suficientes trades). Si falla, penaliza el chunk.

5. Calcula un score por chunk combinando Calmar (Val y Test), Sharpe, retorno anual y penalizaciones por retorno<volatilidad, mismatch de signo entre Test y Val, DD extremos, RR fuera del rango [1.35, 2.40], riesgo fraccional por encima del soft-cap y sobre-operar (trades por año).

6. Devuelve el promedio penalizado por la desviación (mean − λ·std) si al menos el 60% de los chunks aprueban; si no, descarta el trial con un valor muy bajo.

Esta formulación prioriza robustez en VALIDATION, consistencia entre Test y Val, y evita soluciones frágiles con drawdowns o riesgos excesivos.

run\_optuna es un wrapper para lanzar el estudio Optuna sobre la objective anterior. Crea un TPESampler con semilla, usa un NopPruner (sin pruning adicional, ya que dentro del objetivo ya hay reglas duras) y ejecuta study.optimize(...). Devuelve el objeto Study para inspeccionar mejores parámetros, valores, historia de trials, etc.

plotting.py.

Como su nombre lo dice crea funciones de gráfico para revisar rápidamente los indicadores sin manipular datos. plot\_equity\_train\_test\_val es el encargado de dibujar una figura con dos paneles: arriba la curva de equity segmentada por periodos Train/Test/Validation (cada tramo con su color) y abajo el drawdown del histórico. Re-muestrea a “1D” por defecto para suavizar “peines” intradía, formatea el eje Y en dinero (M/K) y traza una línea horizontal del capital inicial. Útil para ver de un vistazo cuándo y dónde rinde más/peor la estrategia, este gráfico nos muestra un que hubiera pasado si... hasta el final del dataset.

plot\_validation\_equity. Igual que la anterior pero enfocado solo a la parte de Validation que es la que nos interesa que nos de rendimiento. También aplica un re-muestreo ligero, usa formato de dinero y la línea de capital inicial. Es la vista limpia para presentar resultados de “hold-out 20%”.

plot\_indicators muestra por separado, los principales indicadores técnicos sobre el dataframe (se asume que tiene al menos Close/High/Low), contiene RSI con bandas inferior/superior del set de parámetros, MACD (línea, señal e histograma), ADX +DI/−DI con umbral de tendencia, Bollinger (banda alta, media, baja) sobre el precio, Stochastic %K y %D con líneas 20/80. Sirve para validar visualmente que las señales tunadas por Optuna “tienen sentido técnico” en el precio. También en esta parte contamos con una pequeña seccion llamada Helpers en donde se encuentran los formateadores de dinero y cortes temporales sencillos yPequeños ajustes opcionales de márgenes para que los gráficos no dejen espacios innecesarios.

position.py

Es un contenedor de estado para una posición abierta. Position Guarda lo mínimo necesario para simular la vida de la posición.

requirements.txt

Es la lista de dependencias mínimas para que el entorno sea reproducible, esto debido a que estámos trabajando con un VENV, entonces al ponerle que instale las paqueterías contenidas aquí no debería de haber problema en correr el código.

signals.py

Crea señales de trading y filtros de régimen.

generate\_signals calcula indicadores con ta y produce una señal discreta signal ∈ {-1,0,1}. La lógica combina:

-RSI (sobrecompra/sobreventa)

-MACD (cruces/confirmación)

-ADX (fuerza de tendencia y opcionalmente +DI/−DI para dirección)

-Bollinger (desviaciones) y Stochastic (momentum).

La señal final prioriza confluencia y coherencia direccional (p. ej., no tomar largos si ADX indica fuerza pero la dirección no acompaña). El resultado es un DF con columnas de indicadores y signal que el backtest entiende.

apply\_regime\_filter(df, sma\_window=200) es un filtro de régimen de mercado: calcula una SMA larga y, si el precio está por debajo, atenúa o bloquea los largos (y viceversa para cortos, según la implementación). La idea es reducir operaciones contra-tendencia en entornos desfavorables, mejorando estabilidad.

trials.py

Es el archivo “de configuración” para separar elección de semillas y defaults del runner que mejor me dieron. Dentro de SEED\_TRIALS me da la lista de diccionarios con puntos de partida (semillas) para que Optuna tenga de donde comenzar a buscar, esto se me vino a la mente debido a que en la clase de MIA al utilizar el PSO al ir simulando le podemos ir dando los parámetros desde donde queremos que inicie para que encuentre el resultado más rápido, por lo que en esta parte de todas las optimizaciones que corrí me realiza combinaciones razonables de RF, SL/TP e indicadores que ya sé que no están tan lejos de un buen resultado. Lo ideal sería encolar 2–3 semillas para acelera la convergencia y reducir la varianza entre corridas.

utils.py

Esta parte contiene utilidades pequeñas y puras para dividir datos y trabajar con fechas.

temporal\_split\_60\_20\_20(df). Parte el DataFrame ordenado por fecha en tramos 60% / 20% / 20% (Train/Test/Val). Es el split “oficial” que usas para el reporte final (Validation), consistente con el backtest y plots.

make\_walkforward\_chunks(df, n\_chunks). Genera n\_chunks consecutivos de walk-forward (cada uno con su train/test/val según tu esquema), garantizando que se recorre todo el rango temporal y que cada chunk conserva el orden temporal (sin leakage). Es la base del objective de Optuna.

infer\_ppy(index). A partir del índice temporal, estima periodos por año (ppy): si la serie es horaria, diaria, etc. Lo usamos para anualizar retornos/volatilidad de forma automática y robusta incluso con huecos.

runner.py.

Este es el encargado de juntar todo y mostrarlo, prácticamente es el que hace la magia, lo puse con la intención de que runall juntara todo y este lo ejecutara. Toma los datos crudos, dispara la optimización y produce el backtest integral con sus gráficos, todo de punta a punta y desde consola. Al iniciarse, parsea argumentos (ruta al CSV, número de trials y chunks del walk-forward, seed, uso de seeds predefinidas, y si se deben mostrar plots. Luego llama a load\_csv\_smart, un lector del CSV que limpia encabezados re-embebidos y filas “raras” en caso de que hubieran estas, y pasa el resultado a clean\_timeseries, que normaliza nombres, convierte Date a índice temporal, tipa OHLC a numérico, ordena por fecha y elimina duplicados conservando el último timestamp. Con el dataset ya “sano”, invoca run\_all: ahí se ejecuta Optuna; al terminar, devuelve los mejores hiperparámetros y la curva/ métricas de Validation. Con esos parámetros “ganadores”, runner.py levanta señales sobre todo el histórico (con filtro de SMA-200) y corre un backtest completo (run\_full\_backtest) para medir cómo habría funcionado la estrategia en producción continua. A efectos de trazabilidad, imprime rangos temporales y conteos de filas de data/señales/equity para confirmar que no hubo recortes, y—si se activan los plots genera tres vistas: (1) equity segmentada Train/Test/Validation con panel de drawdown, (2) la Validation en detalle, y (3) los indicadores técnicos (RSI, MACD, ADX, Bollinger, Stoch) para validar visualmente las señales. Antes de mostrar, ajusta márgenes para eliminar espacios en los ejes.

ACLARACIONES Y DEFINICIONES:

OHLC

OHLC significa Open, High, Low, Close (Apertura, Máximo, Mínimo y Cierre). Es el formato estándar para series de precios financieras y condensa, por barra de tiempo, el rango completo de negociación. Lo usamos porque permite calcular señales y riesgos de forma consistente (p. ej., stops sobre Low/High, medias y osciladores sobre Close) y porque muchos indicadores técnicos dependen explícitamente de estos cuatro puntos.

Walk-Forward

El walk-forward es una validación temporal en bloques: se optimiza en un tramo (train), se comprueba en otro posterior (test) y se evalúa finalmente en un tramo aún más reciente (validation). Repetimos el proceso en “ventanas” deslizantes y promediamos resultados. Lo usamos para simular el flujo real del tiempo, evitar look-ahead bias y medir si los parámetros “sobreviven” a cambios de régimen sin sobreajuste.

Filtro SMA-200

La SMA-200 es la media móvil simple de 200 periodos; se usa como filtro de régimen para distinguir contextos de tendencia vs. lateralidad. La aplicamos para “apagar” señales cuando el entorno no es favorable (o reducir exposición), lo que tiende a mejorar estabilidad y drawdowns en validación sin complicar la lógica con más indicadores.

Parámetros de gestión avanzada

(i) breakeven\_trigger, que mueve el stop al precio de entrada cuando el trade avanza lo suficiente; (ii) trailing stop por ATR (trail\_atr\_mult con ventana ATR=14), que sigue al precio y captura tendencias sin regalar demasiado beneficio; (iii) time\_stop\_bars, cierre por tiempo para evitar quedarnos atrapados en rangos; y (iv) target\_vol (opcional), que ajusta el risk\_fraction según la volatilidad realizada para mantener un riesgo más estable. Estos controles se añadieron porque en validation el rendimiento rara vez era satisfactorio: en lugar de seguir agregando indicadores (que aumentan el riesgo de sobreajuste), preferí fortalecer la gestión del riesgo y de la posición, que es lo que realmente gobierna la distribución de resultados (recorta colas, protege capital y estabiliza el perfil de retornos). Decidí priorizar herramientas que gestionan cómo se gana o se pierde, no más “señales” que podrían no generalizar y la verdad si me obsesioné con poder sacar rendimiento en Validation que fuera arriba de la tasa libre de riesgo (de Estados Unidos).

Indicadores técnicos (RSI, MACD, ADX, Bollinger, Stoch)

* RSI (Relative Strength Index): oscilador de momentum entre 0–100; lo usamos para detectar sobrecompra/sobreventa relativas y sincronizar entradas/salidas en torno a umbrales.
* MACD: diferencia de medias exponenciales con línea de señal e histograma; lo usamos para capturar cambios de impulso y cruces que anticipan giros o continuaciones.
* ADX: mide la fuerza de la tendencia (junto a +DI y −DI); lo usamos como filtro para operar solo cuando la direccionalidad es suficiente.
  + +DI y −DI vienen del sistema direccional de Wilder. Primero se calculan los movimientos direccionales por barra:
    - - DM+ = High\_t − High\_{t−1} si ese valor es mayor que Low\_{t−1} − Low\_t y es positivo; en otro caso, DM+ = 0.
    - - DM− = Low\_{t−1} − Low\_t si ese valor es mayor que High\_t − High\_{t−1} y es positivo; si no, DM− = 0.
  + Luego, se suavizan (estilo Wilder) y se normalizan por el True Range (volatilidad) para obtener +DI y −DI:
    - - +DI ≈ 100 × (DM+ suavizado / TR suavizado)
    - - −DI ≈ 100 × (DM− suavizado / TR suavizado)
* Bandas de Bollinger: media ± k desviaciones; las usamos para reconocer expansión/contracción de volatilidad y posibles reversiones hacia la media.
* Stochastic (K/D): compara el cierre con el rango reciente; lo usamos para afinar timing en extremos y confirmar impulsos.

En conjunto, estos indicadores aportan señal moderada y complementaria (momentum, tendencia, volatilidad y timing), mientras que la gestión avanzada se encarga de moldear el perfil de riesgo/beneficio para que la estrategia sea más robusta en validación.

-Resultados y análisis de desempeño:

equity[0] : 1000000.0

equity[-1]: 1139251.8981487795

equity max: 1179380.7088268544

x-factor : 1.1392518981487796

max drawdown (val): -0.03445945997490263

n\_trades (val): 1223

=== Best value (Optuna, avg score in walk-forward) ===

0.3877398921470038

=== Best Hyperparameters ===

risk\_fraction: 0.1252595478810879

SL: 0.046425109072137656

TP: 0.10600981581711783

rsi\_window: 21

rsi\_lower: 27

rsi\_upper: 73

macd\_fast: 11

macd\_slow: 24

macd\_signal: 8

adx\_window: 22

adx\_threshold: 39

bb\_window: 21

bb\_dev: 2.4

stoch\_window: 11

stoch\_smooth: 4

breakeven\_trigger: 0.3114421972346785

trail\_atr\_mult: 0.6496657165849747

time\_stop\_bars: 34

use\_target\_vol: False

=== Validation Metrics (hold-out 20%) ===

Sharpe: 1.848

Sortino: 2.543

Calmar: 2.380

MaxDD: -3.45%

WinRate: 0.404

AnnualReturn: 8.20%

AnnualVol: 4.49%

FinalCapital: 1,139,251.90

En la validación (20% del historial reservado para probar), la estrategia empezó con $1,000,000 y terminó en $1,139,251.90 (x-factor 1.139), con un máximo descenso desde pico de solo −3.45% y 1,223 operaciones. Su rendimiento anualizado fue 8.20%, mayor que la tasa libre de riesgo de EE. UU., y la volatilidad fue de 4.49%, es decir, el retorno fue mayor que el riesgo. Los ratios Sharpe 1.848 y Sortino 2.543 indican que el desempeño fue bueno tanto en general como evitando caídas fuertes; el Calmar 2.380 refuerza que el crecimiento anual supera varias veces la peor racha. Aunque el WinRate es 40.4% (se acierta menos de la mitad), el tamaño de las ganancias promedio compensa y deja un balance positivo gracias al TP ≈ 10.60% y SL ≈ 4.64% con risk\_fraction ≈ 12.5%. Este fue el mejor caso encontrado por Optuna (best value 0.3877) tras muchas pruebas; en la mayoría de configuraciones la validación no superaba a la tasa libre de riesgo, pero los parámetros de gestión (breakeven, trailing ATR y time-stop) ayudaron a mejorar los resultados sin añadir más indicadores.

-Desempeño de métricas, gráficos y tablas:

=== Monthly Returns (%) ===

Month 1 2 3 4 5 6 7 8 9 10 11 12

Year

2024 0.00 2.68 4.20 0.34 -0.78 0.70 2.04 1.67 0.65 0.48 2.0 0.04

2025 -0.25 1.22 -0.22 1.30 0.38 -1.53 -0.06 -0.88 -0.74 NaN NaN NaN

=== Quarterly Returns (%) ===

Quarter 1 2 3 4

Year

2024 6.99 0.26 4.42 2.54

2025 0.75 0.13 -1.67 NaN

=== Annual Returns (%) ===

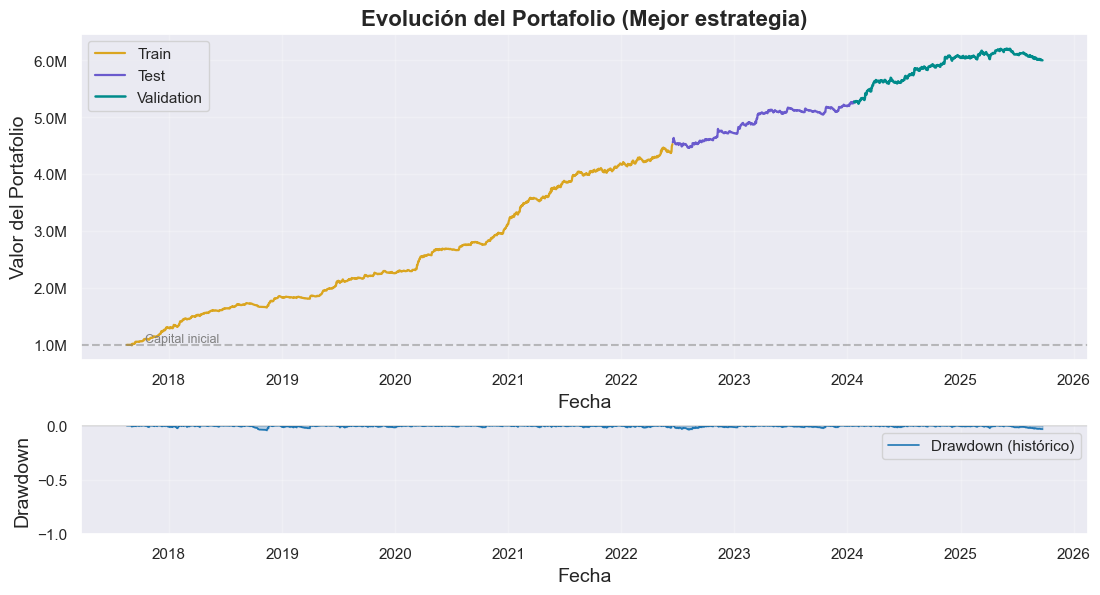
Annual Return

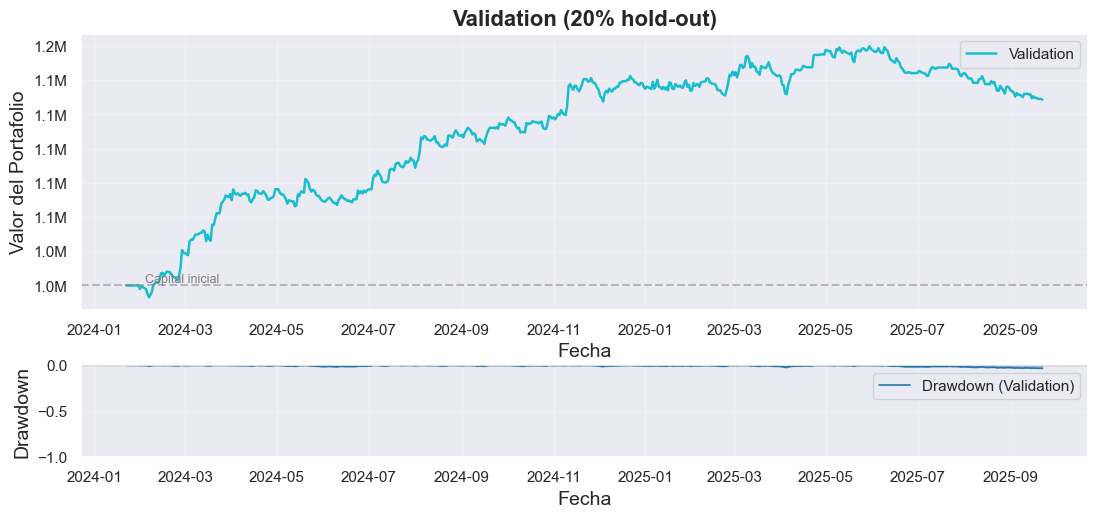
2024 14.85

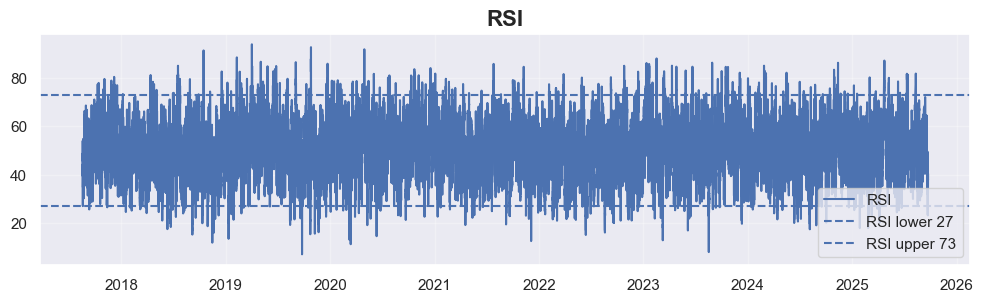
2025 -0.80

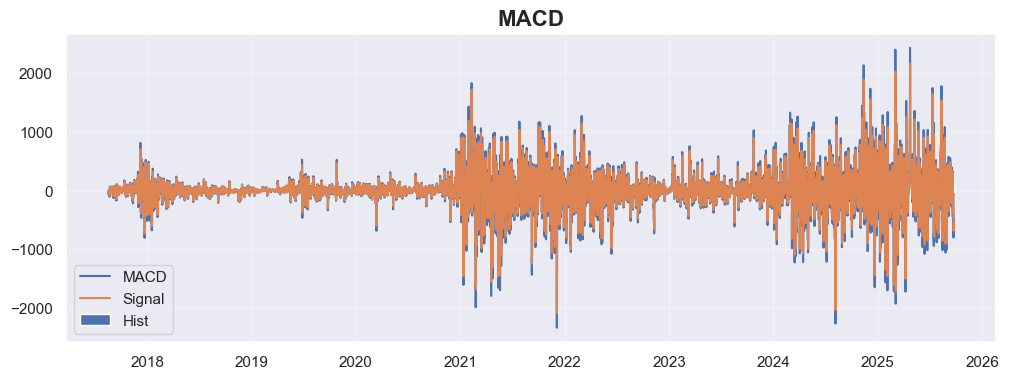
[INFO] Trades (full): 5950

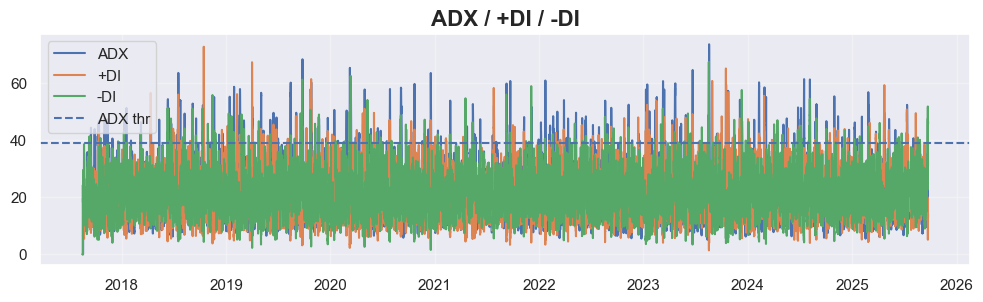
[INFO] Equity final (full): 5,515,589.23

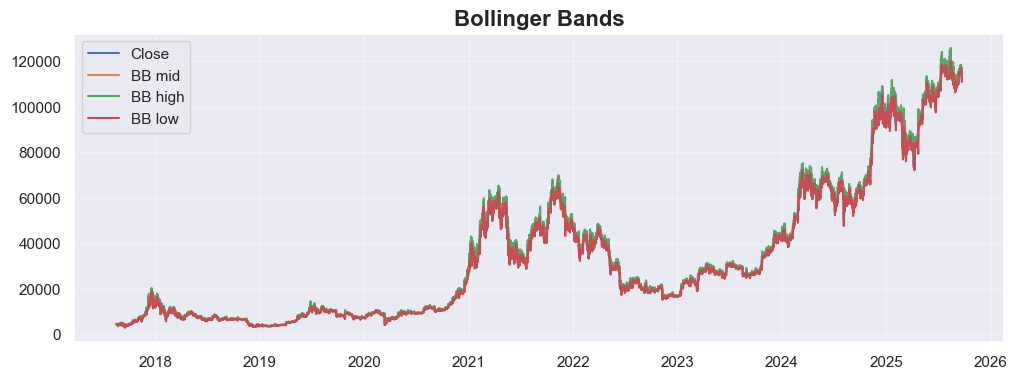












En el hold-out de validation (20%), la cuenta sube de 1,000,000 a 1,139,252 con un drawdown máximo muy contenido (-3.45%) y 1,223 operaciones con un ritmo alto pero controlado, lo que se refleja en métricas sólidas de riesgo: Sharpe 1.85, Sortino 2.54 y Calmar 2.38 para un retorno anualizado de 8.2% con volatilidad anual del 4.49%. Por meses y trimestres, 2024 fue fuerte (+14.9% anual; T1 +7.0%, T3 +4.4%, T4 +2.5%) y 2025 va levemente negativo (-0.8%) con debilidad en Q3 (-1.67%). El gráfico de equity muestra una curva ascendente y relativamente estable, con máximos en aproximadamente 1.18M y caídas poco profundas, coherente con los drawdowns reportados. En el backtest completo (para contexto), se ejecutaron 5,950 trades y la equity final alcanza ~5.52M, lo que sugiere que, aunque la estrategia es activa, mantiene una relación retorno/riesgo saludable y drawdowns moderados en el periodo evaluado.

-Análisis del riesgo y limitaciones:

El principal riesgo es el sobreajuste: como buscamos muchos parámetros, el optimizador puede aprender “peculiaridades” del pasado que no se repiten. Además, las asunciones de ejecución son optimistas (comisiones fijas, sin deslizamiento ni costos de fondeo en cortos, llenados perfectos en SL/TP), y la limpieza de datos pudo eliminar filas válidas o suavizar extremos. El mercado de cripto, por otro lado, cambia de régimen con frecuencia; un filtro SMA-200 o ADX ayuda, pero no garantiza que el comportamiento futuro sea parecido al histórico. Incluso las métricas “bonitas” pueden ocultar riesgos (p. ej., el drawdown calculado con cierres por hora subestima caídas intrabar).

También hay efectos secundarios de la gestión avanzada (breakeven, trailing ATR, time-stop): pueden mejorar una ventana concreta, pero volver la estrategia sensible a “latigazos” de precio. El riesgo de concentración (tamaño por operación ~12.5%) y el overtrading hacen que pequeños cambios en costos o liquidez deterioren el resultado; y persisten los riesgos operativos (exchange, API, custodia) y de reproducibilidad (versiones de librerías o datos). En el peor caso, un evento de cola gorda (flash crash, desconexión) puede saltarse stops y multiplicar pérdidas.

Para mitigar investigando un poco algunas ideas son el poder usar validaciones fuera de muestra adicionales y paper trading prolongado; pruebas de sensibilidad (comisiones/slippage/funding, ±20% en tamaños), ablation de la gestión (quitar/poner reglas para ver su aporte real), y límites operativos (cortes por pérdida diaria, enfriamientos tras rachas). Añadir deslizamiento dependiente de volatilidad, estimar drawdown intradía, contemplar funding en cortos, registrar versiones/datos y guardar los estudios de Optuna. Sólo tras pasar estas pruebas la estrategia debería considerarse para capital real.

-Conclusión:

Concluyendo el proyecto, siento que el mayor logro no fue solo “hacer que corriera”, sino construir un proyecto serio y reproducible: limpié datos, separé en train/test/validation, monté un walk-forward y combiné indicadores técnicos con gestión de riesgo avanzada (breakeven, trailing por ATR, time-stop y sizing). Integré heurísticas y búsqueda de hiperparámetros que ví en MIA, sumé restricciones y penalizaciones que ví en Métodos de Optimización, y entendí de forma práctica cómo funcionan los cortos y qué aporta cada indicador. Me costó mucho ya que hubo muchas malas iteraciones, noches de cómputo y callejones sin salida, pero si pude con eso espero poder con lo que viene. En términos del objetivo académico, sí se cumplió: diseñé una estrategia sistemática basada en análisis técnico, programé un entorno de backtesting, optimicé parámetros con un esquema walk-forward conservador y preparé salidas claras (métricas, tablas y gráficos) para un informe profesional de desempeño. Además, confirmé algo importante: el análisis técnico solo raramente es suficiente; cuando se combina con filtros de régimen, validación honesta y reglas sólidas de riesgo, puede volverse utilizable y mostrar rendimientos superiores al costo del riesgo asumido.

Más allá de la métrica puntual, hoy tengo una noción e idea de cómo se hace un backtesting y que no es tan sencillo como en un inicio creí. Aprendí a cuestionar resultados, a priorizar la robustez sobre el brillo, y a conectar lo visto en clase junto con ayuda de la IA para explorar el espacio de parámetros, hasta Optimización para disciplinar la búsqueda, todo aterrizado en un backtest transparente. Si el resultado cambia en el futuro, creo que tengo la noción de cómo depurarlo, mejorarlo y defenderlo. Y eso, para mí, vale tanto como el mejor gráfico de equity.