

Playbook: Framework Cuantitativo para Criptomonedas (Python y AI)

Fase 1: Configuración del Entorno y Herramientas

La base de un *framework* exitoso comienza con un entorno de trabajo rápido y profesional, y la adopción de librerías avanzadas.

Paso	Elemento Clave	Descripción y Justificación (Fuentes)
<b>1.1 Instalación Base</b>	<b>Anaconda y Jupyter Notebook</b>	Instale Anaconda para gestionar librerías. Lance y trabaje en <b>Jupyter Notebook</b> (o Google Colab, especialmente útil para ML/LSTM).
<b>1.2 Librerías Fundamentales</b>	<b>Pandas, NumPy, Matplotlib</b>	Importe <code>pandas</code> ( <code>pd</code> ) para manipulación de datos, <code>numpy</code> ( <code>np</code> ) para cálculos matemáticos (especialmente logaritmos y raíz cuadrada), y <code>matplotlib</code> ( <code>plt</code> ) para graficar.
<b>1.3 Adquisición de Datos (APIs)</b>	<b>YFinance (Proxy), Alpha Vantage, Polygon.io</b>	Use <code>yfinance</code> o <code>pandas data reader</code> como ejemplo de acceso a datos. Para datos de criptomonedas, el autor menciona que plataformas como <b>Alpha Vantage</b> y <b>Polygon.io</b> ofrecen APIs de stocks y cripto. <b>Invierta en datos de alta calidad</b> , ya que el <i>backtesting</i> correcto lo requiere.
<b>1.4 Rendimiento de Datos</b>	<b>Polars y DuckDB</b>	Para un análisis ultrarrápido de grandes volúmenes de datos (común en cripto), use <b>Polars</b> (reemplazo de alto rendimiento para Pandas) y <b>DuckDB</b> (base de datos relacional <i>in-process</i> rápida). DuckDB puede leer directamente <i>DataFrames</i> de Polars y permite consultas SQL complejas (como VWAP rodante).

Fase 2: Recolección y Limpieza de Datos

El autor subraya que el **80-90% del trabajo en ciencia de datos es la limpieza de datos**.

Paso	Elemento Clave	Descripción y Justificación (Fuentes)
<b>2.1 Armonización de Fechas</b>	<b>Conversión y Fusión</b>	Asegúrese de que todos los <i>DataFrames</i> (precios, factores, etc.) tengan un índice de fecha comparable. Si los tipos de datos de índice son diferentes (ej. <code>Timestamp</code> vs. <code>Period</code> ), cree una <b>columna de cadena común</b> (ej. ' <b>Año-Mes</b> ') para realizar una fusión precisa ( <code>pd.merge</code> ).
<b>2.2 Cálculo de Retornos</b>	<b>Retornos Logarítmicos</b>	Utilice retornos logarítmicos ( <code>np.log(df / df.shift(1))</code> ) en lugar de retornos simples para análisis y agregaciones, ya que el retorno simple puede ser engañoso al calcular medias.
<b>2.3 Manejo de Valores Faltantes</b>	<b>drop na y fillna</b>	Elimine o complete valores <code>NaN</code> (Not a Number) antes de ejecutar regresiones o modelos de ML, ya que Python requiere datos limpios. Puede usar <code>ffill</code> ( <i>forward fill</i> ) o <code>bfill</code> ( <i>backward fill</i> ) para llenar datos faltantes en series de tiempo, lo que es útil para manejar fines de semana o días festivos no operativos.

<b>2.4</b>	<b>Resampling de Frecuencia</b>	Convierta la data diaria a <b>semanal</b> o <b>mensual</b> ( <code>.resample('W').last()</code> ) según sea necesario para la estrategia o el análisis de sensibilidad a factores.
Fase 3: Modelado (ML/AI)		
Para predicciones directas (como el precio de cierre), las Redes Neuronales Recurrentes (RNN), específicamente LSTM, son altamente efectivas para series de tiempo.		
Paso	Elemento Clave	Descripción y Justificación (Fuentes)
<b>3.1 Modelo de Secuencia</b>	<b>LSTM (Long Short-Term Memory)</b>	Utilice la arquitectura <b>LSTM</b> de Keras/TensorFlow. Este modelo está diseñado para almacenar información pasada relevante y desechar información no importante, ideal para series temporales.
<b>3.2 Escalado de Datos</b>	<b>MinMaxScaler (0 a 1)</b>	Este paso es <b>crítico</b> para las redes neuronales. Escale los datos de entrada (precios) al rango de 0 a 1 utilizando <code>sklearn.preprocessing.MinMaxScaler</code> .
<b>3.3 Reformateo Tridimensional</b>	<code>np.reshape</code>	La red LSTM espera que la data de entrada sea tridimensional: (Número de Muestras, Número de Pasos de Tiempo, Número de <i>Features</i> ). Debe reformatear la data de entrada para cumplir con esta forma.
<b>3.4 Arquitectura</b>	<b>Capas y Configuración</b>	Use un modelo <code>Sequential</code> . La primera capa LSTM necesita <code>input_shape</code> . Use <code>return_sequences=True</code> solo para las capas LSTM que preceden a otra LSTM. Compile usando el optimizador <b>Adam</b> y la función de pérdida <b>Mean Squared Error (MSE)</b> .
<b>3.5 Evaluación del Modelo</b>	<b>RMSE</b>	La precisión del modelo se evalúa mediante el <b>Root Mean Squared Error (RMSE)</b> . Un valor más bajo indica un mejor ajuste a los datos de prueba no vistos.
Fase 4: Desarrollo de Estrategias y Backtesting		
Implemente la lógica de <i>backtesting</i> con un enfoque en la validación, similar a cómo se usa <b>VectorBT</b> y <b>Zipline Reloaded</b> .		
4.1 Backtesting Riguroso (Evitar Overfitting)		
Elemento Clave	Metodología	Cita
<b>Validación Estadística</b>	Utilice la librería <b>VectorBT</b> para ejecutar millones de simulaciones de manera ultrarrápida.	
<b>Walk Forward Analysis</b>	Aplique los parámetros optimizados en la data de entrenamiento ( <i>in-sample</i> ) a la data no vista ( <i>out-of-sample</i> ).	
<b>Significancia</b>	Realice un <b>T-Test unilateral</b> (con SciPy) en el <i>Sharpe Ratio out-of-sample</i> . Solo considere la estrategia exitosa si el <b>P-value es menor a 0.05</b> .	
<b>Fuga de Datos (Data Leakage)</b>	Cuando calcule <i>features</i> (como volatilidad), use "bloques de días no superpuestos" (non-overlapping chunks) para evitar el sesgo causado por la memoria del precio.	
4.2 Implementación de Estrategias (Zipline / Lógica)		
• <b>Factor Customizado:</b> Defina un factor (ej. volatilidad semanal) usando la <b>Custom Factor machinery</b> de Zipline.		

- **Pipeline API:** Use la *Pipeline API* para definir el universo negociable (ej. activos con mayor volumen promedio en dólares) y seleccionar el *quantile* deseado (ej. el grupo de criptos con menor volatilidad).
- **Lógica de Negociación:** Programe la función `rebalance` para que se ejecute en un periodo definido (ej. mensualmente). La lógica debe incluir la verificación de que el activo es negociable (`can trade`), liquidación de posiciones no deseadas, y la construcción de la cartera, idealmente con **peso igual (equal weight)**.
- **Ajuste por Sesgo:** Para estrategias de *crossover* (como SMA), considere implementar un **sesgo largo (long bias)** (posición 1 o 0) en lugar de largo/corto (1 o -1), ya que esto puede **reducir la desviación estándar (riesgo)** y mejorar los retornos totales.

Fase 5: Gestión de Riesgos y Evaluación de Rendimiento

El autor enfatiza que debe mirar todas las métricas, no solo el P&L o el *Sharpe Ratio*.

Elemento Clave	Descripción y Cálculo	Cita
<b>Tear Sheet Completo</b>	Use <b>Pyfolio Reloaded</b> con una sola línea de código ( <code>pf.create_full_tear_sheet</code> ) para obtener un análisis profundo de rendimiento y riesgo, incluyendo <i>drawdowns</i> , <i>skew</i> , <i>kurtosis</i> (que mide las colas de distribución).	
<b>Drawdowns</b>	Calcule el <i>drawdown</i> a partir de la diferencia entre el <b>Máximo Acumulativo (cumulative max)</b> y los <b>Retornos Acumulativos</b> . Mida el <i>drawdown</i> máximo en términos de valor monetario y porcentual, ya que es crucial para la psicología del <i>trader</i> .	
<b>Hedging (Factores de Riesgo)</b>	Utilice regresión <b>OLS</b> ( <code>statsmodels.api.OLS</code> ) para modelar los retornos en exceso de su portafolio contra factores de riesgo conocidos (aunque Fama-French es para acciones, debe encontrar factores proxy para cripto). Los coeficientes beta resultantes determinan los <b>pesos de cobertura (hedge weights)</b> necesarios para neutralizar la exposición a ese factor.	
<b>Métricas Clave</b>	<b>Sharpe Ratio</b> (Retorno ajustado al riesgo), <b>Calmar Ratio</b> (Compara curva de capital vs. <i>drawdown</i> ), <b>Alpha</b> (Retornos no explicados por el <i>benchmark</i> ) y <b>Beta</b> (Sensibilidad respecto al <i>benchmark</i> , como el SPY).	

Fase 6: Arquitectura y Automatización (POO y Agentes AI)

Para la escalabilidad y profesionalismo, la lógica de *backtesting* debe estar encapsulada.

Elemento Clave	Metodología	Cita
<b>Programación Orientada a Objetos (POO)</b>	Cree <b>Clases</b> (ej. <code>SMABackTester</code> ) con funciones internas ( <code>__init__</code> , <code>get_data</code> , <code>test_results</code> , <code>plot_results</code> ) para encapsular toda la lógica de la estrategia.	
<b>Reutilización de Código</b>	Guarde la clase en un archivo <code>.py</code> separado. Importe la clase en cualquier <i>notebook</i> o <i>script</i> posterior para probar la estrategia con una sola línea de código.	
<b>AI Agents (Llama Index)</b>	Integre Large Language Models (LLM, como GPT-4 o Claude) utilizando <b>Llama Index</b> . Configure el LLM con <b>tools</b> (ej. <code>Code Interpreter</code> ) que le permitan <b>escribir y ejecutar código</b>	

## Análisis de Documentos con AI

## Ejecución en Vivo

**Python** y consultar bases de datos (DuckDB) basándose en una *prompt* de texto.

Use librerías como **LangChain** o **Llama Index** junto con almacenes de vectores (*Vector Stores*) y *embeddings* para ingerir documentos grandes (ej. PDFs de 151 estrategias, o documentos financieros) y luego pedirle al Agente que razoné y responda preguntas complejas sobre ellos.

Conecte su *framework* a la API de su *broker* (aunque los ejemplos son Interactive Brokers o QuantConnect, el principio se aplica a cualquier plataforma cripto que ofrezca API) para enviar órdenes de *trading* automáticamente.

---

### Resumen y Metáfora

Para tener un *framework* exitoso en el mercado de criptomonedas, debe **maximizar la velocidad de la prueba y minimizar la probabilidad de engañarse a sí mismo**.

Su *framework* no solo debe ser un coche rápido (Python y librerías de alto rendimiento como Polars/DuckDB), sino también un **laboratorio científico** (Validación Estadística con VectorBT) y un **asistente de investigación avanzado** (Agentes de AI de Llama Index).

Si el *backtesting* es como construir una flecha (la estrategia), la **validación estadística es asegurarse de que esa flecha vuele en el aire real** (datos futuros no vistos) y no solo en el ambiente controlado de su simulador.