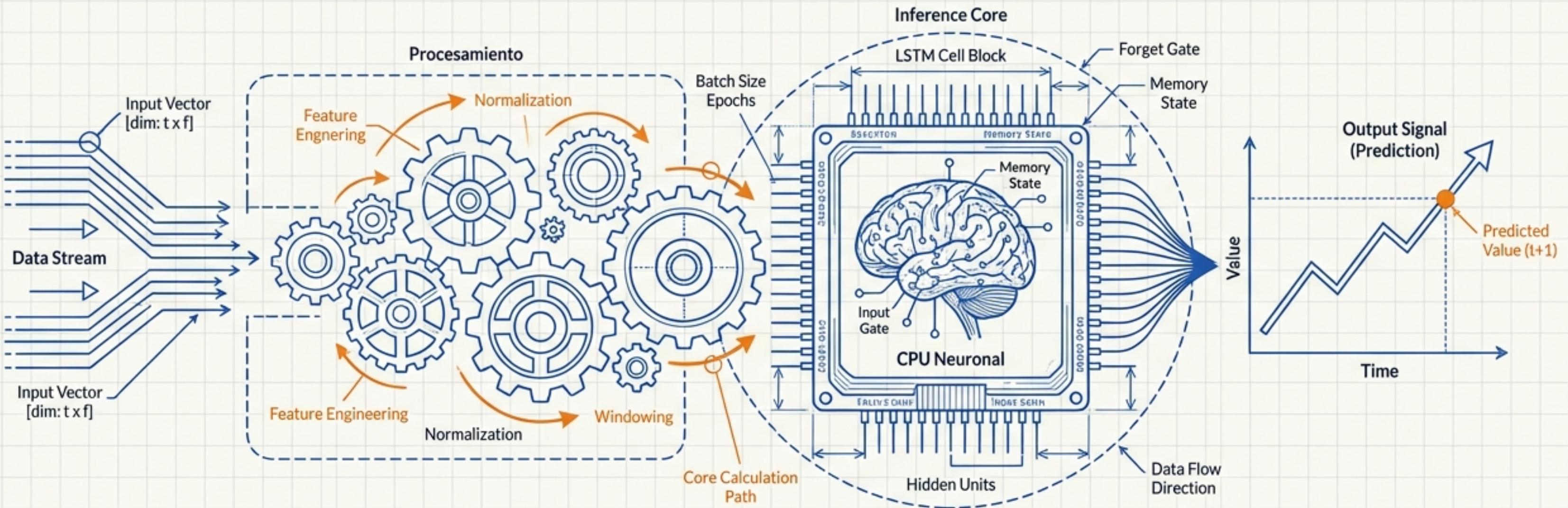


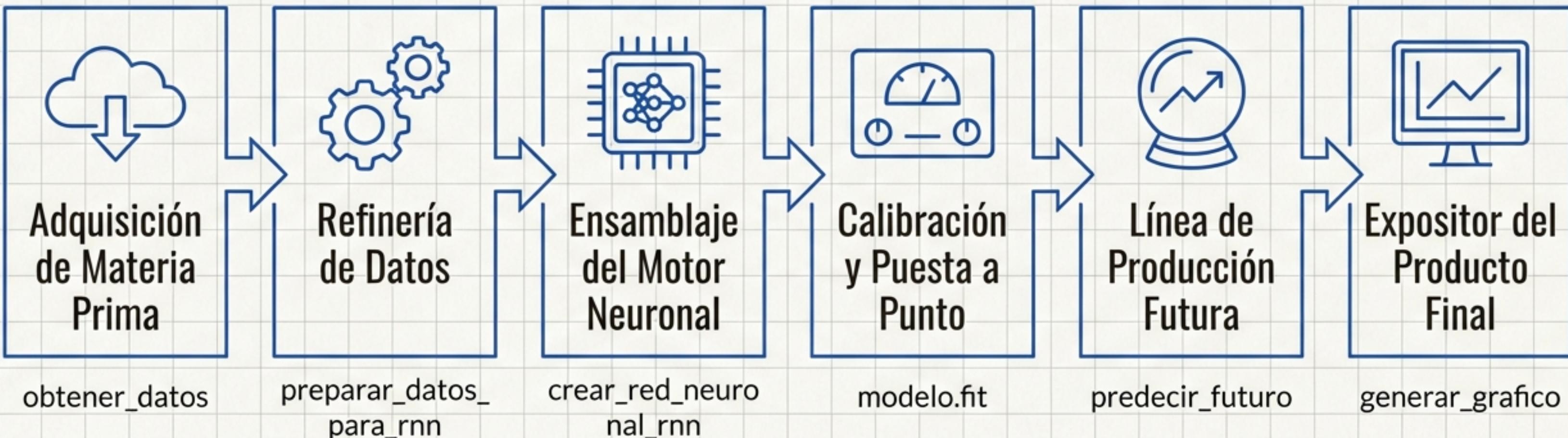
CONSTRUYENDO UNA FÁBRICA DE PREDICCIONES

Un Deep Dive en la Arquitectura de un Modelo LSTM para Series Temporales



Basado en un análisis de un modelo de pronóstico de precios de acciones implementado en Python con Keras y TensorFlow.

EL PLANO MAESTRO DE LA FÁBRICA



EQUIPANDO EL TALLER Y FIJANDO LAS ESPECIFICACIONES

LA CAJA DE HERRAMIENTAS (BIBLIOTECAS)



Cada proyecto comienza con las herramientas adecuadas.

Importamos bibliotecas especializadas para:

- **Numpy:** Operaciones matemáticas de alta velocidad (arrays de C++).
- **Pandas:** Manipulación de datos estructurados (DataFrames para series temporales).
- **yfinance:** Extracción de datos financieros desde la API de Yahoo.
- **TensorFlow/Keras:** El cerebro digital para construir y entrenar la red neuronal.
- **Scikit-learn:** Utilidades de pre-procesado como 'MinmaxScaler'.
- **Matplotlib:** Visualización y generación de gráficos.

GARANTIZANDO LA REPRODUCIBILIDAD (DETERMINISMO)

Especificación Técnica: `seed = 42`

El entrenamiento de IA es estocástico (aleatorio). Sin una 'semilla' fija, los mismos datos producirían resultados diferentes en cada ejecución.

```
seed = 42
```

```
os.environ['PYTHONHASHSEED'] = str(seed)
```

```
random.seed(seed)
```

```
np.random.seed(seed)
```

```
tf.random.set_seed(seed)
```

--- CRÍTICO: Fija los pesos iniciales de las neuronas.

Esto asegura que nuestra 'fábrica' produzca exactamente el mismo resultado bajo las mismas condiciones, siempre.

ETAPA 1: ADQUISICIÓN DE LA MATERIA PRIMA DIGITAL

obtener_datos(empresa, periodo='2y')

Panel de Decisiones de Ingeniería

Decisión 1:

Hiperparámetro: `periodo='2y'

¿Por qué 2 años? Proporciona ~504 puntos de datos, un balance ideal.

- **Riesgo de < 1 año:** El modelo memoriza el ruido en lugar de aprender patrones (Overfitting).
- **Riesgo de > 5 años:** El mercado cambia de 'régimen' (ej. tipos de interés). Datos muy antiguos pueden confundir al modelo sobre la dinámica actual.

Decisión 2:

Parámetro Financiero: `auto_adjust=True`

- **Problema:** Los 'Splits' y 'Dividendos' crean saltos artificiales en el precio (ej. una acción de \$200 pasa a \$50 tras un split 4:1).
- **Solución:** Este parámetro ajusta los precios históricos para crear una serie de tiempo continua y realista, esencial para el análisis de patrones.



ETAPA 2 (PARTE A): LA REFINERÍA DE DATOS - NORMALIZACIÓN

preparar_datos_para_rnn()

Las redes neuronales, como `LSTM`, operan de manera óptima con valores pequeños, típicamente entre 0 y 1.



La Lógica Matemática

- Las "funciones de activación" (como Tanh o Sigmoid) dentro de las neuronas se "saturan" con valores grandes, lo que detiene el aprendizaje.
- `MinmaxScaler` reescaliza cada precio a una posición relativa entre el mínimo y el máximo histórico del set de datos.

Snippet de Código Anotado

```
scaler = MinMaxScaler(feature_range=(0, 1))
datos_array = datos.values.reshape(-1, 1)
datos_escalados = scaler.transform(datos_array)
```

Scikit-learn requiere una columna vertical.

ETAPA 2 (PARTE B): LA REFINERÍA DE DATOS - CREANDO MEMORIA

LA VENTANA DESLIZANTE (SLIDING WINDOW)

Convertimos una secuencia lineal en pares de [ENTRADA] -> [SALIDA] para que el modelo pueda aprender.



Panel de Decisión de Ingeniería

Hiperparámetro: ventana=60 (Time Steps)

- **Significado:** Es la “memoria a corto plazo” de la red. Usa los últimos 60 días para predecir el siguiente.
- **Lógica de Negocio:** 60 días laborables equivalen a un trimestre fiscal, un ciclo natural en los mercados financieros donde se reportan resultados (Earnings Season).

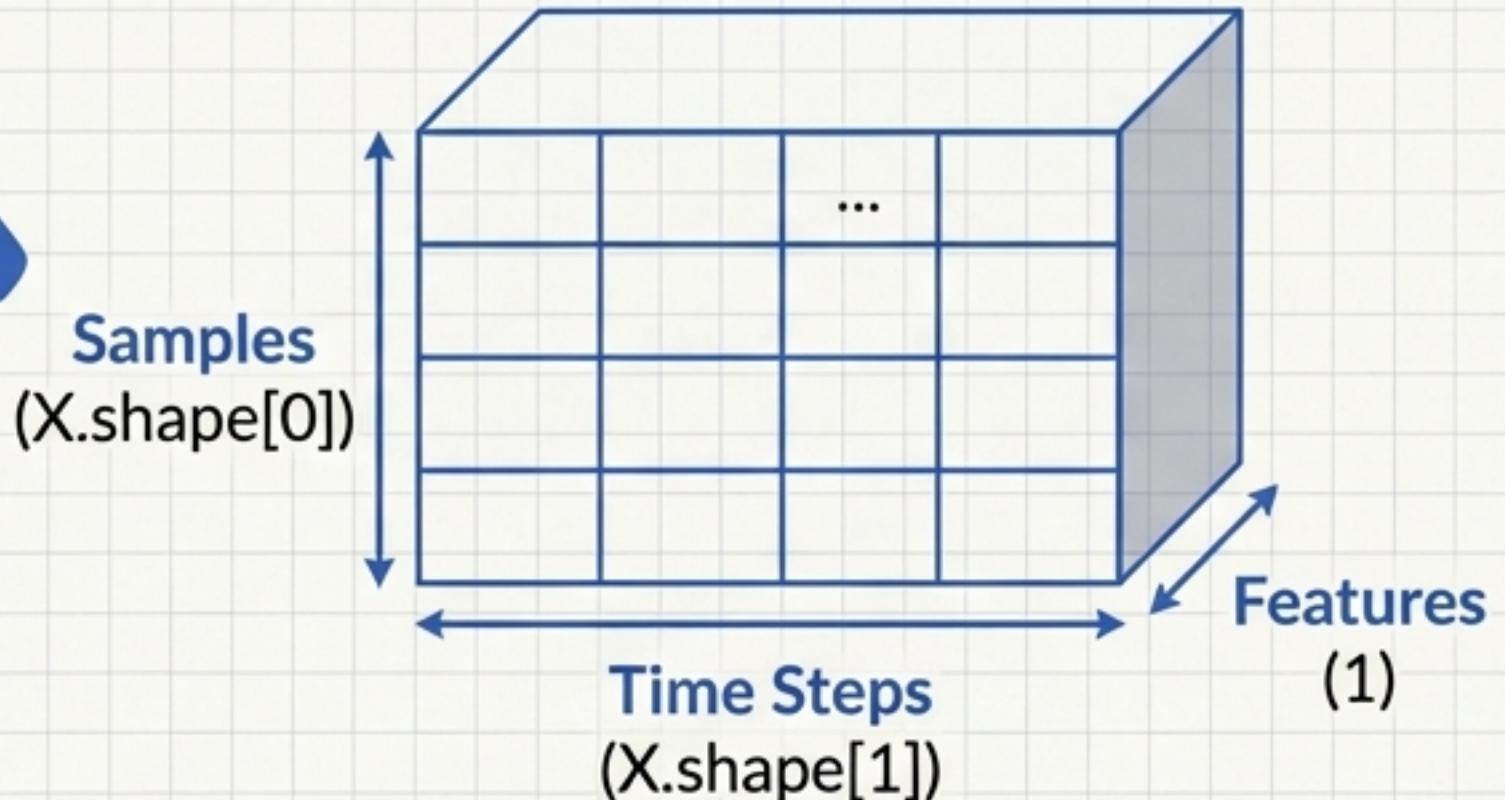
ETAPA 2 (PARTE C): LA REFINERÍA DE DATOS - EMPAQUETADO PARA EL MOTOR

La arquitectura LSTM de Keras/TensorFlow no acepta una tabla 2D. Requiere un 'cubo' de datos tridimensional (un Tensor 3D).

Forma 2D: (Samples, Time Steps)

	Día 1	Día 2	...	Día 60
Ventana 1				
Ventana 2				
...				

np.reshape



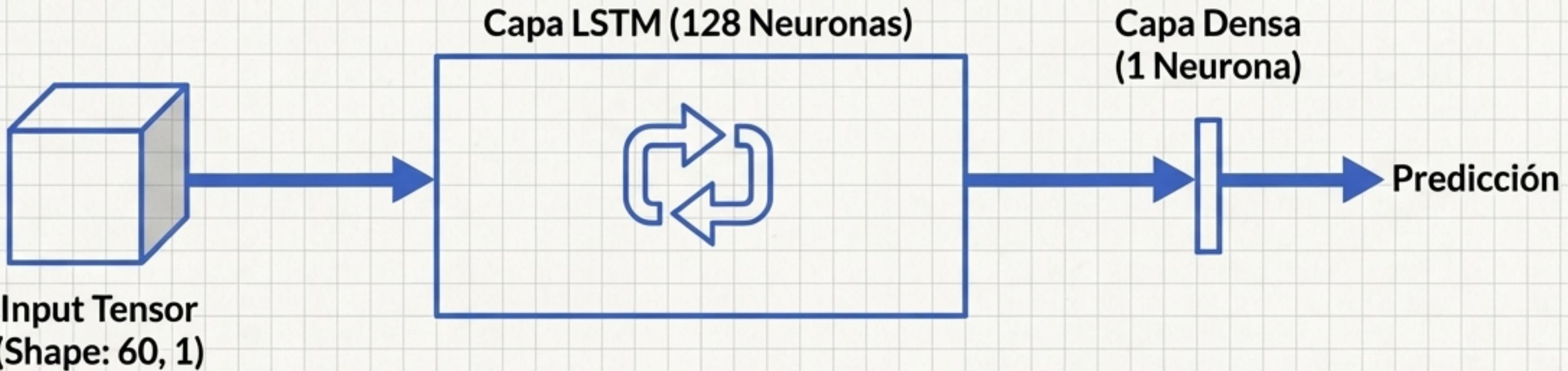
La Arquitectura del Tensor LSTM (Forma Requerida: `(Samples, Time Steps, Features)`)

- **Samples `(X.shape[0])`**: El número total de ventanas de 60 días que hemos creado.
- **Time Steps `(X.shape[1])`**: El tamaño de nuestra ventana o 'memoria'. En este caso, 60.
- **Features `(1)`**: El número de variables que observamos por día. Aquí, solo el precio de cierre ('Close').

```
# X_train shape antes: (444, 60)
X = np.reshape(X, (X.shape[0], X.shape[1], 1))
# X_train shape después: (444, 60, 1)
```

ETAPA 3: ENSAMBLANDO EL MOTOR DE INFERENCIA (LSTM)

`crear_red_neuronal_rnn()`



1. Capa LSTM (`units=128`): El corazón del modelo.

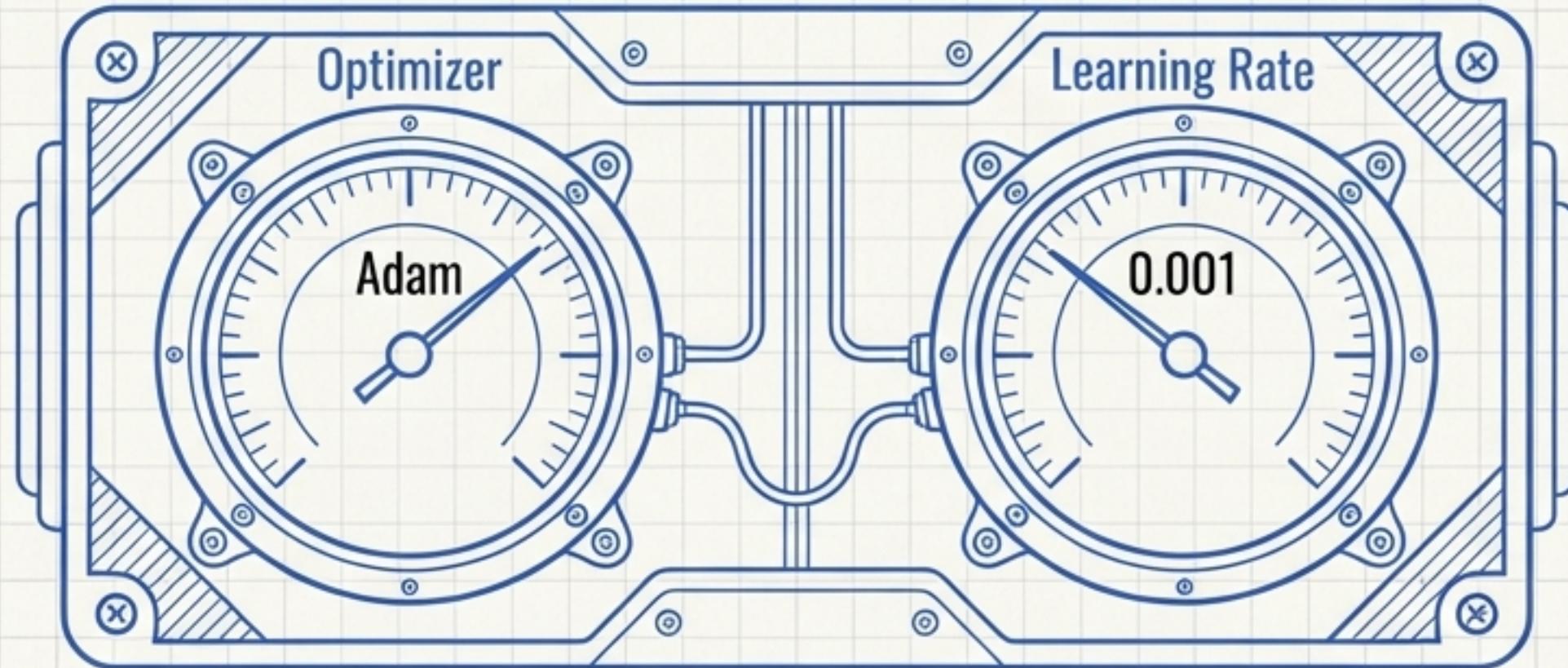
- **¿Qué es?**: Una Red Neuronal Recurrente con una "memoria" interna (Cell State) que le permite recordar patrones a largo plazo, a diferencia de una RNN simple.
- **¿Por qué 128 neuronas?**: Es una medida de la "capacidad" del modelo para aprender funciones complejas. Además, números como 32, 64, 128 son eficientes para el procesamiento en paralelo de las GPUs.

2. Capa Densa (`units=1`): La salida.

- **Función:** Conecta las 128 neuronas de la capa LSTM a una única neurona final.
- **¿Por qué 1 neurona?**: Porque nuestro objetivo es predecir un único valor: el precio del día siguiente.

EL PANEL DE CONTROL: COMPILANDO Y AJUSTANDO EL MOTOR

Función Clave: `modelo.compile()`



Panel de Especificaciones de Entrenamiento

Optimizador: 'Adam' (Adaptive Moment Estimation)

Metáfora: Imagina una bola rodando por una montaña de errores para encontrar el punto más bajo. Adam ajusta inteligentemente la velocidad (learning rate) e inercia (momentum) de la bola para encontrar el mínimo de manera eficiente.

Hiperparámetro: 'learning_rate=0.001'

Texto: ¿Qué es?: El tamaño del 'paso' que da la bola en cada iteración.

- **Muy grande:** La bola puede saltar por encima del valle y nunca encontrar el mínimo.
- **Muy pequeño:** El entrenamiento será extremadamente lento y puede quedarse atascado en mínimos locales.

```
modelo.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error')
```

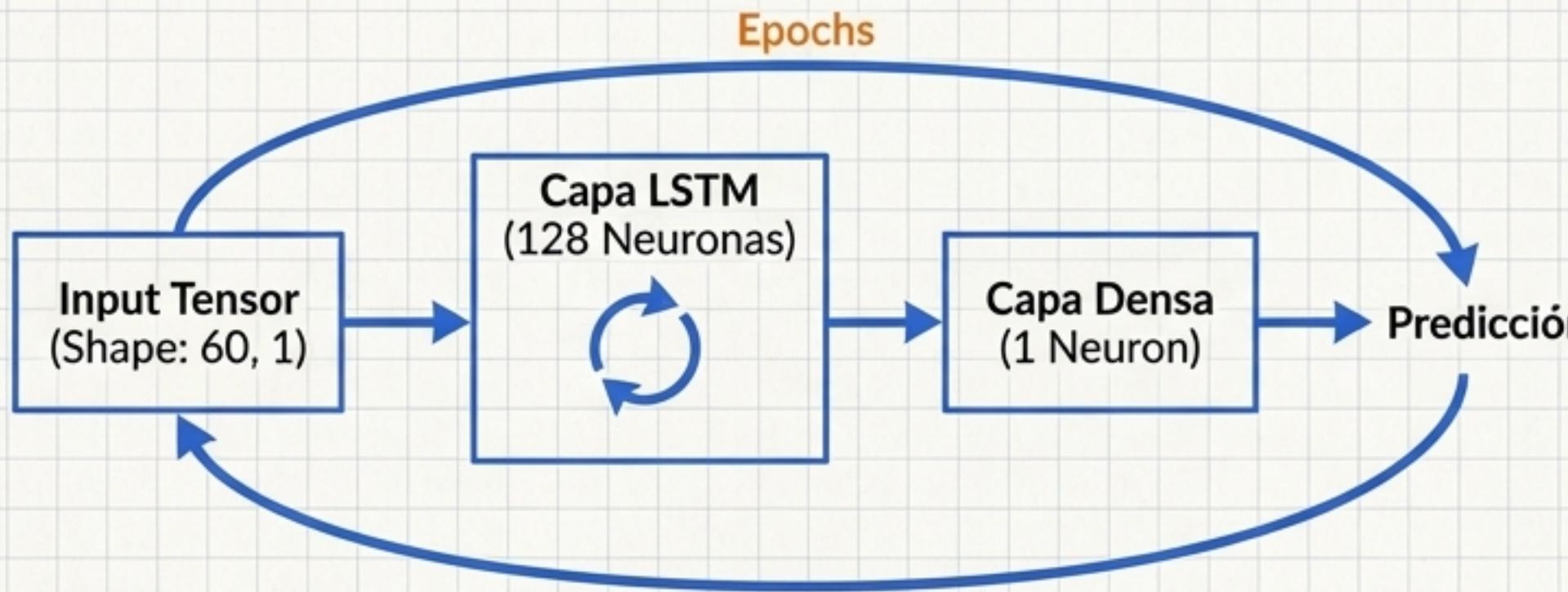
Algoritmo de Optimización

Tasa de Aprendizaje

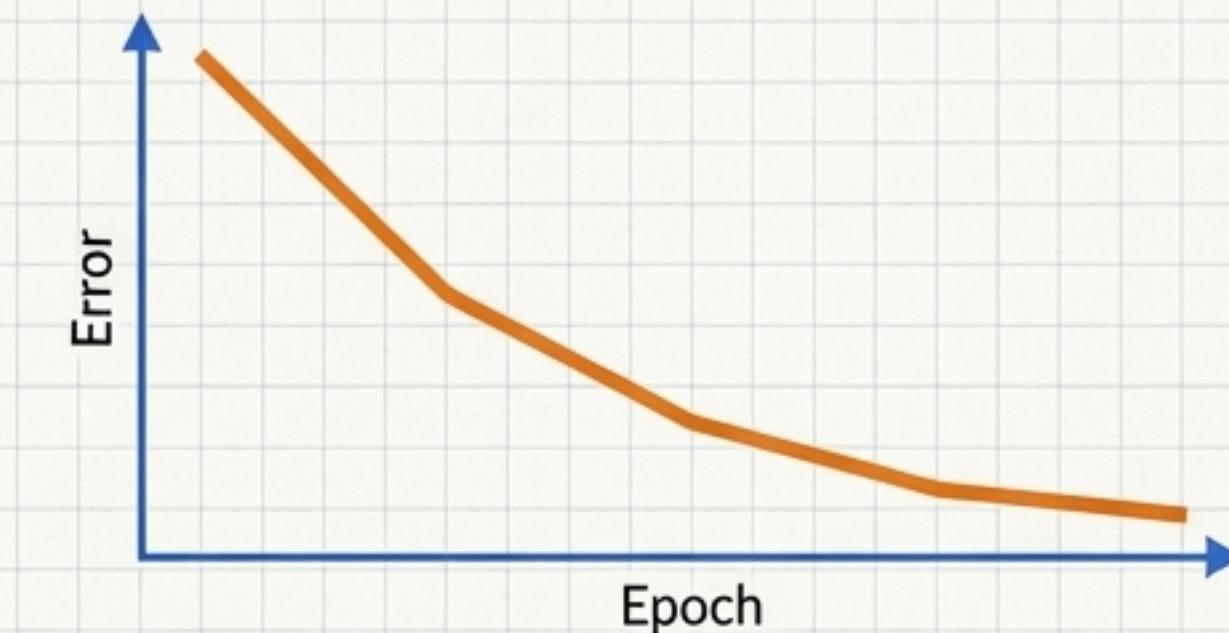
ETAPA 4: CALIBRACIÓN Y PUESTA A PUNTO DEL MOTOR

Función Clave: `modelo.fit()`

El entrenamiento es el proceso donde el modelo itera sobre los datos históricos para minimizar su error de predicción, ajustando los pesos de sus neuronas.



Error de Entrenamiento (Loss)



Parámetros Críticos del Entrenamiento

Hiperparámetro: `epochs=50`

Una 'epoch' es una pasada completa por todo el conjunto de datos de entrenamiento. El modelo 'lee el libro de historia' 50 veces para aprender sus lecciones.

Parámetro de Integridad: `shuffle=False`

! ¡CRÍTICO PARA SERIES TEMPORALES!

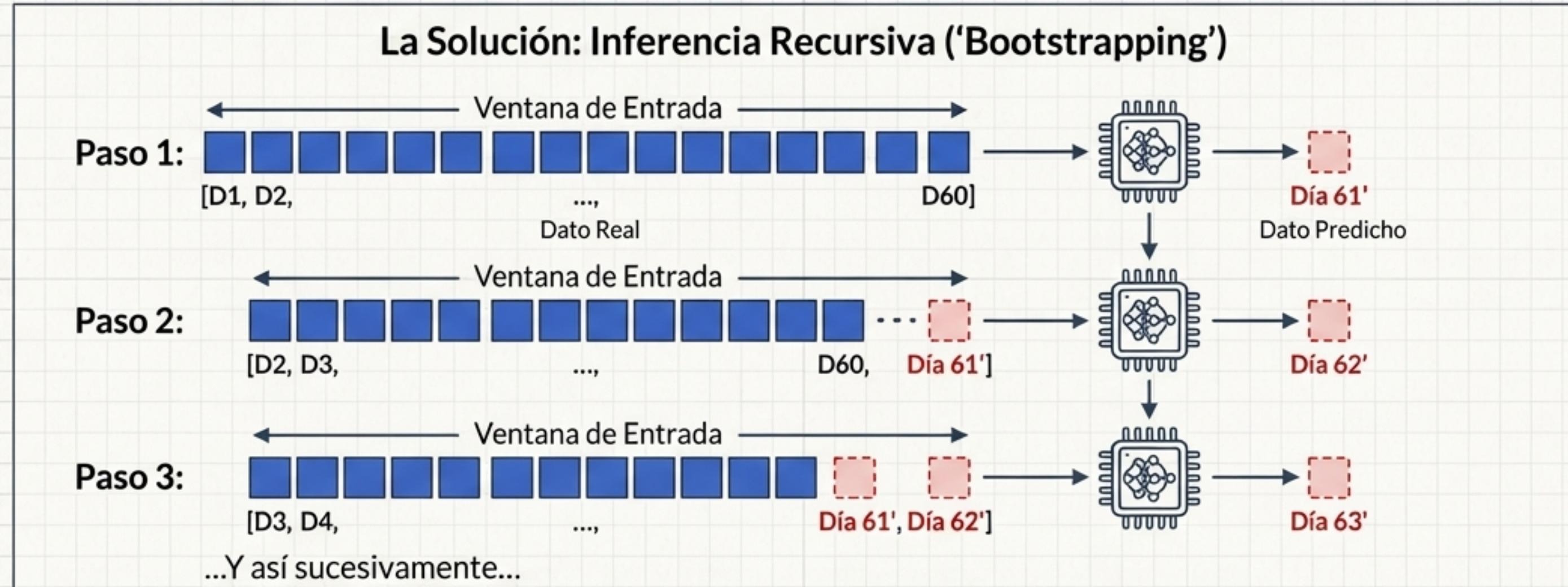
Si `shuffle=True` (el default): Los datos se barajarían. El modelo intentaría predecir el precio del martes basándose en datos del viernes, rompiendo la secuencia temporal.

Con `shuffle=False`: Mantenemos el estricto orden cronológico, asegurando que el modelo aprenda la dependencia del tiempo.

ETAPA 5: LA LÍNEA DE PRODUCCIÓN - GENERANDO EL FUTURO

El Desafío: Para predecir el día T+2, el modelo necesita el dato real del día T+1 en su ventana de entrada. Pero... ¡no tenemos datos reales del futuro!

Función Clave: `predecir_futuro()`



Snippet de Lógica Clave

```
# Elimina el día más antiguo  
secuencia_actual = secuencia_actual[1:]  
# Añade la nueva predicción al final  
secuencia_actual = np.append(secuencia_actual, [[prediccion_dia]])
```



ADVERTENCIA DE INGENIERÍA: EL EFECTO DEL "TELÉFONO ROTO"

La Inferencia Recursiva tiene un coste: **el error se capitaliza.**



- La predicción para el Día 1 se basa en 60 días de datos 100% reales. Es la más fiable.
- La predicción para el Día 2 se basa en 59 días reales y 1 día predicho (que ya tiene un pequeño error).
- La predicción para el Día 60 se basa en una ventana que contiene 59 días predichos, cada uno con su propio error acumulado.

Conclusión Práctica: La confianza en la predicción disminuye exponencialmente a medida que nos adentramos en el futuro. Este modelo es más útil para pronósticos a corto plazo que a largo plazo.

ETAPA 6: EL EXPOSITOR DEL PRODUCTO TERMINADO



Componentes de la Visualización

- Datos Históricos:** Se trazan los últimos 120 días de datos reales para dar contexto. (`datos_cierre.iloc[-120:]`)
- Eje de Tiempo Futuro:** Se generan fechas futuras que son solo días laborables, saltando fines de semana, usando `pandas.bdate_range`.
- Línea de Predicción:** Se traza la predicción de la IA (`predicciones`) contra las nuevas fechas futuras.

Cita del Código

```
plt.plot(datos_recientes.index,  
         datos_recientes.values,  
         label='Histórico Real', color='blue')  
  
plt.plot(fechas_futuras, predicciones,  
         label='Predicción IA', color='red',  
         linestyle='--')
```

color='re'

NotebookLM

DETALLES DE INGENIERÍA AVANZADA: OPTIMIZANDO LA FÁBRICA

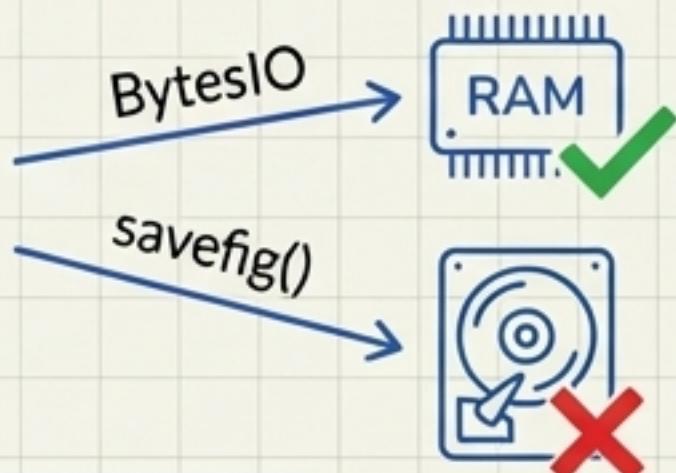
Un buen diseño no solo se enfoca en el modelo, sino también en la eficiencia y robustez del código que lo rodea.

Procesamiento en Memoria vs. Disco

- **Técnica:** Uso de `io.BytesIO` para la serialización del gráfico.
- **Problema:** Escribir archivos temporales en disco (ej. `plt.savefig('grafico.png')`) es lento y problemático en entornos de servidor (conurrencia, permisos).
- **Solución:** `BytesIO` crea un “archivo virtual” en la memoria RAM. Es extremadamente rápido y evita por completo el acceso al disco duro.

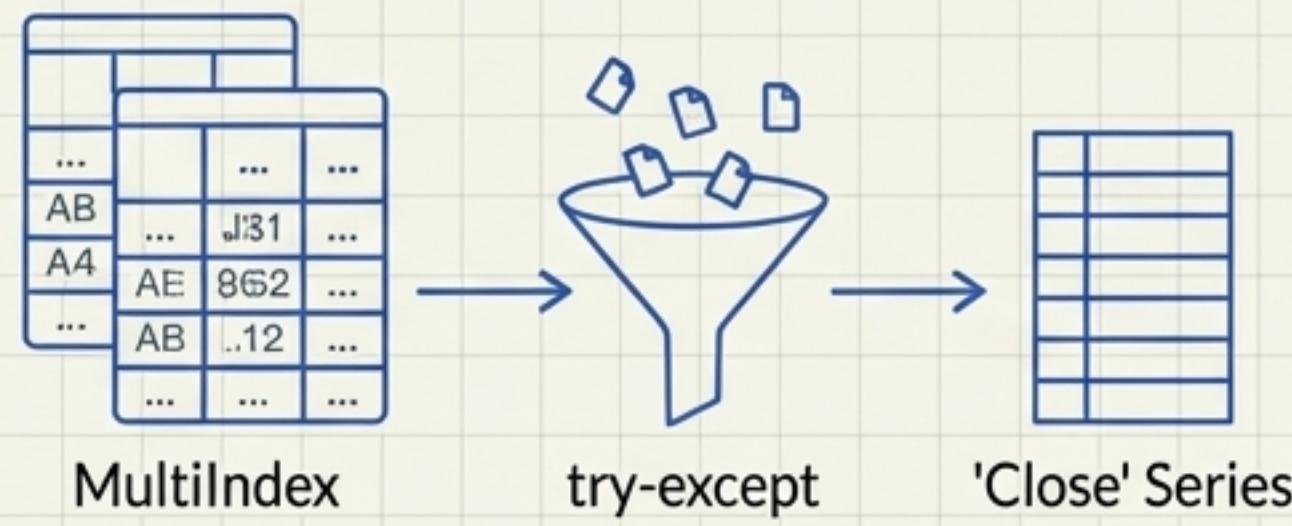
Matplotlib

```
img = BytesIO();
plt.savefig(img, format='png');
img.seek(0)
```

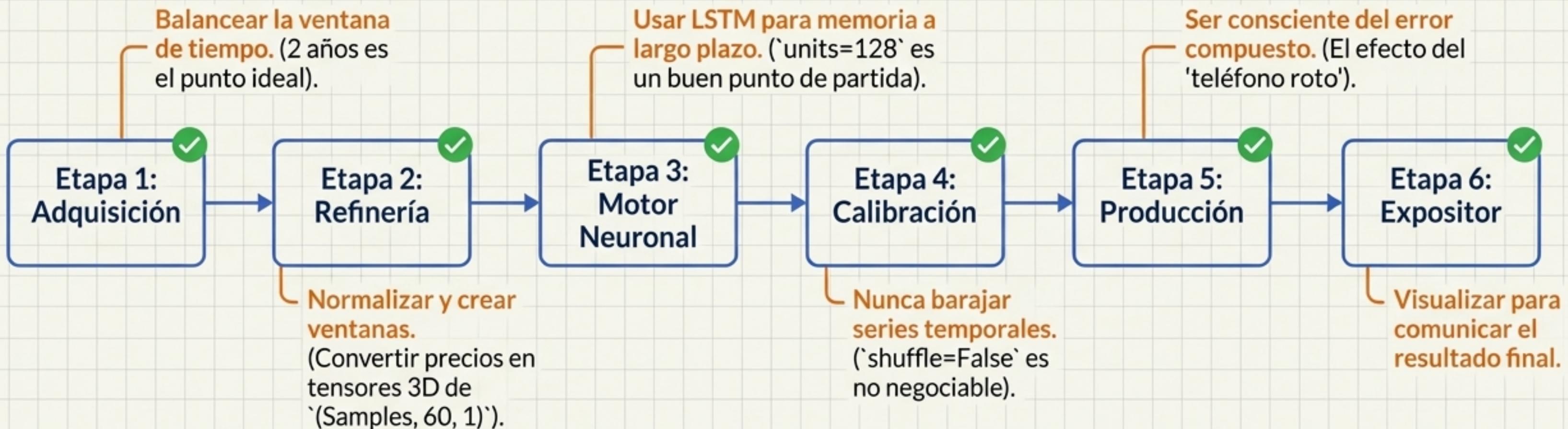


Código Defensivo contra Cambios de API

- **Contexto:** En `obtener_datos`, la API de `yfinance` a veces cambia su formato de salida.
- **Problema:** Un cambio inesperado en la estructura de datos (ej. un `MultiIndex` de Pandas) podría romper todo el pipeline.
- **Solución:** El código incluye una lógica de `try-except` para aplanar la tabla de datos de manera robusta, asegurando que siempre se extraiga la columna 'Close' sin importar el formato.



LA FÁBRICA EN OPERACIÓN: RESUMEN DEL PROCESO



La construcción de un modelo de pronóstico efectivo es un proceso de ingeniería deliberado, donde cada decisión, desde la limpieza de datos hasta la elección de hiperparámetros, contribuye directamente a la calidad del producto final.