Trading retail

Es la operativa en mercados financieros realizada por personas individuales (no instituciones), usando su propio capital y plataformas públicas ofrecidas por brokers.

En otras palabras:

👉 trading hecho por traders "de a pie", no por bancos, fondos o mesas profesionales.

EA

Un **Expert Advisor** es un **programa automático de trading** que corre dentro de **MetaTrader 4 / MetaTrader 5** y que puede:

- 📊 analizar el mercado
- 🧠 aplicar una estrategia (reglas o señales)
- 🟢 **abrir, modificar y cerrar operaciones automáticamente**
- ⚠️ gestionar riesgo (stop loss, take profit, trailing, etc.)

En resumen:

👉 **Un EA es un "trader automático" programado en MQL.**

Broker API

Una **API (Application Programming Interface)** es una interfaz que permite que **tu programa (Python)**:

- consulte precios en tiempo (casi) real,
- vea tu cuenta (balance, posiciones),
- envíe órdenes de compra/venta,
- gestione órdenes abiertas,

sin usar una plataforma gráfica como MetaTrader.

Binance

Qué es

- Un **exchange de criptomonedas** (Bitcoin, Ethereum, etc.).
- No es un broker tradicional de Forex/acciones.

Con Python puedes:

- Descargar datos históricos y en tiempo real
- Enviar órdenes (market, limit, stop)
- Gestionar posiciones y balances

Se usa para:

- **Trading algorítmico en crypto**
- Bots 24/7

Pros / Contras

✅ API muy completa

✅ Ideal para automatización

❌ Alta volatilidad

❌ Riesgo de cambios regulatorios

Interactive Brokers

Qué es

- Broker **institucional/semiprofesional**.
- Acceso a acciones, ETFs, opciones, futuros, Forex.

Con Python puedes:

- Operar directamente desde scripts
- Usar datos de mercado profesionales
- Integrar ML en producción

Se usa para:

- Trading cuantitativo serio

- Portafolios multi-activo

Pros / Contras

✅ Muy profesional

✅ Costos bajos

❌ API más compleja

❌ Curva de aprendizaje fuerte

Alpaca

Qué es

- Broker **orientado a desarrolladores**.
- Principalmente acciones y ETFs de EE. UU.

Con Python puedes:

- Hacer trading algorítmico fácilmente
- Paper trading sin dinero real

Se usa para:

- Aprender y prototipar
- Trading algorítmico en acciones

Pros / Contras

✅ API simple y bien documentada

✅ Paper trading gratuito

❌ Solo mercados específicos (EE. UU.)

OANDA

Qué es

- Broker especializado en **Forex**.
- API bien documentada.

Con Python puedes:

- Operar Forex sin MetaTrader
- Acceder a datos históricos y en vivo

Se usa para:

- Trading algorítmico en Forex vía API

Pros / Contras

✅ Ideal para Forex

✅ API clara

❌ Menos popular que MT en retail

⚖️ Resumen rápido

Plataforma	Mercado	Ideal para
Binance	Crypto	Bots 24/7
Interactive Brokers	Multi-activo	Trading cuantitativo serio
Alpaca	Acciones USA	Aprender + paper trading
OANDA	Forex	Forex sin MetaTrader

🧠 ¿Cuál encaja contigo ahora?

Por todo lo que has dicho antes:

- Estás empezando **con Forex**
- Estás aprendiendo **pipeline ML**
- Luego quieres pasar a **MetaTrader**

👉 **OANDA o MT5** son los más coherentes.

Pero:

- para **aprender ejecución por API**, Alpaca es excelente,
- para **nivel pro**, Interactive Brokers.

📌 **Idea clave para cerrar**

Python + Broker API significa:

Python piensa y también ejecuta.

Python + MetaTrader (EA) significa:

Python piensa, MetaTrader ejecuta.

Ambas son válidas.

La diferencia es **infraestructura, complejidad y mercado**.

Que se aporta y que resultados da un ML

1️⃣ "Desarrollé un modelo"

✓ Correcto

Has definido una función:

$$\hat{y} = f(x_1, x_2, x_3, x_4)$$

2️⃣ "El modelo se entrena y las métricas dicen que es óptimo"

✓ Correcto

Eso significa:

- el modelo **ajusta bien el patrón histórico**
- **bajo los supuestos del entrenamiento**
- **out-of-sample** (si lo hiciste bien)

3️⃣ "Supongamos que el modelo es $Y = x_1 + x_2 + x_3 + x_4$"

✓ Correcto como abstracción

En realidad será algo tipo:

- regresión logística
- árbol

- boosting
  pero conceptualmente es una **función determinista** dada X.

4️⃣ “Leo el mercado y obtengo las X”

✓ Correcto

Esto es **feature engineering en vivo**:

- tomas la **última vela cerrada**
- calculas exactamente las mismas X
- mismo pipeline que en entrenamiento

⚠️ Aquí está el 80% de los errores en trading ML (pero tú lo estás entendiendo bien).

5️⃣ “Le paso las X al modelo y calculo Y”

✓ Correcto

Esto es **inferencia**:

$$\hat{y}_t = f(X_t)$$

6️⃣ “Y puede ser 0 o 1”

✓ Correcto

Eso es una **salida discreta** (clasificación).

7️⃣ “0 = compro, 1 = vendo”

✓ **Conceptualmente correcto**, pero aquí está el **matiz fino**:

Lo más correcto, es decir:

- el modelo devuelve **una clase o una probabilidad**
- **la estrategia traduce eso a una acción**

Ejemplo más formal:

- modelo: P(up | X) = 0.63
- estrategia:
  - si P > 0.6 → BUY
  - si P < 0.4 → SELL
  - si no → NO TRADE

Pero si tú defines:

- 0 = BUY
- 1 = SELL

👉 **eso sigue siendo una señal válida**.

8️⃣ **“Voy a la plataforma y ejecuto la operación”**

✓ Exacto

Eso es **ejecución**.

Y aquí aparece MetaTrader, OANDA, IB, etc.

NO antes.

🔑 **Entonces, ¿qué eran “las señales”?**

Ahora lo dejamos cristalino:

✓ Señal de ML

Es cualquiera de estas:

- una clase (0 / 1)
- una probabilidad
- un score continuo

✓ **Señal de trading**

Es:

- BUY / SELL / HOLD
- derivada de la señal del ML **mediante una regla**

📌 El ML **NO sabe de spreads, stops, ni tamaño de posición**. Eso es responsabilidad de la **estrategia**.

La forma correcta es:

el ML evalúa el estado del mercado;

la estrategia decide la acción;

la plataforma ejecuta.

🧩 **Dónde entra MetaTrader / OANDA ahora (sin confusión)**

Volvemos a tu último paso:

“Voy a la plataforma donde tengo el activo y ejecuto”

Eso se puede hacer de dos formas:

A) Manual (para aprender)

- miras Y
- haces clic BUY / SELL

B) Automática

- Python escribe la señal
- EA / API la lee
- se ejecuta

📌 **El ML no cambia** en ninguno de los dos casos.

✅ **Conclusión clara y definitiva**

- Sí, **eso son señales del ML**
- Sí, **tú estás razonando correctamente**
- No, **no estás confundiendo ML con ejecución**

- Lo que estás describiendo es **exactamente** el flujo correcto de trading algorítmico

Nada de lo que dijiste es incorrecto.

Solo faltaba separar explícitamente:

Modelo → Señal → Estrategia → Ejecución

Python escribe la señal

Esto significa que el resultado del ML se pondrá en un archivo texto, CVS, etc. Para que el EA lo lea dentro del archivo MQL que tiene la estrategia.

🧠 EJEMPLO COMPLETO (HIPOTÉTICO PERO REALISTA)

CONTEXTO

- Mercado: **EUR/USD**
- Timeframe: **1H**
- Capital: 10 000 USD
- Riesgo por trade: 1%

1️⃣ El MODELO (ya terminado)

Supongamos que ya entrenaste un modelo y quedó **congelado**.

🎯 Qué hace el modelo

Pregunta que responde el modelo:

“Dadas las condiciones actuales del mercado, ¿cuál es la probabilidad de que la próxima vela de 1H cierre al alza?”

📥 Variables que LEE del mercado (features)

Estas **SÍ** son las *características del modelo*:

Feature	Qué mide
`ret_1`	Retorno de la última vela cerrada
`ret_6`	Retorno acumulado últimas 6 velas
`sma_ratio`	Relación SMA(10) / SMA(30)
`vol_20`	Volatilidad rolling
`range`	Rango High–Low relativo
`body`	Tamaño del cuerpo de la vela

📌 Nota importante:

- Todas se calculan **con velas cerradas**
- Todas describen **el estado del mercado**
- **Ninguna** depende de tu dinero, broker o riesgo

2️⃣ El MODELO calcula su salida

Con esas X, el modelo calcula:

P(up | X) = 0.63

Eso es **TODO** lo que hace el ML.

❗ El modelo **NO sabe**:

- si tienes 10 000 o 1 000 USD
- si el spread es alto o bajo
- dónde pondrás el stop
- si vas a operar o no

3️⃣ La ESTRATEGIA (aquí está lo que tú confundías)

La estrategia **NO es el modelo**.

La estrategia **usa la salida del modelo**.

📐 **Reglas de estrategia (ejemplo)**

SI P(up) ≥ 0.60 → CANDIDATO A BUY

SI P(up) ≤ 0.40 → CANDIDATO A SELL

SI 0.40 < P(up) < 0.60 → NO TRADE

Además:

SI vol_20 es extrema → NO TRADE

SI spread > 1.5 pips → NO TRADE

👉 Aquí decides **CUÁNDO operar**

👉 El modelo solo evaluó el mercado

🛄 **La OPERATIVA (plataforma / ejecución)**

Ahora sí entramos en la **plataforma** (MetaTrader, OANDA, etc.).

Supón:

- P(up)=0.63
- spread = 0.9 pips
- volatilidad normal

✅ La estrategia autoriza **BUY**

💰 **Ejecución concreta**

En la plataforma haces (o el EA hace):

- **BUY EURUSD**
- Precio: mercado (o limit)
- **Stop Loss**: 1 × ATR = 25 pips
- **Take Profit**: 2 × ATR = 50 pips
- **Tamaño**:
    - riesgo = 1% de 10 000 = 100 USD
    - 25 pips → 0.4 lotes (ejemplo)

📌 Todo esto **NO existía** dentro del modelo ML.

5️⃣ Qué PASA si el modelo dice otra cosa

Caso A

P(up) = 0.52

➞ NO TRADE

➞ No haces nada

➞ No hay operación

Caso B

P(up) = 0.35

➞ SELL

➞ misma lógica, pero inversa

🔑 DÓNDE estaba tu confusión (y es normal)

Tú pensabas:

“Las características ya SON la estrategia”

La corrección es:

Las características describen el mercado.

La estrategia describe cómo actúas ante esa descripción.

Son dos niveles distintos.

🧠 Frase que te va a ordenar todo

Guárdate esto:

El modelo observa.

La estrategia decide.

La plataforma ejecuta.

🧪 Analogía rápida (estadística pura)

- Features → variables explicativas
- Modelo → estimador
- Estrategia → regla de decisión
- Operativa → implementación del estimador bajo restricciones reales

---

✅ Resumen final (sin ambigüedad)

- ✓ Las X **NO son la estrategia**
- ✓ El modelo **NO es la estrategia**
- ✓ La estrategia **USA** la salida del modelo
- ✓ La operativa **EJECUTA** la estrategia

Nada de lo que dijiste era “tonto”.

Solo estabas mezclando **capas**.

El ML me da la probabilidad para yo decidir si Comprar o Vender

🧠 Qué hace cada parte (definitivo)

🧠 Modelo ML

- Entrada: características del mercado (X)
- Salida: probabilidad / score / clase

Ejemplo:

```
P(up | X) = 0.63
```

El modelo **no decide dinero, no decide stops, no decide lotes**.

📐 Estrategia (tu decisión racional)

Tú defines algo como:

```
SI P(up) ≥ 0.60 → el contexto es favorable
SI P(up) < 0.60 → no me interesa
```

Aquí **tú decides**:

- el umbral
- cuándo NO operar
- cómo protegerte

👉 Esto es criterio, no ML.

💰 Ejecución

Si el contexto es favorable:

- defines stop, take, size
- ejecutas en plataforma

🔑 Punto sutil pero importante

El modelo **no dice "debo" en sentido operativo**, dice **"dado lo que veo, esto es más probable"**.

La palabra *"debo"* es una **decisión humana** (o de la estrategia codificada).

📌 Por qué esta separación es poderosa

Porque te permite:

- cambiar stops **sin reentrenar**
- cambiar tamaño **sin reentrenar**
- cambiar broker **sin reentrenar**
- cambiar umbral **sin reentrenar**

Si todo estuviera “dentro del modelo”, estarías atado y frágil.

🧪 Ejemplo ultra-concreto (cerramos el círculo)

Supón:

```
P(up) = 0.61
```

Tu estrategia:

- umbral = 0.60
- volatilidad normal
- spread bajo

Entonces:
👉 **decides BUY**

Si fuera:

```
P(up) = 0.55
```

👉 **decides NO TRADE**

El modelo **no cambió**,

el mercado **no cambió**,

solo **tu criterio** decidió.

🎯 Frase final para que no vuelva la confusión

Guárdate esta:

El modelo propone.

La estrategia dispone.

La plataforma ejecuta.

Y tú estás pensando exactamente así.

Un modelo de ML NO “rompe” los indicadores clásicos. Los *absorbe*, los generaliza o los ignora si no aportan información.

Ahora vamos con la explicación bien estructurada, porque aquí hay mucho mito.

1️⃣ Qué son realmente RSI, Medias, Bollinger

Antes de ML, aclaremos esto:

Un indicador técnico **no es magia**, es solo:

- una **transformación matemática** de precios y volumen
- una forma particular de resumir información pasada

Ejemplos:

- RSI → normalización del momentum
- Medias → suavizado / tendencia
- Bollinger → media + desviación (volatilidad)

👉 Todos son **features hechas a mano** (*hand-crafted features*).

2️⃣ Qué hace el ML con los indicadores

Un modelo de ML:

- **NO cree** en RSI o medias
- **NO sabe** qué es “sobrecompra”
- **NO sigue reglas fijas**

El ML solo ve:

números que pueden o no ayudar a predecir el futuro

Opciones del ML frente a indicadores

1. **Usarlos como features** (muy común)
2. **Reemplazarlos por features más básicas**

3. **Ignorarlos** si no aportan señal

3️⃣ ¿Rompe los esquemas?

Depende de qué entiendas por “romper”.

❌ No los invalida conceptualmente

RSI, medias, Bollinger **siguen siendo válidos**.

🟢 Sí rompe el esquema rígido

Por ejemplo:

- RSI > 70 → vender
- RSI < 30 → comprar

ML NO hace eso.

En vez de reglas duras, aprende algo como:

“cuando RSI está alto *y* la volatilidad es baja *y* el retorno previo fue X, la probabilidad cambia…”

Eso es **más general y más flexible**.

4️⃣ Dos formas reales de usar ML con indicadores

🅰️ ML encima de indicadores (lo más común)

Usas RSI, medias, ATR, etc. como features:

X = [RSI, SMA_ratio, BB_width, ATR, returns…]

El ML decide:

- cuándo sirven
- cuándo no
- con qué peso

👉 Ideal para empezar.

🅱️ ML en vez de indicadores

Usas:

- retornos
- rangos
- volatilidad
- microestructura

Y el modelo aprende patrones sin indicadores clásicos.

👉 Más limpio, pero requiere más datos y cuidado.

5️⃣ Qué pasa en la práctica (importante)

Muchos traders descubren que:

- indicadores "clásicos" **solos** no funcionan bien
- pero **combinados y contextualizados** con ML sí aportan

Ejemplo:

- RSI solo → ruido
- RSI + tendencia + volatilidad → señal condicional

ML **no reemplaza** la idea del indicador, reemplaza la **regla fija** del indicador.

6️⃣ Analogía estadística (muy tu terreno)

Indicadores clásicos son como:

- variables transformadas manualmente

ML es como:

- una regresión multivariada no lineal
- que decide qué variables importan y cuándo

No elimina la estadística. La generaliza.

7️⃣ Resumen claro

- ❌ ML no “mata” RSI, medias o Bollinger
- 🟢 ML los puede usar como features
- 🟢 ML aprende relaciones más complejas
- 🟢 ML elimina reglas rígidas tipo “70/30”

🎯 Frase para cerrar

Los indicadores no desaparecen con ML. Desaparecen las reglas fijas.