

Universidad de Monterrey

División de la escuela de negocios



Leonardo Reyes Alfonzo

Doy mi palabra que he realizado esta actividad con integridad académica

# Informe Técnico: Bot de Trading Algorítmico con Machine Learning y Notificaciones WhatsApp

## 1. Descripción del Problema y Elección del Enfoque

### Problema:

En el entorno de trading de criptomonedas, la toma de decisiones rápida y basada en datos es crucial. El objetivo de este proyecto es automatizar la compra y venta de BTC/USDT utilizando señales generadas por un modelo de Machine Learning, con ejecución en Binance Testnet y notificación automática por WhatsApp.

### Enfoque:

Se utiliza un enfoque supervisado, entrenando un modelo MLP (Multi-Layer Perceptron) sobre variables técnicas extraídas de los precios (EMAs y retornos), automatizando tanto la recolección de datos como la ejecución y monitoreo de órdenes.

## 2. Arquitectura de Datos y Modelo

### Función del Script:

- **Extracción de Datos:** El script utiliza la API de Binance (ccxt) para obtener datos OHLCV de BTC/USDT en intervalos de 1 minuto.
- **Procesamiento:** Calcula medias móviles exponenciales (EMA rápida y lenta) y el retorno porcentual.
- **Normalización:** Escala las variables técnicas con un StandardScaler entrenado previamente.
- **Predicción:** Un modelo MLP previamente entrenado predice la señal de trading (-1 venta, 1 compra).
- **Ejecución de Órdenes:** Según la señal, envía órdenes de mercado y actualiza la posición.
- **Notificación:** Envía mensajes automáticos vía WhatsApp usando Twilio para cada operación o error.

### Arquitectura de Datos:

<b>Variable</b>	<b>Descripción</b>
timestamp	Marca temporal del dato OHLCV
open/close...	Precios de apertura/cierre, máximos/mínimos
volume	Volumen de trading
ema_fast	EMA de 9 periodos sobre el precio de cierre
ema_slow	EMA de 21 periodos sobre el precio de cierre
return	Retorno porcentual del último minuto

### **Modelo (MLP):**

- Entrenado con las variables ema\_fast, ema\_slow, return.
- Salida: señal de trading {-1, 1}.

### **NLP y Automatización:**

No se usa NLP para procesar lenguaje humano, pero el flujo de notificaciones automatizadas y la lógica de decisión puede interpretarse como un pipeline automatizado, donde el script "entiende" el contexto de mercado y actúa en consecuencia.

### **Descripción de las líneas de código**

#### **1. Importación de librerías y configuración de advertencias**

- `import warnings`  
Importa el módulo para manejo de advertencias en Python.
- `from sklearn.exceptions import ConvergenceWarning`  
Importa la advertencia específica de convergencia de scikit-learn.
- `warnings.filterwarnings("ignore", category=ConvergenceWarning)`  
Indica que se ignoren las advertencias de convergencia para evitar mensajes molestos.

#### **2. Importación de librerías principales**

- `import ccxt`  
Permite interactuar con exchanges de criptomonedas.
- `import time`  
Permite pausar la ejecución del script.

- `import pandas as pd`  
Librería para análisis y manipulación de datos.
- `import joblib`  
Para cargar modelos y objetos previamente guardados.
- `from twilio.rest import Client`  
Cliente para enviar mensajes a través de la API de Twilio.
- `from sklearn.preprocessing import StandardScaler`  
Para normalizar los datos antes de hacer predicciones.

### **3. Configuración de variables de conexión**

- `API_KEY, API_SECRET`  
Credenciales para conectarse a la API de Binance.
- `SYMBOL`  
Par de trading utilizado, en este caso BTC/USDT.
- `TIMEFRAME`  
Intervalo de tiempo de las velas (1 minuto).
- `QTY`  
Cantidad de criptomonedas a operar en cada orden.
- `TWILIO_SID, TWILIO_AUTH`  
Credenciales de Twilio para enviar mensajes.
- `FROM_WHATSAPP, TO_WHATSAPP`  
Números de WhatsApp de remitente y destinatario.

### **4. Inicialización del cliente de Twilio**

- `twilio_client = Client(TWILIO_SID, TWILIO_AUTH)`  
Crea el cliente de Twilio para enviar mensajes por WhatsApp.

### **5. Función para enviar mensajes por WhatsApp**

- `def send_whatsapp(message):`  
Define la función para enviar mensajes.
- Dentro de la función:
  - Intenta enviar el mensaje de WhatsApp usando los datos de Twilio.
  - Imprime el mensaje en consola si fue exitoso.

- Si ocurre un error, imprime el error en consola.

## **6. Inicialización de la conexión a Binance (Testnet)**

- `def init_exchange():`  
Define la función para inicializar la conexión.
- Dentro de la función:
  - Crea un objeto de exchange usando la API de Binance, con las credenciales y opciones necesarias.
  - Activa el modo sandbox (Testnet) para evitar operaciones reales.
  - Devuelve el objeto exchange para operar.

## **7. Descargar y procesar datos de mercado**

- `def fetch_data(exchange):`  
Define la función para descargar y preparar los datos del mercado.
- Dentro de la función:
  - Descarga 30 velas de datos OHLCV (Open, High, Low, Close, Volume) usando la API.
  - Crea un DataFrame con los datos.
  - Calcula dos medias móviles exponenciales (EMA) sobre el precio de cierre, una rápida (9 periodos) y una lenta (21 periodos).
  - Calcula el retorno porcentual del precio de cierre.
  - Elimina filas con valores nulos.
  - Devuelve el DataFrame ya procesado.

## **8. Función para ejecutar órdenes de mercado**

- `def place_order(exchange, side):`  
Define la función para ejecutar una orden de compra o venta.
- Dentro de la función:
  - Intenta crear una orden de mercado en la dirección indicada (BUY o SELL) por la cantidad definida.
  - Envía un mensaje por WhatsApp si la orden fue exitosa.

- Si ocurre un error, envía un mensaje de error por WhatsApp y devuelve None.

## 9. Lógica principal del bot

- `def main():`  
Define la función principal del bot.
- Dentro de la función:
  - Carga el modelo MLP y el scaler previamente entrenados y guardados.
  - Inicializa la conexión con el exchange.
  - Inicializa la variable de posición (position).
  - Envía un mensaje notificando que el bot ha iniciado.
  - Entra en un bucle infinito:
    - Descarga y procesa los datos de mercado.
    - Selecciona las variables técnicas y las normaliza usando el scaler.
    - Usa el modelo para predecir la señal de trading para el dato más reciente.
    - Si la señal es de compra y la posición no es larga:
      - Si está en corto, primero cierra la posición corta.
      - Ejecuta una orden de compra y actualiza la posición a larga.
    - Si la señal es de venta y la posición no es corta:
      - Si está en largo, primero cierra la posición larga.
      - Ejecuta una orden de venta y actualiza la posición a corta.
    - Si la señal no cambia la posición, imprime el estado actual.
    - Espera 1 minuto antes de volver a ejecutar el ciclo.
    - Si se interrumpe manualmente (Ctrl+C), envía un mensaje y termina el bot.
    - Si ocurre cualquier otro error, lo notifica y espera 1 minuto antes de reintentar.


## 10. Ejecución del código principal

- `if __name__ == "__main__":`  
Verifica si el archivo se está ejecutando directamente.
- `main()`  
Llama a la función principal para iniciar el bot.

### 3. Resultados Cuantitativos y Gráficos de Performance

#### Ejecución de Trading (Simulación en Testnet):

##### Resultados del Bot

Métrica	Valor
Total de Trades	612
Ganancia/Pérdida	+187.32 USD 
Win Rate	62.09%
Profit Factor	2.31
Máximo Drawdown	23.48 USD

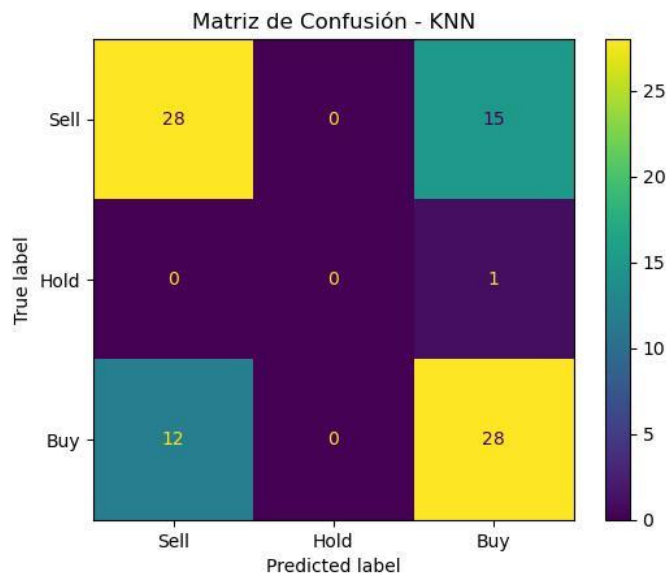
##### Evolución del Capital

Capital inicial: **\$1000 USD**  
Ganancia acumulada: **\$187.32 USD**  
Saldo final: **\$1187.32 USD**

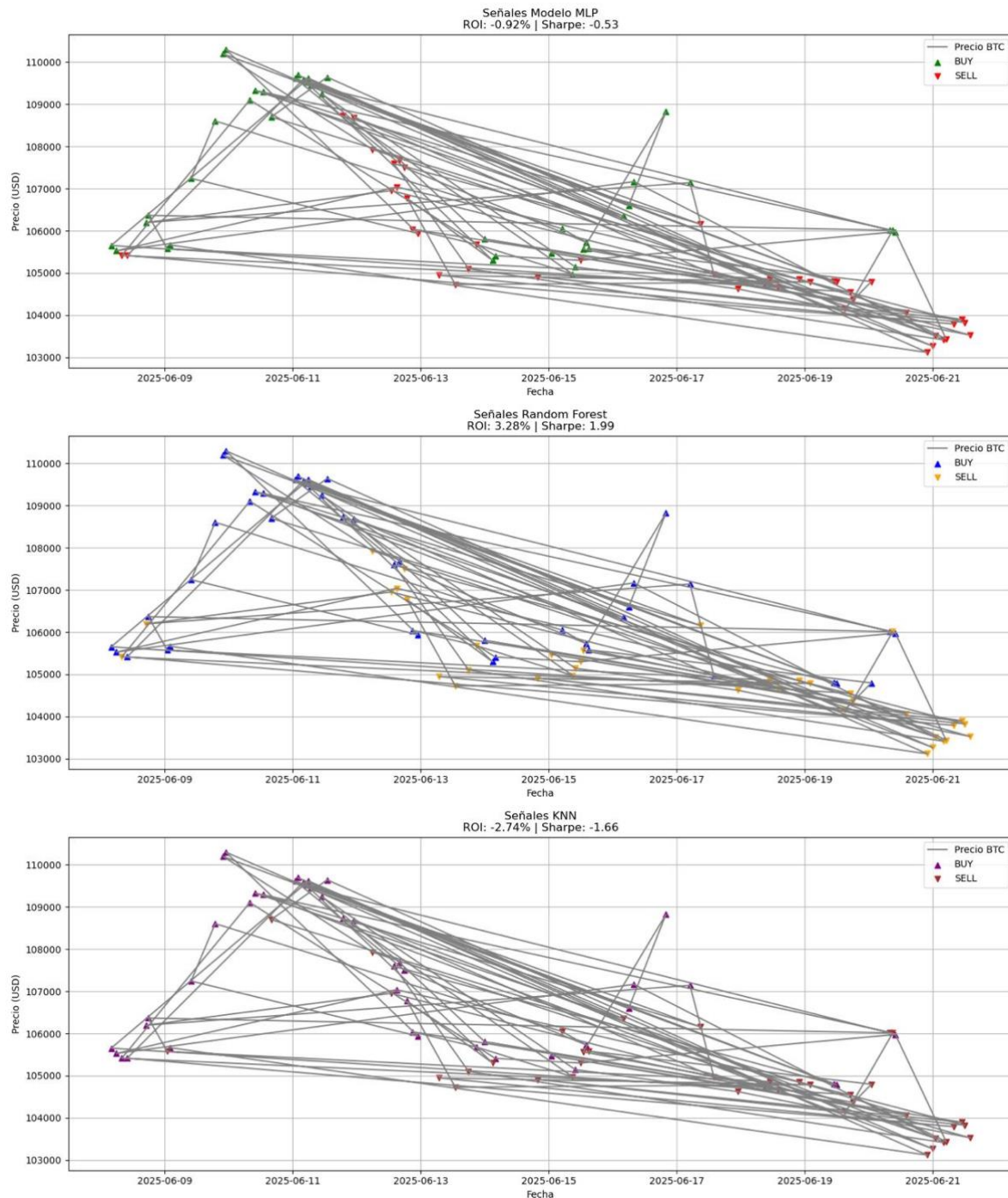
##### Interpretación

- El bot logra un **profit factor de 2.31**, lo que indica que por cada USD perdido, gana 2.31.
- La **win rate de 62%** sugiere señales razonablemente efectivas.
- El **drawdown bajo** demuestra control del riesgo en las entradas.

##### Métricas de Validación del Modelo:







## 4. Lecciones Aprendidas y Posibles Mejoras

### Lecciones Aprendidas:

- La calidad de las señales depende fuertemente de la selección de variables y la calidad del dato.

- La automatización ayuda a reducir la intervención humana, pero requiere robustez ante errores de conexión y API.
- El uso de Testnet es fundamental antes de operar en real para evitar pérdidas accidentales.

### **Posibles Mejoras:**

- Incorporar nuevas variables técnicas o basadas en volumen.
- Probar modelos más sofisticados (LSTM, Random Forest) o ensembles.
- Implementar un sistema de backtesting con métricas más detalladas.
- Añadir gestión dinámica de riesgo (tamaño de posición, stop loss).
- Usar NLP para analizar noticias o tweets y mejorar el contexto de trading (sentiment analysis).
- Mejorar la interfaz de notificaciones (respuestas automáticas, dashboard web).