Bot Cripto - Documentación Técnica

Resumen Ejecutivo

Bot Cripto es un sistema modular de predicción del mercado de criptomonedas diseñado para operar con el par BTC/USDT en intervalos de 5 minutos con un horizonte de predicción de 5 velas (25 minutos hacia adelante). El sistema emplea modelos de aprendizaje automático, detección de regímenes de mercado y gestión de riesgos para generar señales de trading automatizadas.

Arquitectura del Sistema

Pipeline de Alto Nivel

```
Recolección de Datos → Ingeniería de Features → Entrenamiento de Modelos →
Predicción Ensemble → Gestión de Riesgos → Lógica de Decisión →
Ejecución → Monitoreo y Notificaciones
```

Componentes Principales

1. Capa de Datos

- **Proveedores**: Soporte multi-fuente mediante adaptadores
  - Binance (biblioteca CCXT) - Fuente principal de datos cripto
  - yFinance - Pares de divisas (ej: EUR/USD)
- **Almacenamiento**: Formato Parquet para series temporales eficientes
- **Comando de descarga**: `bot-cripto fetch --days 30`

2. Ingeniería de Features

- **Indicadores Técnicos**:
  - RSI (Índice de Fuerza Relativa)
  - MACD (Convergencia/Divergencia de Medias Móviles)
  - ATR (Rango Verdadero Promedio)
  - Métricas de volatilidad rodante
  - Features basadas en volumen
- **Comando**: `bot-cripto features`

3. Arquitectura de Modelos

Tres Modelos Especializados:

1. **Predictor de Tendencia** (`train-trend`)
    - Predice probabilidad de movimiento direccional
    - Clasificación binaria: ARRIBA vs ABAJO
2. **Predictor de Retorno** (`train-return`)
    - Pronostica el retorno porcentual esperado
    - Proporciona estimaciones en percentiles p10, p50, p90
3. **Predictor de Riesgo** (`train-risk`)
    - Estima volatilidad y exposición al riesgo
    - Genera puntuaciones de riesgo (escala 0-1)

Arquitecturas Base:

- `BasePredictor`: Interfaz abstracta
- `BaselineModel`: Modelo de referencia simple
- `TFTPredictor`: Temporal Fusion Transformer (avanzado)

Estrategia de Ensemble:

- `WeightedEnsemble`: Combina predicciones de tendencia/retorno/riesgo
- Fusiona outputs para generar decisión final

4. Motor de Detección de Regímenes

Clasifica las condiciones del mercado en tres estados:

- **TREND** (Tendencia): Movimiento direccional fuerte
    - Trigger: ADX > `REGIME_ADX_TREND_MIN`
    - Estrategia: Seguir señales de momentum
- **RANGE** (Rango): Mercado lateral/errático
    - Trigger: ADX bajo, ATR normal
    - Estrategia: Reducir tamaño de posición o saltar
- **HIGH_VOL** (Alta Volatilidad): Volatilidad/incertidumbre elevada
    - Trigger: ATR > percentil `REGIME_ATR_HIGH_VOL_PCT`
    - Estrategia: Posicionamiento defensivo o salida

Variables de Configuración:

- `REGIME_ADX_TREND_MIN`: ADX mínimo para clasificación de tendencia
- `REGIME_ATR_HIGH_VOL_PCT`: Umbral de percentil ATR para alta volatilidad

5. Motor de Gestión de Riesgos

Dimensionamiento de Posiciones:

- Riesgo base por operación: `RISK_PER_TRADE` (predeterminado: 2% del capital)
- Tamaño máximo de posición: `MAX_POSITION_SIZE` (predeterminado: 10% del capital)
- Ajuste dinámico: `RISK_POSITION_SIZE_MULTIPLIER` escala según puntuación de riesgo

Controles de Drawdown:

- Límite diario: `MAX_DAILY_DRAWDOWN` (ej: -3%)
- Límite semanal: `MAX_WEEKLY_DRAWDOWN` (ej: -7%)
- Stop de emergencia: `HARD_STOP_MAX_LOSS` para salidas de emergencia

Bloqueo por Riesgo:

- Umbral: `RISK_SCORE_BLOCK_THRESHOLD` (ej: 0.35)
- Acción: Previene ejecución de operación si riesgo > umbral
- Flag: `risk_allowed` en output de señal

Persistencia de Estado:

- Archivo: `logs/risk_state_<symbol>.json`
- Rastrea: Drawdown actual, contador de operaciones, P&L diario/semanal

6. Lógica de Decisión

Criterios de Generación de Señales:

```python
# Lógica simplificada
if prob_up >= 0.60 and expected_return >= 0.002:
    decision = "LONG"
elif prob_up <= 0.40 and expected_return <= -0.002:
    decision = "SHORT"
else:
    decision = "HOLD"
```

Cálculo de Confianza:

- Basado en magnitud de probabilidad y estabilidad del régimen
- Mayor confianza en régimen TREND
- Menor confianza en RANGE/HIGH_VOL

Rechazo de Operaciones:

- Puntuación de riesgo excede umbral
- Probabilidad insuficiente (< 60% para LONG, > 40% para SHORT)
- Límites de drawdown superados
- Régimen incorrecto (ej: RANGE cuando solo se permite tendencia)

7. Modos de Ejecución

Paper Trading (Predeterminado: `LIVE_MODE=false`):

- Ejecución simulada con costos realistas
- Rastrea: `logs/paper_risk_state.json`
- Sin riesgo de capital real
- Spread: `SPREAD_BPS` puntos base
- Slippage: `SLIPPAGE_BPS` puntos base

Trading en Vivo (Requiere confirmación explícita):

- Token de seguridad: `LIVE_CONFIRM_TOKEN` debe coincidir
- Límite de pérdida diaria: `LIVE_MAX_DAILY_LOSS`
- Órdenes reales en exchange vía CCXT
- ⚠️ **USAR CON EXTREMA PRECAUCIÓN** ⚠️

Stop Loss & Take Profit:

- Buffer de stop loss: `STOP_LOSS_BUFFER` (ej: 1.5%)
- Buffer de take profit: `TAKE_PROFIT_BUFFER` (ej: 2.0%)

8. Monitoreo y Notificaciones

Integración con Telegram:

- Token del bot: `TELEGRAM_BOT_TOKEN`
- ID del chat: `TELEGRAM_CHAT_ID`
- Limitación de tasa: Previene spam
- Tipos de mensajes:
    - Inicio/finalización de trabajos
    - Ejecuciones de operaciones
    - Errores y advertencias

Seguimiento de Rendimiento:

- Archivo: `logs/performance_history_<symbol>.json`
- Métricas: Tasa de aciertos, ratio Sharpe, drawdown máximo
- Folds de backtest: `bot-cripto backtest --folds 4`

Detección de Drift:

- Comando: `bot-cripto detect-drift --history-file ./logs/performance_history.json`
- Propósito: Identificar degradación del modelo en el tiempo
- Acción: Activar reentrenamiento cuando se detecta drift

Dashboard Watchtower:

- Tecnología: Interfaz basada en Streamlit
- Comando: `bot-cripto dashboard --host 0.0.0.0 --port 8501`
- Características:
    - Monitoreo de señales en tiempo real
    - Gráficos de rendimiento
    - Visualización de métricas de riesgo
    - Seguimiento de versiones de modelos
- Configuración:
    - `WATCHTOWER_DB_PATH`: Ubicación de base de datos SQLite
    - `DASHBOARD_REFRESH_SECONDS`: Intervalo de auto-actualización
    - `DASHBOARD_TARGET_START`: Fecha de inicio de datos históricos

Contrato de Output de Señales

Archivo: `signal.json` (generado por inferencia)

```json
{
  "ts": "2026-02-12T17:30:00Z",          // Timestamp de señal (UTC)
  "symbol": "BTC/USDT",                  // Par de trading
  "timeframe": "5m",                     // Intervalo de vela
  "horizon_steps": 5,                    // Horizonte de predicción (25 min)

  "prob_up": 0.67,                       // Probabilidad de incremento de precio
  "expected_return": 0.008,              // Retorno esperado (0.8%)
  "p10": -0.004,                         // Percentil 10 de retorno
  "p50": 0.006,                          // Retorno mediano
  "p90": 0.013,                          // Percentil 90 de retorno

  "risk_score": 0.22,                    // Nivel de riesgo (0-1)
  "decision": "LONG",                    // LONG | SHORT | HOLD
  "confidence": 0.73,                    // Confianza de señal (0-1)
  "reason": "BUY SIGNAL: Prob 67.0% >= 60.0%, Ret 0.8% >= 0.2%",

  "regime": "TREND",                     // TREND | RANGE | HIGH_VOL
  "position_size": 0.42,                 // Fracción de capital (42%)
  "risk_allowed": true,                  // Chequeo de riesgo pasado

  "version": {
    "git_commit": "abc1234",             // Versión de código
    "model_version": "20260212T180000Z_abc1234,..." // Timestamps de modelos
  }
}
```

Flujos de Trabajo Operacionales

Flujo de Desarrollo/Pruebas

```bash
# 1. Configurar entorno
python -m venv .venv
source .venv/bin/activate
pip install -e ".[dev]"
cp .env.example .env

# 2. Descargar datos históricos
bot-cripto fetch --days 30

# 3. Generar features
bot-cripto features

# 4. Entrenar todos los modelos
bot-cripto train-trend
bot-cripto train-return
bot-cripto train-risk

# 5. Ejecutar inferencia (genera signal.json)
bot-cripto run-inference

# 6. Validar con backtesting
bot-cripto backtest --folds 4

# 7. Verificar drift del modelo
bot-cripto detect-drift --history-file ./logs/performance_history.json

# 8. Lanzar dashboard de monitoreo
bot-cripto dashboard --host 0.0.0.0 --port 8501
```

Despliegue en Producción Linux (Systemd)

Estructura de Directorios Recomendada:

```
/opt/bot-cripto/        # Código de aplicación
/etc/bot-cripto/        # Configuración (bot-cripto.env)
/var/log/bot-cripto/    # Logs y archivos de estado
```

Despliegue de Un Solo Paso:

```bash
cd /opt/bot-cripto
bash scripts/deploy_linux_native.sh
# Seguir instrucciones impresas para instalación de systemd
```

Configuración Manual:

```bash
# 1. Crear virtualenv
cd /opt/bot-cripto
python3 -m venv .venv
source .venv/bin/activate
pip install -e ".[dev]"
cp .env.example .env

# 2. Instalar servicios/timers de systemd
sudo PROJECT_DIR=/opt/bot-cripto bash /opt/bot-cripto/systemd/install_systemd.sh

# 3. Verificar instalación
systemctl status bot-cripto-inference.timer
systemctl status bot-cripto-retrain.timer
systemctl list-timers | grep bot-cripto
```

Activación Manual de Servicios:

```bash
# Forzar reentrenamiento inmediato
sudo systemctl start bot-cripto-retrain.service

# Forzar inferencia inmediata
sudo systemctl start bot-cripto-inference.service
```

Flujo de Trabajo con Docker

Construir Imágenes:

```bash
docker build -f docker/Dockerfile.train -t bot-cripto-train:latest .
docker build -f docker/Dockerfile.infer -t bot-cripto-infer:latest .
```

Ejecutar Inferencia:

```bash
docker run --rm \
  --env-file .env \
  -v ${PWD}/data:/mnt/data \
  -v ${PWD}/models:/mnt/models \
  -v ${PWD}/logs:/mnt/logs \
  bot-cripto-infer:latest
```

Prueba Rápida en Linux

Validación rápida:

```bash
source .venv/bin/activate
bash scripts/smoke_linux.sh
```

Parámetros personalizados:

```bash
SYMBOL="BTC/USDT" DAYS=30 FOLDS=4 bash scripts/smoke_linux.sh
```

Referencia de Configuración

Parámetros Críticos

Variable	Predeterminado	Descripción
SYMBOL	BTC/USDT	Par de trading
TIMEFRAME	5m	Intervalo de vela
HORIZON_STEPS	5	Horizonte de predicción (velas)
REGIME_ADX_TREND_MIN	25	ADX mínimo para régimen de tendencia
REGIME_ATR_HIGH_VOL_PCT	75	Percentil ATR para alta volatilidad
RISK_PER_TRADE	0.02	Riesgo base por operación (2%)
MAX_DAILY_DRAWDOWN	0.03	Límite de pérdida diaria (3%)
MAX_WEEKLY_DRAWDOWN	0.07	Límite de pérdida semanal (7%)
MAX_POSITION_SIZE	0.10	Tamaño máximo de posición (10%)
RISK_SCORE_BLOCK_THRESHOLD	0.35	Umbral de bloqueo por riesgo
RISK_POSITION_SIZE_MULTIPLIER	0.5-1.0	Factor de dimensionamiento dinámico
STOP_LOSS_BUFFER	0.015	Distancia de stop loss (1.5%)
TAKE_PROFIT_BUFFER	0.020	Distancia de take profit (2.0%)
HARD_STOP_MAX_LOSS	0.05	Stop de emergencia (pérdida 5%)
INITIAL_EQUITY	10000	Capital inicial (USD)
SPREAD_BPS	10	Spread bid-ask (10 bps)
SLIPPAGE_BPS	5	Slippage de ejecución (5 bps)

Ajuste de Modelos

Variable	Descripción
`ENABLE_PROBABILITY_CALIBRATION`	Habilitar/deshabilitar calibración de probabilidad
`PROBABILITY_CALIBRATION_METHOD`	Algoritmo de calibración (isotonic, sigmoid)
`TFT_CALIBRATION_MAX_SAMPLES`	Muestras máximas para calibración
`TFT_CALIBRATION_HOLDOUT_RATIO`	Ratio de holdout para validación de calibración
`MODEL_RISK_VOL_REF`	Volatilidad de referencia para modelo de riesgo
`MODEL_RISK_SPREAD_REF`	Spread de referencia para modelo de riesgo

Seguridad en Trading en Vivo

Variable	Descripción
`LIVE_MODE`	Debe ser `true` para trading real
`LIVE_CONFIRM_TOKEN`	Token secreto para prevenir accidentes
`LIVE_MAX_DAILY_LOSS`	Límite adicional de pérdida diaria para modo en vivo

Datos y Monitoreo

Variable	Descripción
`DATA_PROVIDER`	binance, yfinance, etc.
`WATCHTOWER_DB_PATH`	Ruta de base de datos SQLite
`DASHBOARD_REFRESH_SECONDS`	Intervalo de actualización de UI
`DASHBOARD_TARGET_START`	Fecha de inicio de historial del dashboard

Mejores Prácticas de Seguridad

1. Gestión de Credenciales

- ✅ Usar `.env` o `/etc/bot-cripto/bot-cripto.env` para secretos
- ✅ Establecer permisos de archivo: `chmod 600 /etc/bot-cripto/bot-cripto.env`
- ❌ Nunca hacer commit de `.env` al control de versiones
- ❌ Nunca hardcodear API keys en el código

2. Hardening de Producción

- Usar usuario de servicio sin privilegios (ej: `botcripto`)
- Restringir capacidades del servicio systemd
- Habilitar logging de auditoría para todas las operaciones
- Configurar rotación de logs para prevenir agotamiento de disco

3. Checklist de Trading en Vivo

- [ ] Paper trading validado por 30+ días
- [ ] Tasa de aciertos > 50%, ratio Sharpe > 1.0
- [ ] Drawdown máximo aceptable en escenarios de peor caso
- [ ] `LIVE_CONFIRM_TOKEN` establecido a valor seguro aleatorio
- [ ] `LIVE_MAX_DAILY_LOSS` configurado conservadoramente
- [ ] API keys del exchange limitadas solo a trading (sin retiros)
- [ ] Notificaciones de Telegram activas y probadas
- [ ] Dashboard de monitoreo accesible
- [ ] Estrategia de capital de respaldo definida

4. Recuperación ante Desastres

- Backups regulares de:
  - `/opt/bot-cripto/models/` (modelos entrenados)
  - `/var/log/bot-cripto/risk_state_*.json` (estado de riesgo)
  - `/var/log/bot-cripto/performance_history_*.json` (rendimiento)
- Documentar procedimiento de rollback
- Probar recuperación desde backup trimestralmente

Aseguramiento de Calidad

Calidad de Código

```bash
# Linting
ruff check src tests

# Formateo
black --check src tests

# Verificación de tipos
mypy src/bot_cripto

# Tests unitarios
pytest tests -v
```

Validación de Rendimiento

```bash
# Backtest con múltiples folds
bot-cripto backtest --folds 4

# Analizar métricas de rendimiento
# Verificar: Tasa de aciertos, ratio Sharpe, drawdown máximo, factor de ganancia

# Detección de drift
bot-cripto detect-drift --history-file ./logs/performance_history.json
```

Resolución de Problemas

Problemas Comunes

1. Modelos no encontrados

- Síntoma: `FileNotFoundError` durante inferencia
- Solución: Ejecutar pipeline de entrenamiento: `bot-cripto train-trend`, etc.

2. Señal no generada

- Síntoma: Archivo `signal.json` vacío o inexistente
- Solución: Revisar logs para errores, verificar disponibilidad de datos
- Causa común: Datos históricos insuficientes

3. Riesgo bloqueando operaciones

- Síntoma: `risk_allowed: false` en señal
- Solución:
  - Verificar configuración de `RISK_SCORE_BLOCK_THRESHOLD`
  - Revisar predicciones del modelo de riesgo
  - Verificar que no esté en período de drawdown

4. Servicio systemd fallando

- Revisar logs: `journalctl -u bot-cripto-inference.service -n 50`
- Verificar ruta de virtualenv en archivo de servicio
- Verificar permisos de archivo de entorno

5. Dashboard no carga

- Verificar Streamlit instalado: `pip install -e ".[ui]"`
- Verificar disponibilidad de puerto: `netstat -tuln | grep 8501`
- Revisar logs del dashboard para errores

Temas Avanzados

Despliegue en Kubernetes (Opcional)

- Manifiestos disponibles en directorio `k8s/`
- Alternativa a systemd para orquestación de contenedores
- Apropiado para clusters multi-nodo
- Requiere volumen externo para persistencia de modelos

Desarrollo de Modelos Personalizados

- Implementar interfaz `BasePredictor`
- Agregar al ensemble en configuración
- Seguir flujo de entrenamiento/inferencia
- Documentar en `docs/COMO_FUNCIONA_TODO.md`

Soporte Multi-Símbolo

- Configurar archivos de estado separados por símbolo
- Ajustar timers systemd para cada par
- Escalar infraestructura en consecuencia
- Monitorear riesgo de correlación

Recursos de Documentación

Documento	Propósito
`docs/IMPROVEMENTS_APPLIED.md`	Historial de cambios y mejoras
`docs/COMO_FUNCIONA_TODO.md`	Detalles internos del sistema
`docs/RUNBOOK_OPERACIONES.md`	Manual operacional
`docs/WATCHTOWER.md`	Guía de dashboard y monitoreo

Soporte y Mantenimiento

Tareas de Mantenimiento Regular

Diario:

- Monitorear notificaciones de Telegram
- Revisar métricas del dashboard
- Verificar ejecución de timers systemd

Semanal:

- Analizar historial de rendimiento
- Ejecutar detección de drift
- Revisar archivos de estado de riesgo
- Validar rendimiento del modelo vs backtest

Mensual:

- Reentrenar modelos con dataset expandido
- Actualizar dependencias (`pip list --outdated`)
- Revisar y ajustar parámetros de riesgo
- Hacer backup de archivos de estado críticos

Trimestral:

- Auditoría completa del sistema
- Prueba de recuperación ante desastres
- Revisión de seguridad
- Benchmark de rendimiento vs baseline

Seguimiento de Versiones

El output de señal incluye información de versión:

```
json

"version": {
  "git_commit": "abc1234",
  "model_version": "20260212T180000Z_abc1234,..."
}
```

Usar esto para correlacionar rendimiento con versiones específicas de modelo/código.

Glosario

- **ADX**: Average Directional Index - mide fuerza de tendencia
- **ATR**: Average True Range - mide volatilidad
- **Puntos Base (bps)**: 1/100 de 1% (100 bps = 1%)
- **Drawdown**: Caída de pico a valle en capital
- **Horizonte**: Número de velas hacia adelante a predecir
- **MACD**: Indicador de Convergencia/Divergencia de Medias Móviles
- **Régimen**: Clasificación de condición de mercado
- **RSI**: Relative Strength Index - oscilador de momentum
- **Ratio Sharpe**: Métrica de retorno ajustado por riesgo
- **TFT**: Temporal Fusion Transformer - arquitectura de deep learning
- **Tasa de Aciertos**: Porcentaje de operaciones rentables

Ejemplos de Uso Práctico

Escenario 1: Configuración Inicial (Enero 2025)

```bash
# Instalar en servidor Linux
ssh usuario@servidor-trading
cd /opt
sudo git clone https://github.com/tu-repo/bot-cripto.git
cd bot-cripto

# Configurar entorno
python3 -m venv .venv
source .venv/bin/activate
pip install -e ".[dev,ui]"

# Configurar variables de entorno
sudo mkdir -p /etc/bot-cripto
sudo cp .env.example /etc/bot-cripto/bot-cripto.env
sudo chmod 600 /etc/bot-cripto/bot-cripto.env
sudo nano /etc/bot-cripto/bot-cripto.env
# Editar: TELEGRAM_BOT_TOKEN, TELEGRAM_CHAT_ID, etc.

# Preparar directorios de logs
sudo mkdir -p /var/log/bot-cripto
sudo chown $USER:$USER /var/log/bot-cripto

# Descargar datos iniciales (últimos 90 días)
bot-cripto fetch --days 90

# Generar features
bot-cripto features

# Entrenar modelos iniciales
bot-cripto train-trend
bot-cripto train-return
bot-cripto train-risk

# Validar con backtest
bot-cripto backtest --folds 5

# Revisar resultados
cat logs/performance_history_BTC_USDT.json | jq '.metrics'
```

Escenario 2: Paper Trading (Febrero 2025)

bash

```bash
# Asegurar que LIVE_MODE=false en configuración
grep LIVE_MODE /etc/bot-cripto/bot-cripto.env
# Debe mostrar: LIVE_MODE=false

# Ejecutar inferencia manual
bot-cripto run-inference

# Revisar señal generada
cat signal.json | jq '.'

# Ejemplo de output esperado:
{
  "ts": "2025-02-13T15:45:00Z",
  "symbol": "BTC/USDT",
  "decision": "LONG",
  "prob_up": 0.68,
  "expected_return": 0.0075,
  "position_size": 0.38,
  "risk_score": 0.19,
  "regime": "TREND",
  "confidence": 0.71,
  "risk_allowed": true
}

# Instalar servicios systemd para automatización
sudo PROJECT_DIR=/opt/bot-cripto bash systemd/install_systemd.sh

# Verificar que timers estén activos
systemctl list-timers | grep bot-cripto
# Debe mostrar:
# bot-cripto-inference.timer - cada 5 minutos
# bot-cripto-retrain.timer - cada 24 horas

# Monitorear paper trading por 30 días
# Revisar dashboard diariamente en http://servidor:8501
bot-cripto dashboard --host 0.0.0.0 --port 8501
```

Escenario 3: Análisis de Rendimiento (Marzo 2025)

```bash
# Después de 30 días de paper trading
# Generar reporte de rendimiento
cat logs/performance_history_BTC_USDT.json | jq '.metrics | {
  win_rate: .win_rate,
  sharpe_ratio: .sharpe_ratio,
  max_drawdown: .max_drawdown,
  total_return: .total_return,
  total_trades: .total_trades
}'

# Ejemplo de output esperado:
{
  "win_rate": 0.58,
  "sharpe_ratio": 1.34,
  "max_drawdown": -0.042,
  "total_return": 0.087,
  "total_trades": 142
}

# Verificar drift del modelo
bot-cripto detect-drift --history-file logs/performance_history_BTC_USDT.json

# Si no hay drift significativo, proceder
# Si hay drift, reentrenar:
bot-cripto fetch --days 120  # Más datos históricos
bot-cripto features
bot-cripto train-trend
bot-cripto train-return
bot-cripto train-risk
```

Escenario 4: Transición a Vivo (Abril 2025 - SOLO SI VALIDADO)

```bash
# ⚠️ PRECAUCIÓN: Solo proceder si paper trading es exitoso ⚠️

# Verificar métricas mínimas:
# - Win rate > 55%
# - Sharpe ratio > 1.2
# - Max drawdown < 5%
# - Mínimo 100 operaciones en paper

# Configurar modo en vivo
sudo nano /etc/bot-cripto/bot-cripto.env

# Cambiar:
LIVE_MODE=true
LIVE_CONFIRM_TOKEN=tu-token-secreto-aleatorio-aqui
LIVE_MAX_DAILY_LOSS=0.02  # 2% pérdida diaria máxima

# Configurar API keys del exchange (solo permisos de trading)
EXCHANGE_API_KEY=tu-api-key
EXCHANGE_API_SECRET=tu-api-secret

# Reducir tamaño inicial de posición (conservador)
MAX_POSITION_SIZE=0.05  # Reducir a 5% inicialmente
RISK_PER_TRADE=0.01     # Reducir a 1%

# Reiniciar servicios
sudo systemctl restart bot-cripto-inference.service

# Monitorear CONSTANTEMENTE las primeras 48 horas
tail -f /var/log/bot-cripto/inference.log

# Revisar cada operación ejecutada
# Verificar notificaciones de Telegram
# Monitorear dashboard en tiempo real
```

Escenario 5: Mantenimiento Rutinario (Mayo 2025)

```bash
# Revisar estado semanal
systemctl status bot-cripto-inference.timer
systemctl status bot-cripto-retrain.timer

# Verificar logs de errores
journalctl -u bot-cripto-inference.service --since "1 week ago" | grep ERROR

# Limpiar logs antiguos (opcional, si no hay logrotate)
find /var/log/bot-cripto -name "*.log" -mtime +30 -delete

# Actualizar dependencias (con precaución)
source /opt/bot-cripto/.venv/bin/activate
pip list --outdated

# Hacer backup de modelos y estado
sudo tar -czf /backup/bot-cripto-$(date +%Y%m%d).tar.gz \
  /opt/bot-cripto/models \
  /var/log/bot-cripto/risk_state_*.json \
  /var/log/bot-cripto/performance_history_*.json

# Ejecutar drift detection
bot-cripto detect-drift --history-file logs/performance_history_BTC_USDT.json

# Si drift detectado, programar reentrenamiento
# Programar para horario de bajo volumen (ej: 3 AM UTC domingo)
sudo systemctl start bot-cripto-retrain.service
```

Escenario 6: Resolución de Emergencia (Junio 2025)

```bash
# EMERGENCIA: Pérdidas excesivas detectadas

# 1. DETENER INMEDIATAMENTE
sudo systemctl stop bot-cripto-inference.timer
sudo systemctl stop bot-cripto-inference.service

# 2. Revisar estado actual
cat logs/risk_state_BTC_USDT.json | jq '.'

# 3. Cerrar posiciones abiertas manualmente en exchange
# (si es necesario, usar interfaz web del exchange)

# 4. Analizar qué salió mal
tail -n 500 /var/log/bot-cripto/inference.log > /tmp/emergency_analysis.log
cat signal.json | jq '.reason'

# 5. Revisar últimas 20 operaciones
cat logs/performance_history_BTC_USDT.json | jq '.trades[-20:]'

# 6. Determinar causa raíz:
# - ¿Drift del modelo?
# - ¿Cambio drástico de régimen de mercado?
# - ¿Error en configuración?
# - ¿Bug en código?

# 7. Volver a paper trading
sudo nano /etc/bot-cripto/bot-cripto.env
# Cambiar: LIVE_MODE=false

# 8. Reentrenar modelos con datos más recientes
bot-cripto fetch --days 150
bot-cripto features
bot-cripto train-trend
bot-cripto train-return
bot-cripto train-risk

# 9. Validar con backtest exhaustivo
bot-cripto backtest --folds 10

# 10. Solo reactivar después de validación completa
```

Escenario 7: Optimización de Parámetros (Julio 2025)

```bash
# Experimentar con diferentes configuraciones de régimen

# Crear copias de seguridad de configuración actual
cp /etc/bot-cripto/bot-cripto.env /etc/bot-cripto/bot-cripto.env.backup

# Probar configuración más conservadora para mercados volátiles
nano /etc/bot-cripto/bot-cripto.env

# Ajustar:
REGIME_ADX_TREND_MIN=30            # De 25 a 30 (más selectivo)
REGIME_ATR_HIGH_VOL_PCT=70         # De 75 a 70 (más sensible)
RISK_SCORE_BLOCK_THRESHOLD=0.30    # De 0.35 a 0.30 (más restrictivo)
MAX_POSITION_SIZE=0.08             # De 0.10 a 0.08 (más conservador)

# Ejecutar backtest con nueva configuración
bot-cripto backtest --folds 5

# Comparar resultados
# Si mejora métricas de riesgo sin sacrificar mucho retorno, mantener
# Si no, revertir:
cp /etc/bot-cripto/bot-cripto.env.backup /etc/bot-cripto/bot-cripto.env
```

Escenario 8: Expansión Multi-Par (Agosto 2025)

```bash
# Agregar ETH/USDT al sistema

# 1. Crear configuración separada
sudo cp /etc/bot-cripto/bot-cripto.env /etc/bot-cripto/bot-cripto-eth.env
sudo nano /etc/bot-cripto/bot-cripto-eth.env
# Cambiar: SYMBOL=ETH/USDT

# 2. Descargar datos de ETH
SYMBOL=ETH/USDT bot-cripto fetch --days 90

# 3. Generar features
SYMBOL=ETH/USDT bot-cripto features

# 4. Entrenar modelos específicos para ETH
SYMBOL=ETH/USDT bot-cripto train-trend
SYMBOL=ETH/USDT bot-cripto train-return
SYMBOL=ETH/USDT bot-cripto train-risk

# 5. Crear servicios systemd adicionales para ETH
# (modificar archivos en systemd/ para agregar instancias -eth)

# 6. Monitorear correlación entre BTC y ETH
# Asegurar que no hay sobre-exposición
```

Métricas Clave de Éxito

Para Paper Trading (Fase 1: Meses 1-2)

Mínimos Aceptables:

- Tasa de aciertos: ≥ 52%
- Ratio Sharpe: ≥ 0.8
- Drawdown máximo: ≤ 8%
- Total de operaciones: ≥ 100

Objetivos Deseables:

- Tasa de aciertos: ≥ 58%
- Ratio Sharpe: ≥ 1.5
- Drawdown máximo: ≤ 5%
- Factor de ganancia: ≥ 1.3

Para Trading en Vivo (Fase 2: Mes 3+)

Controles Estrictos:

- Pérdida diaria máxima: -2%
- Pérdida semanal máxima: -5%
- Drawdown total máximo: -10%
- Máximo de operaciones simultáneas: 1 (inicialmente)

Señales de Alerta (requieren revisión inmediata):

- 3 pérdidas consecutivas
- Pérdida diaria > 1.5%
- Win rate < 45% en últimas 50 operaciones
- Drift score > umbral crítico

Calendario de Actividades 2025-2026

Q1 2025 (Enero - Marzo)

- **Enero**: Configuración inicial, descarga de datos, entrenamiento baseline
- **Febrero**: Paper trading intensivo, calibración de parámetros
- **Marzo**: Análisis de resultados, optimización de modelos

Q2 2025 (Abril - Junio)

- **Abril**: Transición a trading en vivo (solo si métricas alcanzadas)
- **Mayo**: Monitoreo continuo, ajustes finos
- **Junio**: Evaluación trimestral, reentrenamiento mayor

Q3 2025 (Julio - Septiembre)

- **Julio**: Implementación de mejoras basadas en datos reales
- **Agosto**: Expansión a pares adicionales (si aplicable)
- **Septiembre**: Auditoría de seguridad y rendimiento

Q4 2025 (Octubre - Diciembre)

- **Octubre**: Preparación para condiciones de mercado de fin de año
- **Noviembre**: Evaluación anual completa
- **Diciembre**: Planificación estratégica para 2026

Q1 2026 (Enero - Marzo)

- **Enero**: Reentrenamiento con datos de todo 2025
- **Febrero**: Implementación de arquitecturas mejoradas (si disponibles)
- **Marzo**: Estado actual - Validación continua del sistema

Última Actualización: Febrero 2026
Versión del Sistema: Basada en seguimiento de commits Git
Estado: Producción lista con validación de paper trading requerida
Próxima Revisión Mayor: Abril 2026