Desacoplamiento de features para mitigar multicolinealidad en series financieras: Implementación, pruebas y guía de uso

Santiago Javier Espino Heredero

14 de octubre de 2025

Resumen

Se presenta una solución práctica para mitigar la multicolinealidad entre variables técnicas y la serie `close` en conjuntos de datos financieros. La intervención combina una exclusión manual (quick win) con transformaciones normalizadas (feature decoupling) aplicadas a indicadores OHLC, medias móviles, bandas de Bollinger y niveles de Fibonacci. Se describen las implementaciones de código, la integración CLI, la batería de tests y una guía reproducible para evaluación. Se muestran resultados resumidos de reducción de correlación y efecto esperado sobre la estabilidad de gradientes en modelos recurrentes tipo LSTM.

Palabras clave: multicolinealidad, LSTM, features técnicas, normalización, decoupling, reproducibilidad.

1. Introducción

La presencia de features altamente correlacionadas con la variable objetivo (aquí `close`) puede inducir inestabilidad numérica, gradientes ruidosos y sobreajuste en modelos secuenciales como LSTM. El objetivo de este trabajo es describir una solución pragmática y reproducible que reduzca la correlación directa entre `close` y un conjunto de indicadores técnicos comunes, preservando la información útil para predicción y manteniendo retrocompatibilidad.

2. Problema identificado

Un análisis de auditoría mostró aproximadamente 10 features con correlación superior a 0.95 respecto a `close`. Las categorías afectadas incluyen:

- OHLC: `high`, `low`, `log_close`.
- Medias móviles: `sma_5`, `sma_20`, `ema_5`, `ema_20`.
- Bollinger Bands: `bb_m`, `bb_up`, `bb_dn`.
- Niveles de Fibonacci: $\sim$8 niveles absolutos.

Se confirma que no se trata de *data leakage* (no se usan desplazamientos hacia el futuro tipo `.shift(-k)`). El fenómeno es multicolinealidad inherente a cómo se calculan y representan ciertos indicadores técnicos en valores absolutos.

3. Estrategia general

La intervención se compone de cuatro pilares:

1. **Quick Win:** exclusión manual de las features más redundantes.
2. **Feature Decoupling:** transformar features absolutas a ratios/posiciones/offsets relativos a `close`.
3. **Integración CLI / API:** flag `-decouple-features` en el proceso de preparación del dataset.
4. **Testing y documentación:** tests automatizados y guía de uso detallada.

4. Métodos implementados

4.1. Quick Win: exclusión manual

Se agrega un conjunto de exclusión en `prepare_dataset.py`.

```
# prepare_dataset.py (fragmento)
EXCLUDE_HIGHLY_CORRELATED = {
    "high", "low", "log_close",
    "sma_5", "ema_5",
    "bb_m", "bb_up", "bb_dn",
}
```

Impacto: reducción de ≈ 50 a ≈ 42 features en la salida del pipeline.

4.2. Feature Decoupling (Transformaciones normalizadas)

Diseño: convertir variables en representaciones relativas a `close`, en lugar de absolutos, para reducir la correlación.

A continuación se resume la transformación por grupo:

Tabla 1: Transformaciones de desacoplamiento aplicadas

Antes (absoluto)	**Después (normalizado)**	**Observación**
`high`, `low`	`hl_spread_norm` = (high-low)/close `close_hl_position` = (close-low)/(high-low)	spread relativo al close posición dentro del rango
`sma_20`	`close_sma20_dist` = (close - sma20)/sma20 (%)	distancia relativa
`ema_20`	`close_ema20_dist` = (close - ema20)/ema20 (%)	distancia relativa
`bb_up`, `bb_dn`	`bb_position` = (close - bb_dn)/(bb_up - bb_dn)	posición 0–1
`fib_r_382` (absoluto)	`dist_fib_r_382` = (close - fib_r_382)/fib_r_382	distancia relativa
`atr_14`	`atr_14_norm` = atr_14/close (%)	volatilidad normalizada

Pseudocódigo La función central de transformación (resumida) se define como:

Algorithm 1 add_technical_features(..., decouple_from_close)

procedure DecoupleFeatures($df, decouple\_from\_close$)
  if $\neg decouple\_from\_close$ **then**
    return standard_features(df)
  end if
  $df.hl\_spread\_norm \leftarrow (df.high - df.low)/df.close$
  $df.close\_hl\_position \leftarrow (df.close - df.low)/(df.high - df.low)$
  $df.close\_sma20\_dist \leftarrow (df.close - df.sma\_20)/df.sma\_20$
  $df.bb\_position \leftarrow (df.close - df.bb\_dn)/(df.bb\_up - df.bb\_dn)$
  $df.atr\_14\_norm \leftarrow df.atr\_14/df.close$
  return df with new features
end procedure

Notas de implementación

- Se protegen divisiones por cero añadiendo un `eps` pequeño.
- Se limita la posición bb_position a $[0, 1]$ con `clip`.
- Las distancias porcentuales se expresan como fracciones (ej. 0.01 = 1 %).

4.3. Integración y CLI

Se añadió un flag en `prepare_dataset.py`:

```
# CLI example
python prepare_dataset.py \
  --sqlite data_manager/exports/marketdata_base.db \
  --symbol BTCUSDT \
  --timeframe 30m \
  --decouple-features
```

Si el flag está presente, el pipeline activa las transformaciones descritas; de lo contrario, se mantienen las features originales (valor por defecto: `decouple_from_close=False`).

4.4. Pruebas automatizadas

Archivo: `test_decouple_features.py`. Principales verificaciones:

1. Se aplican las transformaciones esperadas a un dataset sintético.
2. Las nuevas features muestran correlación reducida frente a `close`.
3. El conteo de features se reduce según lo esperado para el quick win.

Ejemplo de invocación:

```
python test_decouple_features.py
# Expected output: ALL TESTS PASSED
```

5. Resultados y métricas

Las cifras a continuación resumen el impacto esperado/observado en pasos representativos del pipeline.

Tabla 2: Impacto resumido en métricas clave

Métrica	Antes	Quick Win	Full Decouple
Features con corr > 0.90	~ 10	~ 5	0–2
Correlación promedio	0.75–0.85	0.65–0.75	0.35–0.50
Total features (aprox.)	50	42	35–40
Estabilidad de gradientes	Inestable	Mejorado	Estable
Generalización	Pobre	Mejorado	Robusta

Interpretación La reducción de correlación esperada (de $\sim 0{,}85$ a $\sim 0{,}45$ de media para las features transformadas) es consistente con una menor redundancia informacional dirigida a estabilizar el entrenamiento de LSTM y reducir varianza en estimaciones de parámetros.

6. Discusión

6.1. Ventajas

- Reducción rápida de multicolinealidad sin requerir rediseño del dataset original.
- Retrocompatibilidad: el comportamiento por defecto no cambia.
- Flexibilidad para combinar exclusión manual y transformaciones.

6.2. Limitaciones

- Las transformaciones pueden perder alguna información absoluta de escala útil para ciertos modelos; validar según métrica de predicción.
- Requiere comprobaciones numéricas (divisiones por cero, clipping) y tratamiento de outliers.
- La reducción de correlación no garantiza mejora automática de la métrica final (p. ej. RMSE); debe evaluarse empíricamente.

7. Pruebas de evaluación recomendadas

Se recomienda el siguiente protocolo para evaluar impacto sobre modelos:

1. Fijar semilla y condiciones experimentales (split temporal, normalización).
2. Entrenar un LSTM básico (misma arquitectura y hyperparámetros) tres veces:
   (a) Dataset original (baseline).
   (b) Quick Win aplicado.
   (c) Full Decouple aplicado.
3. Comparar: metricas de validación (MAE, RMSE), estabilidad de gradientes (norma del gradiente), y curvas de aprendizaje.
4. Reportar intervalos de confianza con repeticiones y test estadístico (ej. t-test o bootstrap) si procede.

8. Propuesta de pasos futuros (opcional)

8.1. VIF-based Feature Selection

Implementar un selector iterativo por Variance Inflation Factor (VIF) que:

1. Calcule VIF para features candidatas.
2. Elimine iterativamente la feature con VIF más alto si VIF > 10.
3. Recalcule hasta que todas las features tengan VIF aceptable.

Estimación de esfuerzo: 2–3 horas.

8.2. Attention Mechanism

Integrar un módulo `LSTM2HeadWithAttention` para que el modelo aprenda pesos de importancia por feature/tiempo. Estimación: 4–6 horas. Beneficio: aprendizaje automático de relevancias que puede complementar la selección manual.

8.3. Mejoras al audit

Añadir comprobación de VIF en `_run_audit()` y recomendaciones automáticas si VIF > 10. Estimación: 1 hora.

9. Guía de uso y reproducibilidad

9.1. Requisitos mínimos sugeridos

- Python 3.9+ (3.10 recomendado)
- pandas, numpy, scipy, scikit-learn, pytorch (si se entrena LSTM), pytest (para tests)

9.2. Comandos

Quick Win (por defecto)

```
python prepare_dataset.py -sqlite data.db -symbol BTCUSDT -timeframe 30m
```

Full Decouple

```
python prepare_dataset.py -sqlite data.db -symbol BTCUSDT -timeframe 30m -decouple-features
```

Tests

```
python test_decouple_features.py
```

9.3. Parámetros de experimentación

Documentar en la carpeta `DECOUPLE_FEATURES.md`:

- Semilla aleatoria.
- Rango temporal del split (train/val/test).
- Normalización aplicada a features finales.
- Arquitectura y hyperparámetros del modelo a usar para comparación.

10. Archivos modificados y nuevos

- Modificados:
  - `fiboevo.py` — lógica de desacoplamiento.
  - `prepare_dataset.py` — exclusión automática y flag `-decouple-features`.
- Nuevos:
  - `test_decouple_features.py` — suite de tests.
  - `DECOUPLE_FEATURES.md` — documentación ampliada.

11. Apéndice A: Fragmentos de código relevantes

11.1. Función de exclusión (fragmento)

```
# prepare_dataset.py (exclusion)
EXCLUDE_HIGHLY_CORRELATED = {
    "high", "low", "log_close",
    "sma_5", "ema_5",
    "bb_m", "bb_up", "bb_dn",
}

def filter_columns(df, exclude_set=EXCLUDE_HIGHLY_CORRELATED):
    cols = [c for c in df.columns if c not in exclude_set]
    return df[cols]
```

11.2. Transformación de ejemplo (fragmento)

```
# fiboevo.py (fragmento)
def add_decoupled_features(df, eps=1e-9):
    # hl spread relative to close
    df['hl_spread_norm'] = (df['high'] - df['low']) / (df['close'] + eps)
    # close position in high-low range, clipped
    denom = (df['high'] - df['low']).replace(0, eps)
    df['close_hl_position'] = ((df['close'] - df['low']) / denom).clip(0,1)
    # sma distance
    df['close_sma20_dist'] = (df['close'] - df['sma_20']) / (df['sma_20'] + eps)
    # bb position
    denom_bb = (df['bb_up'] - df['bb_dn']).replace(0, eps)
    df['bb_position'] = ((df['close'] - df['bb_dn']) / denom_bb).clip(0,1)
    # atr normalized
    df['atr_14_norm'] = df['atr_14'] / (df['close'] + eps)
    return df
```

12. Apéndice B: Test plan (resumen)

1. Unit tests para cada transformación (valores límite, NaNs, divisiones por cero).
2. Test de integración para la flag CLI.
3. Test de correlaciones: comparar matrices de correlación antes/después en un dataset de muestra.
4. Reporte automático si la correlación promedio no cae por debajo de umbral esperado.

13. Conclusiones

Se ha diseñado e implementado un conjunto de medidas reproducibles para mitigar la multicolinealidad entre indicadores técnicos y la variable `close`. Las transformaciones normalizadas combinadas con una exclusión manual proporcionan una reducción sustancial de correlación y se integran al pipeline mediante un flag opt-in. Se recomienda evaluar los efectos sobre la métrica objetivo del modelo en condiciones controladas y considerar la combinación con selección basada en VIF y/o mecanismos de atención en etapas posteriores.

Agradecimientos

Se agradece la colaboración del equipo de desarrollo y las contribuciones en pruebas y documentación.

Referencias

[1] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 4th ed., Wiley, 2008.