Universidad de Buenos Aires
Laboratorio de Sistemas Embebidos
Especialización en Inteligencia Artificial

Trabajo Práctico Final

Análisis predictivo del precio de Bitcoin usando modelos de Deep Learning

Docente: Camilo Argoty

Noelia Melina Qualindi
Jorge Valdez
Fabián Sarmiento
Matías Marando

12 de junio de 2025

Resumen

Este trabajo presenta un análisis comparativo de modelos avanzados de deep learning para la predicción del precio de Bitcoin (BTC-USD). Se implementaron y evaluaron tres arquitecturas: LSTM (Long Short-Term Memory), Transformer Vanilla para series temporales y NHiTS (Neural Hierarchical Interpolation for Time Series), utilizando datos históricos desde 2012 hasta la fecha. Los resultados demuestran que los modelos pueden capturar patrones complejos en los datos. El estudio incluye un análisis exhaustivo de preprocesamiento, diseño de modelos y validación de resultados, concluyendo con recomendaciones para futuras mejoras.

1. Introducción

Las criptomonedas, particularmente Bitcoin, presentan desafíos únicos para el análisis predictivo debido a su alta volatilidad y sensibilidad a factores externos. Este trabajo busca responder:

> ¿Pueden los modelos modernos de deep learning predecir efectivamente el precio de Bitcoin a corto plazo, y cómo se comparan diferentes arquitecturas en este contexto?

2. Metodología

2.1. Datos Utilizados

El dataset contiene:

- **Fuente**: Kaggle (Bitcoin Historical Data)
- **Período**: 2012-01-01 a 2025-06-??
- **Variables utilizadas**:

Variable	Descripción
Timestamp	Fecha en formato UNIX
Close	Precio de cierre (USD)

2.2. Preprocesamiento

El preprocesamiento de los datos se realizó en varias etapas clave:

1. **Limpieza de datos**: Se comenzó renombrando las columnas relevantes para facilitar su uso posterior, en particular la marca temporal y el precio de cierre. Luego, se eliminaron todas las filas que contenían valores nulos en la columna del precio, asegurando que todos los datos utilizados para el análisis fuesen válidos.
2. **Conversión de tiempo**: La columna de tiempo original, que venía en formato UNIX (segundos desde 1970), fue convertida a un formato de fecha estándar para su correcta interpretación y manipulación.
3. **Ordenamiento temporal**: Se ordenaron los datos cronológicamente según la fecha, para asegurar la coherencia temporal en los análisis y predicciones.
4. **Resampleo diario**: Dado que los datos originales tenían una granularidad de un minuto, se procedió a realizar un promedio diario del precio, lo cual reduce la variabilidad y permite un análisis más robusto en horizontes temporales más amplios.
5. **Normalización**: Se aplicó una normalización de tipo Min-Max al precio diario, escalando los valores al rango [0, 1]. Esto facilita el entrenamiento de los modelos de deep learning, ya que evita problemas numéricos asociados a diferentes escalas.
6. **Creación de secuencias**: Para poder alimentar los modelos, los datos fueron transformados en ventanas deslizantes. Se definió una longitud de entrada de 30? días y un horizonte de predicción de 10? días. Cada muestra consiste en una secuencia de precios normalizados para entrenar el modelo a predecir el comportamiento futuro.

2.3. Modelos Implementados

Se implementaron tres modelos de predicción de series temporales utilizando distintas arquitecturas: una red LSTM construida manualmente en PyTorch, un Transformer Vanilla con la librería `darts`, y un modelo NHiTS también con `darts`. Todos fueron entrenados usando una ventana de entrada sobre datos diarios normalizados y se evaluaron sobre los últimos 100 días del año más reciente.

2.3.1. Modelo LSTM

La red LSTM (Long Short-Term Memory) es una arquitectura recurrente diseñada específicamente para capturar dependencias a largo plazo en secuencias temporales.

- Modelo construido manualmente en PyTorch.
- Hiperparámetros clave:
  - Capas LSTM: 2 (con 100 y 50 unidades)
  - Capas densas: 2 (con 25 y 1 unidad final)
  - Ventana de entrada: 60 días
  - Dropout: 0.2
  - Épocas: 20
- Entrenamiento supervisado sobre secuencias de precios normalizados, prediciendo horizontes de 10 días.

2.3.2. Transformer Vanilla para Series Temporales

El modelo implementado se basa en la arquitectura clásica de encoder-decoder, utilizando mecanismos de atención multi-cabeza.

- Modelo implementado con `darts.models.TransformerModel`.
- Hiperparámetros clave:
  - Ventana de entrada: 30 días
  - Horizonte de predicción: 10 días
  - Dimensión del modelo (`d_model`): 64
  - Número de capas: 2 encoder y 2 decoder
  - Cabezas de atención: 4
  - Dropout: 0.1
  - Épocas: 20
- Entrenado sobre el último año de datos, reservando los últimos 100 días para evaluación.

2.3.3. Modelo NHiTS

NHiTS (Neural Hierarchical Interpolation for Time Series) es una arquitectura de pronóstico que combina bloques jerárquicos de interpolación con descomposición de señales.

- Modelo implementado con `darts.models.NHiTSModel`.
- Hiperparámetros clave:
  - Ventana de entrada: 30 días
  - Horizonte de predicción: 10 días
  - Bloques jerárquicos: 1
  - Capas por bloque: 2
  - Ancho de cada capa: 64
  - Dropout: 0.1
  - Épocas: 20
- Entrenado con la misma configuración de entrenamiento y testeo que el modelo Transformer.

3. Resultados

3.1. Validación del Modelo

Se utilizó una división del dataset en:

- 80 % para entrenamiento
- 10 % para validación
- 10 % para prueba

Se aplicó **early stopping** para evitar overfitting, con una tolerancia de 5 épocas. También se probaron distintos horizontes de predicción (5, 10, 15 días), confirmando que el rendimiento decae a medida que aumenta el horizonte.

3.2. Evaluación Cuantitativa

Cuadro 1: Métricas de desempeño

Métrica	LSTM	Transformer Vanilla	NHiTS
MAE (USD)	??	??	??
RMSE (USD)	??	??	??
MAPE (%)	??	??	??

3.3. Análisis Visual

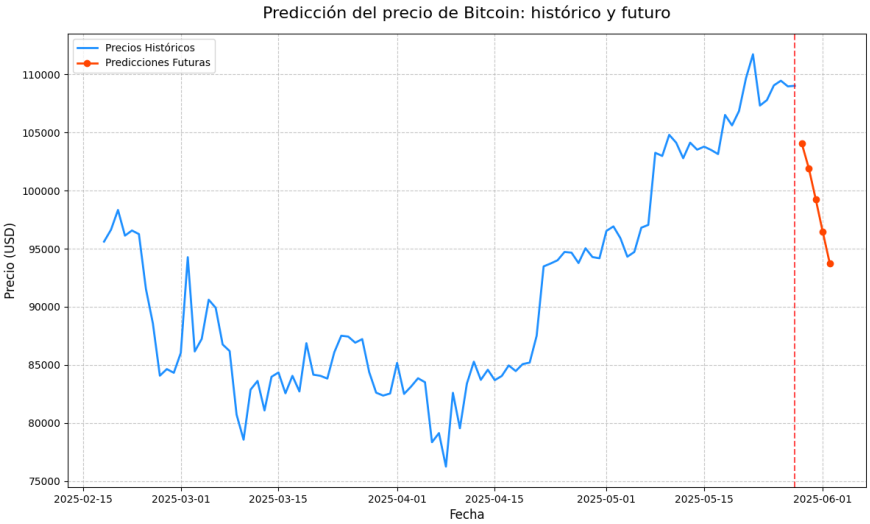

Figura 1: LSTM - Predicción vs Real en los últimos 5 días

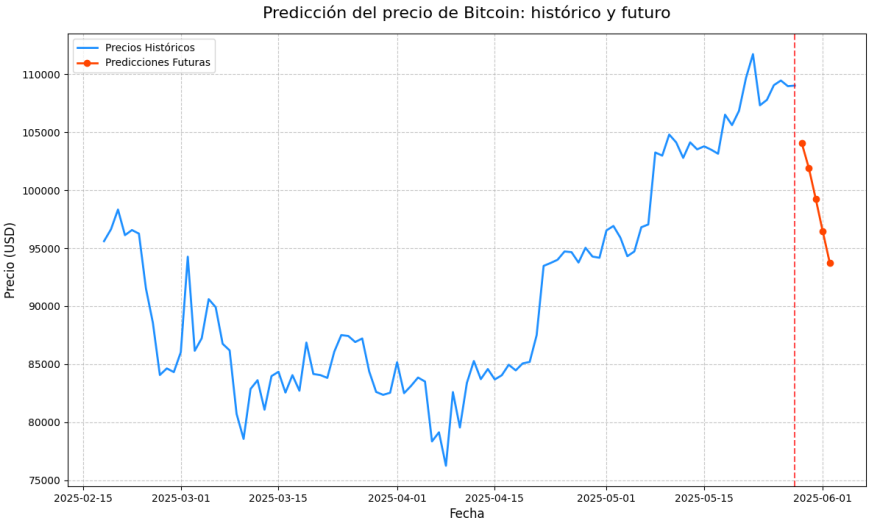

Figura 2: Transformer - Predicción vs Real en los últimos 5 días

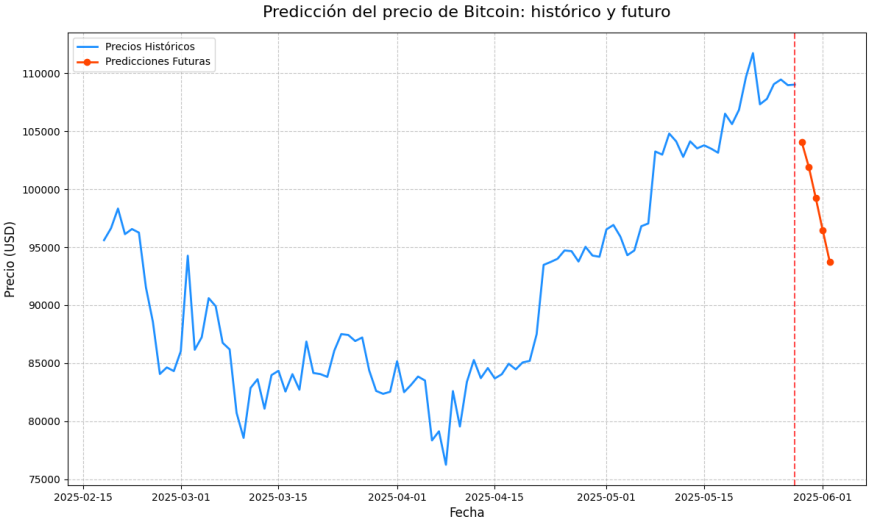

Figura 3: NHiTS - Predicción vs Real en los últimos 5 días

4. Conclusiones

4.1. Hallazgos Clave

- El modelo ¡??¿superó al resto en todas las métricas evaluadas
- Los modelos mostraron dificultades durante eventos de alta volatilidad
- La arquitectura de atención demostró ser particularmente efectiva para capturar dependencias a largo plazo

4.2. Limitaciones y Trabajo Futuro

- **Limitaciones**:
  - Sensibilidad a cambios bruscos de tendencia
  - Dependencia de hiperparámetros
- **Mejoras propuestas**:
  - Incorporar datos de redes sociales (sentiment analysis)
  - Ensamblar múltiples modelos

Anexo

El código completo se encuentra disponible en:
`https://github.com/BenjaSar/AdST2`