Universidad de la República
Facultad de Ingeniería

Análisis predictivo en Bitcoin utilizando técnicas de aprendizaje profundo

Federico González
Tutor: Dr. Pablo Rodríguez Bocca

Resumen

El prominente mercado de las criptomonedas, caracterizado por un alto nivel especulativo y de gran volatilidad, plantea un novedoso y desafiante escenario para la aplicación de métodos de pronósticos sobre series temporales. En este contexto, Bitcoin se destaca por abarcar la mayor parte de la capitalización total del mercado, así como del volumen total de transacciones diarias. El presente proyecto plantea realizar un análisis de la efectividad de la aplicación de algoritmos de aprendizaje automático al problema de generar predicciones de corto plazo sobre el precio de Bitcoin. Se realiza un estudio comparativo de distintos algoritmos de aprendizaje automático, empleando una batería de modelos cuyas implementaciones incluyen: SVR (Support Vector Regression), RNN (Recurrent Neural Networks), CNN (Convolutional Neural Networks) y una arquitectura de red neuronal híbrida que combina capas recurrentes LSTM (Long Short Term Memory) con capas convolucionales unidimensionales. Además, se propone utilizar características basadas en indicadores de análisis técnico, en conjunto con una metodología para la generación de datos de entrenamiento adicionales, con el fin de evaluar su incidencia en la capacidad predictiva de los modelos. La utilización de los modelos de aprendizaje profundo híbridos CNN + LSTM en combinación con la incorporación de las características y datos de entrenamiento adicionales reportaron los mejores resultados del conjunto de modelos evaluados, alcanzando un RMSE (Raíz del Error Cuadrático Medio) de 42.11 (1.10 %) en la generación de pronósticos del precio de Bitcoin para un horizonte de 2 horas sobre un periodo de prueba de 720 puntos correspondientes a cada hora del mes de diciembre de 2018. Durante el desarrollo de esta investigación, se realiza un repaso del estado del arte de las técnicas de aprendizaje automático, con énfasis en su aplicación a problemas de pronósticos sobre series temporales.

Palabras clave: Aprendizaje automático, aprendizaje profundo, regresión, series temporales, Bitcoin, SVR, RNN, CNN, LSTM, GRU.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

El problema de realizar pronósticos sobre series temporales ha sido extensivamente estudiado para mercados financieros maduros tal como lo es la bolsa de valores o el mercado de divisas. Bitcoin [99] presenta un interesante paralelo, como un problema de predicción de series temporales en un mercado que aún se encuentra en una etapa de transición, con un alto nivel de incertidumbre y volatilidad [133]. A su vez, la naturaleza abierta de Bitcoin, plantea un paradigma opuesto al de los mercados financieros tradicionales, operando bajo un sistema descentralizado, en el cual todas las transacciones se publican en un registro abiertamente accesible. Dada la complejidad de la tarea, y la disponibilidad absoluta de su información transaccional histórica, resulta un candidato interesante para la aplicación técnicas de aprendizaje profundo. Además de presentar una oportunidad de investigación intrínsecamente motivante, la capacidad de obtener predicciones con cierto grado de precisión en un mercado de alta volatilidad tiene el potencial de generar rentabilidad mediante su aplicación en estrategias automatizadas de intercambio.

1.2. Propuesta

Se propone evaluar y comparar el desempeño de una gama de diversas técnicas de aprendizaje automático a la tarea de realizar pronósticos a corto plazo del precio de Bitcoin. Entre las soluciones evaluadas se consideran algoritmos tradicionales tales como SVR [29], así como modelos de aprendizaje profundo cuya literatura reporta excelentes resultados en problemas análogos de pronósticos sobre series temporales, como redes neuronales recurrentes LSTM [46, 42], GRU [20] y redes convolucionales unidimensionales [88]. Se propone además la utilización de un modelo de red neuronal híbrido combinando capas convolucionales en conjunto con capas recurrentes LSTM. Asimismo, se plantea mejorar la precisión de los modelos mediante la utilización de características generadas en base a indicadores de análisis técnico, y a partir de la generación de datos de entrenamiento adicionales a través de la inclusión de información del mercado de otras criptomonedas.

1.3. Organización del documento

El documento se divide en siete capítulos:

Capítulo 1 - Introducción

Introducción al presente problema y una breve descripción de la investigación propuesta.

Capítulo 2 - Marco teórico

Se presentan los conceptos relacionados al proyecto, incluyendo una resumida introducción a Bitcoin y un recorrido por las técnicas de aprendizaje automático, con particular énfasis en su aplicación al pronóstico de series temporales y los resultados obtenidos en el estado del arte.

Capítulo 3 - Descripción del problema

Se define el alcance del problema a tratar, el conjunto de datos utilizado y las métricas empleadas para la evaluación de los resultados.

Capítulo 4 - Soluciones evaluadas

Se demuestra cómo se enmarca el problema para su resolución utilizando aprendizaje supervisado. Se presentan los modelos de aprendizaje automático implementados, así como el proceso de generación de características de análisis técnico y de datos de entrenamiento adicionales.

Capítulo 5 - Pruebas realizadas

Se definen las pruebas realizadas con el fin de optimizar cada tipo de modelo y evaluar el desempeño de las soluciones propuestas.

Capítulo 6 - Resultados

Se analizan los resultados obtenidos luego de ejecutar las pruebas.

Capítulo 7 - Conclusiones

Se exponen las conclusiones obtenidas a partir del desarrollo del proyecto y se consideran posibles mejoras y adiciones a las soluciones propuestas.

Capítulo 2

Marco teórico

2.1. Bitcoin

Bitcoin [99] es la primera implementación del concepto presentado por Nick Szabo en 1997 [122] de un sistema de consenso distribuido, mutuamente confiable y sin intermediarios. Es utilizado como un sistema de dinero electrónico y es considerado la primer criptomoneda descentralizada [111]. Está implementado como un protocolo de código abierto y no está respaldado por el sistema financiero; en su lugar es sustentado por la tecnología de blockchain. La unidad de cuenta [90] nativa del sistema de Bitcoin es "un Bitcoin", representado con los símbolos bursátiles de "BTC", "XBT" o "₿"

2.1.1. Blockchain

Un blockchain [99] (o cadena de bloques) es un histórico de todas las transacciones del sistema, análogo a un libro contable público y distribuido. Cada bloque del blockchain de Bitcoin contiene un conjunto de transacciones que son hasheadas y codificadas en un árbol de Merkle [93].

Cuadro 2.1: Cabezal de un bloque del blockchain de Bitcoin.

Campo	Descripción	Tamaño (bytes)
Versión	Número de versión de Bitcoin	4
hashPrevBlock	Hash calculado en base al cabezal del bloque previo	32
hashMerkleRoot	Hash de la raíz del árbol de Merkle de las transacciones de este bloque	32
Timestamp	Timestamp de este bloque en UNIX	4
Bits	Objetivo actual	4
Nonce	Número aleatorio utilizado para generar el hash del bloque	4

Además, el cabezal de un bloque (Cuadro 2.1) incluye un hash criptográfico del cabezal del bloque anterior del blockchain. De esta manera cada bloque está "encadenado" al anterior, generando una cadena desde el "bloque génesis" (o "bloque cero") hasta el bloque actual (Figura 2.1). Este proceso asegura la integridad de toda la cadena. Aunque los bloques no son inalterables, modificar uno implicaría regenerar todos los subsiguientes bloques de la cadena, resultando computacionalmente inviable.

Cada nodo de la red mantiene una copia local de todo el blockchain, de manera que la cadena es verificada de forma independiente y descentralizada. Cada transacción es registrada localmente por cada nodo, y posteriormente transmitida al resto de la red. Cada aproximadamente diez minutos, un nuevo conjunto de transacciones es aceptada por la red e incluida en forma de un bloque en el blockchain, sin necesidad de la intervención de una entidad supervisora centralizada.

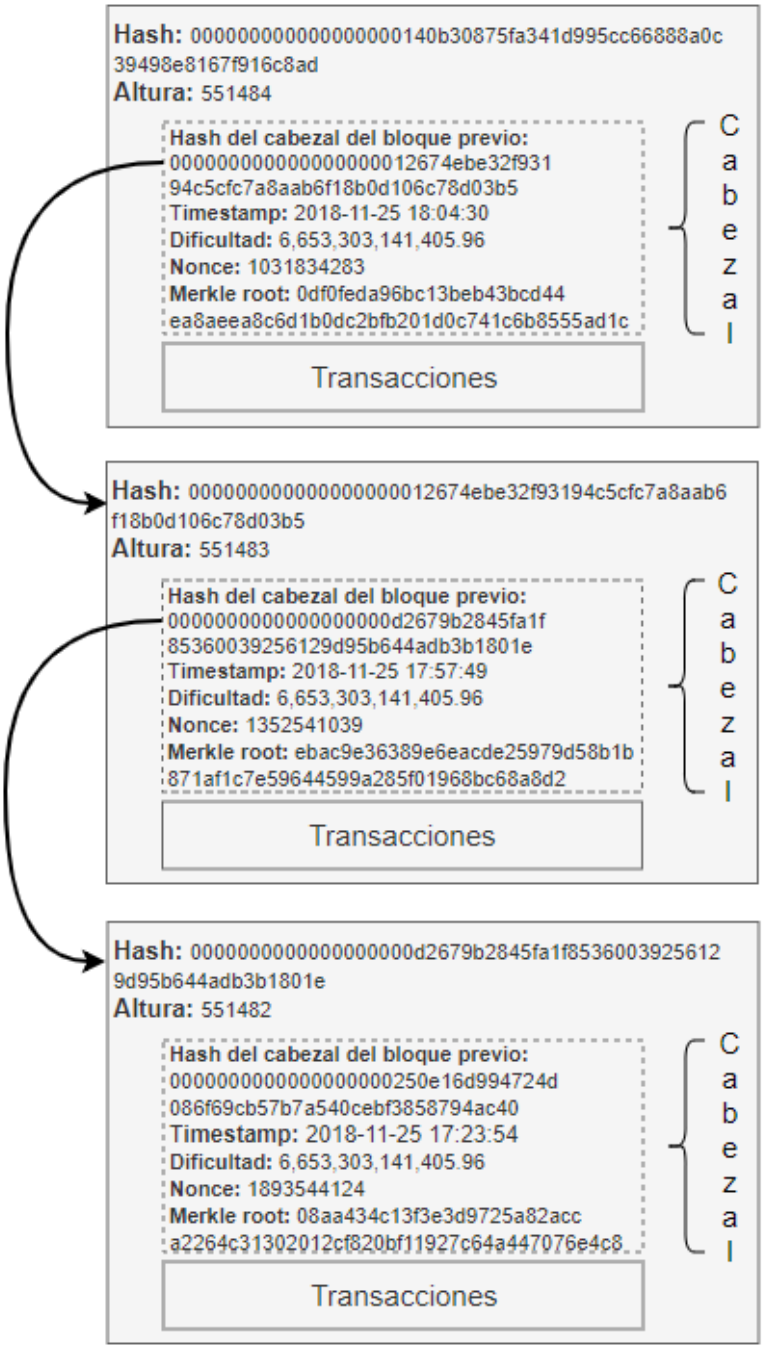


Figura 2.1: Bloques 551482 al 551484 del blockchain de Bitcoin.

2.1.2. Transacciones

Una transacción es una transferencia de una cierta cantidad de Bitcoin de una o más direcciones de entrada a una o más direcciones de salida. Para prevenir el problema del

"doble gasto" [21] cada entrada de una transacción debe hacer referencia o bien a una salida sin gastar de otra transferencia, o a una "generación". Se le denomina "generación" a la primer transferencia de un bloque del blockchain, generada al descubrirse un nuevo bloque. Cuando la salida de una transacción es utilizada como la entrada de otra, esta debe ser consumida en su totalidad. Es por esto que si se desea transferir una cantidad inferior a la disponible en la salida, se debe realizar una transacción extra conocida como "cambio", mediante la cual la cantidad restante es devuelta a la dirección de origen (Figura 2.2).

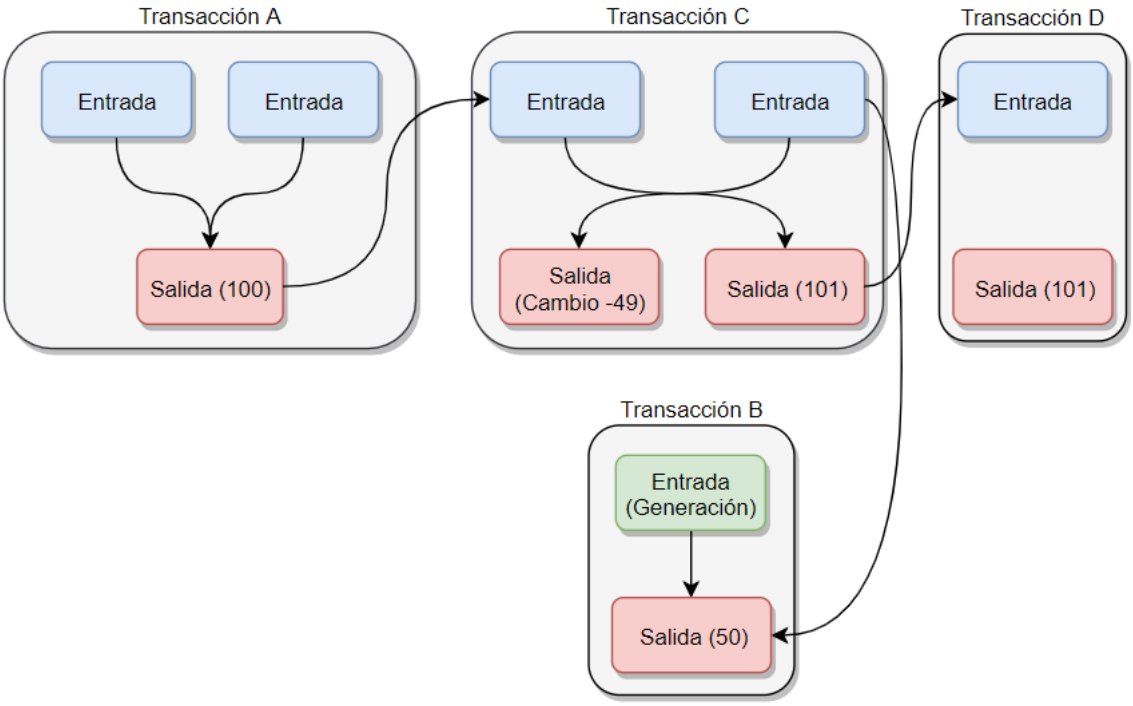


Figura 2.2: "A" le envía 100 BTC a "C" y "C" genera 50 BTC. "C" envía 101 BTC a "D", por lo que debe generar una transacción de "cambio" por 49 BTC. "D le envía 101 BTC a alguien más, pero la transacción no fue verificada aún. En este punto, sólo la salida de "D" y el cambio de "C" pueden ser gastados en otra transacción.

Al estar registradas permanentemente en el blockchain, todas las transacciones desde el comienzo de la historia de Bitcoin son públicamente visibles y fácilmente accesibles a través de exploradores del blockchain en la web [13] o mediante la descarga directa del blockchain en su totalidad [28].

2.1.3. Direcciones

Una dirección es un identificador alfanumérico de 26 a 35 caracteres de largo que representa un posible destino para una transacción. Cada dirección está asociada con una clave privada, la cual permite firmar digitalmente una transacción para utilizar sus fondos. Cada dirección es utilizada una única vez por transacción; esto implica administrar una gran cantidad de direcciones y claves si se desean realizar transacciones con relativa frecuencia. Por este motivo para administrar fondos en Bitcoin se suele hacer uso de *wallets*. Los wallets permiten almacenar todas las claves públicas y privadas asociadas con las direcciones de Bitcoin del usuario de manera de realizar transacciones de forma práctica y conveniente. Existen en forma de software, ya sea como aplicaciones de escritorio, móviles o basadas en la web, o en forma de hardware, como dispositivos físicos que almacenan localmente las claves del usuario y utilizan algún sistema de autenticación como un pin o contraseña.

2.1.4. Anonimato

Hasta recientemente, las transacciones realizadas con Bitcoin eran consideradas anónimas. Sin embargo, en la actualidad, se utiliza el concepto de *pseudo-anonimato* [95, 18]. Si bien no existe una asociación directa entre un Bitcoin y su(s) propietario(s), los flujos de transacciones son globalmente visibles. En consecuencia, es estimado que la identidad de casi el 40 % de los usuarios puede ser identificada, incluso en los casos donde los usuarios hayan adoptado las medidas de privacidad recomendadas por Bitcoin [3]. Uno de los aspectos generados por este pseudo anonimato es el uso de Bitcoin en posibles actividades delictivas o secretas, así como dificultades para hacer cumplir las normativas contra el lavado de dinero [104, 98].

2.1.5. Mercado

El precio de Bitcoin no es fijado por el sistema bancario, financiero o ningún tipo de entidad supervisora centralizada, en su lugar, es determinado absolutamente por la oferta y demanda del mercado.

El carácter descentralizado de Bitcoin y la carencia de seguridad jurídica generan un tipo de mercado mayormente desregulado y volátil. El precio de Bitcoin ha pasado por ciclos de apreciación y depreciación referidos como "burbujas y estallidos" (bubbles and bursts). En 2011 el valor de un Bitcoin incrementó rápidamente de US$0.30 a US$32 para luego caer estrepitosamente a US$2. En abril del 2013 alcanzó un máximo de U$S266 para luego estrellarse en un mínimo de US$50 en noviembre del mismo año. A finales de 2017 alcanzó un máximo histórico de US$19,783.06 y, un año después, en noviembre de 2018 volvió a experimentar una caída histórica llegando a cotizar por debajo de los US$4,000 (Figura 2.3).

Bitcoin tiene una volatilidad siete veces mayor que el oro, ocho veces mayor que S&P 500 y dieciocho veces mayor que el dólar estadounidense [133]. Estas características han propiciado la investigación en el ámbito de las ciencias económicas, realizándose estudios para analizar las características de Bitcoin como una mercancía financiera. En [30] y en [82] se estudia su volatilidad utilizando modelos GARCH [14], concluyéndose que, en general, Bitcoin tiene un lugar en los mercados financieros y en la gestión de carteras de valores, ya que puede clasificarse como algo intermedio entre el oro y el dólar estadounidense en una escala que va desde un medio puro de intercambio hasta una reserva de valor. En [6] se analizó cómo se ajusta el mercado de Bitcoin a ciertas propiedades estadísticas que son características de las monedas tradicionales. Se intentó ajustar la serie temporal del precio de Bitcoin a las distribuciones paramétricas más populares dentro de las finanzas, encontrándose el mejor ajuste utilizando una distribución hiperbólica generalizada [7]. Asimismo se concluye que una inversión en Bitcoin exhibe un nivel muy alto de volatilidad

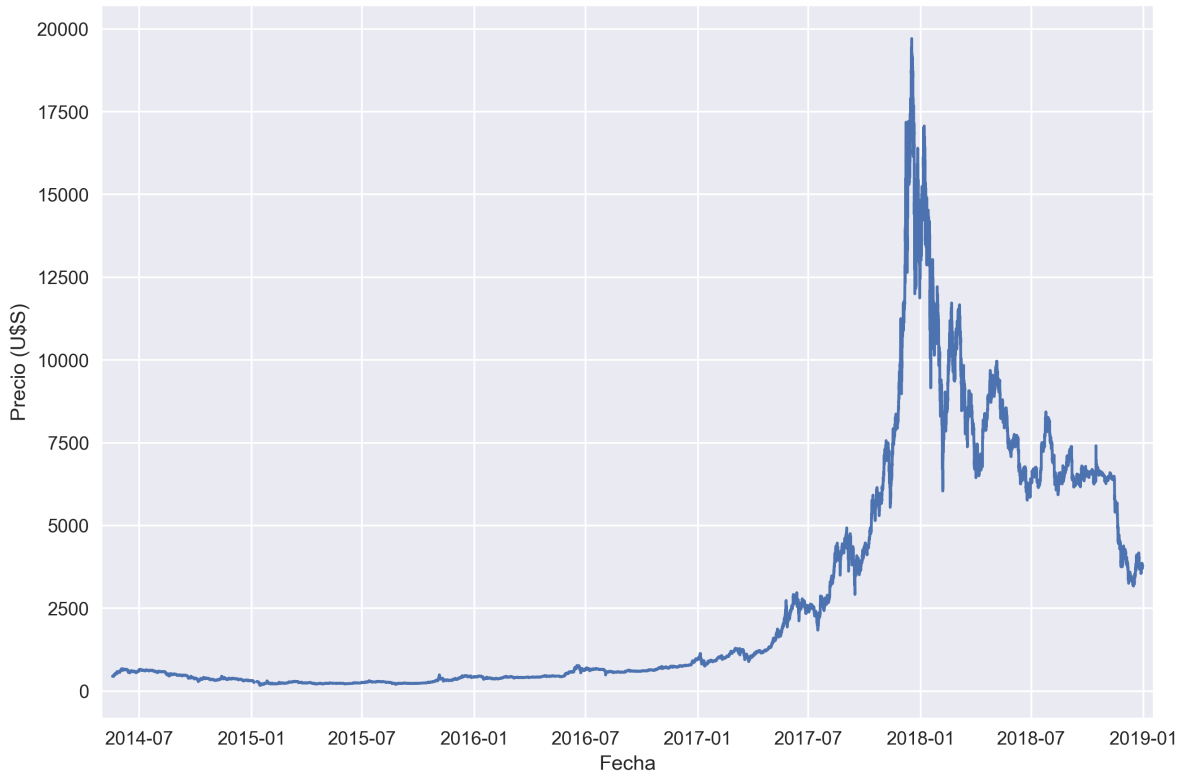


Figura 2.3: Evolución histórica del precio de Bitcoin (Fuente: [10]).

y a su vez un muy alto rendimiento. En [22] se aplica un enfoque similar, desde la óptica de la estadística financiera, y se concluye que si bien el precio de Bitcoin exhibe un alto nivel de volatilidad, este está disminuyendo con el tiempo. Se menciona una desconexión entre el precio del Bitcoin y los fundamentos del mercado financiero. A su vez, se estudió la memoria a largo plazo [8] de la serie temporal del precio, hallando un comportamiento pro-cíclico ($0.5 < H < 1$, H = Exponente de Hurst [48]) hasta el año 2014, momento en el cual se quiebra la tendencia ($H < 0.5$); sugiriendo una evolución a un mercado orientado de manera más informacional. En [16] y [31] se estudia el potencial de Bitcoin como un bien de inversión, alcanzando conclusiones contradictorias. En lo que ambos coinciden es en que puede ser utilizado a modo de diversificar una cartera de valores, y que provee cierta cobertura contra los movimientos diarios del dólar estadounidense y de la bolsa de valores.

De manera análoga a la bolsa de valores, han habido numerosos intentos de predecir los

movimientos del precio de Bitcoin. Con tales fines se han utilizando técnicas tradicionales de estadística sobre series temporales, así como modelos de aprendizaje automático. En las siguientes secciones se realiza un repaso por los enfoques más relevantes.

2.2. Series temporales

Una serie temporal es una serie de datos indexados cronológicamente, comúnmente espaciados uniformemente en el tiempo. Se define como x(t), t = 0, 1, 2, ... Donde t denota el tiempo transcurrido, y x(t) es considerada como una variable aleatoria. El análisis de una serie temporal involucra métodos para extraer información estadísticamente relevante y otro tipo de información representativa sobre las relaciones subyacentes con el objetivo de, entre otros, extrapolar o interpolar los datos y así predecir el comportamiento de la serie en momentos no observados.

Los modelos de series temporales pueden tener diferentes formas y representar diversos procesos estocásticos. Hay dos modelos de series temporales lineales ampliamente utilizados en la literatura, autorregresivos (AR) y de medias móviles (MA) [17, 45]. Combinando ambos ha surgido la familia de modelos basados en el principio de Box-Jenkins, ampliamente conocidos como los modelos de Box-Jenkins, entre ellos: el modelo autorregresivo de promedio móvil (ARMA) [17, 45, 25], el autorregresivo integrado de promedio móvil (ARIMA) [17, 45, 25], el autorregresivo de promedio móvil fraccionalmente integrado (ARFIMA) [36] y el autorregresivo integrado de promedio móvil estacionario (SARIMA) [17].

El problema de predecir el precio de Bitcoin puede describirse como un problema de pronóstico de series temporales, donde la serie de datos cronológicamente indexados es el precio de Bitcoin, y el objetivo es extrapolar para predecir su comportamiento en el futuro. La aplicación de modelos de series temporales a este problema ha sido explorado por varios estudios. En [121, 4] se utilizan modelos ARIMA para realizar predicciones del precio de Bitcoin en una ventana de tiempo de un día. Los resultados obtenidos son

variados, entre un MSE (Mean Squared Error) de 295,797 hasta 16,000. En [5] se utiliza el mismo tipo de modelo pero realizando una predicción en una ventana de tiempo de un mes reportando un MAPE (Mean Absolute Percentage Error) de 5.36 %.

En las últimas décadas, la disponibilidad de grandes volúmenes de datos ha propiciado la intersección entre la informática y la estadística, dando lugar a enfoques probabilísticos en el campo de la inteligencia artificial, surgiendo sistemas capaces de analizar y aprender a partir de gran cantidad de información. En la siguiente sección se exploran estos enfoques, con énfasis en su aplicación a problemas de predicción sobre series temporales.

2.3. Aprendizaje automático

Una posible definición generalizada de un algoritmo de aprendizaje automático es "Un programa que mejora su desempeño en una tarea T, respecto a una medida P, basándose en la experiencia E" [97].

En particular, los algoritmos de aprendizaje automático *supervisado* son aquellos cuya tarea es la de inferir una función $g : X \to Y$ que mapea una entrada a una salida. Para cada observación de un conjunto de *predictores* (comúnmente denominados *características de entrada*, o *variables de entrada*) $x_i \in X, i = 1, ..., n$, hay una *respuesta* asociada $y_i \in Y$. Se desea ajustar un modelo que relacione la respuesta con los predictores, con el objetivo de predecir con precisión la respuesta en instancias no observadas (predicción) o de obtener información acerca de la relación entre la respuesta y los predictores (inferencia).

Las variables de respuesta se pueden caracterizar como cuantitativas o categóricas. Las variables cuantitativas toman valores numéricos, mientras que las categóricas toman valores dentro de un conjunto de clases o categorías. Los problemas con una respuesta cuantitativa son referidos como problemas de *regresión* mientras que los de respuesta categórica se denominan como problemas de *clasificación*.

El problema de predecir el precio de Bitcoin en base a información histórica previa, se puede enmarcar en un problema de aprendizaje automático (según la primer definición):

- T = predecir el precio de Bitcoin en una determinada ventana de tiempo a futuro.
- P = la precisión de la predicción, según alguna métrica (e.g. RMSE).
- E = la información histórica del precio de Bitcoin y/o algún otro dato relacionado relevante.

Además, para cada observación del conjunto de predictores $x_i, i = 1, ..., n$ (el precio de Bitcoin y/o algún otro dato relevante), se cuenta con la respuesta y_i numérica (el precio de Bitcoin en una determinada ventana a futuro). Por lo que se está ante un caso de aprendizaje *supervisado* sobre un problema de *regresión*.

2.3.1. Teorema de "No hay almuerzo gratis"

El teorema de "No hay almuerzo gratis" [135, 134] (No free lunch theorem) establece que no hay un modelo que funcione mejor para todo problema. Las hipótesis supuestas para un buen modelo en el marco de un cierto problema pueden no ser válidas para otro problema, por lo que es frecuente en el contexto de aprendizaje automático probar varios modelos y encontrar el que mejor funcione para un problema en particular. Esto es especialmente cierto en el aprendizaje supervisado; donde comúnmente se utiliza la validación o validación cruzada para evaluar la capacidad predictiva de un conjunto de modelos de complejidad variable. Dependiendo del problema, el "mejor" modelo estará determinado por una combinación de factores tales como velocidad, precisión o complejidad.

2.3.2. Regresión lineal

Aún siendo uno de los enfoques más simples, no deja de ser de los más útiles y ampliamente utilizados en el campo del aprendizaje automático supervisado. La regresión lineal *simple* asume que hay una relación aproximadamente lineal entre un predictor X y la variable de respuesta Y:

$$X \approx \beta_0 + \beta_1 Y$$

Los algoritmos que utilizan regresión lineal, deben hallar los parámetros $\hat{\beta}_0$ y $\hat{\beta}_1$ que "mejor" aproximen a la función real. $x_i \approx \hat{\beta}_0 + \hat{\beta}_1 y_i$ para $i = 1, ..., n$. En otras palabras, se desea obtener la recta de término independiente $\hat{\beta}_0$ y pendiente $\hat{\beta}_1$ tal que esté lo más "cerca" posible de todos los puntos del conjunto de datos. En este contexto, el concepto de distancia puede ser definido de varias formas, una de las definiciones más aplicadas es la de *mínimos cuadrados* (Figura 2.4).

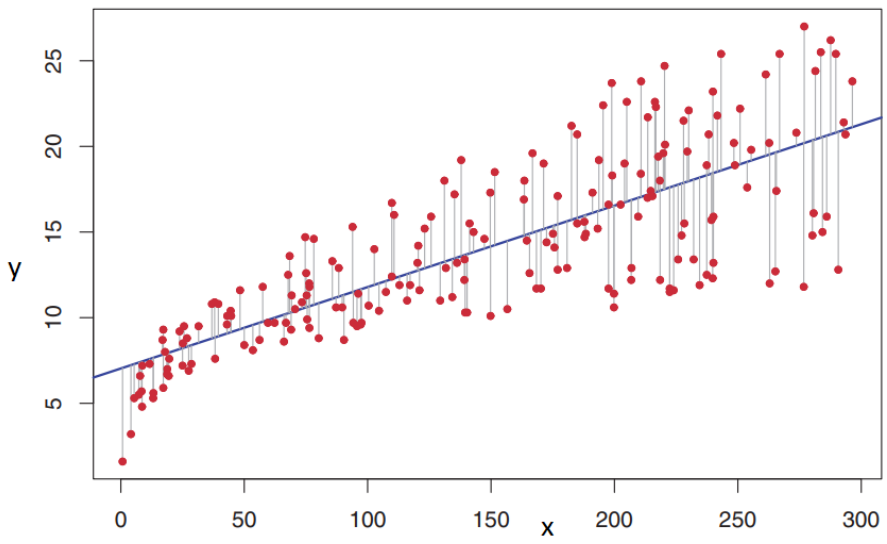


Figura 2.4: Aproximación de un conjunto de datos por regresión lineal utilizando mínimos cuadrados (Fuente: [79]).

Sea $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ la predicción para Y basada en el valor i-ésimo de X. Entonces $e_i = y_i - \hat{y}_i$ representa el i-ésimo residual: esto es, la diferencia entre el i-ésimo valor de respuesta

observado y el i-ésimo valor de respuesta predicho por el modelo lineal. Se define la suma residual de cuadrados (RSS) como:

$$RSS = e_1^2 + e_2^2 + ... + e_n^2$$

Se desea hallar entonces los $\hat{\beta}_0$ y $\hat{\beta}_1$ que minimicen el RSS (Figura 2.5), se puede demostrar [79] que estos valores son:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Donde $\bar{x}$ e $\bar{y}$ son la media de X e Y respectivamente.

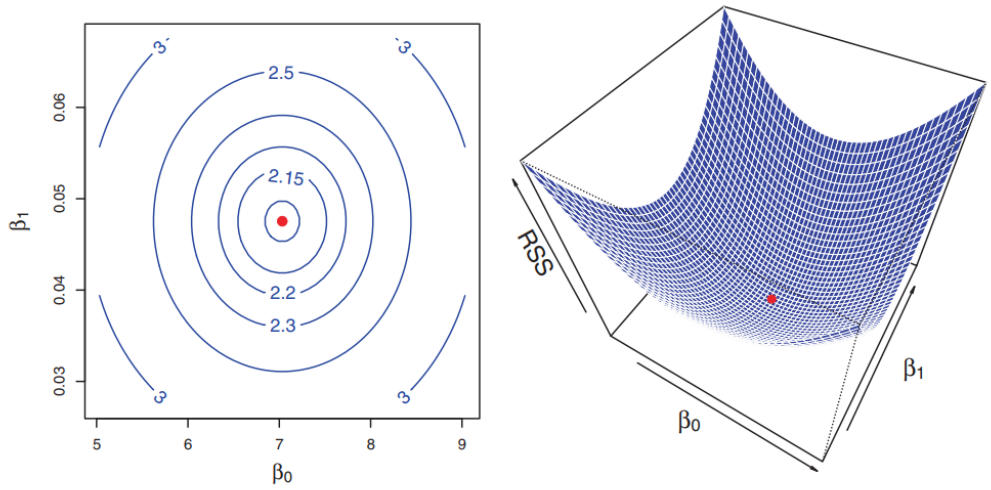


Figura 2.5: Contorno y gráfico tridimensional del RSS en función de los parámetros β_0 y β_1 para el conjunto de datos de la Figura 2.4 (Fuente: [79]).

La regresión lineal simple resulta útil para estudiar la respuesta en base a un único predictor. Sin embargo, se puede generalizar el enfoque al caso de múltiples predictores. El modelo en el caso de regresión lineal múltiple con p predictores queda expresado como:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon$$

Al igual que en el caso de regresión lineal simple, se utilizan los mínimos cuadrados para encontrar los $\hat{\beta}_0, \hat{\beta}_1 ... \hat{\beta}_p$ que mejor se ajusten a los datos (Figura 2.6).

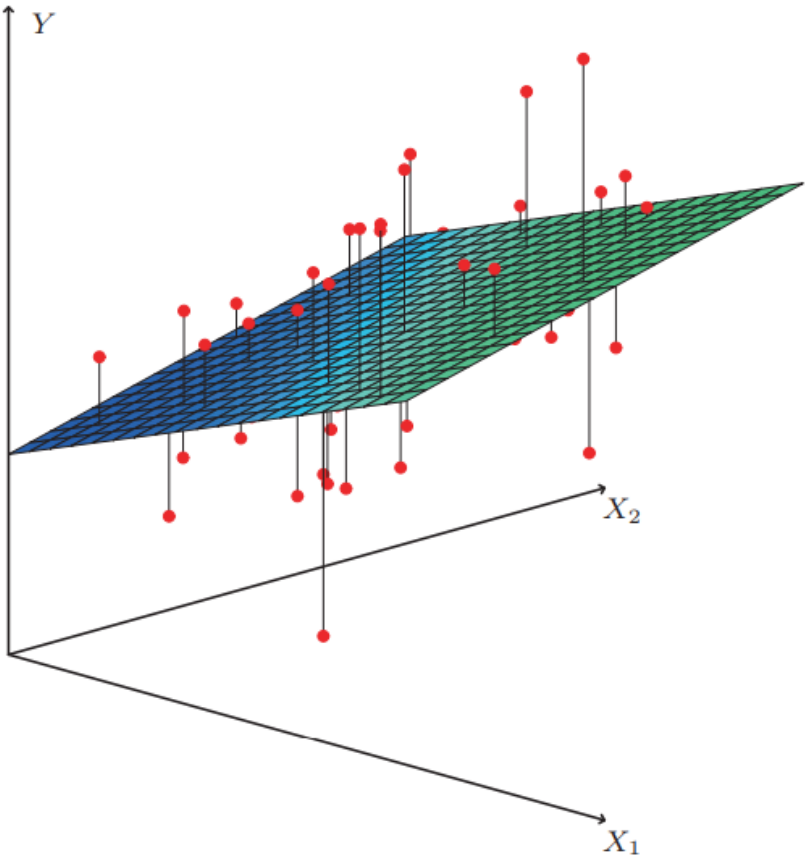


Figura 2.6: En el caso de dos predictores y una respuesta, la linea de regresión se convierte en un plano que minimiza la suma de los cuadrados de las distancias verticales entre el plano y cada observación (Fuente: [79]).

2.3.3. SVM

El algoritmo implementado por las SVM es una generalización no lineal del "Algoritmo de retrato generalizado" desarrollado durante los años sesenta [129]. No fue hasta la década de los noventa cuando alcanzó su presente forma, al ser mayormente estudiado en los laboratorios de AT&T [15, 43, 26, 29]. Posteriormente, con rapidez se convirtió en uno de los algoritmos que mejor se desempeñaba en la tarea de clasificación de objetos y OCR [113, 12]. Originalmente concebido para problemas de clasificación, el método fue

extendido a problemas de regresión, bajo el nombre de SVR [29], reportando excelentes resultados en problemas de predicción sobre series temporales [29, 118]. Hoy en día es considerado una herramienta estándar, ampliamente utilizada en el área del aprendizaje automático.

Dado un conjunto de datos $(x_1, y_1), ..., (x_n, y_n)$ donde $x_i \in \mathbb{R}^l$ denota al vector de predictores e $y_i \in \mathbb{R}$ a la correspondiente variable de respuesta, el objetivo es hallar una función $f(x)$ tal que tiene a lo sumo una desviación ε de la variable de respuesta y_i para todo el conjunto de datos y tal que la función sea lo más "plana" posible.

En la versión lineal:

$$f(x) = \langle w, x_i \rangle + b$$

Donde $\langle \cdot, \cdot \rangle$ denota el producto escalar, y "planitud" en este caso implica un w pequeño. Es decir, se busca una $f(x)$ donde se minimice la norma de w, $\frac{1}{2}||w||^2$, sujeto a:

$$\begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases}$$

No siempre es posible encontrar tal función, por lo que se introducen variables de holgura ξ_i, ξ_i^* (Figura 2.7) de forma que el problema queda formulado como:

Minimizar:

$$\frac{1}{2}||w||^2 + c \sum_{i=1}^{l} (\xi_i + \xi_i^*)$$

Sujeto a:

$$\begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

Donde c determina el balance entre "planitud" de la función y la cantidad de desviaciones mayores a ε que son toleradas.

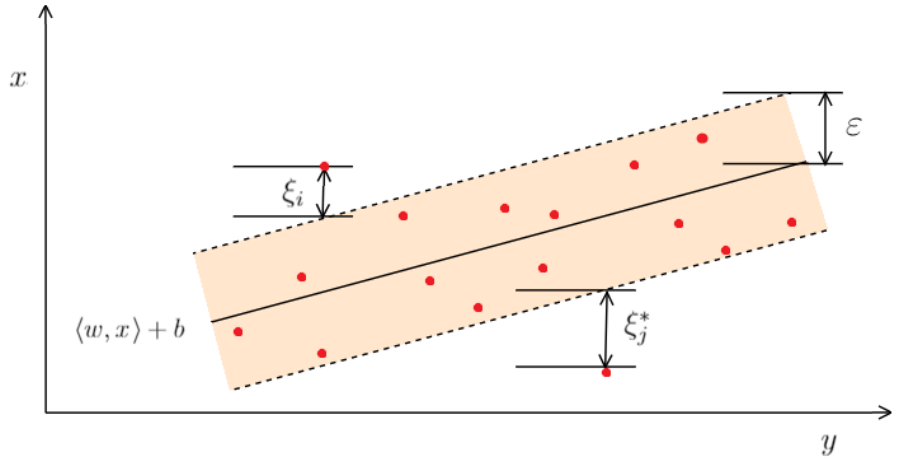


Figura 2.7: SVR lineal con variables de holgura.

Se puede demostrar [116] que el problema se puede reformular de tal manera que w queda expresado como una combinación lineal de los predictores x_i:

$$w = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) x_i$$

Y el problema se traduce a hallar una función f tal que:

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) \langle x_i, x + b \rangle$$

$$\text{maximice:} \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\varepsilon\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

$$\text{sujeto a:} \begin{cases} \sum_{i}^{l}(\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases}$$

El algoritmo puede extenderse a casos no lineales, es decir, donde la relación entre los predictores y la variable no se ajustan a una función lineal $\langle w, x_i\rangle + b \approx y_i$. En este caso se utiliza el "Método Kernel" [2].

El Método Kernel consiste en llevar el conjunto de datos a un espacio de dimensión superior, donde su relación pueda ser expresada por funciones lineales. Para todo x_i y x_j del espacio de entrada $\mathcal{X}$, ciertas funciones $k(x_i, x_j)$ conocidas por el nombre de funciones "kernel" pueden ser expresadas como el producto interno en otro espacio $\mathcal{V}$, $k(x_i, x_j) = \langle\phi(x_i), \phi(x_j)\rangle$, donde ϕ es un mapeo $\phi : \mathcal{X} \to \mathcal{V}$ (e.g. Figura 2.8).

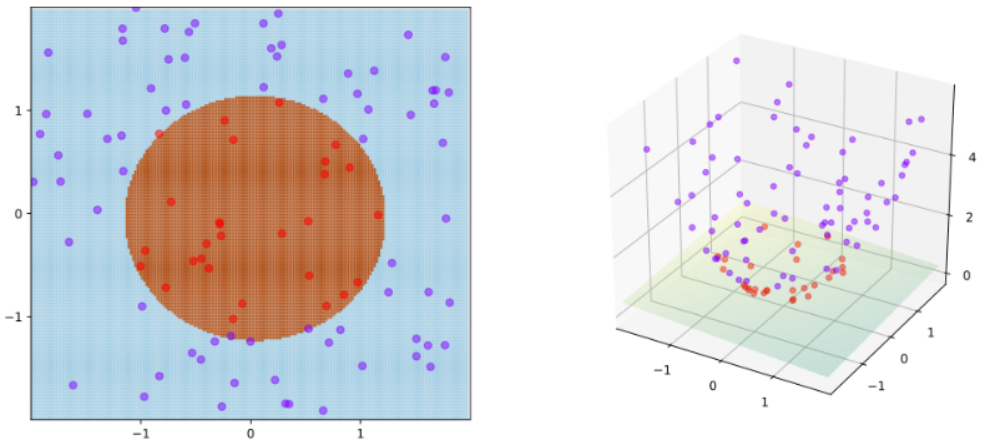


Figura 2.8: Método de Kernel. $\phi(x, y) = (x, y, x_1^2 + x_2^2)$, $k(x, y) = \langle\phi(x), \phi(y)\rangle$ (Fuente: [132]).

De esta forma, el método se puede generalizar a casos no lineales, donde w queda expresado como:

$$w = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\phi(x_i)$$

y la función f como:

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) k(x_i, x) + b$$

Una función de kernel comúnmente empleada para SVR, es la RBF (Radial Basis Function) [130]: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$

2.3.4. Redes neuronales artificiales

Las redes neuronales artificiales (ANN) son sistemas vagamente inspirados por las redes neuronales biológicas (Figura 2.9). Una red neuronal en sí misma no es un algoritmo, sino un marco para que los algoritmos de aprendizaje automático procesen entradas de datos complejos. En la década de 1940, D. O. Hebb [44] concibió una hipótesis de aprendizaje basada en el mecanismo de la plasticidad neuronal, conocido como "aprendizaje de Hebb", una forma de aprendizaje no supervisado. En las siguientes décadas, investigadores comenzaron a aplicar estas ideas a modelos computacionales. Hasta que en 1958 [109], se ideó el perceptrón, un algoritmo para el reconocimiento de patrones. Las primeras redes funcionales multi-capa fueron publicadas en 1965 [78], aunque su investigación se estancó luego de que un estudio sobre aprendizaje [96] descubrió dos problemas clave con este enfoque: los perceptrones eran incapaces de procesar el circuito lógico *or*, y la falta de poder computacional para soportar el procesamiento necesario para utilizar grandes redes neuronales. Un desencadenante clave para el interés renovado en las redes neuronales y el aprendizaje fue el algoritmo de "backpropagation" [131], el cual resolvió efectivamente el problema del *or* al permitir el entrenamiento de redes multi-capa de forma factible y eficiente.

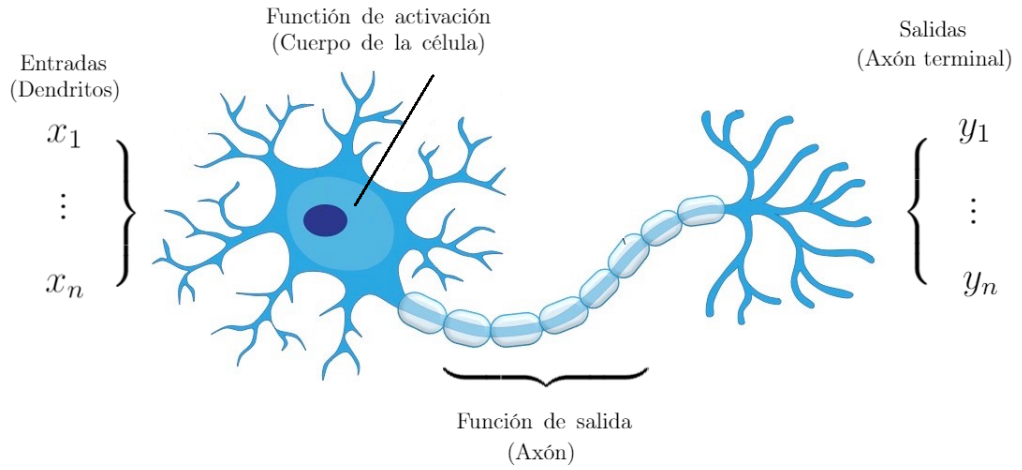


Figura 2.9: Neurona biológica y su correspondencia con una neurona artificial de una ANN.

Una ANN se basa en una colección de nodos interconectados denominados *neuronas artificiales*. Cada conexión, como las sinapsis en un cerebro biológico, puede transmitir una señal de una neurona artificial a otra. Una vez recibida, la señal es procesada y posteriormente transmitida a neuronas artificiales adicionales. Generalmente, cada señal es un número real, y la salida de cada neurona artificial se calcula mediante alguna función no lineal de la suma de sus entradas. Las conexiones entre las neuronas artificiales se llaman *bordes*. Las neuronas y los bordes artificiales suelen tener un peso que se ajusta a medida que avanza el aprendizaje, aumentando o disminuyendo la intensidad de la señal de cada neurona. Típicamente, las neuronas artificiales se agregan en capas, de forma que cada capa puede realizar diferentes tipos de transformaciones en sus entradas. Las señales viajan desde la primera capa (la capa de entrada) hasta la última capa (la capa de salida), potencialmente después de atravesar cada una de ellas varias veces.

Perceptrón

El perceptrón [108] es el modelo más simple de red neuronal, una red constituida por una capa de entrada y un único valor de salida. Esta salida se puede formular como:

$$o = \varphi(\sum_{k}^{n} w_k x_k)$$

Donde x_i denota la i-ésima entrada, w_i su respectivo peso asociado, y $\varphi(x)$ la función de activación. El propósito de la función de activación es el de proporcionar una transición *suave* a medida que cambian los valores de entrada. Esto es, un pequeño cambio en la entrada produce un pequeño cambio en la salida. En la figura 2.10 se muestran algunas funciones de activación comúnmente utilizadas.

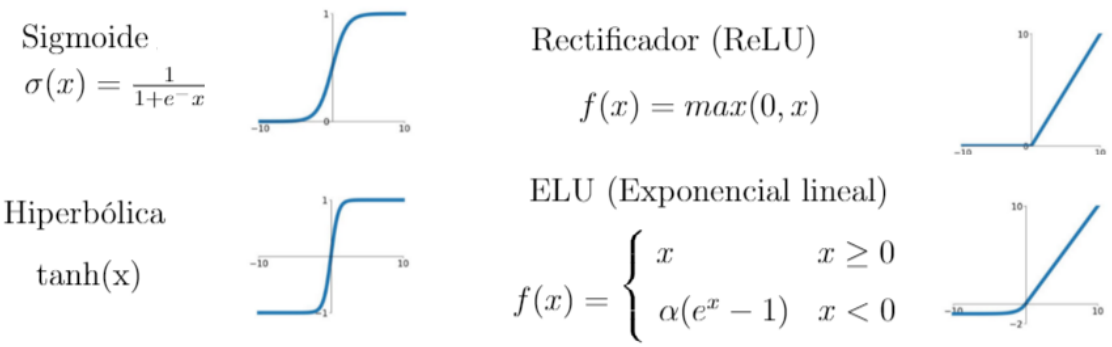


Figura 2.10: Algunas funciones de activación comúnmente utilizadas.

Se puede generalizar la definición a redes neuronales multi-capa, donde la salida de una neurona j queda formulada como una composición de funciones:

$$o_j = \varphi(\sum_{k=1}^{n} w_{kj} o_k)$$

Donde o_k denota la salida de las neuronas de la capa previa, o la entrada en el caso de la primer capa. En una red neuronal multi-capa, a las capas que no son ni de entrada ni de salida se les denomina "capas ocultas". A estas redes neuronales multi-capa se les conoce

con el nombre de "Perceptrón multicapa" (MLP).
Distintas configuraciones de las neuronas, de las capas y de sus respectivas conexiones en una red, dan lugar a una variedad de distintos tipos de redes neuronales.

Aprendizaje

Antes de entrenar la red, los pesos w_i son inicializados con valores aleatorios. Posteriormente, utilizando el conjunto de datos de entrenamiento, los pesos deben ser actualizados para que la red genere la salida deseada. Con tales fines se utiliza el algoritmo de *backpropagation*, método mediante el cual se calcula el gradiente de la función de pérdida respecto a los pesos de la red. Para cada dato del conjunto de entrenamiento, los pesos de la red son actualizados utilizando el *gradiente estocástico de descenso*:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

Donde E representa una función de pérdida de la salida actual de la red y respecto a salida esperada t (e.g.$E = RSE = (y - t)^2$). η representa la tasa de aprendizaje, es decir, la velocidad con la que los pesos serán actualizados en cada iteración.

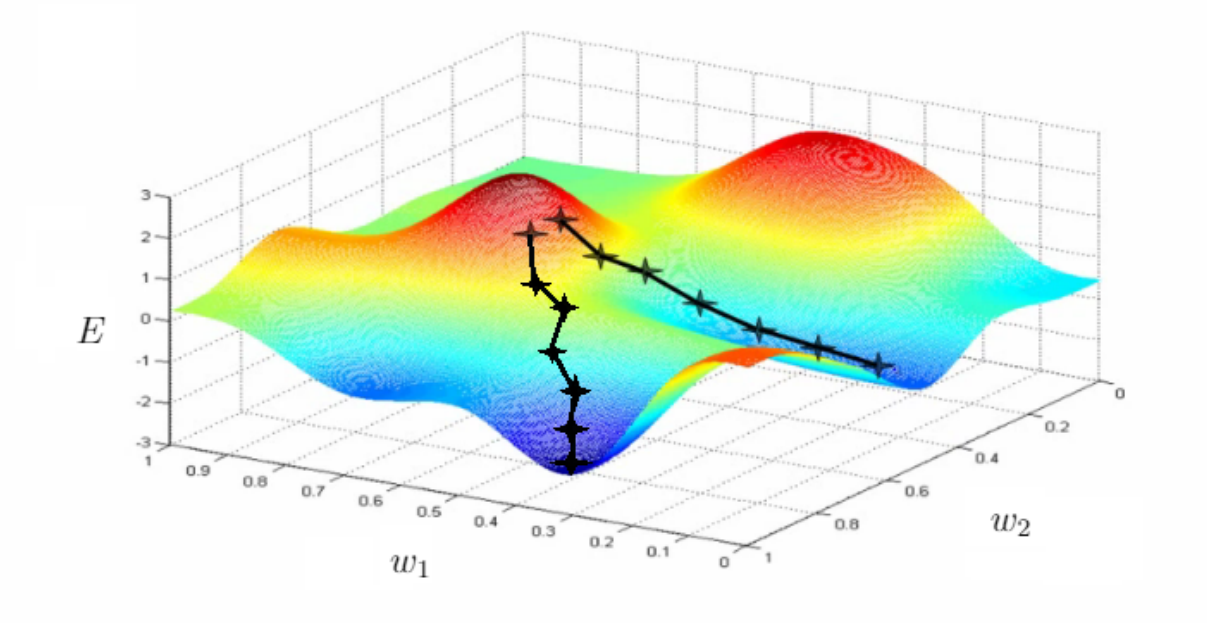


Figura 2.11: Backpropagation con gradiente estocástico de descenso en una neurona con dos pesos. Según los valores iniciales y la tasa de aprendizaje, el gradiente estocástico de descenso puede converger a mínimos locales de E distintos. (Fuente: [119])

Backpropagation con gradiente estocástico de descenso no garantiza encontrar el mínimo global de la función de pérdida, sino únicamente un mínimo local (Figura 2.11). De todas maneras, esta limitación, causada por la no convexidad de las funciones de error, no es un mayor inconveniente en la mayoría de los problemas prácticos [87].

2.3.5. Aprendizaje profundo

El aprendizaje profundo se refiere a la formación de redes neuronales con más de dos capas ocultas. En el pasado, entrenar tales redes se hacía virtualmente imposible a medida que se incrementaba la cantidad de capas. Los dos desafíos más grandes se conocieron como los problemas de "explosión del gradiente" y "desvanecimiento del gradiente". Si bien el problema de la explosión del gradiente fue más fácil de sobrellevar aplicando técnicas simples como el "recorte de gradiente" y la regularización L1 o L2 [125], el problema del desvanecimiento del gradiente se mantuvo intratable durante décadas. En el peor de los casos, este problema impide completamente que una red neuronal continúe aprendiendo

durante el entrenamiento.

La utilización de funciones de activación tradicionales durante backpropagation, como la función de tangente hiperbólica, cuyo gradiente está en el rango $(0, 1)$, tiene el efecto de multiplicar estos pequeños números para calcular los gradientes de las capas anteriores. Esto implica que el gradiente disminuye exponencialmente en la cantidad de capas. En consecuencia las capas anteriores entrenan muy lentamente, si es que lo hacen en absoluto. Sin embargo, las implementaciones modernas de los algoritmos de aprendizaje de redes neuronales permiten entrenar efectivamente redes neuronales muy profundas (hasta cientos de capas). Esto se debe a una combinación de varias mejoras, entre ellas las funciones de activación de la familia ReLU, arquitecturas de redes más sofisticadas (ver secciones 2.3.6-2.3.8), así como a modificaciones avanzadas del algoritmo de descenso de gradiente. Por lo tanto, hoy en día, dado que los problemas de desvanecimiento y explosión del gradiente están mayormente resueltos (o su efecto disminuido), el término "aprendizaje profundo" se refiere al entrenamiento de redes neuronales utilizando el juego de herramientas algorítmico y matemático actual, independientemente de la profundidad de la red neuronal.

2.3.6. Redes neuronales convolucionales (CNN)

Las redes neuronales convolucionales (CNN) [88] son un tipo de red neuronal inicialmente concebido para el reconocimiento de imágenes. Las redes neuronales multi-capa convencionales (MLP), si bien pueden ser utilizadas satisfactoriamente para el procesamiento de imágenes, sufren de la "maldición de la dimensión" [102] (también conocido como efecto Hughes) al intentar procesar imágenes de alta resolución. Además, dicha arquitectura de red no tiene en cuenta la estructura espacial de los datos, ya que trata los píxeles de entrada que están muy separados de la misma manera que los píxeles que están muy juntos. De tal forma se ignora por completo la referencia de localidad de los datos de imagen, tanto computacional como semánticamente.

Las redes neuronales convolucionales mitigan estos desafíos explotando la localidad es-

pacial mediante la aplicación de un patrón de conectividad local entre las neuronas de las capas adyacentes. La arquitectura asegura así que los "filtros" aprendidos producen la respuesta más fuerte a un patrón de entrada espacialmente local. El apilamiento de muchas de estas capas conduce a filtros no lineales que se vuelven cada vez más globales (es decir, responden a una región más grande del espacio de píxeles), de modo que la red crea primero representaciones de pequeñas partes de la entrada, para luego a partir de ellas ensamblar representaciones de áreas más grandes.

En las CNN, cada filtro se replica en todo el campo visual. Estas unidades replicadas comparten la misma parametrización (pesos y sesgo) y forman un mapa de características de forma que todas las neuronas en una capa convolucional dada responden a la misma característica dentro de su campo de respuesta específico. Esto permite que las características se detecten independientemente de su posición en el campo visual. Gracias a estas propiedades, las CNN alcanzan una mejor generalización de los problemas de reconocimiento de imágenes. El intercambio de peso reduce drásticamente el número de parámetros libres aprendidos, lo que reduce los requisitos computacionales para ejecutar la red y permite el entrenamiento de redes más grandes y más potentes.

Capa de convolución

Una capa de convolución consiste de un conjunto de filtros (también llamados kernels), que tienen un campo receptivo pequeño, pero se extienden a través de la profundidad completa del volumen de entrada. Durante el paso hacia adelante, cada filtro se convierte en convolvente a lo ancho y alto del volumen de entrada, calculando el producto de punto entre las entradas del filtro y la entrada, produciendo un mapa de activación bidimensional de ese filtro. Como resultado, la red aprende filtros que se activan cuando detecta algún tipo específico de característica en cualquier posición espacial en la entrada. En la Figura2.12 se muestran algunos ejemplos de filtros aplicados en la capa convolucional.

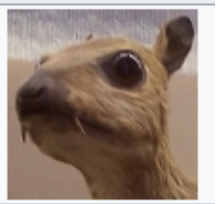
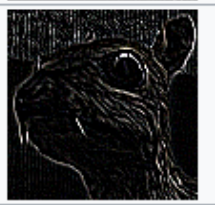
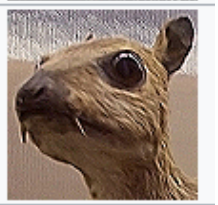
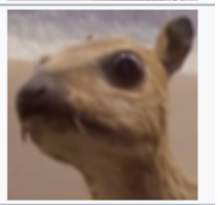
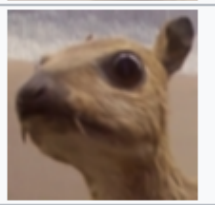
Operación	Filtro	Resultado
Identidad	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Detección de bordes	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Enfoque	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Desenfoque	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Desenfoque gaussiano	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figura 2.12: Características capturadas por distintos filtros aplicados en la capa de convolución (Fuente: [84]).

Dos propiedades importantes de la convolución son el *tamaño del paso* y el *relleno*. El tamaño del paso (conocido como "stride") determina el tamaño de la ventana de desplazamiento del filtro, es decir, cuantos píxeles se desliza el filtro en cada paso de la convolución.

El relleno (conocido como "padding") permite obtener una matriz de salida más grande que la original; es el ancho del cuadrado de celdas adicionales con las que se rodea la imagen antes de procesarla con el filtro. Las celdas agregadas por el relleno generalmente contienen ceros.

Pooling

Luego de cada capa de convolución se suele agregar una capa de pooling (o de reducción de muestreo). Estas capaz permiten reducir el número de parámetros cuando las imágenes son demasiado grandes, disminuyendo la dimensionalidad de cada mapa pero conservando la información importante. En general, se utiliza la operación de "max-pooling", la cual toma el elemento más grande del mapa de características; se ha demostrado [112] experimentalmente que esta operación genera mejores resultados que tomar el promedio ("average-pooling") o la suma ("sum-pooling").

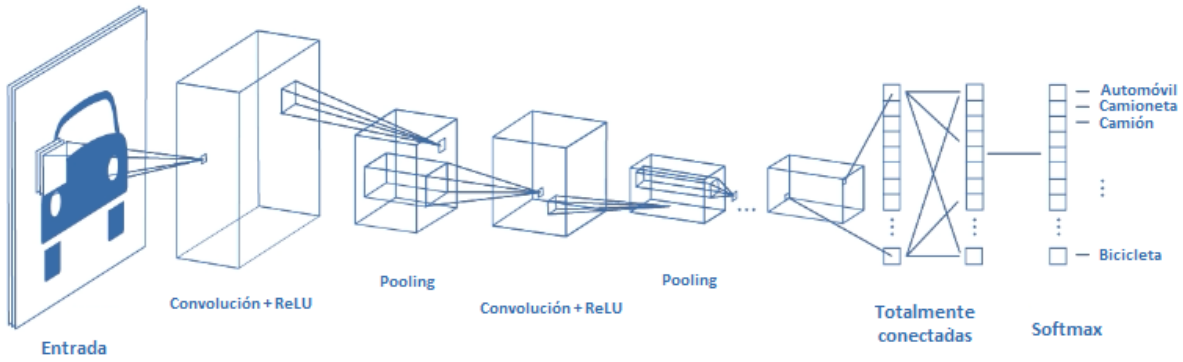


Figura 2.13: Arquitectura de una red convolucional para procesamiento de imágenes. (Fuente: [91])

Luego de varias capas convolucionales y de pooling, el razonamiento de alto nivel en la red neuronal se suele realizar a través de capas totalmente conectadas (MLP). La Figura 2.13 ilustra un ejemplo de este tipo de arquitectura empleado para el procesamiento de imágenes.

Las CNN han reportado excelentes resultados en cuanto a precisión y velocidad en sistemas de reconocimiento de imágenes, tanto de detección de objetos, reconocimiento facial, o reconocimiento a gran escala; alcanzando niveles de precisión similares a los del ser humano [24, 85, 92, 86, 49, 123]. Asimismo, se han utilizado con excelentes resultados a problemas con entradas unidimensionales, tales como interpretación del lenguaje natural [115], reconocimiento de voz [1] o predicciones en series temporales [137].

En [94], se emplea una red convolucional que toma como entrada una matriz bidimensional generada a partir de información de índices financieros relacionados con el precio de Bitcoin, y produce como salida la dirección pronosticada del movimiento diario del precio. Se compara su precisión con respecto a modelos MLP y SVM, obteniendo en todos los casos resultados significativamente superiores a través del modelo CNN.

2.3.7. Redes neuronales recurrentes (RNN)

Las redes neuronales recurrentes (RNN) [110] son una familia de redes neuronales concebidas para el procesamiento de datos secuenciales. Estas redes son capaces de escalar a secuencias mucho más largas de lo que sería práctico para redes sin especialización basada en secuencias. La mayoría de las redes recurrentes también pueden procesar secuencias de longitud variable.

Cada unidad u de una capa recurrente l tiene un estado con valores reales $h_{l,u}$; dicho estado puede ser considerado como la memoria de la unidad. En estas redes, cada unidad en cada capa l recibe dos entradas: un vector de estados de la capa anterior $l-1$ y el vector de estados de esta misma capa del paso temporal anterior (Figura 2.14).

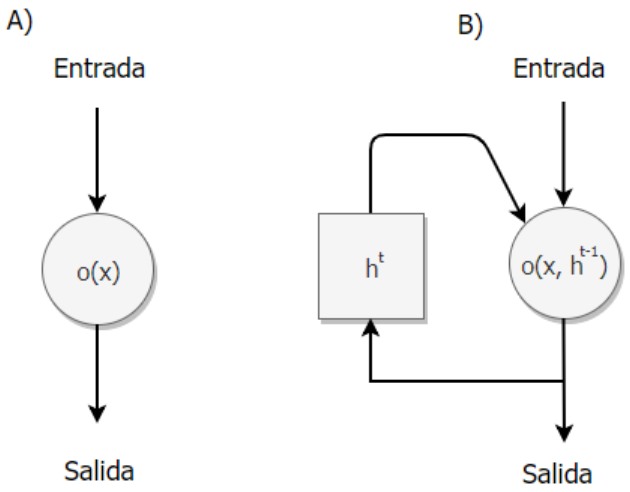


Figura 2.14: A) Perceptrón. B) Capa de red neuronal recurrente, en cada paso la salida depende no sólo de la entrada sino del estado h calculado en base al paso temporal previo.

Dado un conjunto de datos de entrada $[x_1, x_2...x_{t-1}, x_t, x_{t+1}...]$, donde x_i representa un vector de datos de entrada para el i-ésimo paso de una serie temporal, cada vector de datos es leído por la red neuronal secuencialmente en el orden de los pasos del tiempo.

El estado $h^t_{l,u}$ en cada paso de tiempo t en cada unidad u de cada capa l se calcula como:

$$h^t_{l,u} = g_l(w_{l,u}x_t + u_{l,u}h^{t-1} + b_{l,u})$$

Es decir, el estado resultante será una combinación lineal del vector de datos de entrada con el vector del estado del paso anterior de la misma capa, luego de aplicarle la correspondiente función de activación g_l. Al igual que en las MLP, estas capas se pueden apilar generando arquitecturas de redes neuronales profundas.

Gated Recurrent Units (GRU)

Uno de los principales problemas que enfrentan las RNN es el manejo de dependencias a largo plazo. A medida que aumenta la longitud de la secuencia de entrada, los vectores del comienzo de la secuencia tienden a ser "olvidados", debido a que el estado de cada unidad

se ve afectado significativamente por los vectores leídos más recientemente. Esto puede causar que se pierdan vínculos de causa-efecto entre datos distantes en secuencias largas. Las Gated Recurrent Units (GRU) [20] intentan resolver este problema, almacenando información en su "memoria" a largo plazo para uso futuro. La lectura, escritura y borrado de la información almacenada en cada unidad es controlada por funciones de activación que toman valores en el rango $(0, 1)$, la RNN puede "leer" la secuencia de entrada y decidir en cada momento si mantener información específica sobre cada vector. Esa información puede ser utilizada posteriormente por el modelo para procesar vectores ubicados en el final de la secuencia. Esas decisiones son entrenables y se implementan a través del concepto de "puertas". Hay varias arquitecturas de este tipo de unidades, generalmente compuestas de una "puerta de actualización" y una "puerta de olvido" (Figura 2.15).

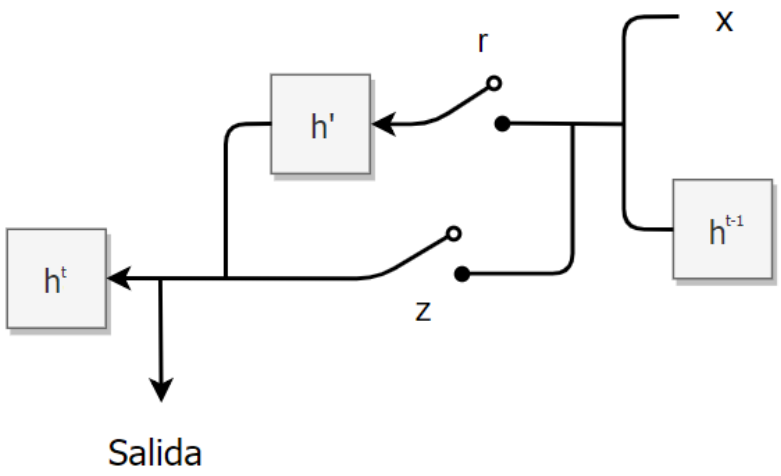


Figura 2.15: GRU. La salida de la unidad en el paso t está determinado por la entrada x, el estado del paso anterior h^{t-1}, el estado actual h', la puerta de olvido r y la puerta de actualización z.

El estado de la celda de memoria h_t de la unidad queda formulado como:

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

$$h'_t = tanh(W_{x_t} + r_t \cdot Uh_{t-1})$$

Donde x_t es el vector de entrada, z_t es la puerta de actualización, r_t es la puerta de olvido, W, U y b son los pesos entrenables de la unidad, y σ_g y σ_h son las funciones de activación sigmoide y tangente hiperbólica respectivamente. Intuitivamente, las puertas z y r determinan el peso que tendrá la información del estado de memoria de los pasos anteriores sobre la salida y el estado de la memoria en el paso actual.

Long Short Term Memory Networks (LSTM)

Al igual que las GRU, las Long Short Term Memory Networks (LSTM) [46, 42] tienen el objetivo de modelar las dependencias a largo plazo de manera efectiva. Las LSTM pueden verse como una versión modificada de las GRU, donde se incorporan dos puertas más, una de entrada y una de salida. La puerta de entrada regula la proporción del estado de la memoria actual que se debe mantener y la puerta de salida regula la cantidad del estado de la memoria que se debe exponer a las siguientes capas de la red. Las LSTM controlan la exposición del contenido de la memoria mientras que las GRU exponen todo el estado de la memoria a las otras unidades en la red.
Una celda LSTM se puede expresar como:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma_{\ g}(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$h_t = o_t \circ \sigma_h(c_t)$$

Donde x_t es el vector de datos de entrada, f_t es es la puerta de olvido, i_t es la puerta de

entrada, o_t es la puerta de salida, h_t es la salida de la unidad LSTM, c_t es la celda de memoria y W, U y b denotan los pesos asociados. El subíndice t de cada variable indica el paso temporal correspondiente.

Las LSTM pueden ser apiladas en arquitecturas de RNN profundas, y como tales han reportado excelentes resultados en tareas como reconocimiento del habla [39, 38], reconocimiento de escritura a mano [40, 41] y predicciones en series temporales [37]. Asimismo, las redes neuronales basadas en GRU han reportado resultados similares a las LSTM [23, 80], con la ventaja de implicar una complejidad computacional inferior.

En [126], se compara la efectividad de utilizar modelos LSTM y GRU para realizar predicciones diarias sobre el precio de Bitcoin, reportándose resultados similares (RMSE=272.96 y RMSE=274.02 respectivamente), aunque se destaca el menor tiempo computacional demandado para el entrenamiento de la red neuronal basada en GRU. Sobre el mismo problema, en [81] se realiza un estudio comparativo entre un modelo estadístico ARIMA y una red neuronal recurrente con tres capas LSTM. Los resultados arrojan una precisión significativamente superior para el modelo LSTM (RMSE=93.27 contra RMSE=1146.07).

2.3.8. Redes neuronales híbridas CNN + LSTM

Se han empleado con éxito redes neuronales híbridas que combinan distintos tipos de capas en su arquitectura. Una posible combinación que ha reportado prometedores resultados en una variedad de aplicaciones, es la conjunción de capas convolucionales con capas recurrentes. Se han utilizado con gran éxito, por ejemplo, en el procesamiento de vídeo [120, 35], en este caso aprovechando la capacidad de las CNN para la interpretación de imágenes, y de las LSTM para procesar secuencias (en este caso, de fotogramas). Análogamente, se han utilizado con éxito para realizar pronósticos sobre series temporales [47, 136], de forma que las capas convolucionales realizan la extracción de características de alto nivel de la serie temporal, y el resultado en forma de secuencia es interpretado por

capas LSTM.

Capítulo 3

Descripción del problema

3.1. Introducción

La serie temporal del precio de Bitcoin es una variable altamente compleja, con un alto nivel de volatilidad y una innumerable variedad de factores que inciden en su valor. En este contexto, se plantea evaluar la capacidad de distintos modelos de aprendizaje automático en la tarea de realizar pronósticos de corto plazo sobre el precio de Bitcoin para un determinado horizonte a futuro. Los modelos deberán aprender a partir de la información histórica, y ser capaces de generalizar para predecir su valor en momentos no observados durante el entrenamiento.

3.2. Datos

Como fuente de datos se utilizó la API pública [10] de la plataforma de intercambio de criptomonedas Binance [9], la cual provee información sobre el mercado de intercambio de Bitcoin, así como de otras criptomonedas desde el año 2015 hasta el presente. El conjunto de datos consiste de una serie de puntos correspondientes a cada minuto del histórico de

transacciones de Bitcoin registrados en la plataforma. Las variables contenidas en cada punto de datos (Cuadro 3.1) conforman lo que se denomina una "vela". El formato de vela expone de forma resumida patrones de intercambio de pequeños intervalos de tiempo (Figura 3.1).

Cuadro 3.1: Información incluida en cada punto del conjunto de datos fuente.

Timestamp	Fecha y hora del intervalo.
Open	Precio de apertura.
Close	Precio de clausura.
Low	Precio mínimo alcanzado.
High	Precio máximo alcanzado.
Volumen	Volumen total de transacciones.

Figura 3.1: Representación gráfica de un día del precio de Bitcoin en formato de vela, agregado por hora. Si el precio de apertura está por encima del precio de cierre, se dibuja una vela de color rojo y de lo contrario de color negro. Las líneas superiores e inferiores, conocidas como sombras, colas o mechas, representan los rangos de precios máximo y mínimo dentro del periodo de tiempo correspondiente.

Al considerar el conjunto de datos de forma cronológicamente ordenada según su fecha y hora (Timestamp), se conforma una serie temporal (Cuadro 3.2).

Cuadro 3.2: Cinco minutos de la serie temporal conformada por el conjunto de datos fuente del histórico de Bitcoin.

Timestamp	Open	Close	Low	High	Volume
2016-11-05 6:01:00	5,707.53	6,951.36	5,707.53	8,765.13	3,695.89
2016-11-05 6:02:00	6,954.62	7,032.08	6,882.06	7,093.23	2,355.27
2016-11-05 6:03:00	7,014.14	7,046.76	6,976.64	7,229.40	4,874.40
2016-11-05 6:04:00	7,039.01	7,076.52	6,981.12	7,175.18	8,426.38
2016-11-05 6:05:00	7,072.85	7,091.20	6,910.59	7,134.41	7,461.64

Los aproximadamente 4 años de información del conjunto fuente empleado suponen un total de 2.430.000 puntos de datos, donde cada uno contiene los valores del precio y volumen correspondientes a intervalos de un minuto de la totalidad del periodo 2015 - 2019.

3.3. Método de evaluación

Se define un método de evaluación que permita comparar la capacidad de predicción de los modelos. Si bien la gran mayoría de la bibliografía relacionada existente define un horizonte de predicción de mediano plazo (1 día o más), se opta por un periodo lo suficientemente pequeño como para que sea útil en el contexto del intercambio de mediana frecuencia, pero a su vez lo suficientemente grande como para que haya una varianza que produzca una dificultad interesante. A tales efectos se plantea utilizar un horizonte de predicción de 2 horas. Los modelos a evaluar realizarán una predicción del precio de cierre para un cierto instante t, utilizando, a lo sumo, la información disponible hasta el instante $t - 2hrs.$. Asimismo, se define el mes de diciembre de 2018 como conjunto de evaluación, lo cual implica que los modelos no tendrán acceso a las observaciones de este periodo de tiempo durante su entrenamiento. La precisión de sus resultados será evaluada utilizando como métrica el RMSE (Root Mean Square Error):

$$RMSE = \sqrt{\overline{(y - o)^2}}$$

Donde y denota el valor de la predicción y o el valor real observado.

Es de especial interés evaluar la calidad de las predicciones en los momentos de mayor volatilidad, donde ocurren las caídas más bruscas y los incrementos más pronunciados del precio. Se identificaron tales momentos en el periodo de evaluación considerado, con el fin de comparar el desempeño de los modelos alrededor de estos puntos de mayor

incertidumbre. Para ello, se calculó la diferencial discreta de primer orden sobre la serie temporal del precio por hora durante el mes de diciembre:

$$\text{diff}(c) = c_t - c_{t-1}$$

Hallando los mínimos y máximos de esta nueva serie, se identifican los momentos donde ocurrieron las mayores caídas y subidas del precio por hora respectivamente (Figura 3.2).

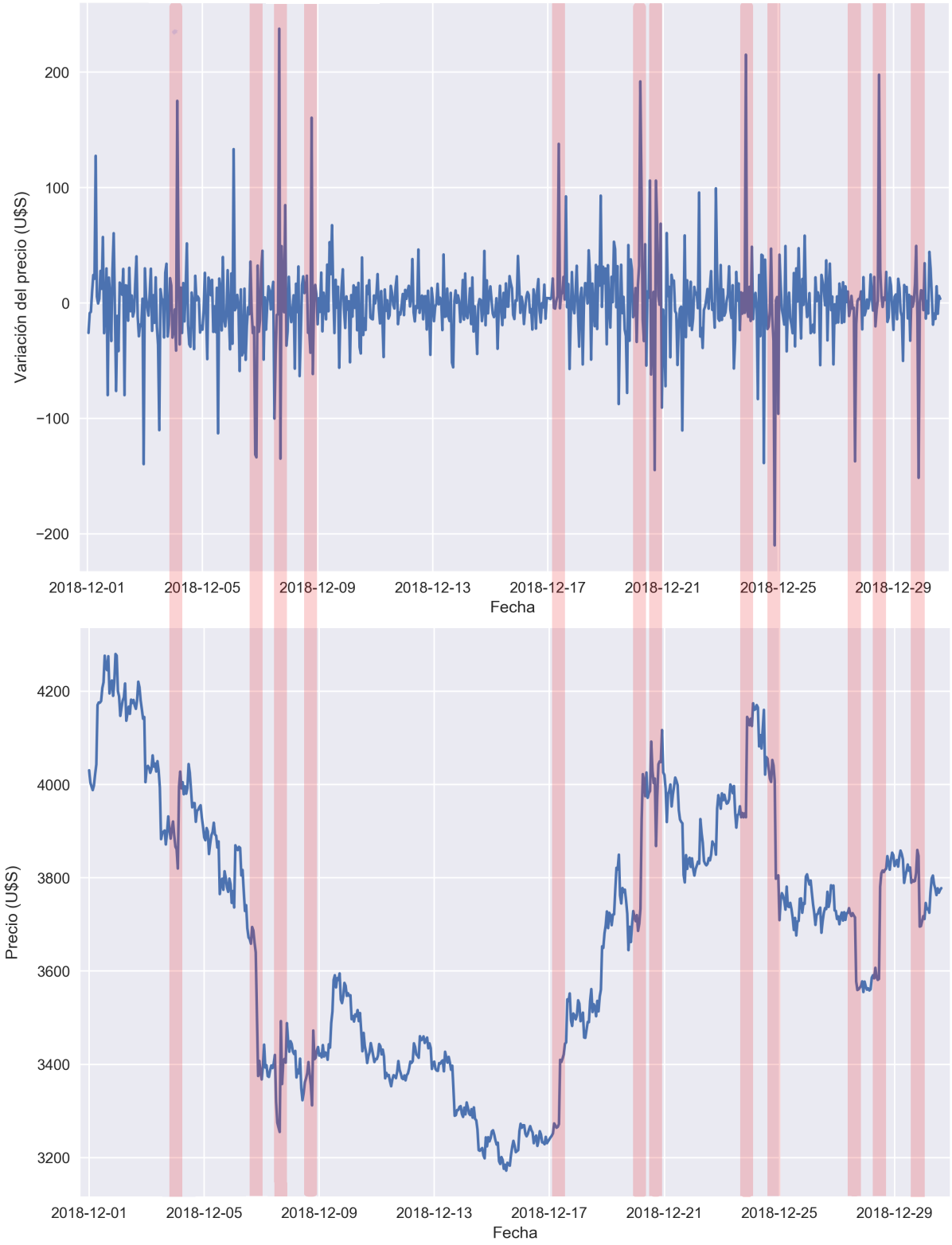


Figura 3.2: Identificación de anomalías. Las mayores caídas/subidas en el precio corresponden con los mínimos/máximos de la diferencial.

Se consideran las 5 mayores caídas y subidas (en adelante referidos como "anomalías"), alrededor de los cuales serán evaluados los modelos utilizando la métrica de RMSE; dicho resultado será referido como RMSEA.

Por último, se desea evaluar la capacidad de los modelos de predecir la "dirección" del cambio de precio, es decir, si va a subir o bajar en el horizonte considerado. Dado el precio de cierre c_t, su correspondiente predicción y_t, y la cantidad de puntos n del periodo de evaluación, se calcula el porcentaje de aciertos de la dirección del movimiento de precio (de ahora en más, "DA") como:

$$DA = 100 * \frac{\sum_t^n \begin{cases} 1 \text{ si } (c_t > c_{t+2hr} \text{ y } c_t > y_t) \text{ ó } (c_t \leq c_{t+2hr} \text{ y } c_t \leq y_t) \\ 0 \text{ en otro caso.} \end{cases}}{n}$$

Es decir, el porcentaje de veces que se predijo correctamente que el precio subiría en el horizonte de 2 horas.

Capítulo 4

Soluciones evaluadas

4.1. Introducción

Se propone plantear el problema como uno de aprendizaje supervisado, utilizando la información histórica del mercado como conjunto de entrenamiento, los datos recientes como características de entrada, y el precio para un horizonte a futuro de 2 horas como la respuesta de salida esperada.

Siguiendo la premisa del teorema de "no hay almuerzo gratis", se evalúa y compara el desempeño de una variedad de modelos de aprendizaje automático, con el objetivo de obtener las mejores predicciones de acuerdo a las métricas definidas.

Todos los modelos y el framework de pruebas fueron desarrollados en Python 3.6 [105], utilizando los wrapper provistos por Keras [83] de las implementaciones de redes neuronales de TensorFlow [124]. Para los modelos SVR se utilizó la implementación de Scikit-Learn [114].

Para cada tipo de modelo considerado, se busca optimizar sus correspondientes hiperparámetros de forma de maximizar la precisión de las predicciones generadas. Parte del

estudio se centrará en analizar la incidencia de cada hiperparámetro en el comportamiento de los modelos.

Una vez acotado el conjunto de hiperparámetros óptimos de cada modelo, se propone mejorar la precisión mediante la utilización de características generadas en base a indicadores de análisis técnico. Posteriormente, y con el mismo objetivo, se plantea la generación de datos de entrenamiento adicionales obtenidos a partir de la información del mercado de otras criptomonedas.

4.2. Procesamiento de datos para aprendizaje supervisado

Con el fin de plantear el problema como uno de aprendizaje supervisado, se debe generar un conjunto de tuplas de la forma (X_i, y_i), tal que X_i representa el vector de las i-ésimas variables de entrada e y_i la i-ésima correspondiente variable de respuesta. En este caso, la variable de respuesta y_t corresponderá al precio de cierre de Bitcoin c_t en un momento t dado. Como variables de entrada se utilizan los datos disponibles hasta 2 horas previas al precio de clausura a pronosticar, incluyendo hasta n puntos de datos correspondientes a n pasos temporales previos de la serie temporal (de ahora en más referidos como *pasos previos*), siendo n uno de los hiperparámetros que se busca optimizar en la construcción de los modelos.
El conjunto de datos queda formulado como un conjunto de tuplas (X_t, y_t) tal que:

$$\begin{cases} X_t = (X_{t-2hr-(n-1)}, X_{t-2hr-(n-2)}, ..., X_{t-2hr-1}, X_{t-2hr}) \\ y_t = c_t \end{cases}$$

En la Figura 4.1 se ejemplifica la creación de una tupla para el aprendizaje automático, empleando como características de entrada 4 pasos previos del precio de clausura con una

granularidad de 1 hora.

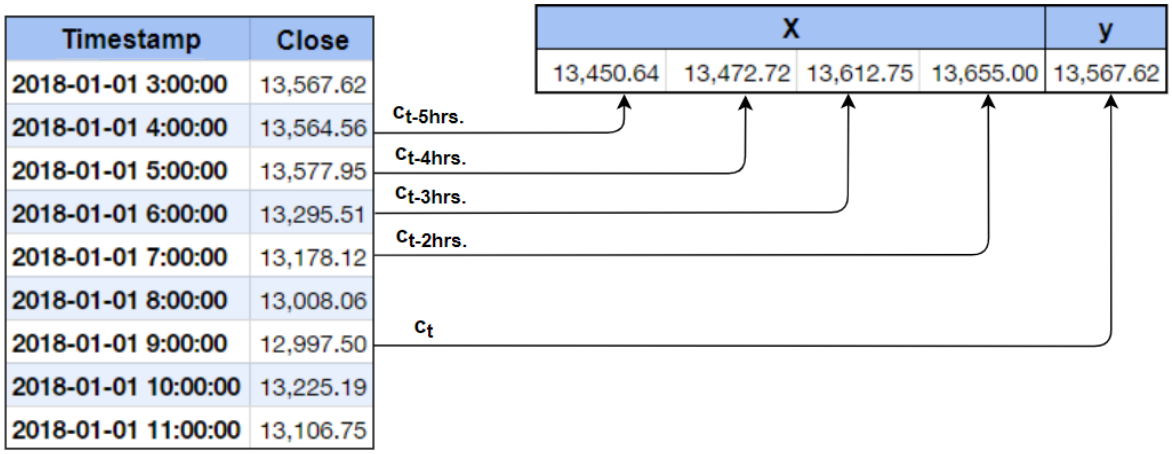


Figura 4.1: Generación de una tupla X, y para el aprendizaje supervisado, utilizando el precio de cierre como única variable de entrada, con $n = 4$ pasos previos de granularidad 1 hora.

Las características a incluir en el vector de variables de entrada X_t y su influencia en el desempeño de los modelos forman parte del análisis a desarrollar en este estudio.

Granularidad

Los datos fuente contienen información agregada por minuto. Parece excesivo emplear una granularidad tan fina habiendo definido el horizonte de predicción en 2 horas. Si bien la elección de la granularidad a utilizar posiblemente implique un impacto en el rendimiento de los modelos, se opta por no sobrecargar la (ya suficientemente extensa) lista de hiperparámetros a optimizar, y se define emplear una granularidad de una hora para todos los casos. Esto representa una reducción de la cantidad de puntos de datos del conjunto de datos fuente, pasándose de un total de 2.430.000 de puntos del conjunto original a 43.200 puntos agregados por hora, de los cuales 720 puntos correspondientes al mes de diciembre de 2018 serán empleados como conjunto de evaluación, y los restantes 42.480 como conjunto de entrenamiento.

Estandarización

La estandarización del conjunto de datos es un requisito común para muchos modelos de aprendizaje automático. Si las variables individuales no se ajustan aproximadamente a una distribución normal, es posible que se obtengan resultados inesperados. Por ejemplo, muchos elementos utilizados en la función objetivo de algunos algoritmos de aprendizaje (como el kernel RBF de SVR o los regularizadores L1 y L2 de los modelos lineales) asumen que todas las variables se centran alrededor de 0 y tienen varianza en el mismo orden. Si una variable tiene una varianza que es órdenes de magnitud mayor que otras, podría dominar la función objetivo, impidiendo que el modelo aprenda de otras variables correctamente como es esperado.

Antes de utilizar los datos como conjunto de entrenamiento, cada una de las variables de entrada, así como la respuesta de salida son estandarizados de forma que:

$$\hat{X} = \frac{X - E[X]}{\sigma(X)}$$

Siendo $\hat{X}$ el valor estandarizado, $E[X]$ la media y $\sigma(X)$ la varianza de la variable X respectivamente.

La Figura 4.2 ilustra el proceso para la variable correspondiente al precio de clausura de Bitcoin, apreciándose que a pesar de la transformación, la "forma" de la gráfica se mantiene incambiada.

Figura 4.2: El precio de Bitcoin antes (arriba) y después (abajo) del proceso de estandarización, con media 0 y varianza 1.

4.3. Modelos

4.3.1. Modelo base

Con el objetivo de obtener una idea de la dificultad del problema planteado, se define un modelo base *ingenuo* que asume que el precio se va a mantener constante. Para todo el conjunto de prueba, el modelo generará como resultado una predicción tal que:

$c_t = c_{t-2hr}$

4.3.2. SVR

Los modelos basados en el algoritmo de SVR son relativamente simples pero en general efectivos. Se implementó un modelo utilizando este algoritmo, proveyendo un robusto punto de partida y de referencia antes de considerar modelos más sofisticados. En particular, se utilizaron implementaciones de SVR con funciones de kernel lineal, polinomial y RBF.

4.3.3. LSTM

Las redes neuronales recurrentes están especialmente equipadas para procesar datos secuenciales, tal como lo es el caso de la serie temporal del precio de Bitcoin. Dentro de esta familia de redes neuronales, las LSTM han reportado excelentes resultados en el procesamiento de series temporales, por lo que resulta natural aplicar este tipo de arquitectura al presente problema.

Se propone utilizar una red neuronal con múltiples capas LSTM, seguidas de una capa de MLP y una capa de salida. La cantidad de capas y de neuronas de cada capa son considerados como hiperparámetros a optimizar.

De manera de utilizar la red efectivamente, cada tupla de entrenamiento proporcionada incluirá las variables de entrada ordenadas de manera temporalmente secuencial.

La Figura 4.3 ilustra un ejemplo de red neuronal con esta arquitectura, donde las características de entrada correspondientes a 4 pasos previos de información son secuencialmente alimentadas a una capa LSTM de entrada, y su correspondiente salida es posteriormente procesada por una capa LSTM oculta adicional, una capa oculta MLP y una capa de salida.

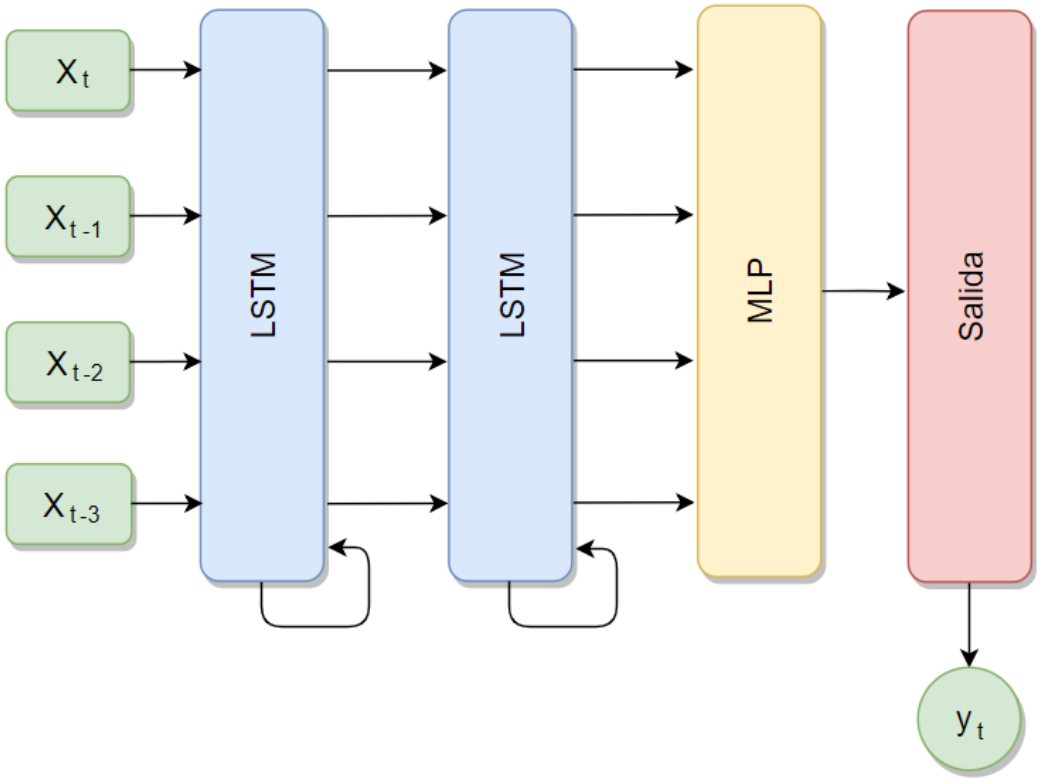


Figura 4.3: Ejemplo de arquitectura de red neuronal con una capa LSTM de entrada, una capa LSTM oculta, y una capa MLP oculta, utilizando 4 pasos previos como entrada.

4.3.4. GRU

Las redes neuronales GRU pueden considerarse una versión simplificada de las LSTM, reportando incluso mejores resultados en una variedad de contextos. Se utilizó el mismo tipo de arquitectura empleado para el modelo LSTM (Figura 4.3) pero sustituyendo las

capas LSTM por unidades GRU en su lugar. Es decir, el conjunto de características de entrada secuencialmente ordenado es alimentado a una serie de una o más capas GRU, seguidas de una capa MLP oculta y una capa de salida.

4.3.5. CNN

Las redes neuronales convolucionales, si bien concebidas para el procesamiento de imágenes, han reportado prometedores resultados en el procesamiento de datos unidimensionales. Esto se debe a su capacidad de reconocer e interpretar patrones espacialmente cercanos, o en el caso de series temporales, temporalmente cercanos. Se utilizó una red neuronal con múltiples capas convolucionales unidimensionales, donde cada variable de entrada es interpretada como un canal independiente, y es proporcionada de manera secuencialmente ordenada. De esta manera, la operación de convolución será aplicada sobre la serie temporal que representa cada variable de entrada.
La cantidad de capas, el tamaño del kernel, y la cantidad de filtros son parte de los hiperparámetros a optimizar.
Es usual utilizar capas de pooling entre cada par de capas de convolución como forma de reducir la cantidad de parámetros del modelo, en especial teniendo en cuenta la gran cantidad de parámetros presentes en problemas de procesamiento de imágenes de alta calidad. En este caso, tratándose de un problema unidimensional, la cantidad de parámetros parece manejable, por lo que se optó por no utilizar este tipo de capas.

La Figura 4.4 muestra un ejemplo de este tipo de modelo, donde las series temporales conformadas por n pasos de cada característica de entrada, son procesadas como canales independientes por una secuencia de una o más capas convolucionales, seguidas de una capa MLP oculta precediendo a la capa de salida.

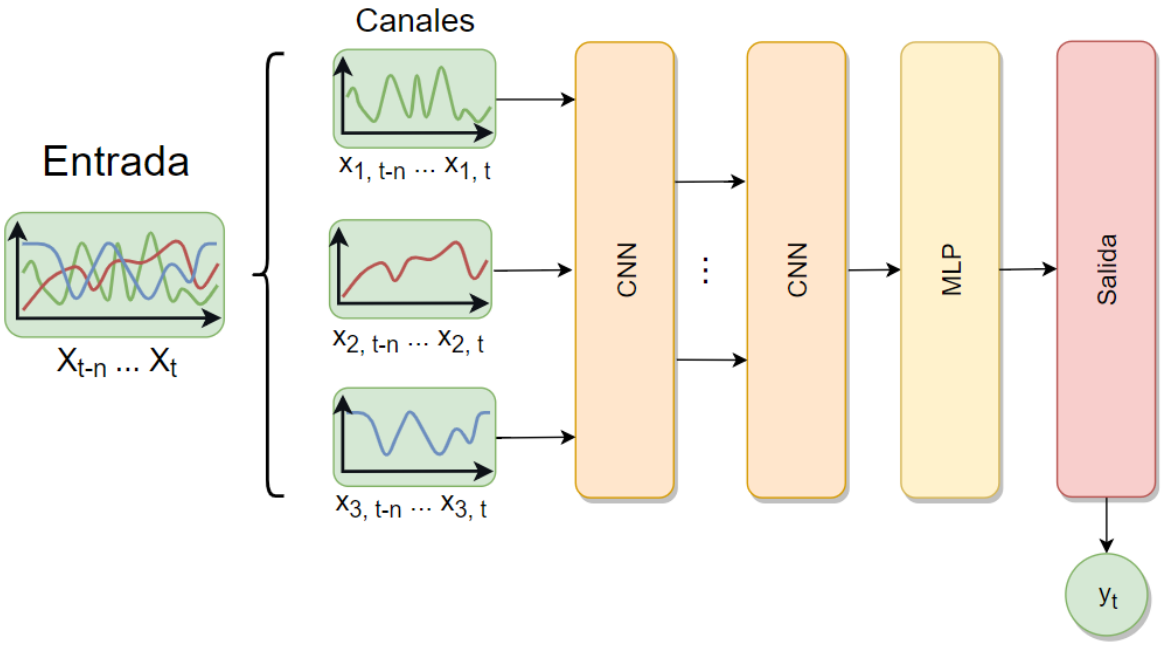


Figura 4.4: Ejemplo de arquitectura de red neuronal con una capa CNN de tres canales de entrada, una capa CNN oculta y una capa MLP oculta.

4.3.6. CNN + LSTM

Se propone utilizar una arquitectura de red neuronal compuesta por una combinación de capas CNN y LSTM. La entrada es agrupada en intervalos de tiempo, los cuales son procesados por una o más capas CNN y el resultado es luego alimentado de manera secuencialmente ordenado a una o más capas LSTM (Figura 4.5). La intención de esta arquitectura es la de explotar las capacidades específicas de cada uno de estos tipos de capa. Las CNN son las encargadas de proveer una interpretación de alto nivel de pequeños intervalos de tiempo de las variables de entrada. Sobre esta información pre-interpretada, se utilizan las capacidades de las capas LSTM para procesar datos secuencialmente ordenados. Seguidamente, antes de la capa de salida, se incluye una capa MLP oculta.

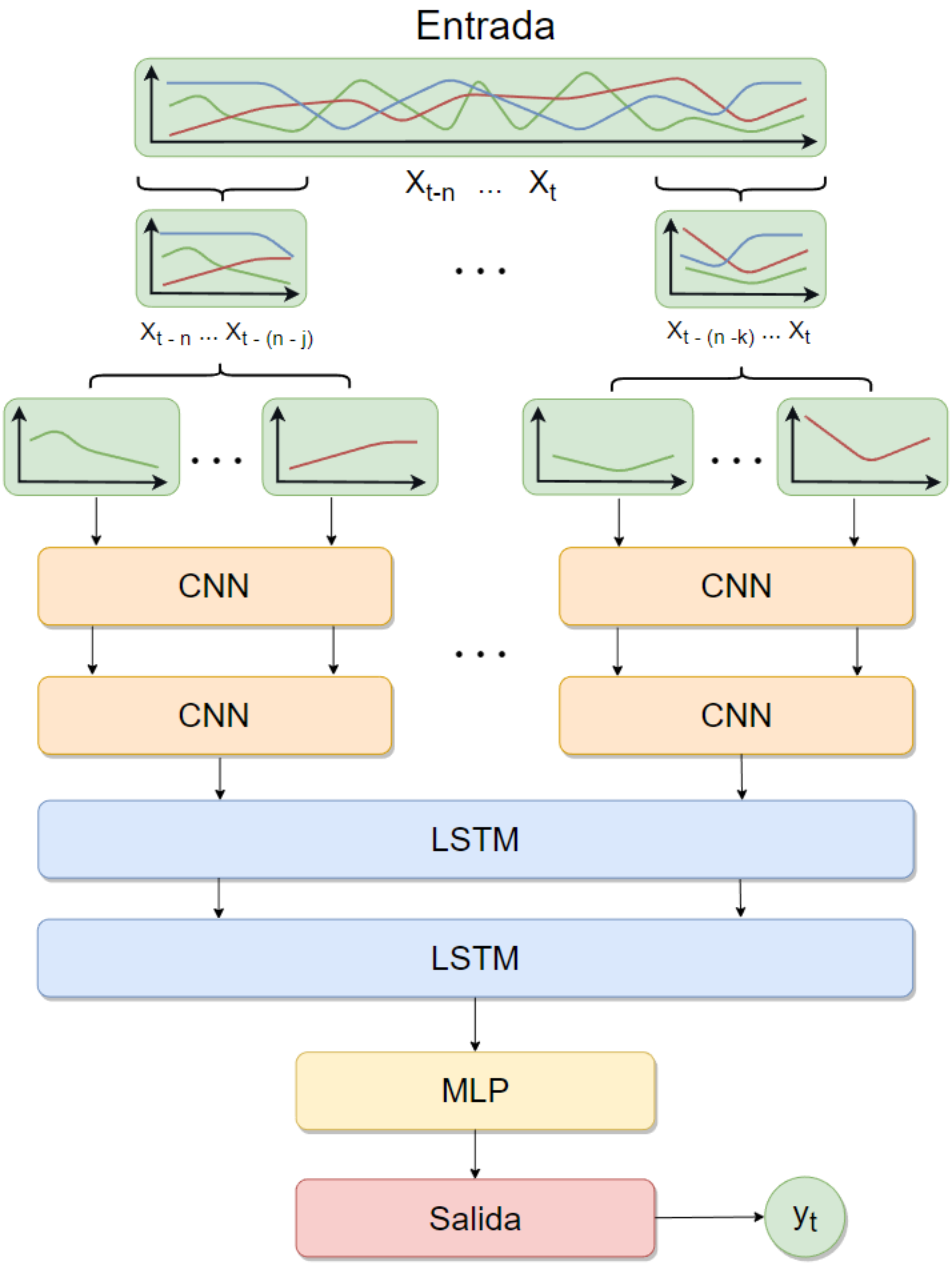


Figura 4.5: Arquitectura de red neuronal CNN + LSTM con dos capas CNN, dos capas LSTM y una capa MLP oculta.

Hay una gran cantidad de hiperparámetros a considerar para esta arquitectura. Además del número de neuronas y capas, será relevante escoger un adecuado tamaño para los intervalos procesados por las CNN, así como de la cantidad de pasos que serán alimentados a las capas LSTM.

4.4. Generación de características de análisis técnico

Además de las variables base incluidas en los datos fuente, se calcularon e incluyeron indicadores basados en fundamentos de análisis técnico. Este tipo de indicadores son usualmente utilizados por corredores de bolsa en el comercio de la bolsa de valores (o más recientemente, en el comercio de criptomonedas). Todos los indicadores son calculados a partir de los datos base, e incluyen características que capturan propiedades de tendencia, momentum, volumen y volatilidad. A continuación se listan la totalidad de los indicadores considerados, incluyendo en cada caso una correspondiente referencia de relevancia.

Tendencia

- Exponential Moving Average (EMA) [58]
- Moving Average Convergence Divergence (MACD) [64]
- Average Directional Movement Index (ADX) [51]
- Vortex Indicator (VI) [74]
- Trix (TRIX) [71]
- Mass Index (MI) [66]
- Commodity Channel Index (CCI) [54]
- Detrended Price Oscillator (DPO) [57]
- KST Oscillator (KST) [63]
- Ichimoku Kinkō Hyō (Ichimoku) [61]

Momentum

- Money Flow Index (MFI) [65]
- Relative Strength Index (RSI) [69]
- True strength index (TSI) [72]
- Ultimate Oscillator (UO) [73]
- Stochastic Oscillator (SR) [70]
- Williams %R (WR) [76]
- Awesome Oscillator (AO) [127]

Volumen

- Accumulation/Distribution Index (ADI) [50]
- On-Balance Volume (OBV) [68]
- Chaikin Money Flow (CMF) [55]
- Force Index (FI) [60]
- Ease of Movement (EoM, EMV) [59]
- Volume-price Trend (VPT) [75]
- Negative Volume Index (NVI) [67]

Volatilidad

- Average True Range (ATR) [52]

- Bollinger Bands (BB) [53]
- Keltner Channel (KC) [62]
- Donchian Channel (DC) [56]

Estos indicadores adicionan una cierta pre-interpretación a los datos base, al tiempo que también permiten incluir información más lejana en el tiempo de forma resumida. En la Figura 4.6 se muestran gráficamente las series temporales generadas por dos de estos indicadores, promedio móvil exponencial de las últimas 36hrs (EMA36) y bandas de Bollinger. Se propone utilizar estos indicadores como características de entrada adicionales, parte del análisis a desarrollar será determinar las combinaciones de estos que produzcan los mejores resultados.

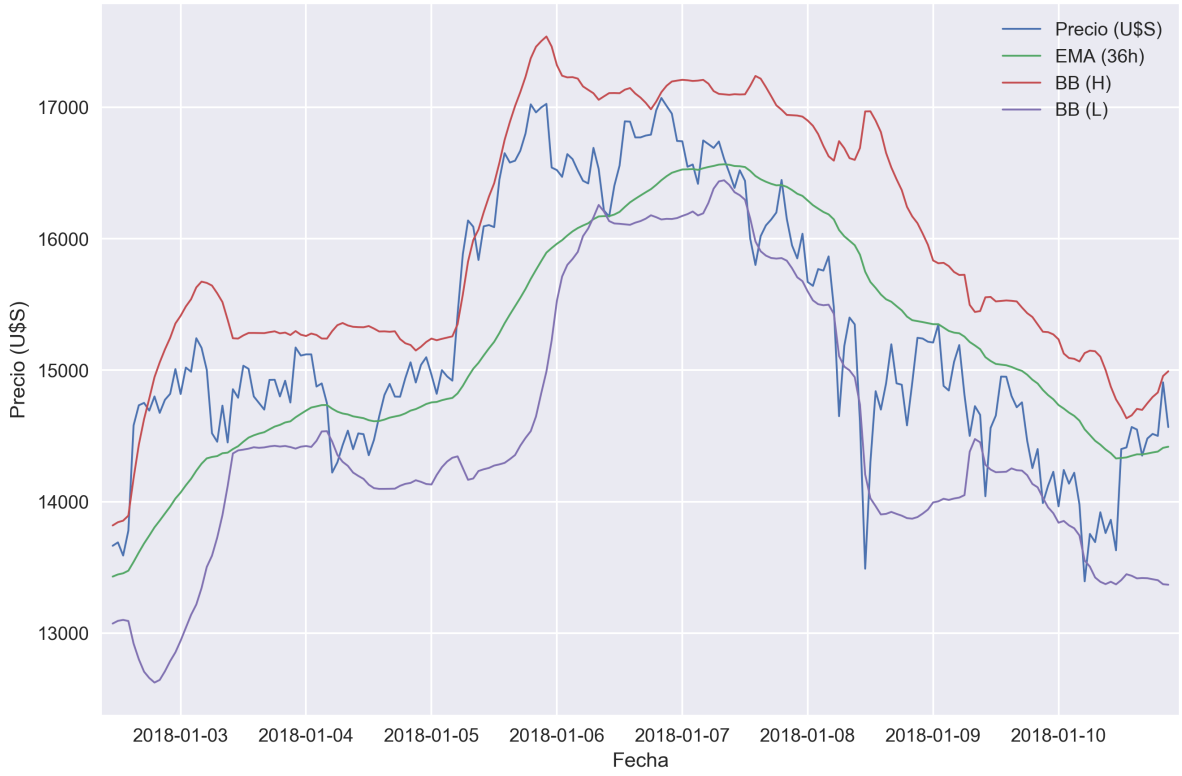


Figura 4.6: Indicadores de análisis técnico, BB (Bollinger Bands) y EMA36 (36hr. Exponential Moving Average)

4.5. Generación de datos de entrenamiento adicionales

Bajo la hipótesis de que las reglas que rigen los movimientos del precio de Bitcoin son también, en cierta medida, extrapolables a los movimientos presentes en el precio de otras criptomonedas (y viceversa), se analiza aumentar el tamaño del conjunto de datos incluyendo el histórico de precios de otras criptomonedas. Una opción podría ser incluir estos datos como nuevas variables de entrada, pero esto incrementaría la complejidad de los modelos significativamente. En su lugar, se opta por incluirlos como si fueran puntos adicionales dentro de la misma serie temporal. Como resultado se obtiene un conjunto de entrenamiento con la misma cantidad de variables, pero más volumen de datos.
Utilizando la misma fuente [10] y formato que para los datos base, se obtiene la información disponible para las criptomonedas con mayor volumen de transacciones: Ethereum (ETH) [33], Litecoin (LTC) [89], Cardano (ADA) [19], NEO (NEO) [100], TRON (TRX) [128], Ripple (XRP) [107], EOS (EOS) [32], Stellar (XLM) [117], IOTA (MIOTA) [77], Bitcoin Cash (BCH) [11], Binance Coin (BNB) [9], Ethereum Classic (ETC) [34], Ontology (ONT) [101] y Quantum (QTUM) [106].

Previo a incluir esta nueva información en el conjunto de datos base, se equipara el precio y el volumen para que en promedio sean de igual magnitud que los de Bitcoin
El precio de clausura equiparado c'_X para una moneda X se calcula como:

$$c'_X = c_X * \frac{\overline{c}_{BTC}}{\overline{c}_X}$$

Donde $\overline{c}_X$ es la media del precio de clausura de la criptomoneda X. El proceso es análogo para el resto de las variables.
Una vez equiparado, se adjuntan los nuevos datos al conjunto original (Figura 4.7).

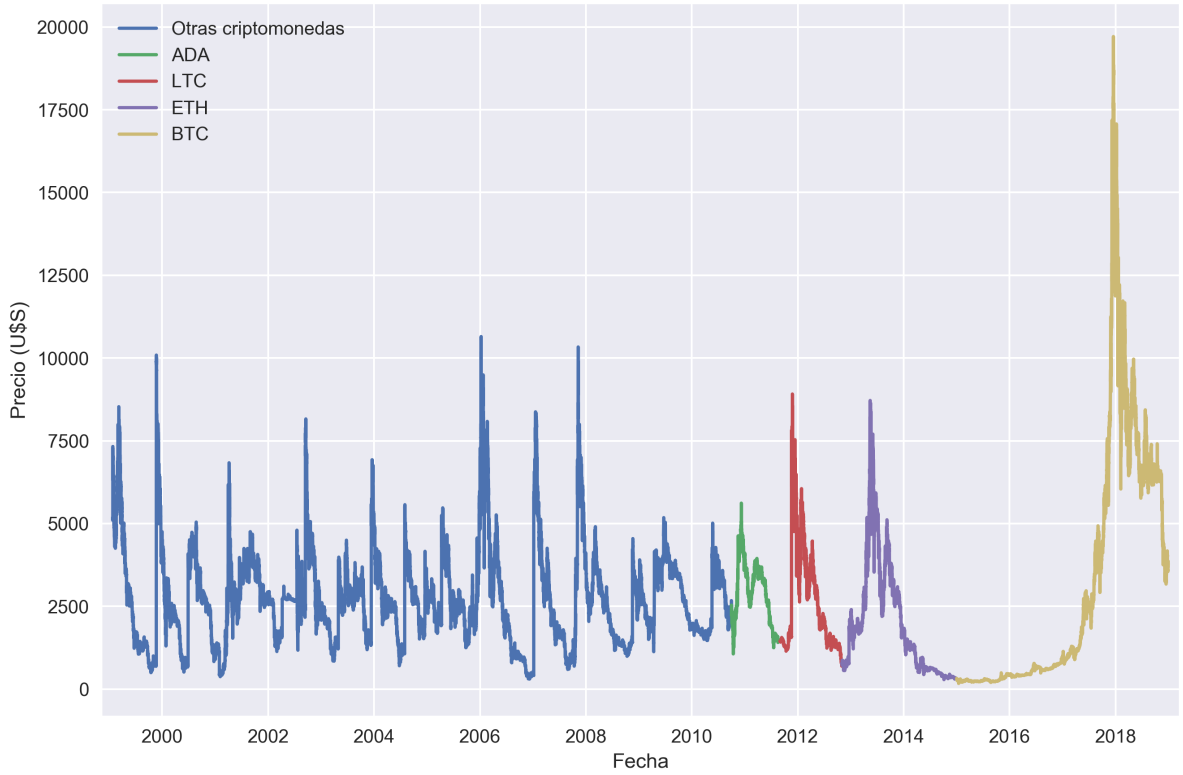


Figura 4.7: Conjunto de datos expandido mediante la incorporación del histórico de otras criptomonedas al conjunto de datos original.

Esta nueva información implica un incremento de la cantidad de datos de aproximadamente un 600 %, pasándose de un conjunto de 43.600 puntos a un total de aproximadamente 250.000 puntos de datos. Teniendo en cuenta además la incorporación de los indicadores de análisis técnico, la pretensión es que los modelos sean capaces de aprender la interdependencia de las variables que conforman las reglas de juego del mercado, independientemente de las especificidades de cada criptomoneda.

Capítulo 5

Pruebas realizadas

Utilizando los criterios de evaluación definidos en 3.3, se plantean los experimentos en tres fases (Cuadro 5.2). Todos los experimentos fueron realizados en un PC de escritorio cuyas especificaciones se detallan en el Cuadro 5.1.

Cuadro 5.1: Especificaciones del PC utilizado para la ejecución de las pruebas.

CPU	Intel Core i7-7700 @ 3.60 GHz
RAM	16.0 GB
GPU	NVIDIA GeForce GTX 760
OS	Windows 10 64-bit

5.1. Fase 1 - Características de entrada básicas

Se entrenan los modelos utilizando el conjunto de datos básico (correspondiente a 4 años del histórico del precio de Bitcoin), empleando como variables de entrada únicamente las características básicas allí presentes (precio de apertura, precio de clausura, precio máximo, precio mínimo y volumen). Se sondea el espacio completo de hiperparámetros de

cada modelo, analizando la incidencia de estos sobre los resultados. Se busca en cada caso hallar la combinación óptima de estos parámetros que maximicen el desempeño según las métricas definidas.

5.2. Fase 2 - Características de entrada extendidas con indicadores de análisis técnico

Se incorporan al conjunto de datos los indicadores de análisis técnico, disponiendo en total de 28 indicadores para ser empleados como variables de entrada adicionales. Partiendo de un espacio más acotado de hiperparámetros, el objetivo en este paso es evaluar la incidencia de la incorporación de estos indicadores y sus posibles combinaciones que, para cada modelo, maximicen la precisión de los resultados.

5.3. Fase 3 - Conjunto de datos de entrenamiento extendido con información de otras criptomonedas

Se parte de las mejores combinaciones de hiperparámetros e indicadores técnicos hallados en las fases anteriores, y se estudia el efecto sobre el rendimiento de los modelos de incluir en el conjunto de datos de entrenamiento los datos adicionales generados a partir de otras criptomonedas.

En el cuadro 5.2 se detallan las características de entrada y conjunto de datos de entrenamiento utilizados en cada fase de experimentación.

Cuadro 5.2: Datos y características utilizados en cada fase.

Fase	Datos	Características de entrada
1	43.600 puntos de datos del histórico de Bitcoin	Base (Open, Close, Low, High, Volume)
2	43.600 puntos de datos del histórico de Bitcoin	Base + Indicadores de análisis técnico
3	43.600 puntos de datos del histórico de Bitcoin + ∼218.000 puntos de datos de otras criptomonedas	Base + Indicadores de análisis técnico

En el cuadro 5.3 se listan los principales hiperparámetros considerados para su optimización por cada tipo de modelo implementado.

Cuadro 5.3: Hiperparámetros específicos a cada modelo.

Modelo	Hiperparámetros
Base	-
SVR	γ, ε, c, pasos previos.
LSTM, GRU	Número de capas ocultas, cantidad de neuronas por capa, pasos previos.
CNN	Tamaño del kernel, cantidad de filtros, número de capas ocultas, pasos previos.
CNN + LSTM	Tamaño del kernel, cantidad de filtros, número de capas ocultas CNN, número de capas ocultas LSTM, pasos previos.

De manera de que los resultados obtenidos como parte de esta investigación sean viables en un ambiente de producción, y de que la experimentación se pueda realizar en un tiempo razonable en el contexto del alcance de este proyecto, se limita el tiempo total de entrenamiento de cada modelo a un máximo de 60 minutos. En consecuencia, ciertos modelos cuyas combinaciones de hiperparámetros y características de entrada impliquen tiempos de entrenamiento por encima de este límite, no serán contemplados.

Capítulo 6

Resultados

6.1. Modelo base

La Figura 6.1 ilustra gráficamente el desfasaje de la predicción del modelo base, correspondiente a un retraso de 2 horas.

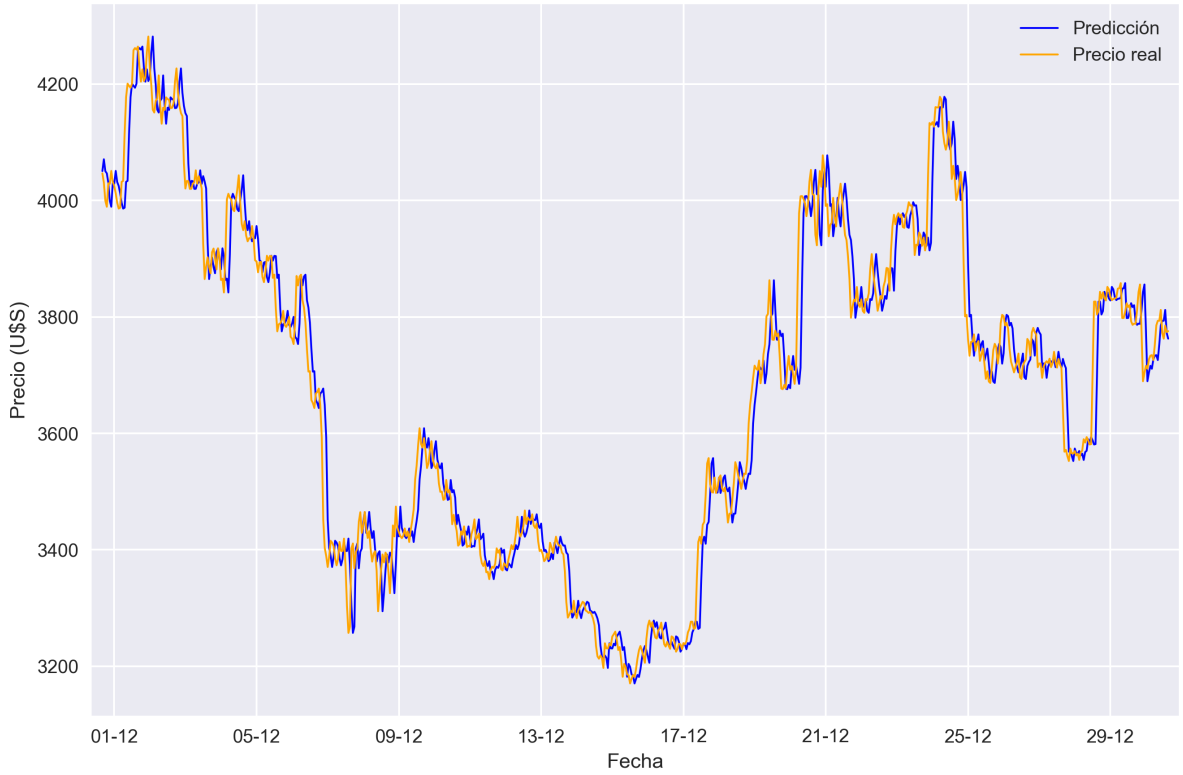


Figura 6.1: Predicciones del modelo base "ingenuo".

Estos resultados (Cuadro 6.1) sientan una base de partida sobre la cual evaluar las predicciones de los modelos de aprendizaje automático desarrollados.

Cuadro 6.1: Resultados del modelo base ingenuo.

RMSE	RMSEA	DA (%)
57.26	148.45	51.12

6.2. Fase 1 - Características de entrada básicas

Se evalúan las predicciones de los modelos implementados, según distintas combinaciones de sus hiperparámetros, utilizando los 4 años de datos disponibles correspondientes al precio de Bitcoin, y las características base presentes en los datos fuente (precio de apertura, precio de clausura, precio mínimo, precio máximo y volumen).

6.2.1. SVR

Kernel RBF (Radial basis function)

Los hiperparámetros a optimizar para este modelo son ε, γ y c. Además, se evalúa cómo se comporta el modelo al variar la cantidad de pasos previos de entrada.

El parámetro γ define qué tan lejos llega la influencia de un solo ejemplo de entrenamiento, con valores bajos implicando "lejos" y valores altos "cerca".

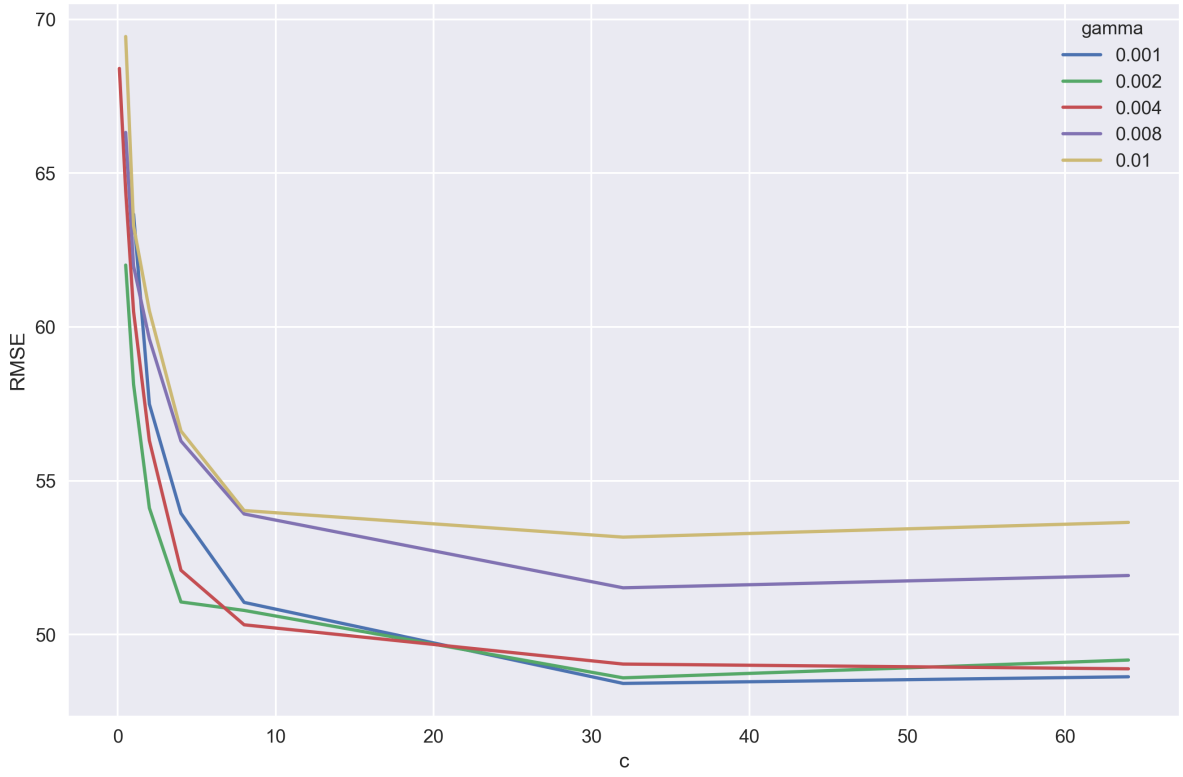


Figura 6.2: SVR (RBF, $\varepsilon = 0.004$).
Influencia de γ y c en la precisión de la predicción.

El parámetro c determina el balance entre la correcta predicción del conjunto de ejemplos de entrenamiento, contra la maximización del margen de la función de decisión. Para valores mayores de c, se aceptará un margen menor si la función de decisión es mejor para

estimar correctamente todos los puntos de entrenamiento. Un valor de c menor, fomentará un margen más grande, por lo tanto, una función de decisión más simple, a costa de la precisión durante el entrenamiento. El efecto que producen los distintos valores de c y γ sobre la precisión de la predicción es ilustrado en la Figura 6.2.

El parámetro ε especifica el rango dentro del cual no se asocia ninguna penalización a la función de pérdida por puntos predichos dentro de una distancia de a lo sumo ε del valor real. Este valor puede afectar el número de vectores de soporte utilizados para construir la función de regresión. Cuanto mayor el valor, menos vectores de soporte se seleccionan. Por otro lado, valores más grandes dan como resultado estimaciones más "planas". Por lo tanto, tanto los valores c como los valores de ε afectan la complejidad del modelo (pero de una manera diferente).

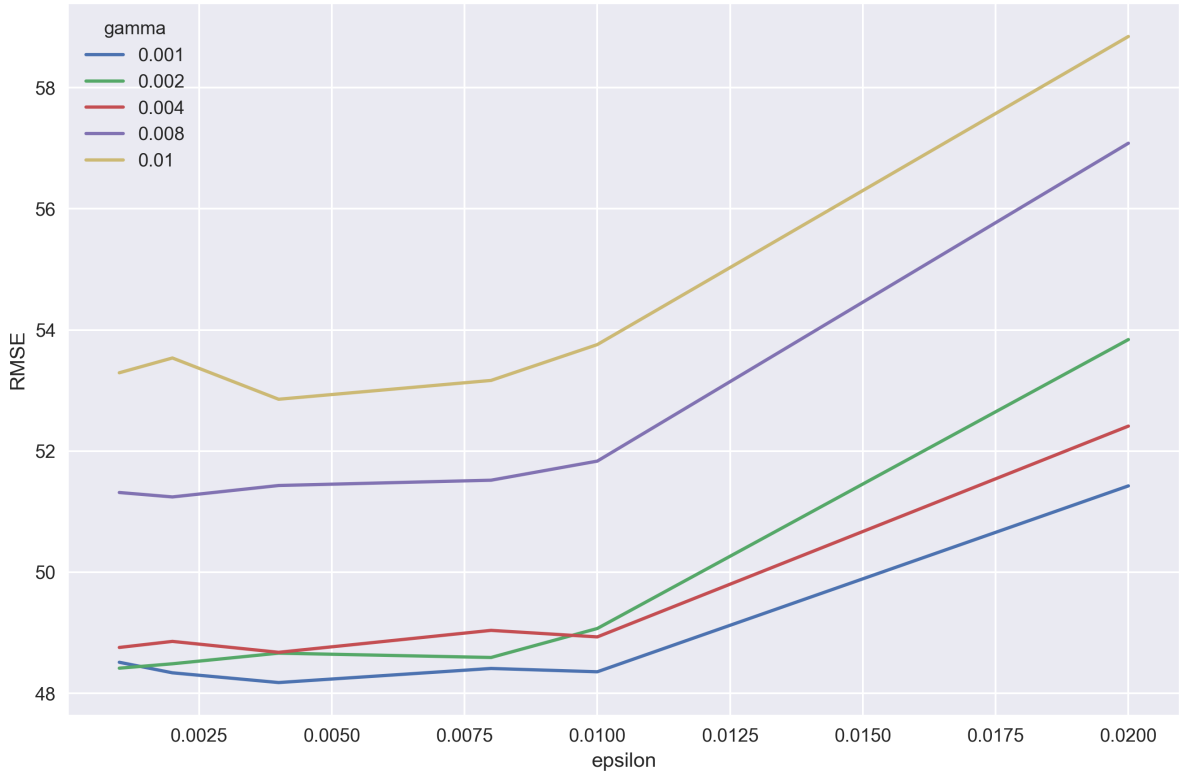


Figura 6.3: SVR (RBF, $c = 32$).
Influencia de γ y ε en la precisión de la predicción.

A partir de valores de ε mayores a 0.01, el RMSE aumenta rápidamente debido a la incapacidad del modelo de penalizar puntos cuyas predicciones están cada vez más lejos de

la realidad (Figura 6.3).

Es interesante notar que sólo con un c lo suficientemente grande, el modelo es lo suficientemente "complejo" para utilizar exitosamente el conjunto de datos de entrenamiento completo (Figura 6.4).

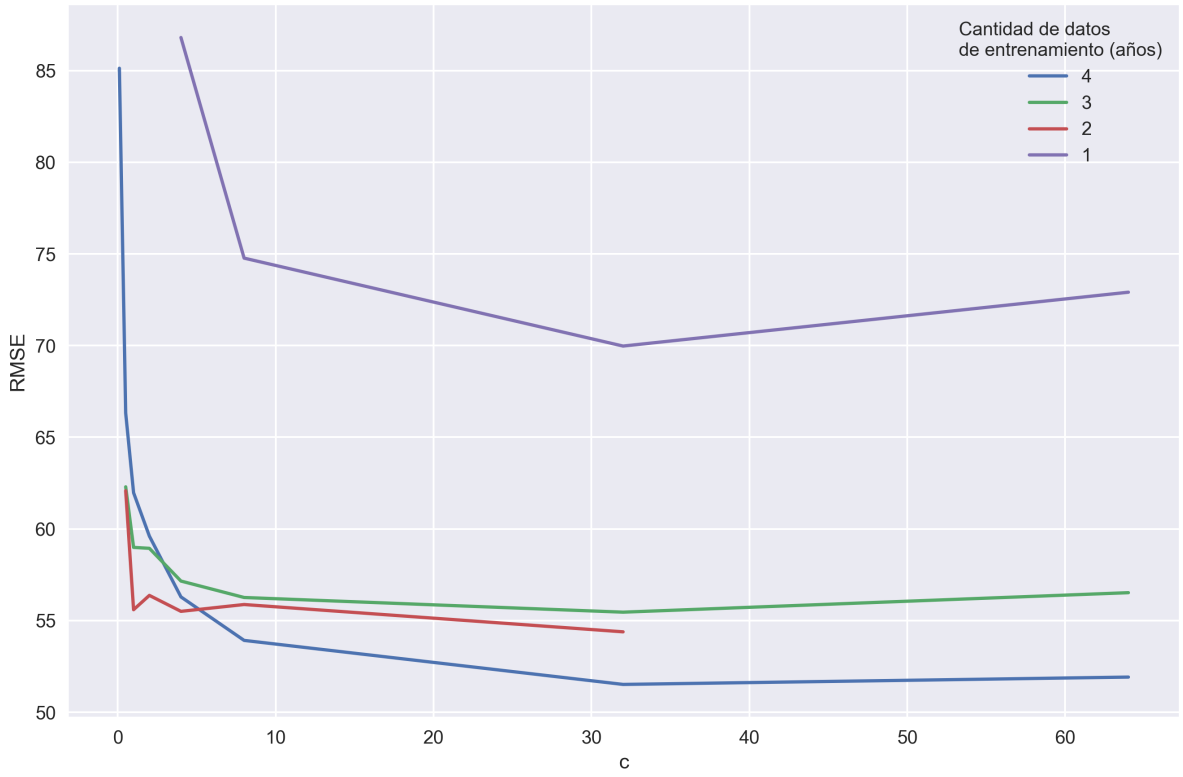


Figura 6.4: SVR (RBF, $\varepsilon = 0.004$, $\gamma = 0.004$).
Influencia de c y la cantidad de datos en la precisión de la predicción.

Para valores menores de c, el modelo es incapaz de generalizar satisfactoriamente utilizando ejemplos de entrenamiento lejanos (con valores de precio muy distintos al conjunto de prueba).

En cuanto a los pasos previos, independientemente de la configuración del resto de los hiperparámetros, para este tipo de modelo, los mejores resultados se alcanzan utilizando 4 pasos previos (es decir, 4 horas previas).

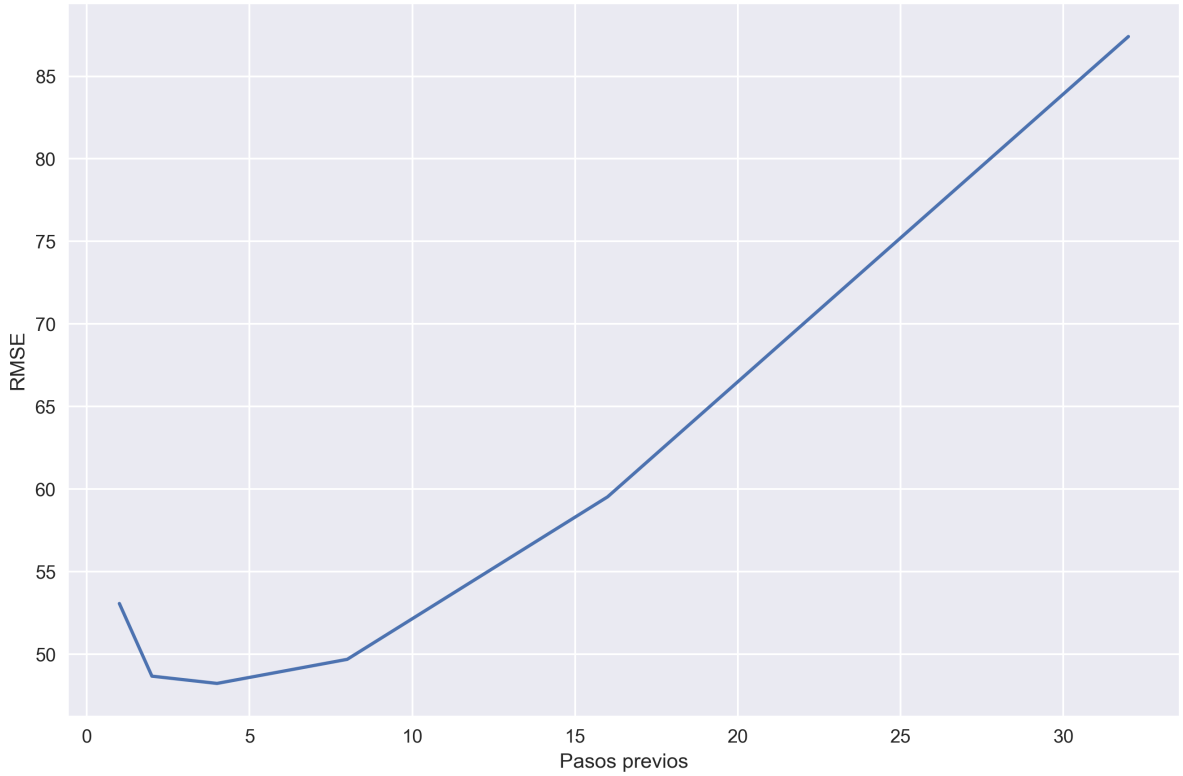


Figura 6.5: SVR (RBF, $\varepsilon = 0.004$, $\gamma = 0.004$).
Influencia de los pasos previos en la precisión de la predicción.

Pasos adicionales empeoran significativamente la predicción, implicando que, o bien no aportan información relevante, o el modelo no es capaz de utilizarlos satisfactoriamente (Figura 6.5).

Cuadro 6.2: Fase 1 - SVR. Resultados correspondientes a la mejor combinación de hiperparámetros hallada (ver Cuadro 6.3).

RMSE	RMSEA	DA (%)
48.05	113.01	79.14

Los resultados (Cuadro 6.2) correspondientes a la mejor combinación de hiperparámetros hallados (Cuadro 6.3) suponen una mejora considerable en relación al modelo base.

Cuadro 6.3: Fase 1 - SVR. Mejor combinación de hiperparámetros.

Pasos previos	4
ε	0.001
γ	0.002
c	32

Otros kernel

No se lograron buenos resultados utilizando otros kernel (lineal, polinómico y sigmoide). Aún con gran esfuerzo de optimización de los parámetros, los mejores modelos apenas mejoraban las predicciones del modelo base.

6.2.2. LSTM

Se busca establecer empíricamente el número de capas ocultas y de neuronas de cada capa que generen los mejores resultados para el conjunto de prueba.

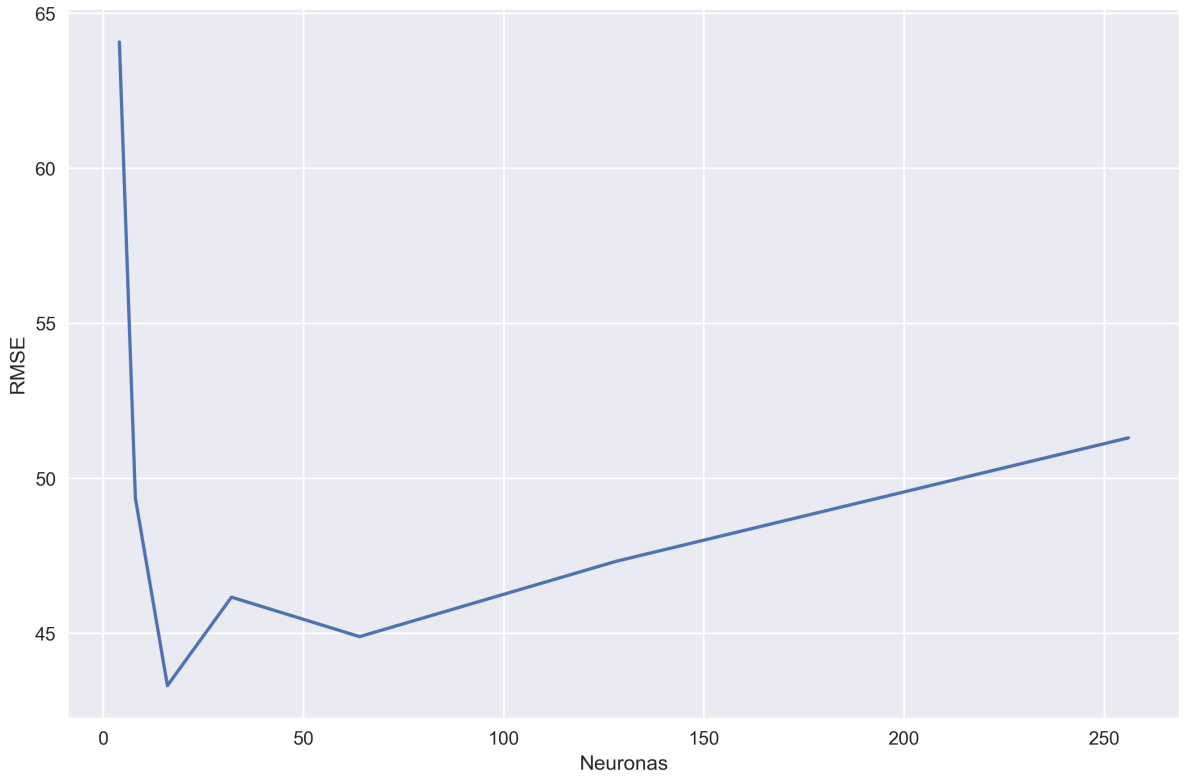


Figura 6.6: LSTM (Una capa LSTM y una capa MLP)
RMSE en función de la cantidad de neuronas de la capa LSTM.

Se observa (Figura 6.6) que empleando una única capa LSTM y una capa MLP, la mejor precisión se obtiene con una relativamente pequeña cantidad de neuronas (16).

A su vez, se destaca que agregar capas LSTM ocultas adicionales no mejora los resultados (Figura 6.7).

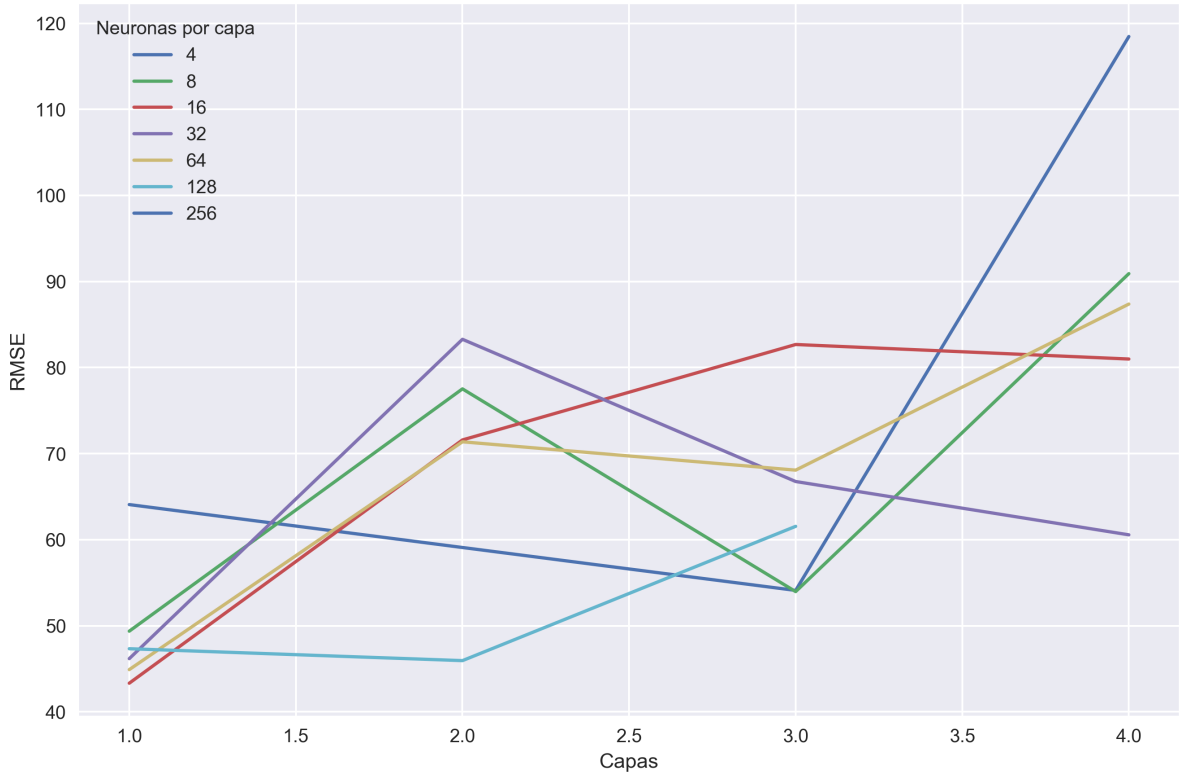


Figura 6.7: LSTM
RMSE en función de la cantidad de capas LSTM y sus respectivas neuronas.

Cabe notar que una única capa LSTM de n pasos contiene en su arquitectura n capas MLP, lo que implica por sí sola un alto grado de complejidad.

Es destacable que a diferencia de lo que ocurría con el modelo SVR, este modelo sí parece aprovechar la información provista en pasos previos adicionales (Figura 6.8). Esto es congruente con la arquitectura de este tipo de red neuronal, especialmente diseñado para utilizar datos secuenciales.

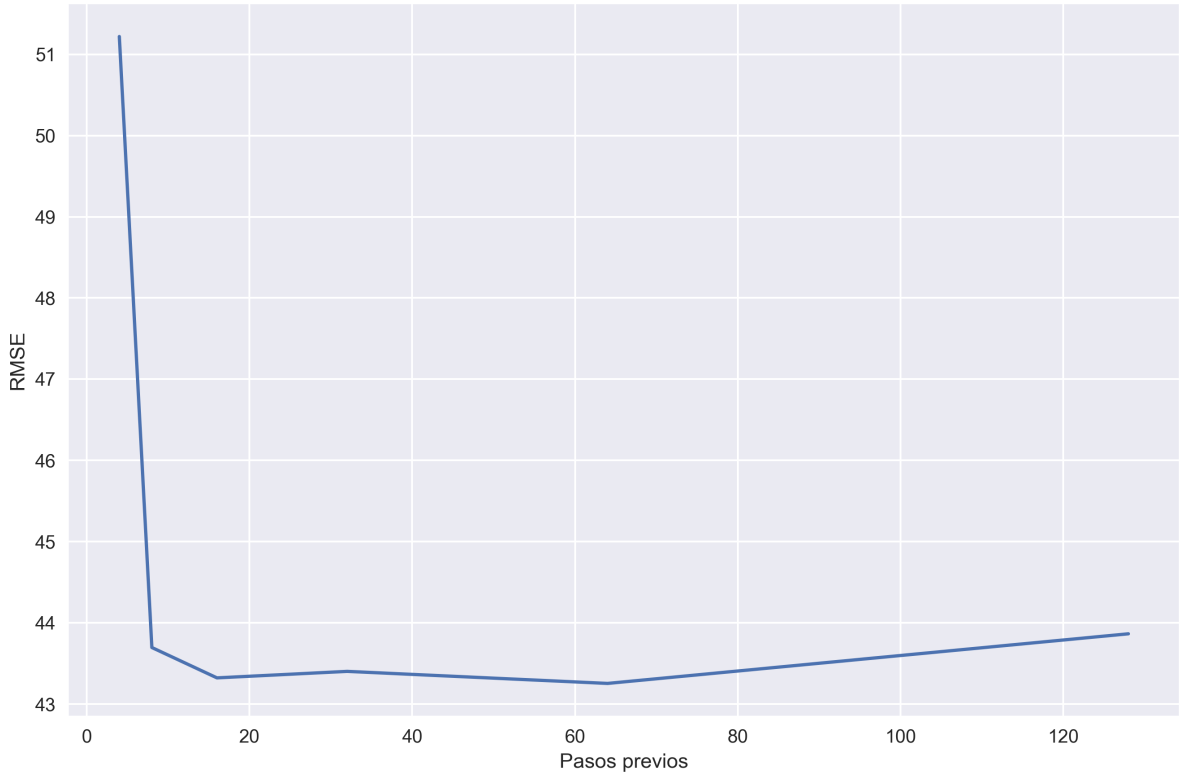


Figura 6.8: LSTM (Una capa LSTM de 16 neuronas y una capa MLP de 32 neuronas) RMSE en función de la cantidad de pasos previos de entrada.

Empleando la combinación de hiperparámetros detallados en el Cuadro 6.5 se obtienen los mejores resultados (Cuadro 6.4); suponiendo una interesante mejora en comparación con el modelo SVR.

Cuadro 6.4: Fase 1 - LSTM. Resultados correspondientes a la mejor combinación de hiperparámetros hallada (ver Cuadro 6.5).

RMSE	RMSEA	DA (%)
43.32	99.88	74.93

Cuadro 6.5: Fase 1 - LSTM. Mejor combinación de hiperparámetros.

Pasos previos	64
Capas LSTM (Neuronas)	1 (16)
Capas MLP (Neuronas)	1 (32)

6.2.3. GRU

El impacto de la cantidad de neuronas y de capas es muy similar al observado para la red LSTM, obteniendo el mejor desempeño utilizando una única capa GRU de 32 neuronas. Asimismo, este modelo es también capaz de utilizar con éxito una gran cantidad de pasos previos, alcanzando la mayor precisión empleando características de entrada correspondientes a 64 horas de datos base previos.

Utilizando los mejores hiperparámetros encontrados (Cuadro 6.7), los resultados para esta versión más simple de RNN suponen una importante mejora sobre el modelo SVR, pero no alcanzan los niveles de precisión obtenidos por el modelo LSTM (Cuadro 6.6).

Cuadro 6.6: Fase 1 - GRU. Resultados correspondientes a la mejor combinación de hiperparámetros hallada (ver Cuadro 6.7).

RMSE	RMSEA	DA (%)
44.65	102.81	73.82

Cuadro 6.7: Fase 1 - GRU. Mejor combinación de hiperparámetros.

Pasos previos	64
Capas GRU (Neuronas)	1 (32)
Capas MLP (Neuronas)	1 (32)

6.2.4. CNN

Se intenta determinar el tamaño óptimo del kernel de convolución y la cantidad de filtros de salida, así como el número de capas convolucionales.

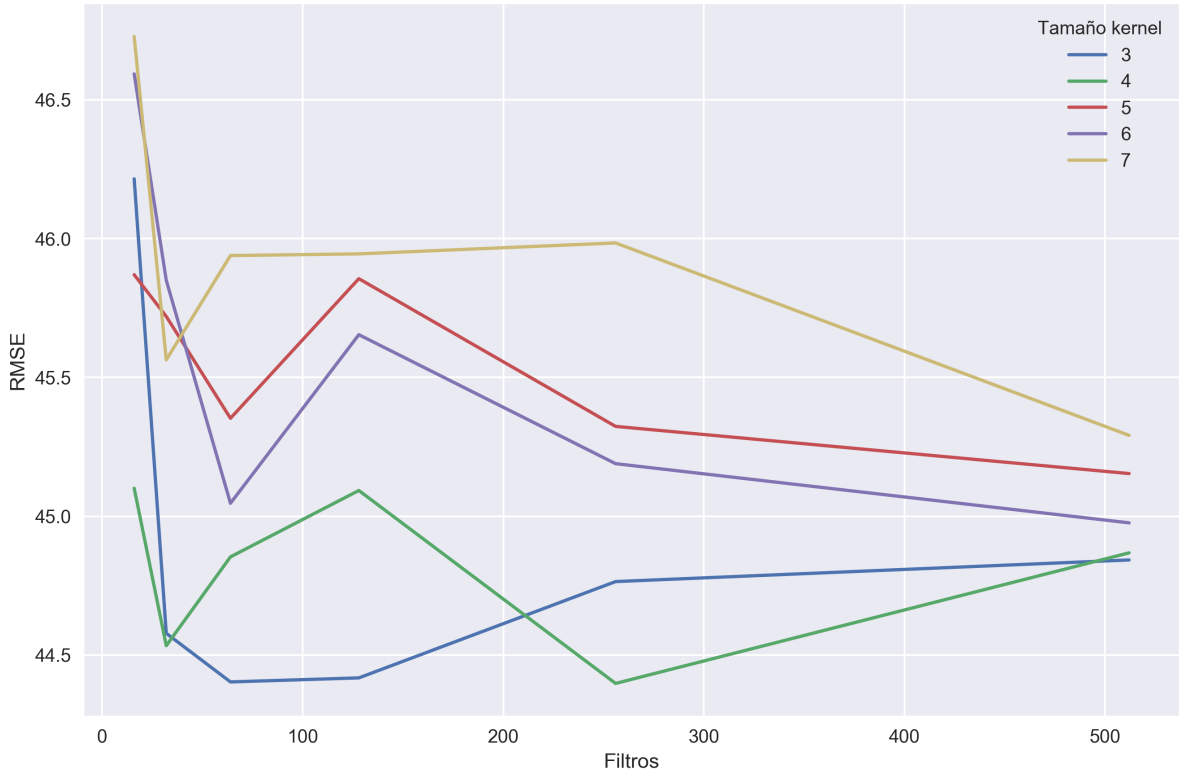


Figura 6.9: CNN (Una capa CNN, una capa MLP de 32 neuronas, 16 pasos previos) RMSE en función de la cantidad de filtros y el tamaño del kernel.

Se observa que se favorece un kernel relativamente pequeño con un alto número de filtros (Figura 6.9). Específicamente, se alcanza la mejor precisión con un tamaño de kernel de 4, y 256 filtros de salida. Una posible interpretación de estos valores es que la red intenta identificar 256 distintos patrones, a partir de intervalos de 4 pasos de largo de las 5 variables de entrada básicas.

Al igual que ocurre con las otras arquitecturas de red neuronal experimentadas hasta el momento, la precisión se degrada al aumentar la cantidad de capas ocultas, independientemente de la cantidad de filtros o el tamaño de kernel.

Empleando la mejor configuración de hiperparámetros hallados en este fase para este tipo de modelo (Cuadro 6.9), los resultados no llegan a ser tan precisos como los obtenidos por el modelo LSTM (Cuadro 6.8).

Cuadro 6.8: Fase 1 - CNN. Resultados correspondientes a la mejor combinación de hiperparámetros hallada (ver Cuadro 6.9).

RMSE	RMSEA	DA (%)
44.22	101.18	73.98

Cuadro 6.9: Fase 1 - CNN. Mejor combinación de hiperparámetros.

Pasos previos	32
Capas CNN (Filtros - Tamaño kernel)	1 (256 - 4)
Capas MLP (Neuronas)	1 (16)

6.2.5. CNN + LSTM

En este caso, el espacio de combinaciones de hiperparámetros es mucho mayor que en los modelos anteriores. Asimismo, la complejidad de la red implica tiempos de entrenamiento mucho mayores, por lo que resulta inviable un análisis profundo del impacto de cada hiperparámetro.

A pesar de estas limitaciones, se lograron obtener resultados a la par del modelo LSTM (Cuadro 6.10), incluyendo en al arquitectura de la red neuronal una única capa CNN, una capa LSTM y una MLP (Cuadro 6.11).

Cuadro 6.10: Fase 1 - CNN + LSTM. Resultados correspondientes a la mejor combinación de hiperparámetros hallada (ver Cuadro 6.11).

RMSE	RMSEA	DA (%)
43.44	98.48	75.04

Cuadro 6.11: Fase 1 - CNN + LSTM. Mejor combinación de hiperparámetros.

Pasos previos	64
Capas CNN (Filtros - Kernel)	1 (256 - 4)
Capas LSTM (Neuronas)	1 (16)
Capas MLP (Neuronas)	1 (32)

6.2.6. Resumen de resultados de la fase

En el Cuadro 6.12 se resumen los resultados obtenidos utilizando la mejor combinación de hiperparámetros hallada para cada modelo considerado.

Cuadro 6.12: Fase 1. Resultados obtenidos utilizando la mejor combinación de hiperparámetros de cada modelo.

Modelo	Hiperparámetros	RMSE	RMSEA	DA (%)
SVR	$\varepsilon = 0.001$, $\gamma = 0.002$, c = 32 Pasos previos = 4	48.05	113.01	70.14
LSTM	Capas LSTM (Neuronas) = 1 (16) Capas MLP (Neuronas) = 1 (32) Pasos previos = 64	**43.32**	99.88	74.93
GRU	Capas GRU (Neuronas) = 1 (32) Capas MLP (Neuronas) = 1 (32) Pasos previos = 64	44.65	102.81	73.82
CNN	Capas CNN (Filtros - Kernel) = 1 (256 - 4) Capas MLP (Neuronas) = 1 (16) Pasos previos = 32	44.22	101.18	73.98
CNN + LSTM	Capas CNN (Filtros - Kernel) = 1 (256 - 4) Capas LSTM (Neuronas) = 1 (16) Capas MLP (Neuronas) = 1 (32) Pasos previos = 64	43.44	**98.48**	**75.04**

Se destaca en esta fase la superioridad de la precisión obtenida por los modelos de redes neuronales, en particular el mejor RMSE es alcanzado por el modelo LSTM, mientras que la mayor precisión alrededor de las anomalías (RMSEA) y la mejor precisión de la dirección del movimiento del precio (DA) fue obtenida por el modelo híbrido CNN + LSTM.

6.3. Fase 2 - Características de entrada extendidas con indicadores de análisis técnico

Se incorporan al conjunto de datos los indicadores de análisis técnico. Se dispone en total de 28 indicadores para ser empleados como variables de entrada adicionales. Un número tan alto de parámetros implicaría modelos demasiado complejos, por lo que parte del objetivo de la experimentación en esta fase es encontrar las mejores combinaciones de un número reducido de indicadores que maximice la precisión de los resultados. Se parte de la base de que estas combinaciones pueden diferir para cada tipo de modelo.

6.3.1. SVR

En primer lugar, se experimenta utilizando cada indicador técnico por separado, junto al resto de las variables de entrada básicas. Como resultado se obtiene una lista de los indicadores que, por si solos, mejoran el resultado de la predicción, estos son: DCH, KCC, UO, TSI, MFI, RSI, AROON, KST, EM y CMF. Posteriormente, se prueban sus posibles combinaciones, en busca de la que produzca el mejor desempeño (Figura 6.10).

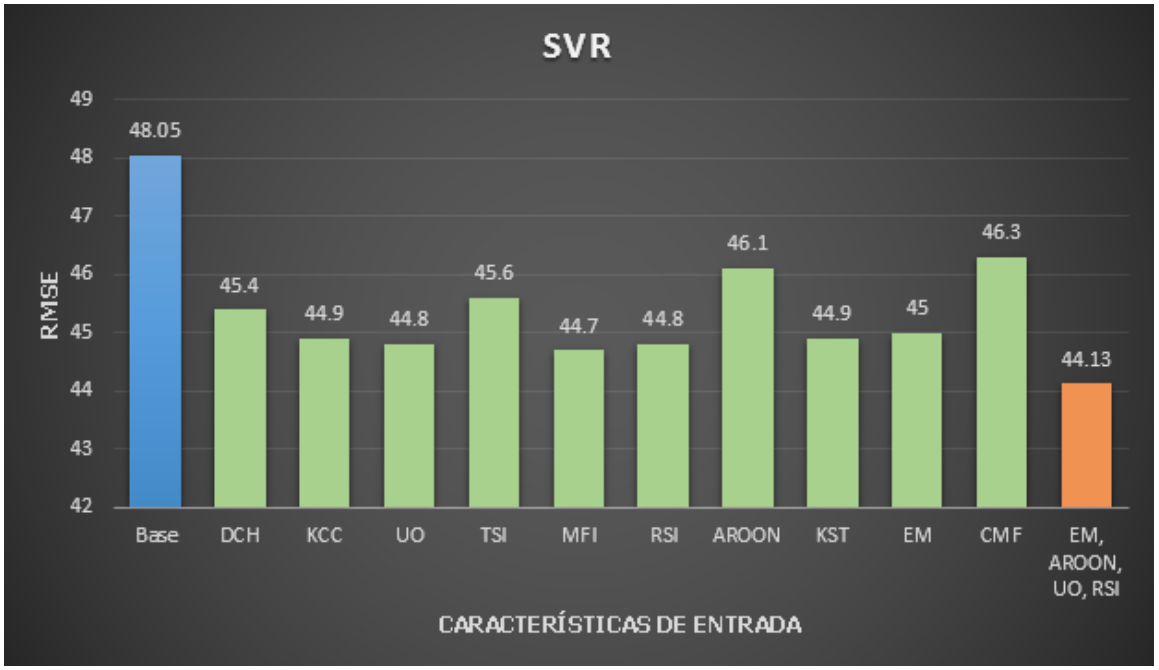


Figura 6.10: SVR - Fase 2. RMSE en función de las características de entrada.

La mejor precisión se alcanza utilizando una combinación de un pequeño subconjunto de indicadores: EM, AROON, UO y RSI, en adición a las características base, mejorando por un gran margen al desempeño de la fase anterior (Cuadro 6.13).

Cuadro 6.13: Fase 2 - SVR. Resultados correspondientes a la mejor combinación de hiperparámetros y características de entrada hallada (ver Cuadro 6.14).

RMSE	RMSEA	DA (%)
44.13	102.55	73.70

Una observación interesante es que en este caso el modelo se desempeña mejor con una mayor cantidad de pasos previos que en la primer fase (8 contra 4); el resto de los hiperparámetros se mantienen incambiados (Cuadro 6.14).

Cuadro 6.14: Fase 2 - SVR. Mejor combinación de hiperparámetros y características de entrada.

Pasos previos	8
ε	0.001
γ	0.002
c	32
Características de entrada	EM, AROON, UO, RSI, Base

6.3.2. LSTM

Los indicadores que contribuyeron en mayor medida a la calidad de la predicción conforman un subconjunto bastante diferente al obtenido para los modelos anteriores (SVR): BB, EMA, MACD, TSI, DC, RSI, ICH y CMF. Nuevamente, el mejor resultado fue alcanzado empleando una pequeña combinación de características, en conjunto con las base: BB, EMA y MACD (Figura 6.11).

Además, se destaca que fue necesario duplicar la cantidad de neuronas respecto a la fase anterior para que el modelo fuera capaz de explotar satisfactoriamente las nuevas características (Cuadro 6.16).

Figura 6.11: LSTM - Fase 2
RMSE en función de las características de entrada.

Los resultados arrojan una mejora con respecto a la primer fase (Cuadro 6.15).

Cuadro 6.15: Fase 2 - LSTM. Resultados correspondientes a la mejor combinación de hiperparámetros y características de entrada hallada (ver Cuadro 6.16).

RMSE	RMSEA	DA (%)
42.92	99.86	75.05

Nuevamente, al igual que lo observado en la fase anterior, un aumento en el número de capas ocultas no se traduce en una mejora de los resultados.

Cuadro 6.16: Fase 2 - LSTM. Mejor combinación de hiperparámetros y características de entrada.

Pasos previos	64
Capas LSTM (Neuronas)	1 (32)
Capas MLP (Neuronas)	1 (32)
Características de entrada	BB, EMA, MACD, Base

6.3.3. GRU

Se obtiene una significante mejora en comparación con la fase anterior mediante la inclusión de los indicadores detallados en la Figura 6.12.

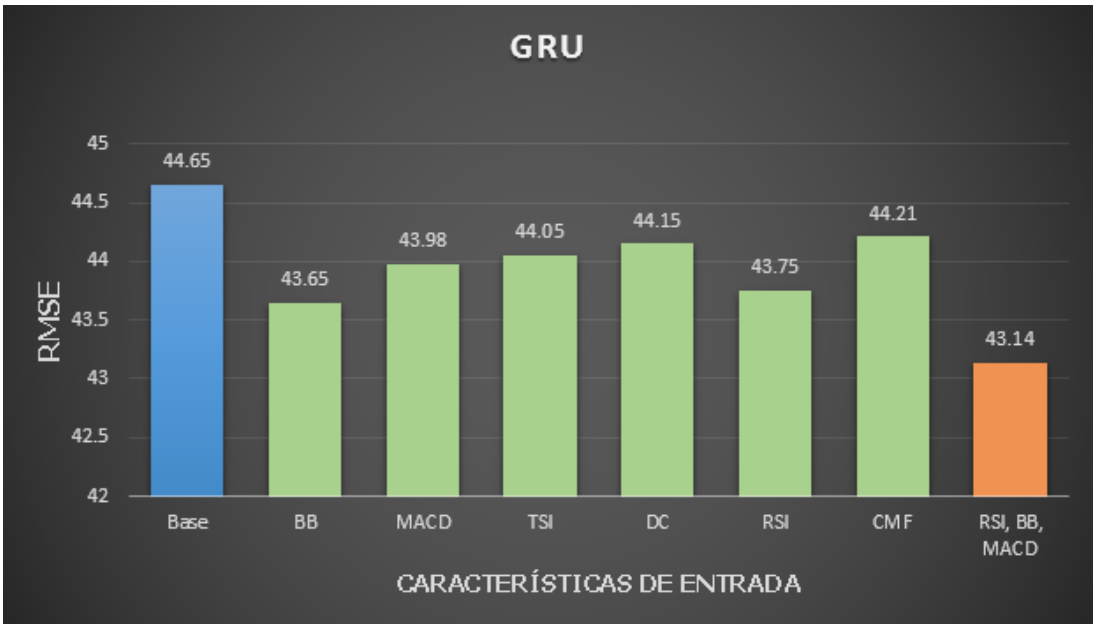


Figura 6.12: GRU - Fase 2
RMSE en función de las características de entrada.

Al igual que en el modelo LSTM, se debió incrementar la cantidad de neuronas para acompañar el incremento de la cantidad de características de entrada. En este caso además, se registró una mejor precisión al incorporar capas GRU ocultas adicionales (Figura 6.13).

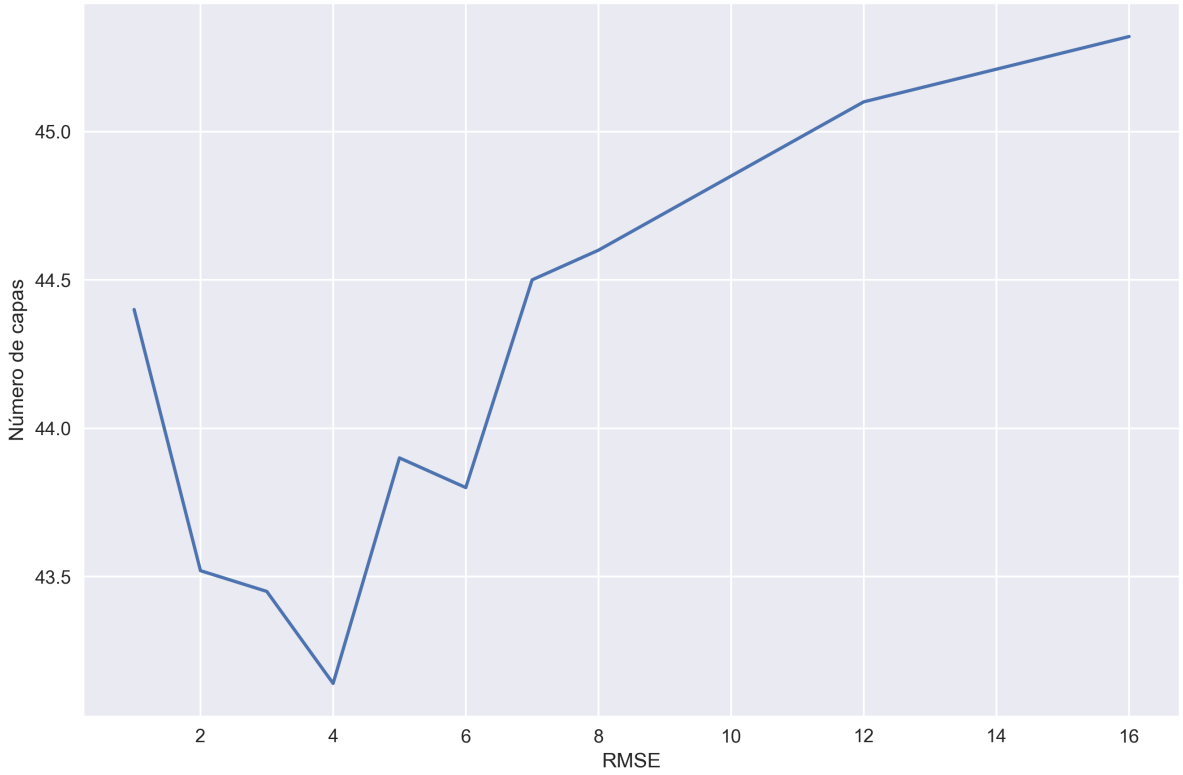


Figura 6.13: GRU - Fase 2
RMSE en función de la cantidad de la cantidad de capas GRU.

La mejora es significativa en comparación con la fase anterior, alcanzando para la mejor combinación de hiperparámetros y características de entrada (Cuadro 6.18) niveles de precisión similares al modelo LSTM (Cuadro 6.17).

Cuadro 6.17: Fase 2 - GRU. Resultados correspondientes a la mejor combinación de hiperparámetros y características de entrada hallada (ver Cuadro 6.18).

RMSE	RMSEA	DA (%)
42.91	100.05	75.05

Cuadro 6.18: Fase 2 - GRU. Mejor combinación de hiperparámetros y características de entrada.

Pasos previos	64
Capas GRU (Neuronas)	4 (64, 64, 32, 32)
Capas MLP (Neuronas)	1 (32)
Características de entrada	RSI, BB, MACD, Base

6.3.4. CNN

Sólo un pequeño conjunto de características de análisis técnico generó una mejora en el desempeño del modelo (Figura 6.14).

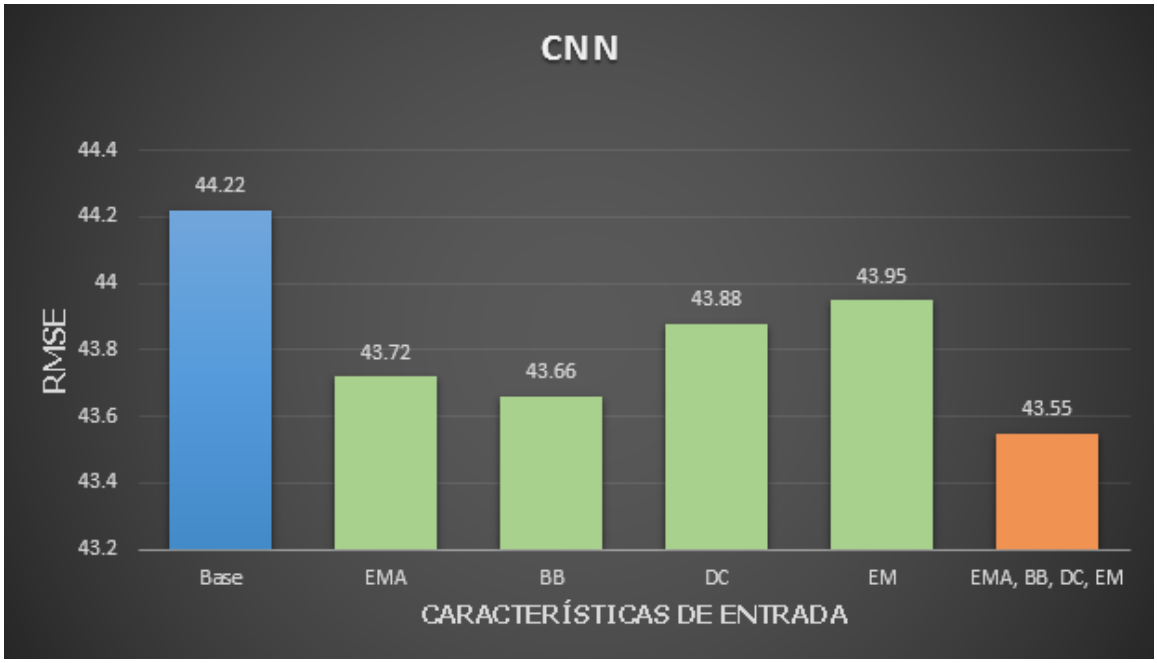


Figura 6.14: CNN - Fase 2
RMSE en función de las características de entrada.

Los hiperparámetros óptimos se mantuvieron constantes respecto a la fase anterior (Cuadro 6.20), con una mejora en la precisión de los resultados no tan significativa como la observada para los modelos de RNN; LSTM y GRU (Cuadro 6.19).

Cuadro 6.19: Fase 2 - CNN. Resultados correspondientes a la mejor combinación de hiperparámetros y características de entrada hallada (ver Cuadro 6.20).

RMSE	RMSEA	DA (%)
43.55	100.73	74.92

Cuadro 6.20: CNN - Fase 2. Mejor combinación de hiperparámetros y características de entrada.

Pasos previos	32
Capas CNN (Filtros - Tamaño kernel)	1 (256 - 4)
Capas MLP (Neuronas)	1 (16)
Características de entrada	EMA, BB, DC, EM, Base

6.3.5. CNN + LSTM

Incrementando la cantidad de filtros de salida de la capa convolucional, y las neuronas de la capa LSTM (ver arquitectura detallada de la red neuronal en el Cuadro 6.22), se observa en este caso una substancial mejora al incluir las nuevas características (Figura 6.15).

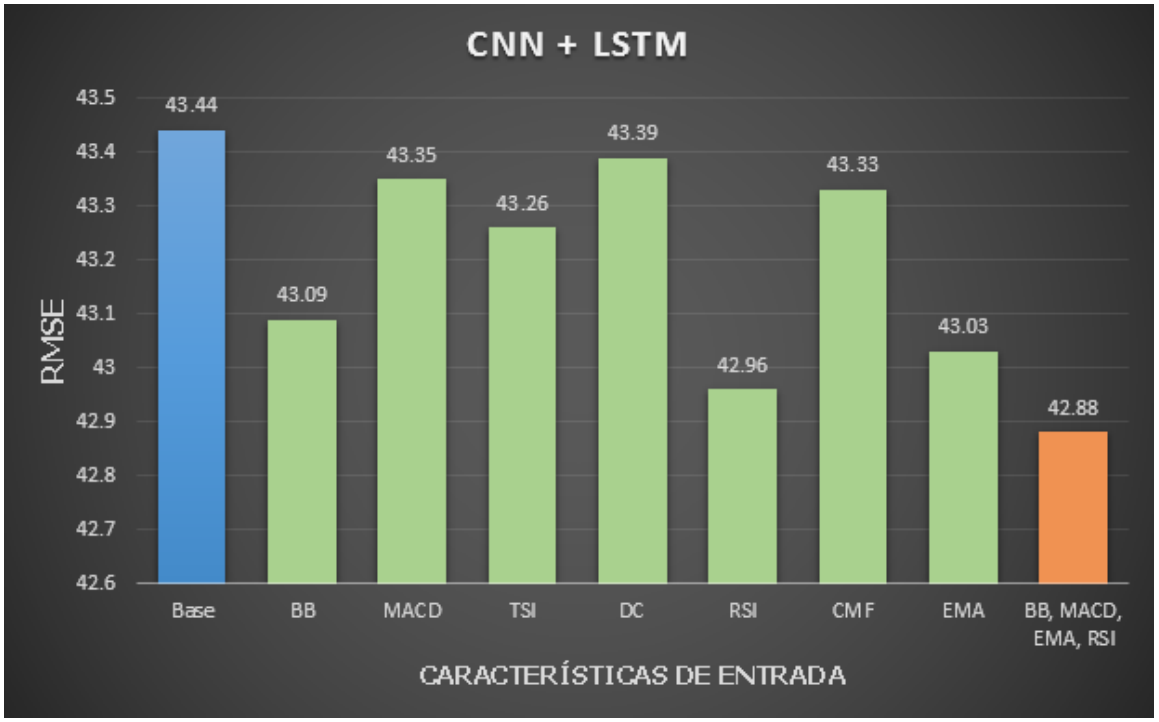


Figura 6.15: CNN + LSTM - Fase 2
RMSE en función de las características de entrada.

Los resultados arrojan la mejor precisión de todos los modelos evaluados en esta fase (Cuadro 6.21).

Cuadro 6.21: Fase 2 - CNN + LSTM. Resultados correspondientes a la mejor combinación de hiperparámetros y características de entrada hallada (ver Cuadro 6.22).

RMSE	RMSEA	DA (%)
42.88	98.32	75.44

Cuadro 6.22: CNN + LSTM - Fase 2. Mejor combinación de hiperparámetros y características de entrada.

Pasos previos	64
Capas CNN (Filtros - Kernel)	1 (512 - 4)
Capas LSTM (Neuronas)	1 (32)
Capas MLP (Neuronas)	1 (32)
Características de entrada	BB, MACD, EMA, RSI, Base

6.3.6. Resumen de resultados de la fase

En el Cuadro 6.23 se resumen los resultados obtenidos utilizando la mejor combinación de hiperparámetros y características de entrada hallada para cada modelo considerado.

Cuadro 6.23: Fase 2. Resultados obtenidos utilizando la mejor combinación de hiperparámetros y características de entrada de cada modelo. (N = Número de neuronas de la capa, F = Número de filtros de la capa convolucional, TK = Tamaño del kernel de la capa convolucional.)

Modelo	Hiperparámetros	Características de entrada	RMSE	RMSEA	DA (%)
SVR	$\varepsilon = 0.001$, $\gamma = 0.002$, c = 32 Pasos previos = 8	EM, AROON, UO, RSI, Base	44.13	102.55	73.70
LSTM	Capas LSTM (N) = 1 (32) Capas MLP (N) = 1 (32) Pasos previos = 64	BB, EMA, MACD, Base	42.92	99.86	75.05
GRU	Capas GRU (N) = 4 (64, 64, 32, 32) Capas MLP (N) = 1 (32) Pasos previos = 64	RSI, BB, MACD, Base	42.91	100.05	75.05
CNN	Capas CNN (F - TK)= 1 (256 - 4) Capas MLP (N) = 1 (16) Pasos previos = 32	EMA, BB, DC, EM, Base	43.55	100.74	74.92
CNN + LSTM	Capas CNN (F - TK) = 1 (512 - 4) Capas LSTM (N) = 1 (32) Capas MLP (N) = 1 (32) Pasos previos = 64	BB, MACD, EMA, RSI, Base	**42.88**	**98.32**	**75.44**

Se observa una mejora con respecto a la fase anterior para todos los modelos considerados, destacándose en particular la precisión superior obtenida por el modelo híbrido CNN + LSTM.

6.4. Fase 3 - Conjunto de datos de entrenamiento extendido con información de otras criptomonedas

Se estudia el efecto sobre el desempeño de los modelos de incluir en el conjunto de datos de entrenamiento los datos adicionales generados a partir de otras criptomonedas.

6.4.1. SVR

Independientemente de la configuración de los hiperparámetros, se obtiene una insignificante mejora respecto a la fase interior (Cuadro 6.24).

Cuadro 6.24: Fase 3 - SVR. Resultados correspondientes a la mejor combinación de hiperparámetros y características de entrada utilizando el conjunto de datos de entrenamiento extendido (ver Cuadro 6.14).

RMSE	RMSEA	DA (%)
44.09	102.13	73.73

La información adicional no parece tener una incidencia significativa para este modelo.

6.4.2. LSTM

Con las combinaciones de hiperparámetros utilizados en las fases anteriores, los resultados no muestran una mejora significativa.

Sin embargo, incorporando capas LSTM adicionales, e incrementando la cantidad de neuronas de las mismas (Cuadro 6.26), la red neuronal es capaz de sacar provecho del conjunto de datos de entrenamiento extendido, obteniendo una significativa mejora respecto a las fases anteriores (Cuadro 6.25).

Cuadro 6.25: Fase 3 - LSTM. Resultados correspondientes a la mejor combinación de hiperparámetros y características de entrada utilizando el conjunto de datos de entrenamiento extendido (ver Cuadro 6.26).

RMSE	RMSEA	DA (%)
42.55	98.88	75.85

Cuadro 6.26: LSTM - Fase 3. Mejor combinación de hiperparámetros y características de entrada utilizando el conjunto de datos de entrenamiento extendido.

Pasos previos	64
Capas LSTM (Neuronas)	3 (128, 128, 128)
Capas MLP (Neuronas)	1 (32)
Características de entrada	BB, EMA, MACD, Base

6.4.3. GRU

Se observan resultados análogos a los obtenidos para el modelo LSTM, siendo necesario aumentar la complejidad de la red para mejorar la precisión (Cuadro 6.28), obteniendo una mejora en la precisión con respecto a la fase anterior (Cuadro 6.27).

Cuadro 6.27: Fase 3 - GRU. Resultados correspondientes a la mejor combinación de hiperparámetros y características de entrada utilizando el conjunto de datos de entrenamiento extendido (ver Cuadro 6.28).

RMSE	RMSEA	DA (%)
42.69	100.05	75.12

Cuadro 6.28: GRU - Fase 3. Mejor combinación de hiperparámetros y características de entrada utilizando el conjunto de datos de entrenamiento extendido.

Pasos previos	64
Capas GRU (Neuronas)	4 (128, 128, 128, 128)
Capas MLP (Neuronas)	1 (32)
Características de entrada	RSI, BB, MACD, Base

6.4.4. CNN

Los hiperparámetros se mantienen incambiados con respecto a la fase anterior (Cuadro 6.30), nuevamente registrándose una mejora de la precisión (Cuadro 6.29).

Cuadro 6.29: Fase 3 - CNN. Resultados correspondientes a la mejor combinación de hiperparámetros y características de entrada utilizando el conjunto de datos de entrenamiento extendido (ver Cuadro 6.30).

RMSE	**RMSEA**	**DA (%)**
42.89	98.76	75.30

Cuadro 6.30: CNN - Fase 3. Mejor combinación de hiperparámetros y características de entrada utilizando el conjunto de datos de entrenamiento extendido.

Pasos previos	32
Capas CNN (Filtros - Tamaño kernel)	1 (256 - 4)
Capas MLP (Neuronas)	1 (16)
Características de entrada	EMA, BB, DC, EM, Base

6.4.5. CNN + LSTM

Una vez incorporados los nuevos datos de entrenamiento, y ajustados los hiperparámetros del modelo de acuerdo a lo especificado en el Cuadro 6.32, se obtienen los mejores resultados de todo el conjunto de experimentos (Cuadro 6.31).

Cuadro 6.31: Fase 3 - CNN. Resultados correspondientes a la mejor combinación de hiperparámetros y características de entrada utilizando el conjunto de datos de entrenamiento extendido (ver Cuadro 6.32).

RMSE	RMSEA	DA (%)
42.11	97.64	77.13

Cuadro 6.32: CNN + LSTM - Fase 3. Mejor combinación de hiperparámetros y características de entrada utilizando el conjunto de datos de entrenamiento extendido.

Pasos previos	128
Capas CNN (Filtros - Kernel)	3 (512 - 4, 256 - 4, 256 - 4)
Capas LSTM (Neuronas)	3 (128, 128, 64)
Capas MLP (Neuronas)	1 (32)
Características de entrada	BB, MACD, EMA, RSI, Base

6.4.6. Resumen de resultados de la fase

En todos los modelos de redes neuronales se observó una mejora en los resultados al incorporar los nuevos datos en el conjunto de entrenamiento. La mejor precisión es registrada por el modelo híbrido de red neuronal CNN y LSTM. En el Cuadro 6.33 se resume los resultados de la fase, utilizando el conjunto de entrenamiento extendido y la mejor configuración de hiperparámetros y características correspondientes a cada modelo.

Cuadro 6.33: Fase 3. Resultados obtenidos utilizando el conjunto de entrenamiento extendido y la mejor combinación de hiperparámetros y características de entrada de cada modelo. (N = Número de neuronas de la capa, F = Número de filtros de la capa convolucional, TK = Tamaño del kernel de la capa convolucional.)

Modelo	Hiperparámetros	Características de entrada	RMSE	RMSEA	DA (%)
SVR	$\varepsilon = 0.001$, $\gamma = 0.002$, c = 32 Pasos previos = 8	EM, AROON, UO, RSI, Base	44.09	102.13	73.73
LSTM	Capas LSTM (N) = 3 (128, 128, 128) Capas MLP (N) = 1 (32) Pasos previos = 64	BB, EMA, MACD, Base	42.55	98.88	75.85
GRU	Capas GRU (N) = 4 (128, 128, 128, 128) Capas MLP (N) = 1 (32) Pasos previos = 64	RSI, BB, MACD, Base	42.69	100.05	75.12
CNN	Capas CNN (F-TK)= 1 (256-4) Capas MLP (N) = 1 (16) Pasos previos = 32	EMA, BB, DC, EM, Base	42.89	98.76	75.30
CNN + LSTM	Capas CNN (F-TK) = 3 (512-4, 256-4, 256-4) Capas LSTM (N) = 3 (128, 128, 64) Capas MLP (N) = 1 (32) Pasos previos = 128	BB, MACD, EMA, RSI, Base	**42.11**	**97.64**	**77.13**

Se destaca el hecho de que los datos de entrenamiento complementarios permitieran explotar las capacidades de arquitecturas de redes neuronales más profundas, de manera que la inclusión de capas y de neuronas adicionales se tradujeran en un mejor desempeño para la mayoría de los modelos (LSTM, GRU y CNN + LSTM).

6.5. Resumen de resultados

En la primer fase de experimentación se confirmó la efectividad de los métodos de aprendizaje automático implementados, observando en cada caso una mejora substancial con respecto al modelo base. En particular el modelo LSTM generó el mejor registro de RMSE, no observándose una clara ventaja en la utilización del modelo de red neuronal híbrido CNN + LSTM en esta fase inicial.

En las posteriores fases se corroboró la efectividad de complementar el conjunto de entrenamiento mediante el uso de características de entrada basadas en indicadores de análisis técnico, y a través de la incorporación de datos de entrenamiento adicionales correspondientes al histórico de otras criptomonedas. Dicho conjunto de entrenamiento expandido permitió que un incremento en la complejidad de los modelos de redes neuronales, en cuanto a cantidad de capas y de neuronas, se tradujera en una mejora en la precisión de los resultados. En las figuras 6.16-6.20 se ilustra un comparativo de los resultados del mejor modelo de cada tipo para cada fase de prueba, observándose en todos los casos una mejora en el desempeño en cada fase sucesiva.

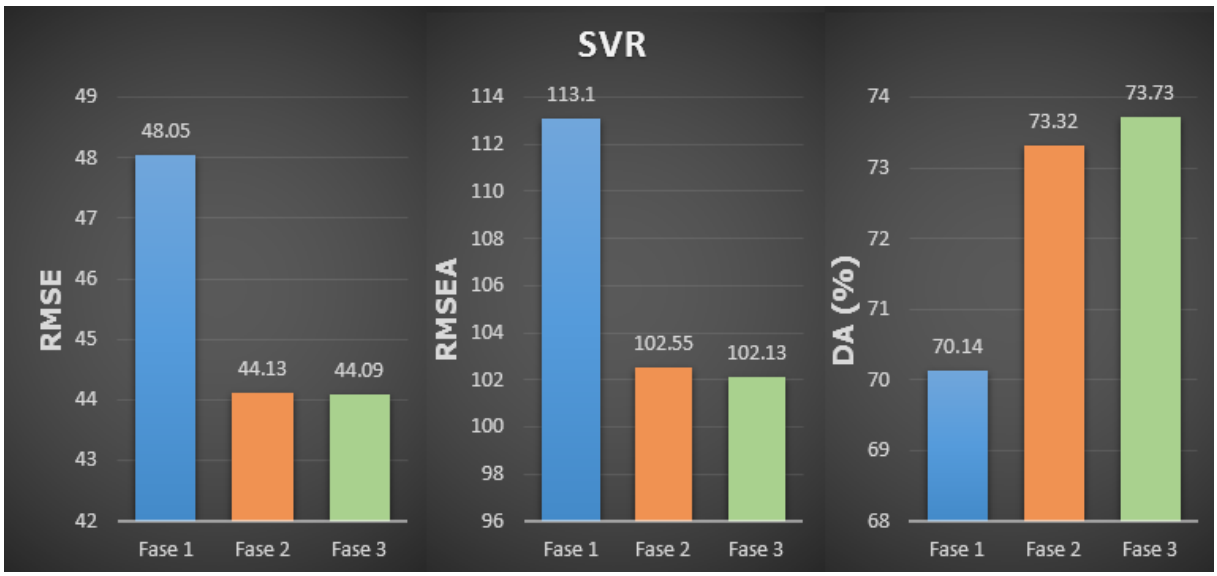


Figura 6.16: Resultados de los mejores modelos SVR de cada fase.

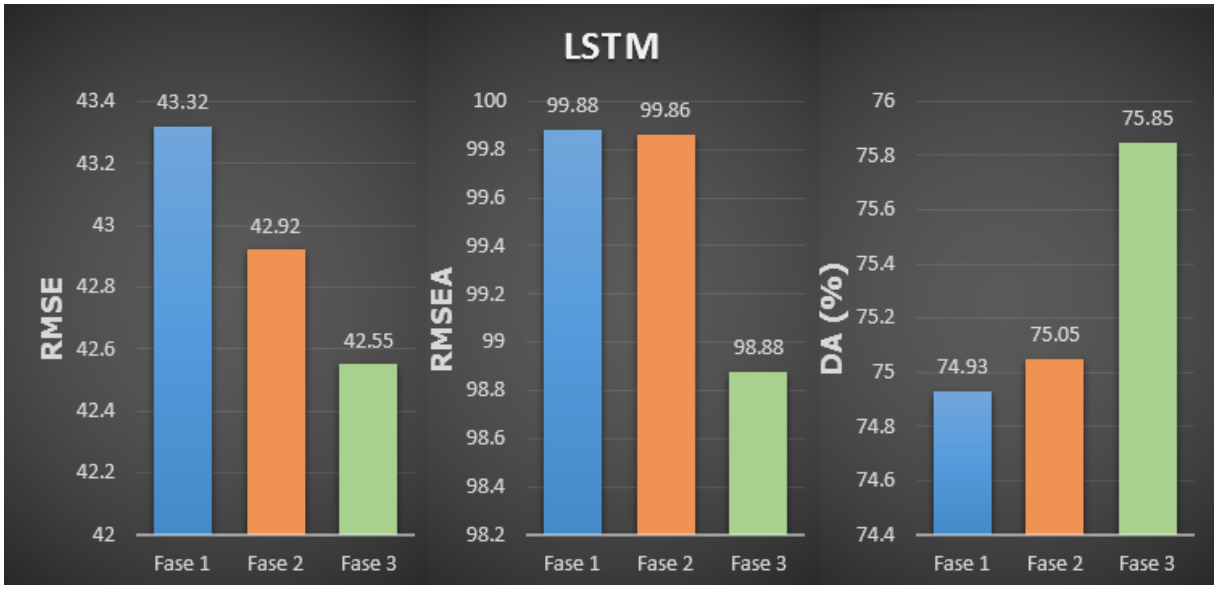


Figura 6.17: Resultados de los mejores modelos LSTM de cada fase.

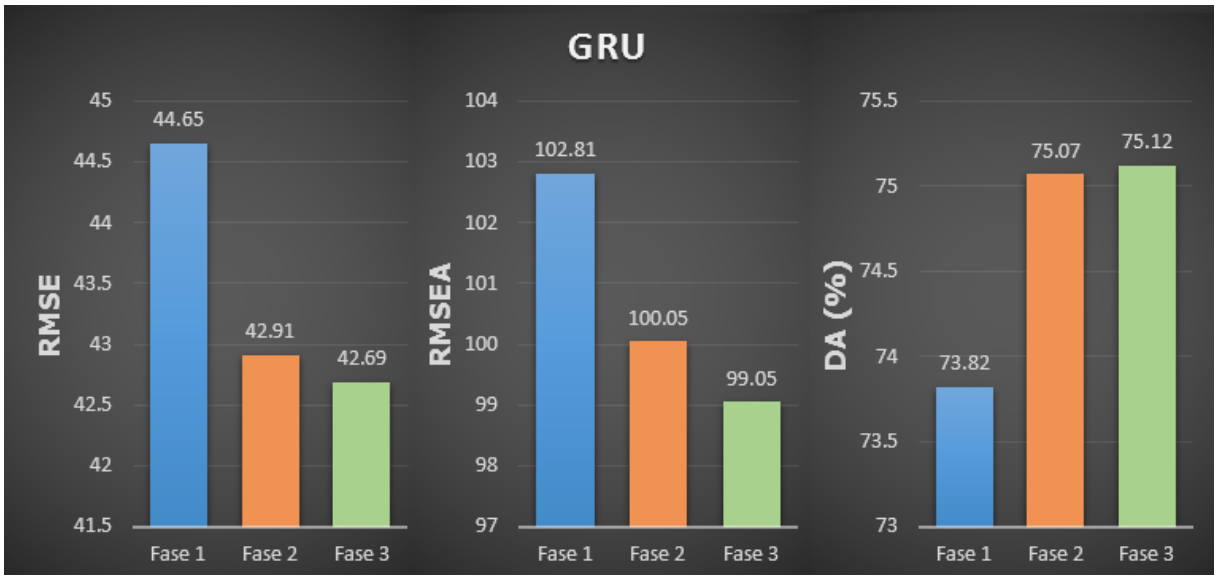


Figura 6.18: Resultados de los mejores modelos GRU de cada fase.

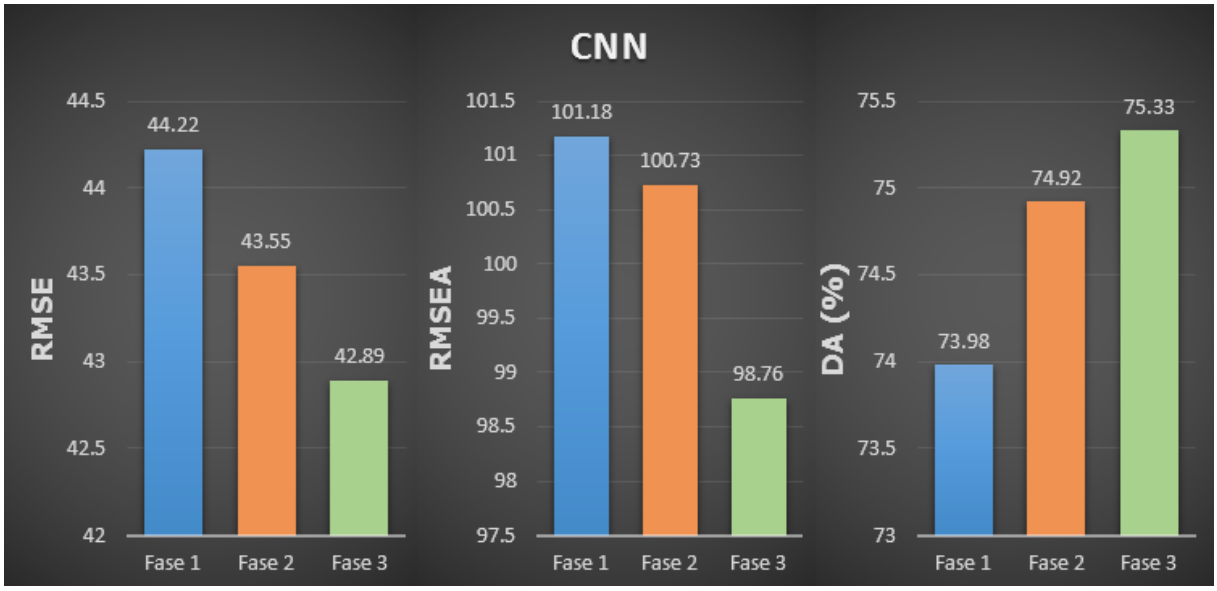


Figura 6.19: Resultados de los mejores modelos CNN de cada fase.

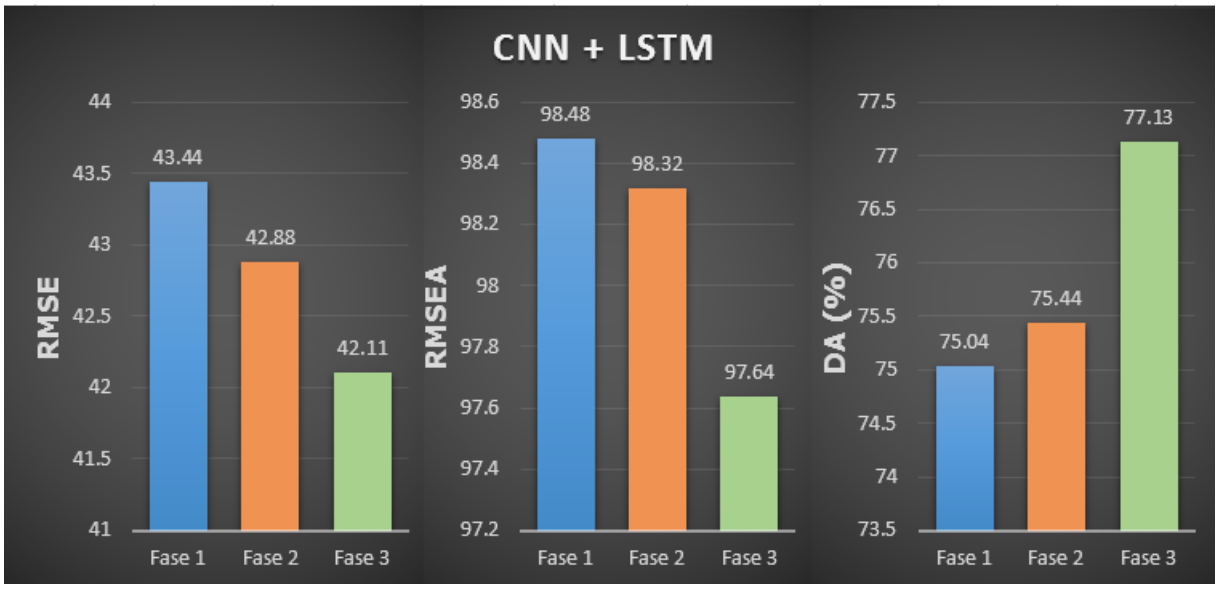


Figura 6.20: Resultados de los mejores modelos CNN + LSTM de cada fase.

En la Figura 6.21 se comparan los resultados correspondientes al mejor modelo de cada tipo para la fase final de experimentación, destacándose que la mayor precisión según todas las métricas definidas fue alcanzada por el modelo de red neuronal híbrido que combina en su arquitectura 3 capas CNN, 3 capas LSTM y una capa MLP (Figura 6.22).

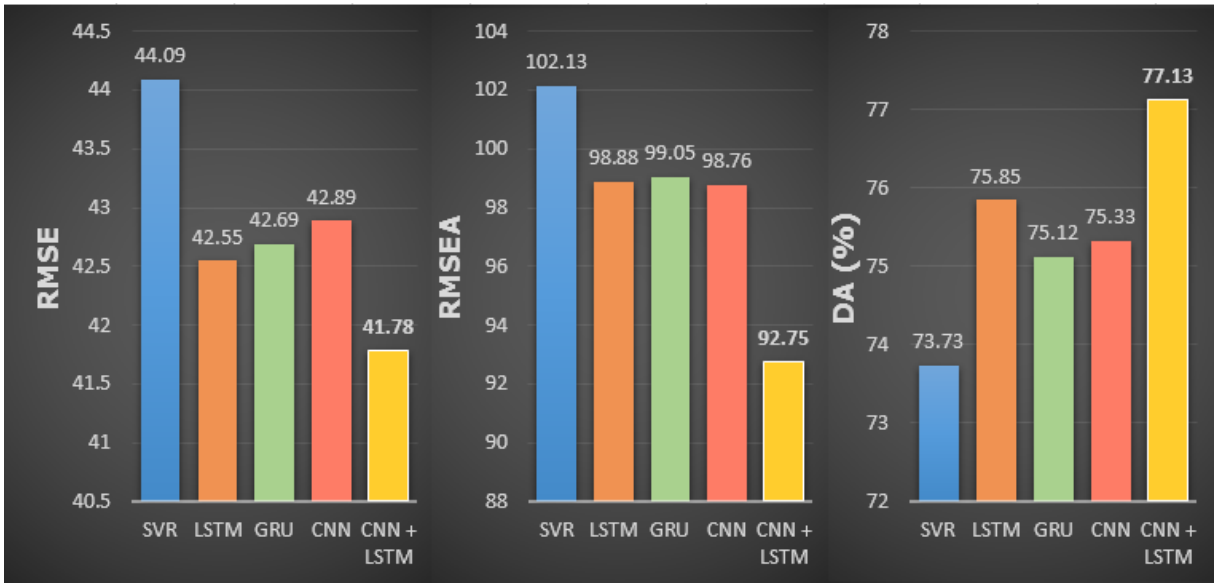


Figura 6.21: Comparación de la precisión de cada tipo de modelo en la última fase de experimentación.

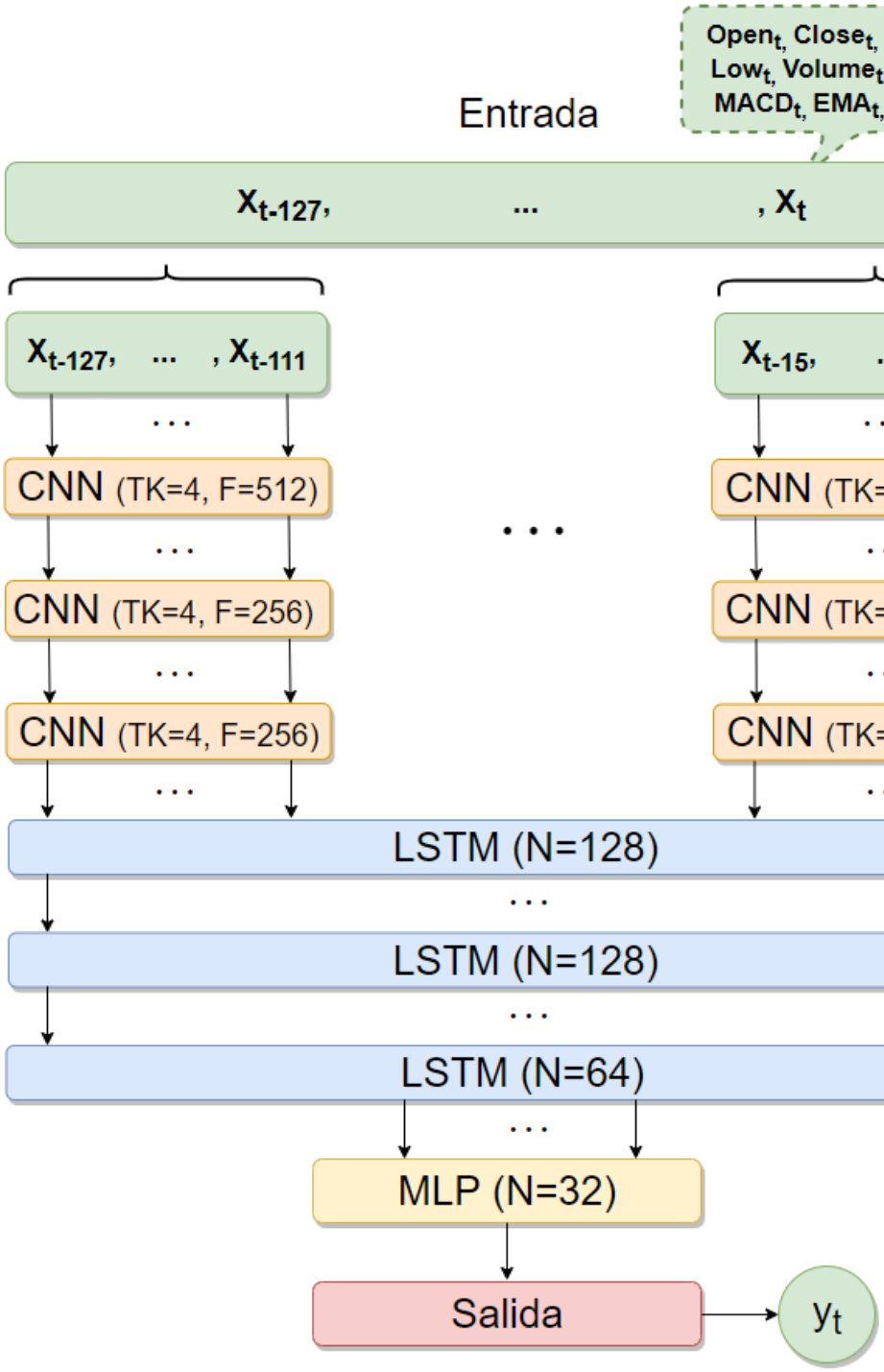


Figura 6.22: Arquitectura del mejor modelo hallado en las fases de pruebas. La red neuronal consiste de 3 capas CNN, 3 capas LSTM y una capa MLP, utilizando como características de entrada los datos base (Open, Close, High, Low, Volume) en conjunto con los indicadores de análisis técnico BB, MACD, EMA y RSI. TK = Tamaño del kernel, F = Número de filtros de salida, N = Cantidad de neuronas de la capa.

Capítulo 7

Conclusiones

Se realizó un repaso de las principales técnicas de aprendizaje automático, con hincapié en los modelos aplicables a problemas de pronósticos sobre series temporales. En base al análisis de resultados reportados en el estado del arte, se definió un conjunto de modelos a emplear en la resolución del problema.

Se desarrolló satisfactoriamente una serie de experimentos para evaluar la efectividad de cada tipo de modelo en la tarea de realizar pronósticos a corto plazo sobre el precio de Bitcoin.

Todas las implementaciones reportaron una precisión significativamente mayor al modelo base. Se destaca la efectividad de los modelos de aprendizaje profundo empleados: redes neuronales recurrentes LSTM, GRU, redes neuronales convolucionales y, en particular, el desempeño comparativamente superior alcanzado por la arquitectura híbrida que combina capas convolucionales y LSTM.

Para todos los tipos de modelo, se confirmó que la aplicación de una pre-interpretación de los datos base, mediante la incorporación de indicadores de análisis técnico entre las características de entrada, mejoró considerablemente los resultados.

Finalmente, para el caso de los modelos basados en redes neuronales, se comprobó la efectividad de incrementar el volumen de datos de entrenamiento a través de la incorporación

de información histórica de otras criptomonedas.

Se observa que de forma de explotar la adición de características y volumen de datos adicionales, se debió incrementar la complejidad de cada modelo (en cuanto a capas y cantidad de neuronas). En la misma linea, se aprecia que en la primer iteración (datos base) el modelo LSTM reportó los mejores resultados. Sin embargo, una vez incorporada la información adicional, el modelo híbrido CNN + LSTM alcanzó por un gran margen el mejor nivel de precisión en todas las métricas definidas:

- **RMSE**: Un valor de 42.11 significó una mejora del 26 % respecto al modelo base. Esto representa un error medio del 1.10 % sobre el precio de clausura en el periodo de evaluación considerado, correspondiente al mes de diciembre de 2018.
- **RMSEA**: En el entorno de los puntos de mayor varianza del periodo de evaluación, la diferencia en la precisión con respecto al modelo base es aún mayor, constituyendo una mejora del 34 %.
- **DA**: El resultado implica que el modelo fue capaz de predecir correctamente la dirección de la variación del precio un 77.13 % de las veces.

Se concluye que la aplicación de técnicas de aprendizaje profundo al pronóstico del precio de Bitcoin es, cuanto menos, prometedora. Esto es especialmente cierto considerando el amplio abanico de posibilidades que restan aún por explorarse siguiendo esta linea de investigación.

7.1. Trabajos futuros

Se consideran posibles adiciones y alternativas a explorar con el objetivo de profundizar aún más en la resolución del problema.

Optimización adicional de hiperparámetros

En especial para los modelos de alta complejidad, la gran cantidad de posibles combinaciones de parámetros hace de su optimización una tarea extremadamente demandante en cuanto a tiempo y poder computacional. Con tiempo y/o poder de procesamiento adicional, sería posible seguir refinando los hiperparámetros de forma de continuar mejorando los resultados obtenidos en este proyecto.

Granularidad de los datos

El conjunto de datos fuente utilizado provee la información del histórico del precio de Bitcoin y otras criptomonedas agregada por minuto. Con el fin de reducir el volumen de datos y la cantidad de pruebas a realizar como parte de este estudio, se optó por utilizar el conjunto de datos con una granularidad de una hora. Sin embargo, resulta de interés experimentar con distintos niveles de granularidad y evaluar su posible impacto en el desempeño de los modelos.

Ensemble

Dada la diversidad de modelos implementados, es posible que combinarlos en un *ensemble* [27] produzca resultados aún superiores. Una forma simple pero generalmente eficaz de hacerlo, es entrenando un regresor lineal, el cual utilizando como entrada la predicción obtenida para cada modelo, produzca como resultado una combinación lineal de las mismas.

Características adicionales

Sería interesante experimentar utilizando otras características como variables de entrada, por ejemplo, las presentes en cada bloque del blockchain (dificultad, costo de transacción,

cantidad de transacciones, etc.). Adicionalmente, podría emplearse la información de otras criptomonedas como variables de entrada, de forma de comprobar si lo que ocurre con el precio de otras criptomonedas es relevante para determinar el precio de Bitcoin en el corto plazo, o viceversa.

Análisis de sentimiento

Hay una infinidad de factores externos que inciden en el precio de Bitcoin. Un análisis de sentimiento [103] aplicado sobre noticias y/o redes sociales podría potencialmente revelar algo de información respecto a tales factores. Dicho análisis debería realizarse en función del tiempo, de forma de ser capaces de incluirlo como otra de las tantas series temporales que se utilizan como características de entrada.

Análisis de redes

En la propuesta inicial de este proyecto se planteó emplear técnicas de análisis de redes con el cometido de extraer información subyacente a la red conformada por el grafo de transacciones de Bitcoin que pudieran utilizarse como características de entrada adicionales. En esta linea, una de las posibilidades consideradas implicaba la identificación de nodos de la red cuyo comportamiento se anticipara a las fluctuaciones del precio de Bitcoin. El gran volumen de datos y de poder computacional que demanda el procesamiento del grafo de transacciones de Bitcoin implicó dejar este enfoque fuera del alcance del proyecto. Con recursos y tiempo adicionales sería de interés explorar la aplicación de este tipo de técnicas.

Estrategia de intercambio automático

Una vez alcanzado cierto nivel de precisión en la generación de predicciones, una posible aplicación de dichos resultados es en el intercambio de criptomonedas. Específicamente,

sería interesante evaluar el potencial de desarrollar una estrategia de intercambio automático que sea capaz de sacar provecho de los pronósticos generados. La implementación de dicha estrategia, podría potencialmente estar basada en técnicas de aprendizaje automático.

Bibliografía

[1] Ossama Abdel-Hamid y col. «Convolutional neural networks for speech recognition». En: *IEEE/ACM Transactions on audio, speech, and language processing* 22.10 (2014), págs. 1533-1545.

[2] Mark A Aizerman. «Theoretical foundations of the potential function method in pattern recognition learning». En: *Automation and remote control* 25 (1964), págs. 821-837.

[3] Elli Androulaki y col. «Evaluating user privacy in bitcoin». En: *International Conference on Financial Cryptography and Data Security*. Springer. 2013, págs. 34-51.

[4] Amin Azari. «Bitcoin Price Prediction: An ARIMA Approach». En: *KTH Royal Institute of Technology* (2018).

[5] Nashirah Abu Bakar y Sofian Rosbi. «Autoregressive Integrated Moving Average (ARIMA) Model for Forecasting Cryptocurrency Exchange Rate in High Volatility Environment: A New Insight of Bitcoin Transaction». En: *International Journal of Advanced Engineering Research and Science* 4.11 (2017).

[6] Aurelio F Bariviera y col. «Some stylized facts of the Bitcoin market». En: *Physica A: Statistical Mechanics and its Applications* 484 (2017), págs. 82-90.

[7] Ole Barndorff-Nielsen. «Exponentially decreasing distributions for the logarithm of particle size». En: *Proc. R. Soc. Lond. A* 353.1674 (1977), págs. 401-419.

[8] Jan Beran. *Statistics for long-memory processes*. Vol. 1. ISBN: 978-0412049019. Routledge, 2017.

[9] *Binance*. `https://www.binance.com`. Accedido: 2019-01-03.

[10] *Binance API*. `https://support.binance.com/hc/en-us/articles/115000840592-Binance-API-Beta`. Accedido: 2019-01-03.

[11] *Bitcoin Cash*. `https://www.bitcoincash.org/`. Accedido: 2019-01-28.

[12] Volker Blanz y col. «Comparison of view-based object recognition algorithms using realistic 3D models». En: *International Conference on Artificial Neural Networks*. Springer. 1996, págs. 251-256.

[13] *Blockchain Explorer*. `https://www.blockchain.com/explorer`. Accedido: 2018-11-22.

[14] Tim Bollerslev. «Generalized autoregressive conditional heteroskedasticity». En: *Journal of econometrics* 31.3 (1986), págs. 307-327.

[15] Bernhard E Boser, Isabelle M Guyon y Vladimir N Vapnik. «A training algorithm for optimal margin classifiers». En: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM. 1992, págs. 144-152.

[16] Elie Bouri y col. «On the hedge and safe haven properties of Bitcoin: Is it really more than a diversifier?» En: *Finance Research Letters* 20 (2017), págs. 192-198.

[17] George EP Box y col. *Time series analysis: forecasting and control*. ISBN: 978-0470272848. John Wiley & Sons, 2015.

[18] Jerry Brito y Andrea Castillo. *Bitcoin: A primer for policymakers*. Vol. 1. Mercatus Center at George Mason University, 2013.

[19] *Cardano*. `https://www.cardano.org/`. Accedido: 2019-01-28.

[20] Kyunghyun Cho y col. «Learning phrase representations using RNN encoder-decoder for statistical machine translation». En: *arXiv preprint arXiv:1406.1078* (2014).

[21] Usman Chohan. «The Double Spending Problem and Cryptocurrencies». En: *SSRN* (2017).

[22] Jeffrey Chu, Saralees Nadarajah y Stephen Chan. «Statistical analysis of the exchange rate of bitcoin». En: *PloS one* 10.7 (2015), e0133678.

[23] Junyoung Chung y col. «Empirical evaluation of gated recurrent neural networks on sequence modeling». En: *arXiv preprint arXiv:1412.3555* (2014).

[24] Dan Cireşan, Ueli Meier y Jürgen Schmidhuber. «Multi-column deep neural networks for image classification». En: *arXiv preprint arXiv:1202.2745* (2012).

[25] John H Cochrane. «Time series for macroeconomics and finance». En: *Manuscript, University of Chicago* (2005).

[26] Corinna Cortes y Vladimir Vapnik. «Support-vector networks». En: *Machine learning* 20.3 (1995), págs. 273-297.

[27] Thomas G Dietterich. «Ensemble methods in machine learning». En: *International workshop on multiple classifier systems*. Springer. 2000, págs. 1-15.

[28] *Download Bitcoin Core*. `https://bitcoin.org/en/download`. Accedido: 2018-11-22.

[29] Harris Drucker y col. «Support vector regression machines». En: *Advances in neural information processing systems*. 1997, págs. 155-161.

[30] Anne Haubo Dyhrberg. «Bitcoin, gold and the dollar–A GARCH volatility analysis». En: *Finance Research Letters* 16 (2016), págs. 85-92.

[31] Anne Haubo Dyhrberg. «Hedging capabilities of bitcoin. Is it the virtual gold?» En: *Finance Research Letters* 16 (2016), págs. 139-144.

[32] *EOS*. `https://eos.io/`. Accedido: 2019-01-28.

[33] *Ethereum*. `https://www.ethereum.org/`. Accedido: 2019-01-28.

[34] *Ethereum Classic*. `https://ethereumclassic.org/`. Accedido: 2019-01-28.

[35] Yin Fan y col. «Video-based emotion recognition using CNN-RNN and C3D hybrid networks». En: *Proceedings of the 18th ACM International Conference on Multimodal Interaction*. ACM. 2016, págs. 445-450.

[36] John Galbraith y Victoria Zinde-Walsh. «Autoregression-based estimators for ARFIMA models». En: (2001).

[37] Felix A Gers, Douglas Eck y Jürgen Schmidhuber. «Applying LSTM to time series predictable through time-window approaches». En: *Neural Nets WIRN Vietri-01*. Springer, 2002, págs. 193-200.

[38] Alex Graves, Navdeep Jaitly y Abdel-rahman Mohamed. «Hybrid speech recognition with deep bidirectional LSTM». En: *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE. 2013, págs. 273-278.

[39] Alex Graves, Abdel-rahman Mohamed y Geoffrey Hinton. «Speech recognition with deep recurrent neural networks». En: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE. 2013, págs. 6645-6649.

[40] Alex Graves y Jürgen Schmidhuber. «Offline handwriting recognition with multidimensional recurrent neural networks». En: *Advances in neural information processing systems*. 2009, págs. 545-552.

[41] Alex Graves y col. «Unconstrained on-line handwriting recognition with recurrent neural networks». En: *Advances in neural information processing systems*. 2008, págs. 577-584.

[42] Klaus Greff y col. «LSTM: A search space odyssey». En: *IEEE transactions on neural networks and learning systems* 28.10 (2017), págs. 2222-2232.

[43] Isabelle Guyon, B Boser y Vladimir Vapnik. «Automatic capacity tuning of very large VC-dimension classifiers». En: *Advances in neural information processing systems*. 1993, págs. 147-155.

[44] Donald O. Hebb y D. O. Hebb. *The Organization of Behavior*. ISBN: 978-0471367277. 1949.

[45] Keith W Hipel y A Ian McLeod. *Time series modelling of water resources and environmental systems*. Vol. 45. Elsevier, 1994.

[46] Sepp Hochreiter y Jürgen Schmidhuber. «Long short-term memory». En: *Neural computation* 9.8 (1997), págs. 1735-1780.

[47] Chiou-Jye Huang y Ping-Huan Kuo. «A deep cnn-lstm model for particulate matter (PM2. 5) forecasting in smart cities». En: *Sensors* 18.7 (2018), pág. 2220.

[48] Harold Edwin Hurst. «Long-term storage capacity of reservoirs». En: *Trans. Amer. Soc. Civil Eng.* 116 (1951), págs. 770-799.

[49] *ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)*. `http://www.image-net.org/challenges/LSVRC/2014/results`. Accedido: 2018-12-23.

[50] *Investopedia - ADI*. `https://www.investopedia.com/terms/a/accumulationdistribution.asp`. Accedido: 2019-01-29.

[51] *Investopedia - ADX*. `https://www.investopedia.com/articles/trading/07/adx-trend-indicator.asp`. Accedido: 2019-01-29.

[52] *Investopedia - ATR*. `https://www.investopedia.com/terms/a/atr.asp`. Accedido: 2019-01-29.

[53] *Investopedia - BB*. `https://www.investopedia.com/terms/b/bollingerbands.asp`. Accedido: 2019-01-29.

[54] *Investopedia - CCI*. `https://www.investopedia.com/terms/c/commoditychannelindex.asp`. Accedido: 2019-01-29.

[55] *Investopedia - CMF*. `https://www.investopedia.com/terms/m/moneyflow.asp`. Accedido: 2019-01-29.

[56] *Investopedia - DC*. `https://www.investopedia.com/terms/d/donchianchannels.asp`. Accedido: 2019-01-29.

[57] *Investopedia - DPO*. https://www.investopedia.com/terms/d/detrended-price-oscillator-dpo.asp. Accedido: 2019-01-29.

[58] *Investopedia - EMA*. https://www.investopedia.com/terms/e/ema.asp. Accedido: 2019-01-29.

[59] *Investopedia - EoM*. https://www.investopedia.com/terms/e/easeofmovement.asp. Accedido: 2019-01-29.

[60] *Investopedia - FI*. https://www.investopedia.com/terms/f/force-index.asp. Accedido: 2019-01-29.

[61] *Investopedia - Ichimoku*. https://www.investopedia.com/terms/i/ichimokuchart.asp. Accedido: 2019-01-29.

[62] *Investopedia - KC*. https://www.investopedia.com/terms/k/keltnerchannel.asp. Accedido: 2019-01-29.

[63] *Investopedia - KST*. https://www.investopedia.com/terms/k/know-sure-thing-kst.asp. Accedido: 2019-01-29.

[64] *Investopedia - MACD*. https://www.investopedia.com/terms/m/macd.asp. Accedido: 2019-01-29.

[65] *Investopedia - MFI*. https://www.investopedia.com/terms/m/mfi.asp. Accedido: 2019-01-29.

[66] *Investopedia - MI*. https://www.investopedia.com/terms/m/mass-index.asp. Accedido: 2019-01-29.

[67] *Investopedia - NVI*. https://www.investopedia.com/terms/n/nvi.asp. Accedido: 2019-01-29.

[68] *Investopedia - OBV*. https://www.investopedia.com/terms/o/onbalancevolume.asp. Accedido: 2019-01-29.

[69] *Investopedia - RSI*. https://www.investopedia.com/terms/r/rsi.asp. Accedido: 2019-01-29.

[70] *Investopedia - SR*. https://www.investopedia.com/terms/s/stochasticoscillator.asp. Accedido: 2019-01-29.

[71] *Investopedia - TRIX*. https://www.investopedia.com/terms/t/trix.asp. Accedido: 2019-01-29.

[72] *Investopedia - TSI*. https://www.investopedia.com/terms/t/tsi.asp. Accedido: 2019-01-29.

[73] *Investopedia - UO*. https://www.investopedia.com/terms/u/ultimateoscillator.asp. Accedido: 2019-01-29.

[74] *Investopedia - VI*. https://www.investopedia.com/terms/v/vortex-indicator-vi.asp. Accedido: 2019-01-29.

[75] *Investopedia - VPT*. https://www.investopedia.com/terms/v/vptindicator.asp. Accedido: 2019-01-29.

[76] *Investopedia - WR*. https://www.investopedia.com/terms/w/williamsr.asp. Accedido: 2019-01-29.

[77] *IOTA*. https://www.iota.org/. Accedido: 2019-01-28.

[78] Grigorevich Ivakhnenko Alekse y Grigorevich Lapa Valentin. *Cybernetic predicting devices*. CCM Information Corporation, 1965.

[79] Gareth James y col. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[80] Rafal Jozefowicz, Wojciech Zaremba e Ilya Sutskever. «An empirical exploration of recurrent network architectures». En: *International Conference on Machine Learning*. 2015, págs. 2342-2350.

[81] Ebru Şeyma Karakoyun y Ali Osman Çıbıkdiken. «Comparison of ARIMA Time Series Model and LSTM Deep Learning Algorithm for Bitcoin Price Forecasting». En: *Proceedings of MAC 2018 in Prague* (2018), pág. 171.

[82] Paraskevi Katsiampa. «Volatility estimation for Bitcoin: A comparison of GARCH models». En: *Economics Letters* 158 (2017), págs. 3-6.

[83] *Keras*. `https://keras.io/`. Accedido: 2019-01-29.

[84] *Kernel (Image processing), Wikipedia*. `https://en.wikipedia.org/wiki/Kernel_(image_processing)`. Accedido: 2018-11-22.

[85] Steve Lawrence y col. «Face recognition: A convolutional neural-network approach». En: *IEEE transactions on neural networks* 8.1 (1997), págs. 98-113.

[86] Patrick Le Callet, Christian Viard-Gaudin y Dominique Barba. «A convolutional neural network approach for objective video quality assessment». En: *IEEE Transactions on Neural Networks* 17.5 (2006), págs. 1316-1327.

[87] Yann LeCun, Yoshua Bengio y Geoffrey Hinton. «Deep learning». En: *nature* 521.7553 (2015), pág. 436.

[88] Yann LeCun y col. «Gradient-based learning applied to document recognition». En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324.

[89] *Litecoin*. `https://litecoin.org/`. Accedido: 2019-01-28.

[90] N Gregory Mankiw. *Principles of macroeconomics*. ISBN: 978-0538453066. Cengage Learning, 2014.

[91] *MathWorks - Introduction to Deep Learning: What Are Convolutional Neural Networks?* `https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html`. Accedido: 2018-12-23.

[92] Masakazu Matsugu y col. «Subject independent facial expression recognition with robust face detection using a convolutional neural network». En: *Neural Networks* 16.5-6 (2003), págs. 555-559.

[93] Ralph C Merkle. «A digital signature based on a conventional encryption function». En: *Conference on the theory and application of cryptographic techniques*. Springer. 1987, págs. 369-378.

[94] John Mern, Spenser Anderson y John Poothokaran. «Using Bitcoin Ledger Network Data to Predict the Price of Bitcoin». En: *Stanford University Department of Aeronautics and Astronautics* (2017).

[95] Ian Miers y col. «Zerocoin: Anonymous distributed e-cash from bitcoin». En: *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE. 2013, págs. 397-411.

[96] Marvin Minsky y Seymour Papert. *Perceptron Expanded Edition*. ISBN: 9780262631112. 1969.

[97] TM Mitchell. «Machine Learning, McGraw-Hill Higher Education». En: *New York* (1997).

[98] Malte Moser, Rainer Bohme y Dominic Breuker. «An inquiry into money laundering tools in the Bitcoin ecosystem». En: *eCrime Researchers Summit (eCRS), 2013*. IEEE. 2013, págs. 1-14.

[99] Satoshi Nakamoto. «Bitcoin: A peer-to-peer electronic cash system». En: *www.bitcoin.org* (2008).

[100] *NEO*. `https://neo.org/`. Accedido: 2019-01-28.

[101] *Ontology*. `https://ont.io/`. Accedido: 2019-01-28.

[102] Thomas Oommen y col. «An objective analysis of support vector machine based classification for remote sensing». En: *Mathematical geosciences* 40.4 (2008), págs. 409-424.

[103] Bo Pang, Lillian Lee y col. «Opinion mining and sentiment analysis». En: *Foundations and Trends® in Information Retrieval* 2.1–2 (2008), págs. 1-135.

[104] Morgen E Peck. «The cryptoanarchists' answer to cash». En: *IEEE Spectrum* 49.6 (2012).

[105] *Python*. `https://www.python.org`. Accedido: 2019-02-03.

[106] *Quantum*. `https://qtum.org/`. Accedido: 2019-01-28.

[107] *Ripple*. `https://ripple.com/xrp/`. Accedido: 2019-01-28.

[108] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[109] Frank Rosenblatt. «The perceptron: a probabilistic model for information storage and organization in the brain.» En: *Psychological review* 65.6 (1958), pág. 386.

[110] David E Rumelhart, Geoffrey E Hinton y Ronald J Williams. «Learning representations by back-propagating errors». En: *nature* 323.6088 (1986), pág. 533.

[111] Katherine Sagona-Stophel. «Bitcoin 101 white paper». En: *www.bitcoin.org* 20 (2015).

[112] Dominik Scherer, Andreas Müller y Sven Behnke. «Evaluation of pooling operations in convolutional architectures for object recognition». En: *Artificial Neural Networks–ICANN 2010*. Springer, 2010, págs. 92-101.

[113] Bernhard Schölkopf, Chris Burges y Vladimir Vapnik. «Incorporating invariances in support vector learning machines». En: *International Conference on Artificial Neural Networks*. Springer. 1996, págs. 47-52.

[114] *Scikit-Learn*. `https://scikit-learn.org`. Accedido: 2019-01-29.

[115] Yelong Shen y col. «Learning semantic representations using convolutional neural networks for web search». En: *Proceedings of the 23rd International Conference on World Wide Web*. ACM. 2014, págs. 373-374.

[116] Alex J Smola y Bernhard Schölkopf. «A tutorial on support vector regression». En: *Statistics and computing* 14.3 (2004), págs. 199-222.

[117] *Stellar*. `https://www.stellar.org/`. Accedido: 2019-01-28.

[118] Mark Stitson y col. «Support vector regression with ANOVA decomposition kernels». En: *Advances in kernel methods—Support vector learning* (1999), págs. 285-292.

[119] *Stochastic gradient descent in plain English - Medium*. `https://medium.com/@julian.harris/stochastic-gradient-descent-in-plain-english-9e6c10cdba97r`. Accedido: 2019-23-02.

[120] Jiajun Sun, Jing Wang y TC Yeh. *Video understanding: from video classification to captioning*. 2017.

[121] Dian Utami Sutiksno y col. «Forecasting Historical Data of Bitcoin using ARIMA and α-Sutte Indicator». En: *Journal of Physics: Conference Series*. Vol. 1028. 1. IOP Publishing. 2018, pág. 012194.

[122] Nick Szabo. «Formalizing and securing relationships on public networks». En: *First Monday* 2.9 (1997).

[123] Christian Szegedy y col. «Going deeper with convolutions». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, págs. 1-9.

[124] *TensorFlow*. `https://www.tensorflow.org/`. Accedido: 2019-01-29.

[125] Andrei Nikolaevich Tikhonov. «On the solution of ill-posed problems and the method of regularization». En: *Doklady Akademii Nauk*. Vol. 151. 3. Russian Academy of Sciences. 1963, págs. 501-504.

[126] DOUGLAS GARCIA TORRES y HONGLIANG QIU. «Applying Recurrent Neural Networks for Multivariate Time Series Forecasting of Volatile Financial Data». En: (2018).

[127] *TradingView - AO*. `https://www.tradingview.com/ideas/awesomeoscillator/`. Accedido: 2019-01-29.

[128] *TRON*. `https://tron.network/`. Accedido: 2019-01-28.

[129] VN Vapnik y A Ya Lerner. «Recognition of patterns with help of generalized portraits». En: *Avtomat. i Telemekh* 24.6 (1963), págs. 774-780.

[130] Jean-Philippe Vert, Koji Tsuda y Bernhard Schölkopf. «A primer on kernel methods». En: *Kernel methods in computational biology* 47 (2004), págs. 35-70.

[131] Paul Werbos. «Beyond Regression:"New Tools for Prediction and Analysis in the Behavioral Sciences». En: *Ph. D. dissertation, Harvard University* (1974).

[132] *Wikipedia*. `https://en.wikipedia.org/wiki/File:Kernel_trick_idea.svg`. Accedido: 2018-11-22.

[133] Mark T Williams. «Virtual Currencies–Bitcoin Risk». En: *world bank conference, Washington, DC*. Vol. 21. 2014.

[134] David H Wolpert. «The lack of a priori distinctions between learning algorithms». En: *Neural computation* 8.7 (1996), págs. 1341-1390.

[135] David H Wolpert y William G Macready. «No free lunch theorems for optimization». En: *IEEE transactions on evolutionary computation* 1.1 (1997), págs. 67-82.

[136] Yuankai Wu y Huachun Tan. «Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework». En: *arXiv preprint arXiv:1612.01022* (2016).

[137] Jianbo Yang y col. «Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.» En: *Ijcai*. Vol. 15. 2015, págs. 3995-4001.