

,,

UNIVERSIDAD ANDRES BELLO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE CIENCIAS DE LA INGENIERÍA
MAGÍSTER EN CIENCIAS DE LA COMPUTACIÓN

Modelo de decisión lingüístico enriquecido con LSTM para la selección de criptomonedas

Seminario Magister en Ciencias de la Computación

Autor: Pablo Poblete Arrué
Profesor Guía: Dra. Romina Torres

Viña del Mar, Chile, 2021.

Contenidos

1 Introducción **2**
1.1 Mercado Eficiente e Inversionistas . . . 2
1.2 Fundamentación del Problema . . . 4
1.3 Discusión bibliográfica . . . 5
1.4 Propuesta . . . 12
1.5 Metodología . . . 14
1.5.1 Metodología para la implementación de modelos . . . 17
1.5.2 Knoledge Discovery y Data Mining . . . 19
1.5.3 Metodología de Data Science . . . 20
1.5.4 Feature Engineering . . . 21
1.6 Contribución del trabajo . . . 23
1.6.1 Enfoque DLDM . . . 23
1.6.2 Contribución esperada . . . 23

2 Marco Teórico **25**
2.1 Linguistic Decision model o MCDM . . . 25
2.2 Métodos de pronóstico . . . 27
2.3 Pronósticos en series temporales financieras . . . 28
2.3.1 Serie temporal financiera . . . 29
2.3.2 Estacionariedad . . . 30
2.3.3 Recursive multi-step forecasting . . . 31
2.3.4 Medias móviles . . . 31
2.3.5 Suavizado exponencial . . . 31
2.3.6 Modelos regresión para series de tiempo . . . 32
2.3.7 Modelos de predicción en series temporales . . . 32
2.3.8 Modelos Autorregresivos . . . 32
2.3.9 Modelos ARIMA . . . 32
2.4 Red Neuronal Artificial (ANN) . . . 33
2.4.1 Perceptrón . . . 34
2.4.2 Red neuronal feed-forward . . . 37
2.4.2.1 Función de activación . . . 39
2.4.2.2 Aproximación de la función XOR . . . 40
2.4.2.3 Retropropagación en el MLP . . . 41
2.4.2.4 Tipos de descenso de gradiente . . . 46
2.4.2.5 El algoritmo . . . 47
2.4.3 Redes neuronales recurrentes (RNN) . . . 48
2.4.4 Redes LSTM . . . 50

3 Materiales y Métodos **53**
3.1 Materiales . . . 53
3.1.1 Archivos planos criptomonedas . . . 53
3.1.2 Datasets en entrenamiento de modelos . . . 53
3.1.3 Datasets de atributos para modelo lingüístico . . . 54
3.2 Método . . . 55
3.2.1 Etapa 1: Modelos LSTM . . . 57
3.2.1.1 Paso 1: Series temporales OHLC y Log Market Cap. . . . 58
3.2.1.2 Paso 2: Escalamiento de características . . . 58
3.2.1.3 Paso 3: Estructurar dataset . . . 59
3.2.1.4 Paso 4: Definir, configurar y entrenar modelos LSTM . . . 60
3.2.1.5 Paso 5: Obtener predicciones . . . 62
3.2.1.6 Paso 6: Generar datasets de atributos . . . 63
3.2.1.7 Paso 7: Preproceso de atributos . . . 64
3.2.2 Etapa 2: Modelo de Decisión Lingüístico . . . 65
3.2.2.1 Paso 1: Construir el Modelo de Decisión Lingüístico . . . 65
3.2.2.2 Paso 2: Normalización de las Evaluaciones . . . 66
3.2.2.3 Paso 3: Obtención de Variables Lingüísticas . . . 67
3.2.2.4 Paso 4: Obtener Matrices Lingüísticas temporales . . . 68
3.2.2.5 Paso 5: Agregación de Información Lingüística de los Atributos 68
3.2.2.6 Paso 6: Agregación dinámica de la información temporal . . 69
3.2.2.7 Paso 7: Ranking de las Alternativas . . . 69
3.2.3 Etapa 3: Experimentación con estrategias simuladas . . . 70
3.2.3.1 Estrategias de inversión simuladas . . . 70
3.2.3.2 Proceso de inversión . . . 70

4 Resultados **73**
4.1 Entrenamiento de modelos LSTM . . . 73
4.2 Selección de modelos LSTM . . . 75
4.3 Experimentación en proceso de inversión . . . 76
4.3.1 Datasets de atributos . . . 76
4.3.2 Resultado en experimentos con proceso de inversión simulado . . . 78
4.3.2.1 Experimento que superan máximo retorno del benchmark . 78
4.3.2.2 Experimentos con retorno de 95% o superior . . . 81
4.3.2.3 Rendimiento del benchmark estrategias . . . 81

5 Conclusiones y trabajo futuro **83**
5.1 Conclusiones . . . 83
5.2 Trabajo futuro . . . 84

Lista de Figuras

1 Metodología general aplicada en este trabajo . . . 14
2 Metodología para la selección modelos LSTM . . . 16
3 Metodología aplicada en enfoque DLDM . . . 17
4 Proceso ML en predicción de criptomonedas [20] . . . 18
5 Proceso KDD [21] . . . 19
6 Metodología de data science [2] . . . 20
7 Feature engineering [54] . . . 22
8 Metodología genérica para análisis de series temporales . . . 29
9 Modelo matemático de una neurona . . . 33
10 Funciones linealmente separables v/s XOR . . . 36
11 Red neuronal simple . . . 37
12 Perceptrón de capa única . . . 38
13 Perceptrón multicapa . . . 38
14 MLP que resuelve problema de clasificación **XOR** . . . 40
15 Grafo resumido de la red **XOR** . . . 41
16 Derivadas parciales . . . 45
17 Arquitectura genérica de una RNN . . . 49
18 Despliegue de una RNN . . . 49
19 Estructura interna red neuronal LSTM [57] . . . 51
20 Diseño conceptual . . . 55
21 Proceso general . . . 56
22 Términos lingüísticos triangulares para variables lingüísticas de beneficio (22a) y costo (22b) . . . 68

Lista de Tablas

1 Estructura de los archivos planos . . . 53
2 Series temporales entrenamiento y validación . . . 54
3 Atributos para modelo DLDM . . . 54
4 Modelos genéricos de dos capas ocultas . . . 61
5 Modelos genéricos de una capa oculta . . . 61
6 Resumen modelos de dos capas ocultas . . . 61
7 Resumen modelos de una capa oculta . . . 62
8 Atributos en proceso de predicción experimentos . . . 63
9 Scalers seleccionados para experimentos . . . 66
10 Carteras en experimentación . . . 71
11 Vista parcial MAE de los modelo LSTM . . . 74
12 Modelos LSTM con MAE promediado . . . 75
13 Modelos seleccionados . . . 76
14 Datasets de atributos . . . 77

15 Rendimiento estrategia DLDM-1, Torres *et al.* [65] 78
16 Rendimiento estrategia DLDM por modelo 79
17 Estrategia DLDM vs DLDM-1 en benchamark [65] 80
18 Rendimiento estrategia DLDM con **Modelo 128_1_50** 81
19 Rendimiento estrategia DLDM de 95% o superior 81
20 Estrategia DLDM vs benchamark de estrategias 82

Abstract

El análisis fundamental y el análisis técnico son técnicas ampliamente utilizadas para seleccionar la alternativa de inversión que presenta mayor potencial de retorno en un horizonte de tiempo dado. Lamentablemente, en mercados con un gran número de alternativas disponibles como es el de las criptomonedas, cuyo volumen estimado supera las 10 mil transándose activamente y con inversionistas no necesariamente expertos, estas técnicas resultan inaplicables, si es necesario considerar todo el mercado a la vez. Este trabajo presenta un método para apoyar a los inversionistas en la toma de decisiones de compra de criptomonedas con una estrategia basada en comprar y vender frecuentemente (*'buy-and-sell'*), que consiste en un ranking dinámico diario utilizando un modelo de decisión lingüística que considera pronósticos en cada atributo de entrada para seleccionar la criptomoneda con mayor potencial de retorno como próximo objetivo de inversión. Este método es ilustrado en un caso simulado sobre un dataset experimental con observaciones diarias para 68 criptomonedas entre el 1 de marzo de 2018 al 31 de mayo de 2018. Los resultados muestran que el modelo LDM que utiliza las combinaciones de atributos Day profitability, Day variability y como tercer atributo ya sea Open, Close, Low o High, superan al final del periodo de inversión el retorno de lo publicado en el estado del arte en más de 300% y, si sólo se consideran los atributos Day profitability y Day variability lo supera en más de 1700%. Claramente, este método supera estrategias alternativas como 1) la basada en selección aleatoria de alternativa de inversión o 2) comprar y mantener Bitcoin o 3) invertir equitativamente y mantener inversión en todas la criptomonedas del mercado, las cuales generan retornos del 3%, -34% y 54% respectivamente.

1 Introducción

1.1 Mercado Eficiente e Inversionistas

Un mercado eficiente (EM) [18] es aquel en el cual, en todo momento los precios de los instrumentos financieros negociables reflejan toda la información disponible [42] sobre su valor fundamental [3]. Un enfoque más moderno [19] dice que los precios reflejan toda la información [71] hasta el punto de equilibrio entre el costo marginal de conseguir nueva información y el beneficio marginal de su obtención [29].

Se reconocen tres niveles de eficiencia en el mercado [71, 72], con diferentes implicancias en el funcionamiento de los mercados [8, 18, 42]: débil, semi-fuerte y fuerte. En la forma débil se verifica que los precios incorporan toda la información pasada, en la forma semi-fuerte se verifica que los precios incorporan toda la información pública así como la pasada, en tanto que la forma fuerte verifica que los precios incorporan tanto la información privada como la pública y por tanto nadie puede obtener un rendimiento superior al mercado.

Para los inversionistas la hipótesis de los mercados eficientes (EMH) tiene importantes implicancias ya que de ella se desprende que el trading especulativo conlleva un alto costo [3] por las cuantiosas pérdidas que el juego especulativo puede acarrear. En cualquier caso es un tema de debate y permanentemente se está intentando probar o refutar su validez. Diversas investigaciones indican que ya sea en mercados en desarrollo o en mercados financieros que están sometidos a estrés y en operaciones de corto plazo, se verifican oportunidades para los traders ya sea aprovechando ventajas de arbitraje y diversas anomalías transitorias del mercado o bien a través de las fluctuaciones producto de nueva información en proceso de asimilación.

A pesar de no descartar la EMH, resulta interesante constatar que los profesionales de las finanzas realmente no creen [9] en la forma fuerte de la misma ya que para ellos, de ser efectiva no existirían los fondos de cobertura ni los administradores activos, quienes a través de un enfoque cuantitativo construyen carteras (conjuntos de instrumentos de inversión) con el apoyo de profesionales altamente calificados en constante búsqueda de información relevante sobre los activos y mercados en que operan.

Conviene mencionar que en mercados erráticos la diversificación resulta irrelevante por el exceso de volatilidad [22], no así en mercados en los cuales opera una ley Gaussiana de grandes números [71] que da más certeza al retorno de la cartera a medida que se diversifica, dado que a pesar de no tener capacidad de predecir el comportamiento individual, sí se podría predecir el comportamiento promedio.

En lo referente a investigaciones tendientes a validar o refutar teorías de economía financiera, la hipótesis de los mercados eficientes (EMH) [18, 72] como teoría del comportamiento de los mercados sigue siendo objeto de intenso debate [9, 29, 71] por tratarse de una simplificación, excesiva para algunos, de la complejidad del comportamiento de los mismos. Los esfuerzos para probar y también para refutar la teoría son numerosos y no necesariamente concluyentes [71, 8] para ninguna de las posiciones en pugna, ya que la evidencia en contra no resulta demasiado convincente. Las crisis financieras globales, en particular la del año 2008 [29] han incrementado el escrutinio sobre sus postulados.

Bajo una hipótesis de mercado eficiente (EMH), los precios deben seguir un proceso estocástico sin memoria. Uno de los modelos más comunes es el modelo de caminata aleatoria, [60, 71, 72] por ello, si los precios siguen una trayectoria aleatoria es difícil predecir su trayectoria futura, de lo cual se desprende que un inversionista no podría superar una estrategia simplista del tipo comprar y mantener ('*buy-and-hold*'). Ahora bien, si las series temporales exhiben memoria de largo alcance entonces la predicción puede basarse en precios pasados [43].

En un sentido amplio se distinguen dos grandes grupos de inversionistas: los expertos y los no expertos. Un inversionista experto es un profesional con capacidad de modelar su proceso de toma de decisiones [9], logrando que sea más bien racional pues está basado tanto en datos como en modelos con diferentes grados de sofisticación.

Por otro lado, los no expertos también conocidos como '*noise traders*' [9, 45, 71], tienden a tomar decisiones menos racionales ya que su proceso de toma de decisiones carece de modelamiento y, frecuentemente transan confiando en creencias, corazonadas e información errónea, incompleta y/o sesgada. De esta manera '*noise trading*' es transar sobre ruido como si este fuese información [6] y, generalmente la obtienen a través de los medios, el boca a boca y de analistas, entre muchas fuentes alternativas.

Se ha descubierto que los '*noise traders*' tienen un efecto significativo en el precio de los activos [56]. Los modelos plantean que algunos de ellos transan por necesidades que no necesariamente están correlacionadas con la información sobre los fundamentales de los activos en los que negocian. En estas dinámicas, pequeñas dosis de información pueden resultar en importantes variaciones en los precios, lo cual ocurre cuando las personas siguen a la masa en lugar de basar su toma de decisiones en información propia, situación que es explicada por los modelos de pastoreo (*Herding models*) [56]. Además, con una gran cantidad de '*noise traders*' en los mercados, ahora es valioso contar con información para transar, incluso si se trata de información costosa de obtener, puesto que se abre la posibilidad de negociar con ella [56]. Por esto se considera que los '*noise traders*' son quienes rentabilizan el juego de los cazadores de oportunidades [9]. Es decir los inversionistas racionales sólo comprarán con descuento para compensar el riesgo [29]. Diversos investigadores se refieren en sus trabajos a la utilidad de los '*noise traders*' para la estabilidad de los mercados, por ejemplo, de acuerdo a Gao et al. [24] el incremento en la cantidad de '*noise traders*' hace a los mercados más resilientes ante shocks, dado que al empujar los precios lejos de los fundamentales, crean oportunidades de aprendizaje, incrementando la proporción de traders informados, con destrezas de negociación, lo que finalmente redundaría en proteger al mercado. Asimismo [45] indica que los '*noise traders*' como grupo pueden sobrevivir y llegar a dominar a los inversores racionales en bienestar, incluso cuando en promedio, un inversionista racional domina a cualquier '*noise trader*'.

Actualmente, producto de la gran cantidad de '*noise traders*' en los mercados, se reconoce el valor de transar con información, incluso si se trata de información costosa de obtener [56].

La gestión activa de carteras, el enfoque cuantitativo y la aplicación del concepto de diversificación [22] tiene por objetivo maximizar el rendimiento de una cartera sujeto a un nivel fijo de riesgo o bien minimizar el nivel de riesgo para un rendimiento esperado. Una cartera diversificada reduce su volatilidad mediante la eventual eliminación del riesgo

no sistemático [22], en tanto que el riesgo sistemático o de mercado no se puede eliminar. Adicionalmente, la propiedad de martingala [60, 71] del proceso de precios permite anticipar que en periodos largos, incluso con gestión activa de carteras [22], no será posible obtener ganancias que superen significativamente al mercado. Se ha planteado que si bien no es posible vencer al mercado en todo momento, los rendimientos de las carteras siguen una distribución normal y por lo cual sería un proceso susceptible a la aplicación de técnicas estadísticas y numéricas.

1.2 Fundamentación del Problema

Una criptomoneda (criptodivisa o criptoactivo) es un medio de intercambio descentralizado que utiliza funciones criptográficas para realizar transacciones financieras [20]. Estimaciones actuales cifran en aproximadamente 17 mil las criptomonedas que se transan públicamente [58] y su número continua incrementándose. Por ejemplo la criptomoneda Bitcoin (BTC) es un sistema de moneda digital que funciona sin una autoridad o gobierno central [28], como es el caso de las divisas fiduciarias. Así mismo, la comercialización de criptomonedas, se define como el acto de comprar y vender criptomonedas con la intención de obtener ganancias [20]. Para apoyar este proceso existe una abundante oferta de software para la implementación de la infraestructura de transacción de criptomonedas: Capfolio, 3Commas, CCXT, Blackbird, StockSharp, Freqtrade, Cryptosignal, Ctubio, Catalyst, Golang Crypto Trading Bot [28].

La creciente complejidad, diversidad e interacción de los mercados de criptomonedas hacen que tomar decisiones de inversión sea un proceso expuesto a niveles de incertidumbre en aumento, incluso para inversionistas profesionales con acceso a herramientas y procesos sofisticados de toma de decisiones. Esto da cuenta de un problema cuyo análisis excede en complejidad a las capacidades de herramientas de análisis gráfico, métodos del análisis técnico o métodos basados en el análisis de los fundamentales.

Las comunidades profesionales de analistas financieros conocidos como “fundamentalistas” y la de analistas técnicos generalmente conocidos como “chartistas” así como los dedicados a la gestión cuantitativa de activos, difieren en sus métodos pero coinciden en la creencia que a través de la detección de tendencias pueden tener capacidad de predecir [71] futuras variaciones. Los analistas financieros (fundamentalistas) las buscan en los reportes financieros complementados con pronósticos macroeconómicos y econométricos [71]. Por su lado, el analista técnico (chartista) las busca examinando las series de tiempo [71] de los precios pasados, aplicando diversos métodos de análisis técnico y/o análisis de gráficos y tendencias. En tanto el gestor cuantitativo aplica los principios de la teoría financiera y técnicas cuantitativas [71] para construir y administrar carteras [22].

Es preciso mencionar que la forma débil de EMH descartaría el análisis técnico [71] como un método plausible dada la imposibilidad de predecir a partir de la información histórica de los precios, que es justamente la creencia de los chartistas. Por otra parte, la forma semi-fuerte de eficiencia entra en conflicto con los analistas financieros y los departamentos de estudios[71] ya que descarta la posibilidad de obtener mejores retornos que el mercado a partir de información pasada de los precios y de la información y estados financieros de las empresas.

Por último la forma fuerte de eficiencia entra en conflicto con la gestión cuantitativa de activos[71], ya que ésta descarta la posibilidad de obtener rendimientos superiores al mercado puesto que toda la información privada y pública estaría incorporada en el precio y, sin embargo los gestores de carteras creen que a través de sofisticados métodos de análisis pueden incrementar los rendimiento y además anticipar el futuro devenir de los mercados. Una nota de cautela en este último punto es la existencia de información privada, a la cual no se tiene acceso pues se trata de pronósticos de mercado y estudios que se manejan de manera confidencial por gestores profesionales.

El retorno de la inversión en estos mercados extremadamente competitivos, requiere contar con procedimientos de selección de objetivos de inversión entre un gran número de alternativas posibles, constituyéndose así en uno de los procesos de toma de decisiones más importantes y complejos, en lo que se refiere a la selección de instrumentos de inversión [14] en un escenario de abrumadora oferta de criptomonedas.

Ahora bien, los econometristas podrían argumentar que en un mundo en evolución es irracional tener expectativas racionales [38], pero es evidente la necesidad de contar con un proceso o metodología que permita al inversionista no especialista, por una parte seleccionar la o las criptomonedas con mayor potencial de rendimiento (objetivo de inversión), a través de un proceso formal, estable y semiautomatizado de decisión, que se vea potenciado por incorporación de pronósticos para las series temporales de cada criptomoneda analizada, cuyo fin último es determinar ex-ante las criptomonedas con mayor potencial de rendimiento para implementar sus iniciativas de inversión. Aún más, dado que los '*noise traders*' tienden a transar con información incompleta y errónea, se hace evidente la necesidad de diseñar, implementar y validar experimentalmente un método formal, estable y susceptible de ser automatizado para apoyar a estos inversionistas en la selección de criptomonedas para implementar su estrategia de inversión y que, a lo menos, presente mejor rendimiento que una estrategia de comprar y mantener (*buy-and-hold*) o que una estrategia aleatoria de compra y venta (*buy-and-sell*).

1.3 Discusión bibliográfica

Numerosos trabajos e investigaciones dan cuenta del gran interés que despierta esta área de investigación. En tales trabajos se comparan y/o propone aplicar variados métodos de econometría y estadística para el análisis de series temporales, algoritmos de machine learning y algoritmos de aprendizaje profundo, entre otras propuestas.

Numerosas investigaciones han enfocado sus esfuerzos en comparar variadas estrategias para predecir de manera más precisa el precio de una criptomoneda. Entre ellas está la implementación de modelos basados en el perceptrón multicapa (MLP), que se utiliza con mayor frecuencia en tareas de clasificación, por tratarse de una red neuronal de retroalimentación simple que se ve afectada por el desvanecimiento del gradiente [84]. Por el contrario, las redes neuronales *Long Short-Term Memory* (LSTM) como una arquitectura de red neuronal recurrente (RNN) ampliamente utilizada, han demostrado ser superiores a las RNN simples por su capacidad de recordar selectivamente patrones durante mucho tiempo, este es el caso de datos dependientes del tiempo como en los precios de mercado

de las criptomonedas [34] y en los problemas del área financiera [20]. Esta capacidad de reconocimiento de asociaciones de largo plazo las hace especialmente útiles en tareas secuenciales [7] como las relacionadas al reconocimiento de lenguaje natural y también en el análisis de series de tiempo, por lo cual autores como [44, 82, 84] destacan que resultan útiles en la implementación de estrategias cuantitativas, que descansan en modelos para el pronóstico de precios (problema de regresión) o el pronósticos de tendencias (problema de clasificación).

Las redes neuronales estocásticas para la predicción del precio de criptomonedas propuestas por Jay *et al.* [34], consisten en modelos estocásticos MLP y LSTM para predecir el precio de las criptomonedas Bitcoin, Ethereum y Litecoin. Un proceso estocástico corresponde a "conjuntos de variables aleatorias que evolucionan en el tiempo de manera arbitraria". Luego, si se considera un punto de inicio y_0, una caminata aleatoria se podría definir como la "ruta formada por una secuencia de pasos ξ_i IID (Independientes e Idénticamente Distribuidos)", que matemáticamente equivale a $y_t = y_0 + \sum_{i=1}^{t} \xi_i$. Con ello y luego de algunas operaciones simples, se llega a que el siguiente estado del sistema está dado por $y_t = y_{t-1} + \xi_t$, expresión cuyo cómputo resulta más simple que la precedente en la implementación del módulo estocástico que es clave en los modelos propuestos. El conjunto de datos considera tres fuentes de datos: estadísticas de mercado que incluyen las series temporales con los precios de apertura (O) y cierre (C) y el precio máximo (H) y mínimo (L), todo lo cual se resume como series temporales OHLC, además considera la información de la red de blockchain e información de sentimiento social como volumen de tweets y tendencias de Google. De acuerdo a los autores, en diversos escenarios de valores iniciales de los hiperparámetros, casi todas las versiones estocásticas de los modelos de redes neuronales experimentadas superaron a las versiones deterministas, cuantificando dicha mejoría en 1.56% en promedio en el caso de conjuntos de datos completamente normalizados y en 3.91% promedio para aquellos selectivamente normalizados.

Por su parte, Lahmiri *et al.* [41], implementan modelos de aprendizaje profundo para el pronóstico de precios de las criptomonedas Bitcoin, Digital Cash and Ripple, concluyendo que las redes neuronales LSTM superan significativamente a las redes neuronales de regresión generalizada (GRNN) empleadas en los mercados de criptomonedas, que además utilizan como *benchmark* en su investigación. Las puntuaciones en métrica RMSE para los modelos LSTM de Bitcoin, Digital cash y Ripple totalizan 2.75×10^3, 19.2923 y 0.0499 respectivamente versus 8.80×10^3, 50.2418 y 0.3115 respectivamente para los modelos GRNN. En contrapartida el entrenamiento de las LSTM requiere más tiempo. Las GRNN son en realidad un algoritmo de aprendizaje de un sólo paso con una estructura altamente paralela [63] que pueden ser utilizadas en problemas de regresión, predicción y clasificación, que además pueden ofrecer buenas soluciones en sistemas dinámicos en linea. Sin embargo, los autores señalan que estos modelos no lograron aproximar patrones ocultos no lineales globales y, dado que están basados en un kernel Gaussiano, resultan más adecuadas para la aproximación local de señales no estacionarias.

La investigación publicada por Yao *et al.* [84], presenta un método para la predicción del precio de una criptomoneda que utiliza técnicas de aprendizaje profundo, considerando para ello factores como capitalización de mercado, volumen, provisión circulante, y provisión

máxima de la moneda. En su propuesta, el flujo incluye cuatro capas LSTM, implementadas en Python y Keras con Tensorflow como backend, con un modelo de flujo secuencial compuesto por capas LSTM y capas densas. En esta configuración experimental, el entrenamiento consideró 10000 épocas y 5 tamaños de lote. Este modelo fue aplicado tanto a Bitcoin como a Litecoin. El conjunto de datos de Bitcoins incluyó las transacciones OHLC minuto a minuto en el periodo comprendido entre enero 2012 y marzo 2018, en tanto que el dataset para Litecoin comprendió el período 27 de diciembre de 2013 a 2 de junio de 2018. En opinión de los autores, esta investigación permite confirmar la aplicabilidad de las redes neuronales LSTM para producir predicciones precisas del precio de las criptomonedas, las cuales exhibieron MAE de 9.48 y RMSE de 15.84.

Pai *et al.* [48] reportan que los modelos ARIMA tienen peores propiedades de pronóstico que las redes neuronales, según los resultados de un análisis que involucró criptomonedas como Ethereum (ETH), Ripple (XRP), Litecoin (LTC), DASH, NEM (XEM), Siacoin (SC), Monero (XMR) y Stellar (STR). Por ello proponen una arquitectura RNN compuesta de 4 capas para la predicción del precio de cierre de las siguentes criptomonedas: Bitcoin (BTC), Ethereum (ETH), Litecoin (LTC) y Ripple (XRP). La primera capa corresponde a la capa de entrada. La segunda y tercera son capas LSTM, en ambos casos con dimensionalidad de (1, 256) en espacio de entrada y de 256 para espacio de salida con optimizador Adam y métrica MSE. La última es una capa densa cuya salida es la predicción. Las series temporales provienen de valores OHLC incluyendo volumen y capitalización de mercado, de las cuales se utilizaron precio de cierre y selectivamente el resto de las series temporales para la aplicación del modelo. Los resultados publicados muestran predicciones con errores mínimo en cada caso: 1.4925×10^{-4}, 6.0898×10^{-5}, 5.0990×10^{-5}, 2.4237×10^{-4} respectivamente. El paper no hace mención a las épocas de entrenamiento, ni a la aplicación de estrategias para selección y/o sintonización de hiperparámetros, tampoco hace referencia a la utilización de estrategias para controlar subajustes y sobreajustes.

Chen *et al.* [12] el Bitcoin ha ido ganando aceptación como activo de inversión gracias al auge de precios de las criptomonedas y, por ello contar con pronósticos es una necesidad en la implementación de decisiones de inversión. Los autores plantean un problema de clasificación binaria, en el cual se desea predecir si el precio de la criptomoneda cae o se incrementa, aplicando diversas técnicas de modelamiento según la dimensionalidad de características. Consideran precios a intervalo diario (baja frecuencia), es decir cada 60×24 mínutos y, en intervalos de 5 minutos (alta frecuencia). El análisis de los precios de baja frecuencia del Bitcoin, incluye mayor cantidad de características (dimensionalidad). Por su parte, los precios de alta frecuencia incluyen menos características. Sus resultados indican que la predicción asociada a precios de baja frecuencia de Bitcoin con métodos estadísticos, incluida la regresión logística y el análisis discriminante lineal (Linear Discriminant Analysis) logran una precisión del 66%, superando a los algoritmos de aprendizaje de máquinas más complicados que logran un 65.3%. Por el contrario, en la predicción asociada a precios de alta frecuencia del Bitcoin, los modelos de aprendizaje automático que incluyen Random Forest, XGBoost, Quadratic Discriminant Analysis, Support Vector Machine y redes neuronales LSTM, superan a los métodos estadísticos y su precisión alcanza el 67.2%. La alta dimensionalidad de características en los métodos estadísticos tiene por objetivo

evitar sobreajuste.

Por su parte Kumar *et al.* [40] analiza cómo las técnicas de aprendizaje profundo ayudan en la predicción de precios de Ethereum. Las técnicas consideradas en el estudio son: Perceptron multicapa (MLP) y las Redes neurales LSTM. Las métricas consideradas y descritas son MAPE (Mean Absolute Percentage Error), MAE, MSE y RMSE. Los datasets utilizados fueron obtenidos de los repositorios de coindesk y coinmarket. Los modelos fueron evaluados con datos de frecuencias diaria, hora y minuto. Los datos diarios cubren el período desde agosto 2015 a agosto de 2018 y corresponden a series temporales OHLC y volumen diario de la criptomoneda e incluyen 1200 días de precios. La frecuencia en base a hora corresponde al periodo que comprende julio de 2017 a agosto 2018 y consiste de 1500 muestras de datos. En frecuencia por minutos se incluyen 400 mil filas de datos entre el 01-01-2018 y el 30-10-2018. Los resultados muestran que el modelo de LSTM es más robusto y preciso para dependencias de largo plazo en comparación con MLP. Para conjunto de datos diario y modelo LSTM, las métricas MSE, MAE, MAPE y RMSE fueron: 0.018, 0.013, 3.67 y 20.53 respectivamente. En el caso hora fueron: 0.0130, 0.0072, 1.38 y 7.12 respectivamente y en el caso minuto fueron: 0.0024, 0.0014, 2.21 y 18.16 respectivamente.

Un estudio comparativo realizado por Ji *et al.* [36] que involucró desarrollo de modelos Deep Neural Networks (DNN), LSTM, CNN (Convolutional Neural Networks), ResNet (deep Residual Networks) y sus combinaciones para estimar el precio del Bitcoin del siguiente día, el conjunto de datos incluyó 2590 días de observaciones para un total de 18 características, entre las cuales se incluye el precio del Bitcoin en dólares, las cuales fueron seleccionadas mediante análisis de correlación. Para la predicción del precio del Bitcoin del siguiente día, se analiza una secuencia de tamaño m. El problema de clasificación relativo a determinar si el precio del siguiente día subirá o bajará utiliza como referente el precio actual. Es decir, si S es el conjunto de datos original, existirán un total de $2590-m+1$ secuencias, estos es desde $S[1:m]$ a $S[2590-m+1:2590]$. El primer 80% de ellas fueron reservadas para entrenamiento de los modelos y el resto se destinó a validación de los mismos. Por ello, la predicción del precio del Bitcoin para el $(i+m)$-ésimo día involucra la secuencia $S[i:i+m-1]$, en tanto que determinar si el precio subirá o bajará ese día se realiza respecto al $(i+m-1)$-ésimo día. En este estudio se validaron diversos escenarios, por ejemplo $m = 5, 10, 20, 50, 100$, diversos método de partición de los conjunto de datos, comparación de métodos de normalización. El método de inversión evaluado consistió en la estrategia de comprar todos los fondos en Bitcoin o mantener en caso de predicción al alza y vender en caso contrario o esperar. En sus resultados indican que los modelos LSTM superan ligeramente al resto de los modelos evaluados en el problema de regresión, particular para tamaños de secuencia 10, 20 y 50 con MAPE 3.96%, 4.46% y 6.68% respectivamente, no así en el problema de clasificación, en cuyo caso los modelos DNN superan ligeramente al resto de los modelos.

La investigación presentada por Dutta *et al.* [17] compara diferentes enfoques mediante métrica RMSE. Los modelos implementados para la predicción del precio del Bitcoin consisten en redes neuronales simples (NN), LSTM y GRU (Gated Recurring Unit), las cuales se entrenaron con hiperparámetros optimizados. Es interesante destacar que las GRU realizan menos operaciones con tensores, lo cual las hace más rápidas de entrenar que las LSTM. Se procedió a seleccionar el modelo con menor RMSE (Root Mean Squared Error)

para la implementación de la estrategia de inversión. El dataset incluyó 15 características, cuya selección incluyó pruebas de correlación y análisis del factor VIF (Variance Inflation Factor). El conjunto de entrenamiento considera observaciones entre 1-01-2010 y 30-06-2018. El conjunto de validación comprende observaciones entre 1-07-2018 y 31-12-2018. El conjunto de testeo comprende observaciones entre 1-01-2019 y 30-06-2019. Se evaluaron configuraciones de *loopback* de 15, 30, 45 y 60 días para predecir el precio del siguiente día, aplicando una tasa de *dropout* de 0.1. El modelo LSTM incluyó sólo una capa con 50 unidades ocultas y una capa densa con 1 nodo. La arquitectura GRU incluyó dos capas con 50 y 10 nodos respectivamente y un *dropout* de 0.1 y una capa densa con 1 nodo. Se implementaron dos estrategias de negociación (*trading*) para los resultados. El tamaño de lote óptimo se fijó 126 para las redes neuronales y en 100 para los modelos RNN. Como resultado se observa que los modelos RNN se desempeñan mejor que las NN tradicionales en la predicción del precio del Bitcoin. Asimismo para *loopback* de 30 días, la arquitectura GRU con y sin *dropout* superó ligeramente en testeo a LSTM en predicción de precios con un RMSE en test de 0.017 y 0.019 respectivamente v/s un RMSE en test de 0.024 para el modelo LSTM con sólo una capa oculta.

Wu *et al.* [73] en su estudio destacan que las redes neuronales LSTM son un método de punta en el aprendizaje de secuencias para el pronóstico de series de tiempo, sin embargo su aplicación al pronóstico de series temporales financieras y, especialmente en el ámbito de las criptomonedas aún se encuentra poco estudiado. Por ello, los autores proponen un nuevo marco de pronóstico basado en modelos LSTM, aplicable al pronósticos del precio diario del Bitcoin, para lo cual evalúan dos variantes de modelos LSTM. La primera variante, se trata de un modelo LSTM convencional. La segunda variante es un modelo LSTM complementado con AR(2) (modelo autoregresivo de segundo orden) que permite determinar las variables de entrada óptimas para construir el modelo de pronóstico de series de tiempo LSTM, sin un proceso de prueba y error y que se basa en la función del coeficiente de autocorrelación (ACF) y función de autocorrelación parcial (PACF). Cabe observar que este método sugerirá el valor del *time-lag* en función de la correlación que muestra la función de autocorrelación [64]. El rendimiento de los modelos propuestos se evaluó utilizando datos de precios diarios de Bitcoin desde el 1-01-2018 al 28-07-2018 con un total de 209 registros. Las métricas consideradas en las evaluaciones correspondieron a: MSE (Mean Squared Error), RMSE (Root Mean Square Error), MAPE (Mean Absolute Percentage Error) y MAE (Mean Absolute Error). El modelo LSTM consta de 1000 neuronas, una capa densa con función de activación lineal, la función de costo es MAE y optimizador "Adam" (Adaptive Moment estimation). El conjunto de entrenamiento quedó constituido por el 66% de las observaciones, 138 registros, en tanto que las restantes observaciones, 34% es decir 71 registros, conformaron el conjunto de validación. El entrenamiento se realizó en 300 épocas con un tamaño de lote definido en 72. Los resultados confirmaron que el modelo LSTM convencional es superado ligeramente por el modelo LSTM con AR(2) (Modelo Autoregresivo de orden 2). Los valores predichos por ambos modelos están cerca de los valores reales y la dirección en los cambios de tendencia también es muy consistente. Este estudio se enfoca en la predicción del precio de una única criptomoneda, aunque fácilmente se podría extender para considerar una muestra mayor.

El estudio realizado por Huisu *et al.* [30] analiza series temporales de precios de Bitcoin

a través de variables relacionadas con blockchain, información macroeconómica y tipos de cambio globales. Para ello adoptan un diseño con ventana móvil de longitud p, l.o cual en el caso de N muestras genera $N - p$ series temporales para entrenar modelos LSTM. Analizan tres casos de estudio. Caso 1 con precio del bitcoin como variable objetivo y métricas: MAPE y RMSE. Caso 2 considera logaritmo del precio del bitcoin como variable objetivo y métricas: MAPE y RMSE. Caso 3 con logaritmo de la volatilidad como variable objetivo y métrica RMSE. En cada uno de ellos analizan diferentes combinaciones de longitud de secuencia y tamaño de entrenamiento. Todas las series temporales corresponden a datos diarios entre septiembre de 2011 y agosto de 2017, con 2171 pasos de tiempo en total, a los cuales aplicaron preprocesamiento con escalamiento min-max como método de normalización. Los resultados obtenidos indican que con una (longitud de secuencia, longitud de entrenamiento) iguales a (200, 150) el modelo propuesto puede predecir el precio de Bitcoin y además supera a los restantes modelos comparados.

La principal limitación de los métodos de aprendizaje profundo para regresión es que no toman en cuenta la variabilidad de la predicción [7]. Motivo por el cual estos algoritmos son incapaces de evaluar qué tan seguros están en relación a sus predicciones, ya que estos modelos no predicen una distribución de probabilidades para la variable dependiente o un rango de posibles valores para la misma. Por ello, Brando *et al.* [7] propone una solución basada en modelos de aprendizaje profundo para resolver de manera efectiva la incertidumbre de la predicción a través del aprendizaje necesario tanto para producir la predicción y su varianza, realizando estimaciones de máxima verosimilitud del modelo resultante que contiene una red para generar predicciones y otra para la varianza de la predicción. Dicha investigación presenta el ajuste de redes tanto para los casos heterocedástico y homocedástico.

La línea de investigación referida a los modelos de mixtura y su aplicación en diversos problemas de clasificación y regresión, en las cuales componentes individuales del modelo se especializan en diferentes partes de los datos. El modelo puede adaptarse a datos no estacionarios gracias a los pesos derivados del conjunto de datos [28]. En su estudio concluyen que el modelo de mixtura temporal propuesto presenta las siguiente ventajas: Mayor precisión en la mayoría de de los casos en comparación a modelos convencionales de aprendizaje, permite interpretar el efecto del libre de órdenes en la volatilidad, el modelo es robusto y adaptativo respecto a datos en series de tiempo, podría utilizarse como un marco de trabajo genérico y flexible para pronóstico e interpretación aplicable al Bitcoin.

Una estrategia experimental que combina una red neuronal convolucional y una LSTM (CNN-LSTM) permite concluir que el modelo puede ser utilizado exitosamente en la creación de estrategias factibles, robustas y altamente rentables, superando los retornos de estrategias básicas como Momentum y Benchmark index [44]. En términos generales la configuración experimental considera: Entrenamiento CNN 500 épocas, LSTM 100 épocas; Función de activación: CNN ReLU, Tanh, LSTM Tanh; Optimizador Adam en ambos casos; Función objetivo: Cross-entropy en ambos casos. Los autores reportan que la red neuronal LSTM permite predecir precios futuros con alta precisión y que estas predicciones las utilizan como señales temporales para mejorar la selección de acciones del modelo CNN.

Para pronosticar la tasa de intercambio de divisas existen dos métodos fundamentalmente

diferentes. Uno de ellos es construir un modelo causal que describa las relaciones entre tasas de intercambio y otras variables macroeconómicas en relación a una teoría económica. En tanto que el otro enfoque se sustenta en el estudio de series de tiempo para hacer predicciones basadas en el procesamiento y análisis de observaciones pasadas. En este caso los modelos comunes son los modelos de series de tiempo ARIMA [55], GARCH [69] y redes neuronales. En este contexto, para realizar pronósticos de corto plazo, 5 a 30 días, se ha propuesto un modelo modificado de árbol binario autoregresivo (BART). BART combina algoritmos de clasificación clásicos, árboles de regresión y modelos autoregresivos ARIMA [15] para realizar predicciones en una cartera de critomonedas: Bitcoin (BTC), Ethereum (ETH) y Ripple (XRP). El diseño experimental consideró el cálculo de la serie de tiempo para los precios de cierre de estas criptomonedas en el período 01/01/2017 al 01/03/2019 a partir de datos de Yahoo Finance. Los pronósticos obtenidos con el algoritmo propuesto (BART) se compararon con los obtenidos a partir de parámetros específicos con ARIMA y ARFIMA. Los autores concluyen que el método BART muestra mayor eficiencia en comparación a las técnicas tradicionales de series de tiempo.

Para el caso de series temporales no estacionarias de criptomonedas se ha propuesto un marco de trabajo denominado RSM (Random sampling method). Este enfoque está inspirado en el campo del aprendizaje profundo y con él entrenaron redes para aprender a comparar secuencias de patrones pertenecientes a diferentes clases, lo cual demostró ser útil en el caso de las series de tiempo de criptomonedas por su carácter no estacionario [61]. Para comparar el rendimiento de RSM se utilizó como línea base un modelo MLP y una red LSTM, ambos con dos capas cada una con 32 unidades ocultas. Los autores reportan que este método muestra resultados que superan a línea base en todas las métricas.

La implementación de una red neuronal recurrente con optimización Bayesiana y una red LSTM superan los pronósticos con ARIMA (Autoregressive Integrated Moving Average). Los resultados mencionan que la implementación del modelo RNN con 20 nodos por capa es suficiente, puesto que demasiados nodos incrementan significativamente el riesgo de sobreajuste y también el tiempo requerido para entrenar la red. Tanh mostró buen rendimiento respecto a las otras funciones de activación. Se determinó que el tamaño del batch tiene poco efecto en la precisión sin embargo su efecto es considerable en el tiempo de entrenamiento cuando el tamaño del lote es muy pequeño. Se utilizó el método de Keras callback para detener el entrenamiento del modelo si la validación no mejora después de 5 épocas, ya que demasiadas épocas pueden resultar en sobre ajuste [46]. Se determinó que la red LSTM se desempeña pobremente en ventanas móviles pequeñas y que es más efectiva con una ventana móvil de 100 días con dos capas ocultas con 20 nodos ocultos. En este caso la convergencia se obtuvo entre 50 y 100 épocas con detención anticipada. Como conclusión los autores mencionan que los modelos RNN y LSTM son evidentemente más efectivos en predicciones en Bitcoin y que las redes LSTM desarrollan mayor capacidad de reconocimiento de dependencias de largo plazo.

Se ha realizado experimentación y comparación de los métodos de regresión Theil-Sen y Huber disponibles en la biblioteca scikit-learn que presentan diversas fortalezas ante la presencia de outliers con redes neuronales LSTM y GRU en Keras. Esta investigación muestra que los mejores resultados se obtienen con la implementación GRU, ya que exhibe

mejor precisión sin embargo supera con creces los tiempos de cálculo exhibidos por el algoritmo de regresión Huber que es el que menor tiempo de proceso exhibe. También se concluye que las características basadas en datos OHLC pueden ser insuficientes para predecir los precios del Bitcoin ya que en las alzas y bajas de la criptomoneda influyen múltiples factores tales como la reacción a las redes sociales, las políticas y leyes anunciadas por los países, etc. [52].

La investigación realizada por Torres et al. [65] propone un método ex-post para seleccionar la o las criptomonedas con mejor potencial de retorno en un proceso de inversión simulado con un enfoque de decisiones lingüísticas dinámicas (DLDM). Para ello, basaron su propuesta sobre el análisis de tres atributos preestablecidos, correspondientes a: **rentabilidad diaria**, calculada como la razón entre el precio de cierre y el precio de apertura de cada criptomoneda; **variación diaria**, calculada como la razón entre el precio mayor y el precio menor de cada criptomoneda y **capitalización de mercado** en logaritmo como apreciación de la fortaleza de cada criptomoneda en el mercado.

A partir de las publicaciones y estudios analizados en el estado del arte, es posible concluir que la mayoría de los estudios que consideran la utilización de redes neurales LSTM, se enfocan en la predicción del precio de una única criptomoneda, aunque eventualmente algunos de ellos se podrían extender para considerar una muestra mayor de criptomonedas. Es interesante destacar que algunas de estas investigaciones, adicionalmente analizan el problema de clasificación asociado al pronóstico de baja o alza del precio de la o las criptomonedas incluidas en dichos estudio. Sin embargo, en general se constata la ausencia de métodos para apoyar las decisiones de inversión en un escenario de múltiples criptomonedas y esa es una carencia crítica en todos estos estudios. Una excepción a esta situación se encuentra en el trabajo de Torres *et al.*, el cual por otro lado carece de componentes de predicción y sólo analiza una combinación de atributos definida a priori sin profundizar en los posibles efectos positivos o negativos que otras combinaciones de atributos podrían tener, es decir, no analiza el posible impacto de otras combinaciones de atributos ni el impacto que diversos esquemas de preprocesamiento podrían eventualmente tener sobre la generación del ranking dinámico.

1.4 Propuesta

Una vez seleccionado el mercado en que se desea o se tiene interés en operar, todo inversionista enfrenta la difícil decisión de seleccionar, a partir de un gran número de opciones, el o los instrumentos con mayor potencial de retorno para invertir en ellos. La abrumadora oferta disponible en el mercado de las criptomonedas, hace que este proceso de toma de decisiones sea aún más complejo. Es por ello que resulta evidente la necesidad de diseñar, implementar y validar experimentalmente un método susceptible de automatizar, para apoyar a todo el espectro de inversionistas en dichas decisiones, en función de las recomendaciones generadas por tal método. Esta necesidad es aún más relevante en el caso de los '*noise traders*' y de los inversionistas del segmento minorista en su conjunto.

Las series temporales OHLC además de la capitalización de mercado de cada criptomoneda incluida en la muestra, constituyen el insumo básico para el método propuesto

en este trabajo, que es construir el ranking dinámico utilizando un ensamblado de modelos de decisión lingüísticos predictivos, que consideren el pronóstico de cada variable de entrada para seleccionar la alternativa que entregue mayor potencial de retorno en el siguiente período de inversión, que en este trabajo se limita al día siguiente ya que la actualización de la decisión de inversión es diaria.

Observación, como ya se indicó OHLC representa los precios de apertura (Open), máximo (High), mínimo (Low) y cierre (Close), en este caso del precio de una criptomoneda en un día de negociación

Para evaluar los resultados obtenidos a partir de la aplicación del método propuesto, se comparará el retorno obtenido al final del período de inversión simulado aplicando este método, respecto al retorno obtenido tanto por un benchmark interno como por un benchmark externo. En el caso del benchmark interno, la comparación será contra el retorno obtenido por un benchmark de estrategias de inversión, las cuales consisten en: (1) una estrategia de comprar y mantener (*buy-and-hold*), tanto en un instrumento específico que en este caso es el Bitcoin, como en el mercado en su totalidad con inversión equitativa en cada criptomoneda de la muestra que compone dicho mercado y, (2) una estrategia aleatoria de compra y venta (*buy-and-sell*), por lo cual la selección de la criptomoneda en que se invertirá en en cada actualización diaria de la decisión no es producto de criterios formales o de un proceso modelado, sino que sólo depende del azar.

El benchmark externo corresponde a los retornos obtenidos al final del período de inversión simulado mediante la estrategia DLDM-1 implementada en el trabajo de Torres *et al.* [65]. De esta manera los resultados obtenidos en los diferentes escenarios de experimentación, luego de aplicar el método propuesto en esta tesis, serán evaluados contra el benchmark externo. Los escenarios a seleccionar corresponden a aquellos que eventualmente superen el retorno exhibido por este benchmark.

Complementando lo expuesto hasta ahora, la selección y aplicación de modelos LSTM para obtener pronósticos, en conjunto con la posterior combinación y preproceso de las series temporales complementadas con dichos pronósticos, hacen posible configurar datasets de atributos-preproceso para evaluar escenarios no contemplados en [65], con los cuales se espera potenciar el enfoque de agregación lingüística y ranking dinámico de criptomonedas por la incorporación de tales pronósticos. En este sentido la hipótesis de trabajo considerada en esta tesis plantea que:

Hipótesis

Generar un ensamblado de modelos para realizar el ranking de las monedas para el próximo día que utilicen como atributos de entrada combinaciones de los pronósticos de los atributos mediante el uso de redes LSTM sobre las series de tiempo de los atributos, permitirá en un periodo de inversión de varios días un retorno mayor que no usarlo.

El **objetivo general** de esta tesis es desarrollar y validar un método para apoyar durante un periodo de días la toma de decisión de inversión en la criptomoneda que entrega mejor retorno de la inversión con el fin de obtener un retorno igual o mejor que el estado del arte,

basado en un ensamblado de modelos linguísticos predictivos que usan los pronósticos de sus atributos para el siguiente día usando LSTM.

Para alcanzar este objetivo general se proponen los siguientes objetivos específicos:

- **[O1]** Configurar, entrenar y seleccionar redes neuronales LSTM con la arquitectura más simple posible y menor MAE promedio para cada atributo y cada criptomoneda incluida en este estudio para generar predicciones respecto a los posibles valores futuros en cada serie que luego serán utilizados en los procesos de agregación lingüística y ranking dinámico.
- **[O2]** Extender el proceso de toma de decisiones multi atributo y multi periodo con enfoque de agregación lingüística y ranking dinámico de criptomonedas propuesto en Torres *et al* [65] por la generación de un ensamblado de modelos a partir de las combinaciones de los pronósticos de los atributos utilizando métodos de escalamiento apropiado.
- **[O3]** Evaluar y cuantificar retorno generado por la inversión materializada a partir de las recomendaciones del ranking dinámico v/s el retorno obtenido en el proceso de inversión documentado en el benchmark externo [65] así como el logrado por el benchmark de estrategias.

1.5 Metodología

La figura 1 presenta la metodología general aplicada en el desarrollo de este trabajo. En ella se aprecian los siguientes procesos generales y flujo de los mismos: (1) Implementar modelos LSTM para cada criptomoneda y serie temporal OHLC y series temporales generadas, (2) Generar combinaciones de atributo-pronóstico con y sin preproceso y, finalmente, (3) Implementar el enfoque DLDM de decisiones lingüísticas dinámicas y, finalmente (4) Evaluar el rendimiento de estrategia de inversión respecto al benchmark de estrategias y respecto al benchmark externo.

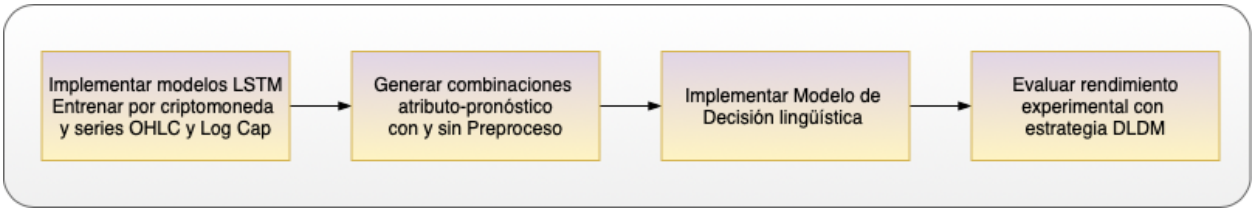


***Figura* 1:** Metodología general aplicada en este trabajo

Cada una de estas actividades macro es en realidad una vista de alto nivel de los procesos específicos inherentes a cada etapa de investigación, las cuales en detalle implican la aplicación de metodologías y diseños específicos para abordar la ejecución de las mismas.

Es por ellos que la aplicación de estos pasos metodológicos requiere la utilización coordinada de diversos pasos metodológicos, sin embargo, desde una perspectiva amplia la

metodología aplicada en este trabajo descansa en dos pilares fundamentales. El primero de ellos, según lo detalla la figura 2, se refiere al entrenamiento y selección de las arquitecturas y configuraciones de redes neuronales LSTM que exhiban el mejor rendimiento, considerando como función objetivo el error absoluto medio (MAE). Con este propósito se realizó la implementación y evaluación de un elevado número de modelos, uno por cada una de las 68 criptomonedas (mercado) y cada serie temporal OHLC, a las cuales se agregan las siguientes series temporales derivadas: Rentabilidad diaria (Day Profitability), Variación diaria (Day Variability) y Logaritmo de la capitalización diaria de mercado (log(Capitalization)). Este proceso involucra, a lo menos, las siguientes etapas:

- Selección de arquitecturas y configuraciones para los modelos de la red neuronal LSTM.
- Fijar la ventana móvil y los pasos de tiempo para configurar los datatests de entrenamiento, verificación y experimentos.
- Configurar los modelos por cada criptomoneda y serie temporal (OHLC y calculadas).
- Entrenar los modelos, monitorear MAE y seleccionar en cada caso el mejor modelo en el proceso de validación aplicando condición de término anticipado.
- Tabular la métrica MAE para cada modelo previamente seleccionado.
- Seleccionar los cinco modelos con el menor MAE promedio para la posterior generación de las predicciones y configuración de los datasets de atributos para los experimentos con el método DLDM.

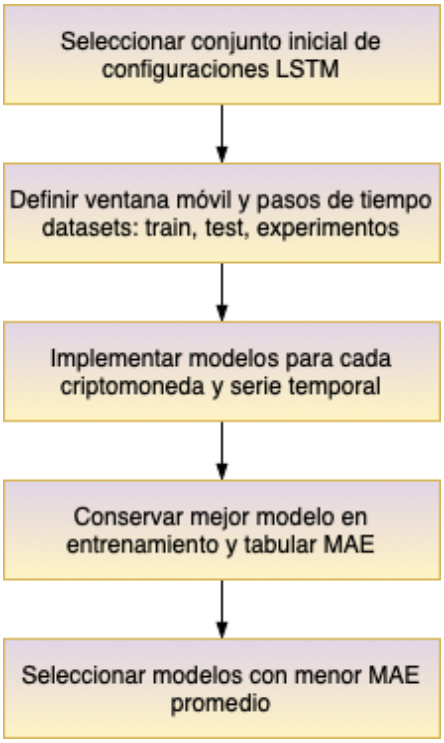


***Figura* 2:** Metodología para la selección modelos LSTM

El segundo pilar fundamental se detalla en la figura 3 y dice relación con el enfoque de decisión lingüística multiatributo y multiperiodo que conduce a la clasificación (ranking) dinámica de criptomonedas. En este caso particular, se realiza un elevado número de experimentos con todos los datasets posibles de tres atributos, preprocesados o no o con un atributo de valor fijo. Para cada experimento se tabula el rendimiento mínimo, máximo y rendimiento al final del periodo de inversión. Asimismo se tabula el día en que se obtuvo el rendimiento mínimo y máximo. Finalmente se seleccionan los experimentos cuyos datasets y esquema de preproceso producen el máximo retorno al final del periodo, en comparación al grupo y respecto al benchmark.

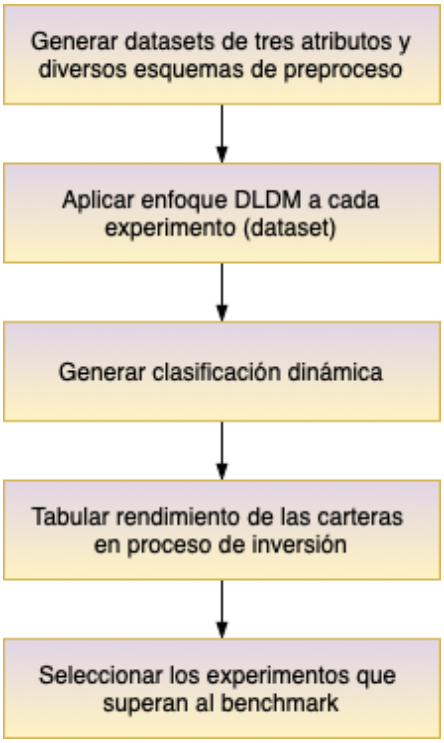


***Figura* 3:** Metodología aplicada en enfoque DLDM

1.5.1 Metodología para la implementación de modelos

El proceso de utilizar la tecnología asociada al aprendizaje automático para la predicción en criptomonedas conlleva el entrenamiento de modelos mediante grandes conjuntos de datos históricos (in-sample). Mediante el aprendizaje se espera lograr generalizar patrones para luego aplicarlos a datos no vistos (out-of-sample) con el propósito de expresar a través de la evaluación de la función objetivo la mejor predicción posible. Indudablemente, la meta última es la inferencia de señales de trading a partir de los indicadores del mercado para anticipar retornos futuros de los activos bajo seguimiento [20]. Con estas predicciones se intenta anticipar resultados out-of-sample de tipo numérico, como por ejemplo el precio, o bien de tipo categórico como por ejemplo tendencias.

Para poner en contexto el desarrollo del modelo, es necesario considerar que este proceso se materializa a partir de las características aportadas por los inputs del proceso y la función objetivo. La definición de las características (inputs) requerirá conocimientos de análisis fundamental y técnico. Con ello las fuentes de datos se podrían estructurar en múltiples grupos, según corresponda a los indicadores, por ejemplo: Económicos (como por ejemplo producto interno bruto, tasa de interés, inflación), Sociales (como por ejemplo evolución y tendencia sentimental en redes sociales), Técnicos (como por ejemplo precio, volumen, OHLC), Estacionales (como por ejemplo hora del día o día de la semana), etc.

Finalmente a través de la función objetivo se establece el criterio para determinar si el modelo ha aprendido la tarea bajo análisis. La siguiente figura detalla el flujo del proceso

descrito:



***Figura* 4:** Proceso ML en predicción de criptomonedas [20]

Desarrollar el modelo de acuerdo al flujo presentado requerirá la ejecución de un conjunto de actividades que a nivel de detalle implican tomar decisiones sobre los datos y su transformación hasta obtener las features que alimentarán el modelo. Una vez seleccionado el modelo, será necesario definir u obtener a partir de un optimizador los parámetros que ajustarán el proceso de aprendizaje. Una vez superadas las fases previas corresponde tomar las decisiones necesarias para el despliegue del modelo en condiciones de predecir valores out-of-sample. Desde una perspectiva de alto nivel estas fases son las siguientes:

Fase análisis y preparación de datos

El análisis y preparación de datos implica extraer y preparar los datos que formarán el dataset, por otro lado la normalización se refiere al proceso de escalamiento de datos mediante alguna de las técnicas disponibles para ello y, su propósito es evitar que el rendimiento de los modelos se vea afectado por la variación en el rango de los datos o por la dispersión de los mismos. Finalmente se seleccionan las features a utilizar en los modelos.

Fase elección y desarrollo de modelos

En esta fase es necesario seleccionar el o los modelos a desarrollar, obtener los conjuntos de entrenamiento y verificación, adicionalmente se podrían activar sucesivos procesos de optimización de hiper parámetros. En función del resultado del proceso de evaluación podría ser necesario repetir el proceso de optimización.

Fase despliegue del modelo

Finalmente, el o los modelos entrenados quedan en condiciones de ser utilizados en procesos de predicción, lo cual culmina con la obtención de métricas de rendimiento.

Los métodos hasta aquí descritos han sido presentados de manera más bien general y para operacionalizar los aspectos más relevantes del proceso mediante el cual se obtendrá el o los modelos que finalmente serán utilizados en la etapa de predicción. En este caso se tratará de modelos de redes neuronales LSTM, por lo cual diversas metodologías de machine learning son también aplicables en este escenario. Es por ello que se han seleccionado las metodologías y técnicas que se describen a continuación.

1.5.2 Knoledge Discovery y Data Mining

El campo del descubrimiento de conocimiento en bases de datos se desarrolló debido a la creciente necesidad de técnicas y herramientas para asistir en la extracción de información útil a partir de crecientes volúmenes de datos. En el paper que dio inicio a este marco de trabajo se define KDD como el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y comprensibles en los datos [21]. DaimlerChrysler, NCR y SPSS desarrollan el Cross Industry Standard Process for Data Mining (CRISP-DM) método que se convierte en el estándar para la aplicación de minería de datos al ámbito de los negocios [70]. También se extiende este método a una perspectiva empresarial de outsourcing en un modelo de tres etapas [70].

El proceso KDD se caracteriza por su naturaleza interactiva como se aprecia en la siguiente figura.

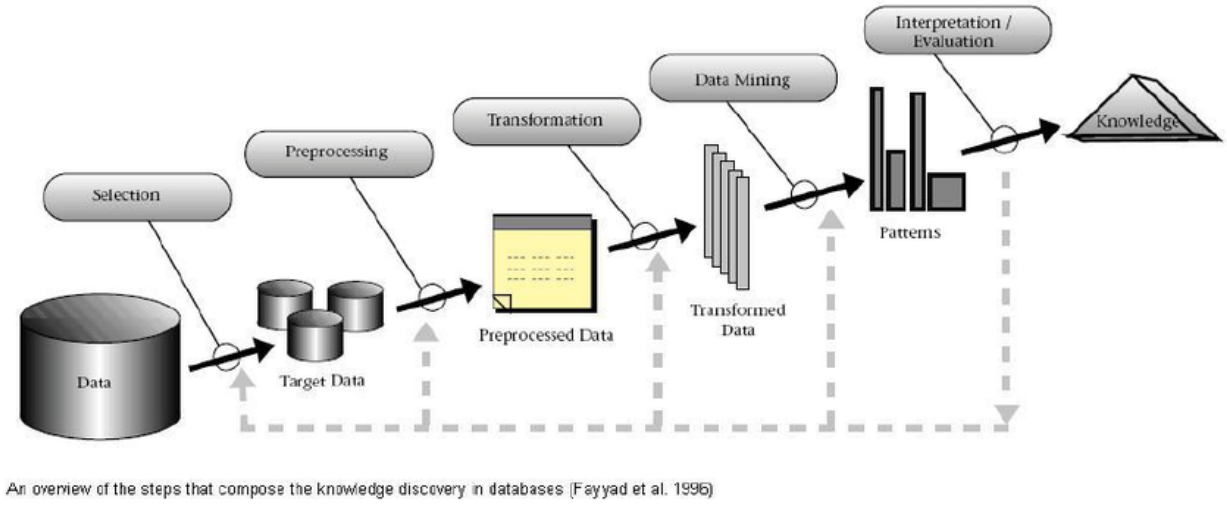


***Figura* 5:** Proceso KDD [21]

La descripción detallada de las etapas básicas de este proceso se encuentra en las referencias [21] y son las que se enumeran a continuación:

- Comprender dominio y objetivos
- Crear el dataset de trabajo
- Limpieza y preprocesamiento de los datos
- Reducción y proyección de datos
- Seleccionar método de minería de datos según objetivos
- Interpretar patrones minados

- Consolidar conocimiento descubierto

En el descubrimiento de información se distinguen objetivos de **verificación** de las hipótesis, **predicción** de la conducta futura de algunas entidades y **descripción** en que el sistema elabora y comunica los patrones para consumo humano.

Los objetivos de predicción y descripción se desarrollan a partir de los siguientes métodos primarios: clasificación, regresión, clustering, sumarización, modelado de dependencias (significantes entre variables) y detección de cambios y desviaciones.

1.5.3 Metodología de Data Science

En el dominio de la ciencia de datos, la resolución de problemas y la respuesta a preguntas mediante el análisis de datos es una práctica estándar. A menudo, los científicos de datos construyen un modelo para predecir resultados o descubrir patrones subyacentes, con el objetivo de obtener conocimientos. Luego, las organizaciones pueden utilizar estos conocimientos para tomar acciones que idealmente mejoren los resultados futuros [2].

La metodología presentada por IBM [2] consta de 10 etapas que forman un proceso iterativo para descubrir conocimientos a partir de los datos . Cada etapa juega un papel vital en el contexto de la metodología general que se aprecia en la siguiente figura.

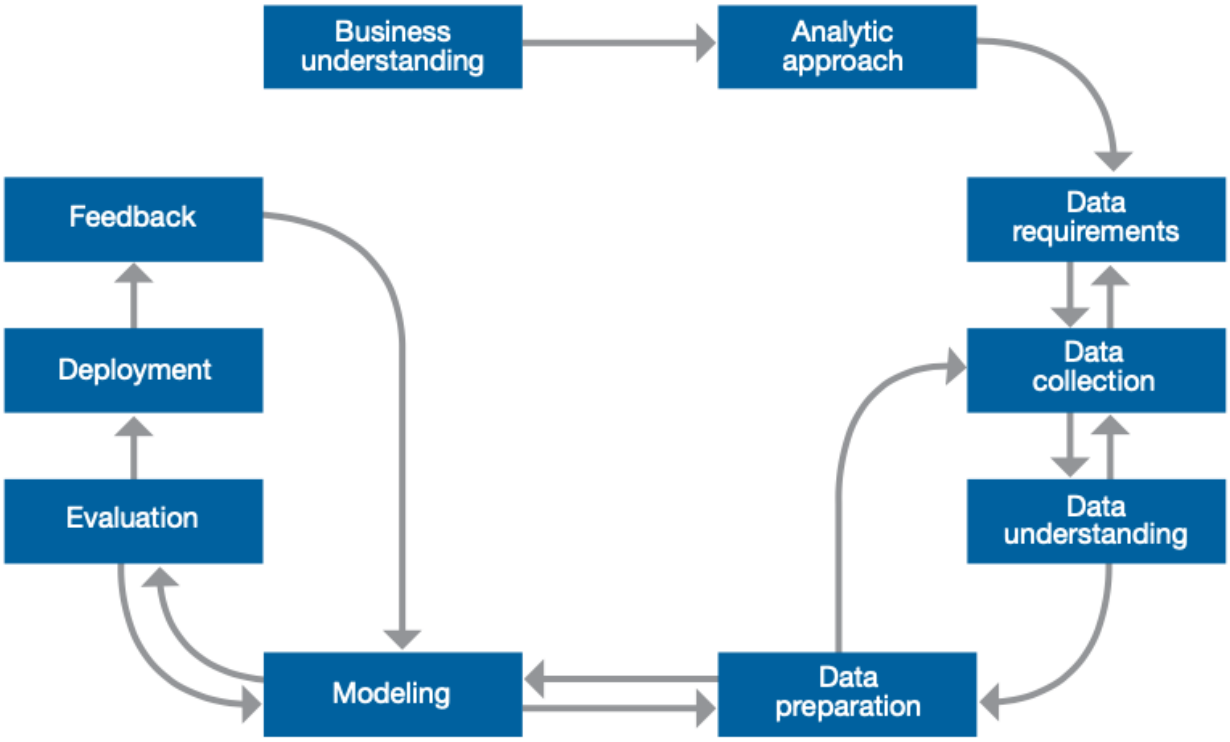


***Figura* 6:** Metodología de data science [2]

La descripción detallada de las etapas de esta metodología se encuentra en las referencias [2] y son las que se enumeran a continuación:

- Comprensión perspectiva empresarial
- Enfoque analítico
- Requerimientos de datos
- Recolección de los datos
- Comprensión de los datos
- Preparación de los dato
- Modelamiento
- Evaluación
- Despliegue
- Retroalimentación

1.5.4 Feature Engineering

Es el proceso de utilizar el conocimiento del dominio de los datos para crear características que hacen que un algoritmo de aprendizaje funcione. Para trabajar con algoritmos de aprendizaje predictivo o clasificación, los datos de entrada deben transformarse a los formatos necesarios para que el algoritmo los entienda y proporcione la clase precisa a la que pertenece una entidad y, de esa manera tener la capacidad de generar la predicción futura. Por lo general, es la parte que más tiempo consume en la construcción de clasificadores [54].

Las tareas principales de este proceso son la construcción de características (features) relevantes adicionales a partir de los conjuntos de características existentes y luego la selección de características para descartar aquellas redundantes, irrelevantes o altamente correlacionadas. Las características seleccionadas corresponden a un conjunto mínimo de features independientes que explican los patrones en los datos y permiten predecir y/o clasificar con éxito [54]. Este proceso se aprecia en la siguiente figura.

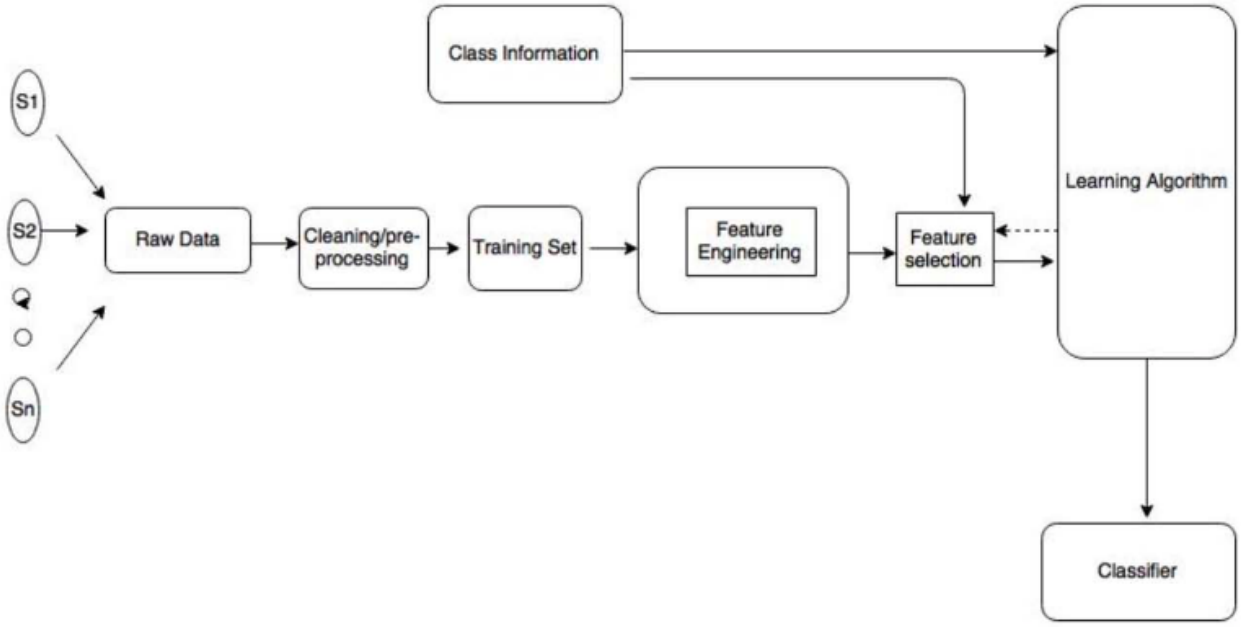


***Figura* 7:** Feature engineering [54]

Conforme a lo planteado por los autores y según se aprecia en la figura, la ingeniería de características (features) se realiza después de la depuración y preparación de datos, antes de entrenar el modelo. El objetivo principal es proporcionar una mejor representación de los datos de manera que el algoritmo tenga éxito en el aprendizaje predictivo [54].

La descripción detallada de los pasos de esta proceso se encuentra en las referencias [54] y son los que se enumeran a continuación:

- Recolección de datos en bruto
- Preprocesamiento de datos
- Ingeniería de características (features)

  Los autores destacan que a pesar que los datos se encuentren limpios y preprocesados aún están en bruto ya que contienen gran cantidad de datos inutilizables, lo cual requiere su transformación para crear features útiles para el modelamiento. Algunos de los métodos aplicables en este caso son: PCA, Kernel PCA, Discretización, auto encoding, suavizado, etc.
- Selección de características
- Modelamiento y medidas de rendimiento
- Clasificación

1.6 Contribución del trabajo

1.6.1 Enfoque DLDM

Como ya se mencionó, una propuesta interesante se encuentra en el enfoque dinámico de decisión lingüística para un escenario de inversión en criptomonedas [65], el cual se basa en un enfoque de toma de decisiones lingüísticas dinámicas (DLDM) para la construcción de modelos de decisión que extiende el enfoque MP-MADM, cuyo objetivo es presentar un método de selección de criptomonedas como objetivos de inversión en función de un ranking dinámico de las mismas. Sin embargo, este trabajo presenta un enfoque más bien ex-post, que carece de componentes de predicción para acceder a estimaciones con potencial de soportar un proceso de toma de decisiones de inversión, en función a estimaciones del rendimiento esperado para cada criptomoneda en cada etapa del proceso de inversión. Por otra parte y como ya se mencionó, esta implementación asume un enfoque que descansa en una combinación arbitraria de atributos, definida a priori y no analiza el posible impacto de otras combinaciones de atributos ni el impacto que diversos esquemas de preprocesamiento pueden tener sobre la generación del ranking dinámico.

El potencial del método de toma de decisiones presentado en [65], resulta suficientemente relevante para la implementación de procesos de toma de decisiones de inversión en criptomonedas y podría verse potenciado al implementar soluciones especificas para subsanar las observaciones y deficiencias ya mencionadas, que en definitiva corresponde al objetivo de esta tesis.

Precisamente, debido a su potencial, esta publicación [65] y sus resultado en el proceso de inversión simulado implementado en la misma, se utilizan como benchmark, contra el cual se compararán los resultados experimentales que se obtendrán con el desarrollo de esta tesis.

1.6.2 Contribución esperada

Por ello la contribución que se espera realizar con este trabajo es proponer un proceso articulado, cuyo objetivo final es la toma de decisiones basadas en datos, a través de un método formal y estructurado que articula: (a) Aplicación de modelos predictivos basados en redes neuronales LSTM para generar pronósticos para las series temporales OHLC de cada criptomoneda, (b) Generación de datasets de atributos específicos con aplicación o no de técnicas de ingeniería de features para alimentar el proceso de toma de decisiones con enfoque DLDM, todo lo cual permite (c) Configurar un método cuantitativo a través del cual es posible implementar una estrategia de inversión basada en las recomendaciones ex-ante, obtenidas desde el ranking dinámico de criptomonedas que se espera mejore los resultados, respecto al benchmark externo y respecto al benchmark interno de estrategias inversión en un proceso de inversión simulado.

En relación a los modelos predictivos, las contribuciones de este trabajo dicen relación con haber implementado un gran número de modelos de redes neuronales LSTM basadas en múltiples arquitecturas y la evaluación de los mismos para cada criptomoneda y serie temporal tanto OHLC como generada durande el proceso. Este proceso de evaluación arrojó

como resultado un método, a través del cual es posible comparar y seleccionar diversas implementaciones de modelos de redes neuronales LSTM, ya sea para para seleccionar modelos genéricos en un mercado de criptomonedas en función del MAE promedio o bien, para optar por el camino más complejo de seleccionar el mejor modelo para cada criptomoneda y serie temporal. Estas evaluaciones se realizaron en los siguientes escenarios:

- Arquitecturas de modelos con una y dos capas ocultas LSTM.
- La primera capa oculta puede contener 128, 64 o 32 unidades ocultas.
- En el caso de las arquitecturas con dos capas ocultas, la segunda contendrá la mitad de unidades ocultas presentes en la primera capa oculta.
- En el caso de una capa oculta, el proceso de entrenamiento consideró detención anticipada en: 50 y 100 épocas respectivamente.
- En el caso de dos capas ocultas, el proceso de entrenamiento consideró detención anticipada en: 5, 10, 20 50 y 100 épocas respectivamente.

En cada uno de estos escenarios se entrenó el modelo con un número potencial de 500 épocas para cada criptomoneda y serie temporal, con monitoreo y tabulación del MAE en validación y selección del mejor modelo para cada caso.

En relación a la generación de los datasets de atributos, la principal contribución dice relación la aplicación de técnicas de ingeniería de features que probaron contribuir positivamente en la estructuración de datasets de atributos que mejoran el proceso de clasificación dinámico del enfoque DLDM propuesto en el benchmark externo. Esta intervención a nivel de atributos probó tener capacidad de mejorar notoriamente los resultados de la clasificación dinámica, a la vez que permitió seleccionar la técnica con mejores resultados comparativos.

En cuanto a la estrategia de inversión basada en las recomendaciones generadas por el ranking dinámico, generado para todas las posibles combinaciones de tres atributos, los resultados muestran que este enfoque permite seleccionar datasets de atributos-preproceso que generan mejores resultados en el proceso de inversión simulado en comparación al benchmark externo. De igual manera, estos resultados superan ampliamente a la estrategia de selección aleatoria (Estrategia Random) de objetivos de inversión (criptomonedas) en un esquema 'buy-and-sell' asimismo, supera a las estrategia de inversión 'buy-and-hold' ya sea centrada en un instrumento particular (Estrategia BTC) o bien en el mercado en su totalidad (Estrategia Whole).

2 Marco Teórico

2.1 Linguistic Decision model o MCDM

Los enfoques de toma de decisiones con múltiples atributos son un aporte valioso en la evaluación y clasificación de alternativas en base a múltiples criterios. Uno de los métodos de toma de decisiones mejor conocidos es MCDM (Multi-Criteria Decision-Making) [5][13][31] también denominado MCDA(Multi-Criteria Decision-Analysis)[85], el cual consiste en la evaluación y clasificación de alternativas en función de múltiples criterios en conflicto, sistematizando un proceso de toma de decisiones basado en datos, racional y eficiente para seleccionar la mejor opción posible en un escenario de múltiples opciones, criterios y prioridades [47]. Es una metodología ampliamente aceptada, abundantemente aplicada, con disponibilidad de paquetes de software, profusamente citada en publicaciones científicas [1] y cuenta con un abundante número de libros sobre la materia [23].

Atendiendo fundamentalmente al número de alternativas en consideración [31][37][66][86], es posible clasificar[1] estos métodos en: MADM (Multi-Attribute Decision-Making) y MODM (Multi-Objective Decision-Making)

Típicamente un enfoque MADM[31] permite evaluar un conjunto de alternativas en base a múltiples criterios[51], es decir es un enfoque de toma de decisiones que usualmente se aplica sobre un número finito[78] y predeterminado de alternativas [26][83] en espacios de decisión discreto y su foco es seleccionar y clasificar [51] alternativas predeterminadas finitas. Se ha utilizado exitosamente en todos los aspectos de la vida organizacional para el análisis de miles de decisiones[13]. Incluye una amplia gama de métodos y técnicas dependiendo de las características del problema a resolver, entre ellas se encuentran: métodos basados en puntuación obtenida a partir de la ponderación de criterios como SAW[67] y COPRAS[53], métodos jerárquicos como AHP [59][67] y MACBETH[5], métodos basados en la distancia a solución ideal (similitud) y no ideal como TOPSIS [31][67] y VIKOR [59][67] y, toda la familia de métodos de outranking de clasificación de preferencias como PROMETHEE[5] y ELECTRE[5], estos últimos métodos son particularmente interesantes pues tienen la capacidad de tratar información incompleta y difusa [53]. Dada la complejidad de los problemas que suele presentar la realidad, se ha planteado que estos enfoques en su perspectiva convencional carecen de un paradigma para problemas específicos del mundo real, como serían decisiones grupales con preferencias inciertas [67].

En contraste, el enfoque MODM se orienta a espacios de decisión continuo, en los cuales se busca la mejor solución de acuerdo a funciones objetivo e involucra un número infinito o muy grande de alternativas que se definen implícitamente a través de reglas [83]. Resolver estos problemas a través de métodos clásicos de optimización involucra modificarlos para el caso multi objetivo. Los modelos de programación matemática son, en este caso, el enfoque mejor conocido, pero también lo son los numerosos enfoques de inteligencia artificial [31] en los cuales encontraremos todo tipo de métodos de optimización multi objetivo como: heurísticas y meta-heurísticas expresadas como algoritmos inspirados en la naturaleza[81] (algoritmos evolutivos, optimización por colonias, optimización por enjambres y algoritmos genéticos, algunos ejemplos serían Tabu-search y Simulated-annealing), redes neuronales y

lógica difusa entre otras alternativas.

Estos métodos tienen en común las siguientes características: (1) un conjunto cuantificable de objetivos, (2) un conjunto bien definido de restricciones y (3) un proceso implícito o explícito [83] para compensar información entre los objetivos cuantificables declarados y también entre objetivos no cuantificables declarados o no [26]. El contexto en que se aplican estos métodos supera a los problemas de selección que se abordan con MADM y se orientan mejor a problemas de diseño.

Los problemas de la vida real a menudo involucran criterios interdependientes y por lo mismo requieren métodos interdependientes para su solución. La tendencia ha evolucionado desde clasificar o seleccionar la alternativa de mayor preferencia a mejorar su desempeño para alcanzar el nivel que se aspira lograr, es por ello que el método Hybrid Dynamic MADM [51] intenta eliminar las limitaciones de los métodos MADM convencionales para dar respuesta a la necesidad de resolver problemas del mundo real que se caracterizan por ser complejos, dinámicos e interdependientes. Para ello combina DEMATEL [66] que es una técnica analítica para construir un modelo estructural que calcula relaciones causa efecto para dimensionar la relación de influencia entre criterios con ANP (Analytic Network Process) [66] para obtener ponderaciones influyentes más precisas y, finalmente aplica VIKOR [66] para calcular cercanía con la solución ideal.

La utilización de operadores de agregación [77, 85], agregación lingüística [65, 75] y conjuntos difusos y los métodos MP-MADM [77] son herramientas centrales a la hora de tomar decisiones sobre selección de uno o más instrumentos de inversión a partir de un gran número de alternativas posibles. También se ha reconocido que los problemas de selección financiera deben incorporar operadores de agregación para mejorar el proceso de modelado [85]. La toma de decisiones en múltiples períodos con múltiples atributos (MP-MADM), es definida [77] como el proceso en el cual la información, incluyendo los pesos y valores de los atributos, es proporcionada a los tomadores de decisiones en diferentes períodos. De esta manera introduce también el operador Dynamic Weighted Averaging (DWA), presentando los métodos aritméticos aplicables en su cálculo, conocidos como: (1) Método basado en series aritméticas (Arithmetic series based method), (2) Método basado en series geométricas (Geometric series based method) y (3) Método basado en distribución normal (Normal distribution based method), los cuales no se detallan ya que su formulación matemática esta disponible en [77].

Concretamente, la formulación del enfoque MP-MADM considera un conjunto discreto de alternativas factibles, un conjunto finito de atributos y un proceso de normalización de la matriz de decisión para permitir la comparación inter-atributo que finalmente permitirá clasificar las alternativas. A partir de este trabajo y para extender el método MP-MADM, se presenta el enfoque de decisiones lingüísticas dinámicas DLDM (Dynamic Linguistic Decision Making) [65] cuyo resultado corresponde a la clasificación de alternativas de acuerdo a la puntuación obtenida en un proceso diario de agregación lingüística. En este contexto en [16] se indica que una variable lingüística es aquella que puede tomar por valor algún término del lenguaje natural y que desempeñan el papel de etiquetas en un conjunto difuso.

La mayoría de las plataformas y servicios en línea se han enfocado en la utilización del análisis técnico como método para estudiar y determinar patrones en el historial de

transacciones de las criptomonedas, lo cual es visto como método práctico para identificar y seleccionar objetivos de inversión. Pero, sin duda una de las tecnologías emergentes que ha ganado notoriedad en este ámbito es el aprendizaje automático, que comprende por ejemplo algoritmos de: clasificación, clustering, regresión, aprendizaje profundo [84] y aprendizaje por refuerzo [28] entre otros. Existe abundante investigación relacionada, incluyendo por cierto el análisis de sentimiento social [11] como factor que puede ayudar a identificar tendencias de mercado para la generación de estrategias de trading [28, 35, 68].

2.2 Métodos de pronóstico

Unos de los aspectos estudiados en este trabajo tiene relación con los métodos de pronóstico, por su relevancia para la formulación de una metodología aplicable a la predicción de precios futuros. Aunque en este caso interesa realizar predicciones en los mercados de criptomonedas, es preciso reconocer que en este tipo de aplicaciones la experiencia mayoritariamente se relaciona a mercados financieros y de renta variable como los accionarios. Para analizar las características generales y determinar la aplicabilidad de estos métodos, los agruparemos en métodos de tipo cualitativo y métodos de tipo cuantitativo.

Desde una perspectiva amplia y general, en los métodos cualitativos encontraremos aquellos basados en paneles de expertos, quienes en función de conocimientos especializados, experiencia y, principalmente por voto de mayoría arriban a un pronóstico (por ejemplo método Delphi [32]).

Evidentemente, los métodos cuantitativos son el foco principal de este trabajo, puesto que proveen una aproximación algorítmica al problema de la generación de pronósticos, estimaciones y predicciones, por lo cual es necesario un conocimiento general sobre ellos y los métodos estadísticos, matemáticos y elaborados algoritmos aplicados para arribar a tales pronósticos. En este grupo analizaremos de manera muy general los métodos de econometría, por su relevancia en el estudio de las series temporales financieras y su aplicación al problema de la predicción del siguiente paso en la serie, en los cuales el enfoque es más bien recursivo. Adicionalmente se describirán los métodos basados en aprendizaje, en particular los de tipo supervisado, capaces de aprender a partir de una muestra previa de datos y resultados históricos, en los cuales es posible considerar toda clase de algoritmos de aprendizaje de máquina y, por cierto los métodos de regresión, además de los más sofisticados basados en redes neuronales y aprendizaje profundo. En este último caso, las redes neuronales de tipo LSTM (Long Short-Term Memory) corresponden al método a partir del cual se obtendrán las predicciones que serán el insumo inicial del proceso de ranking dinámico de criptomonedas que se aplica en este trabajo. Es necesario recordar que las redes neuronales LSTM son un tipo especial de red neuronal recurrente (RNN), capaces de aprender dependencias a largo plazo ya que pueden retener información por lagos períodos de tiempo y han demostrado ser extremadamente útiles en el aprendizaje de secuencias y series.

2.3 Pronósticos en series temporales financieras

El problema de la toma de decisiones en entornos financieros y en mercados de renta variable es objeto de extensas investigaciones, cuyo objetivo es modelar matemáticamente el proceso de generación de pronósticos a partir del comportamiento reflejado en los datos históricos de la o las variables estudiadas (instrumentos de renta variable, criptomonedas, etc.). Como una nota de cautela, es preciso recordar que la forma débil de EMH [8, 18, 42, 71, 72] descarta la posibilidad de obtener predicciones de precios futuros a partir de los precios históricos, ya que el comportamiento de los retornos seguiría un modelo de trayectoria aleatoria (Random walk) [60, 71, 72], de lo cual se desprende que si los precios siguen una trayectoria aleatoria es difícil predecir su trayectoria futura y, por tanto, un inversionista no podría superar una estrategia buy-and-hold. Bajo este criterio se acepta que una estrategia pasiva está a destinada a vencer a una gestión activa. Estos postulados son frecuentemente desafiados y puestos a prueba en diversos estudios en los cuales se analizan mercados específicos en busca de confirmación o pruebas que permitan refutarlos [***citas**]. Por otra parte, es interesante constatar que los profesionales de las finanzas no adhieren completamente a los postulados de la EMH [9], entre otras razones porque de ser completamente efectiva, el enfoque cuantitativo no sería posible.

Las series temporales en entornos financieros han sido profusamente estudiadas, por lo cual existe una amplia batería de métodos estadísticos y técnicas de econometría para la obtención de pronósticos a partir de ellas. Dada la relevancia de las series temporales, se ha considerado necesario mencionar los principales métodos econométricos y las condiciones que deben cumplirse para su correcta aplicación, todo lo cual permitirá concluir que las redes neuronales recurrentes de tipo LSTM son una alternativa válida y práctica en la generación de predicciones a partir de series temporales. El siguiente diagrama resume la metodología genérica presentada en [49] para el análisis de series temporales, el cual utilizaremos como referencia para esta parte de la discusión.

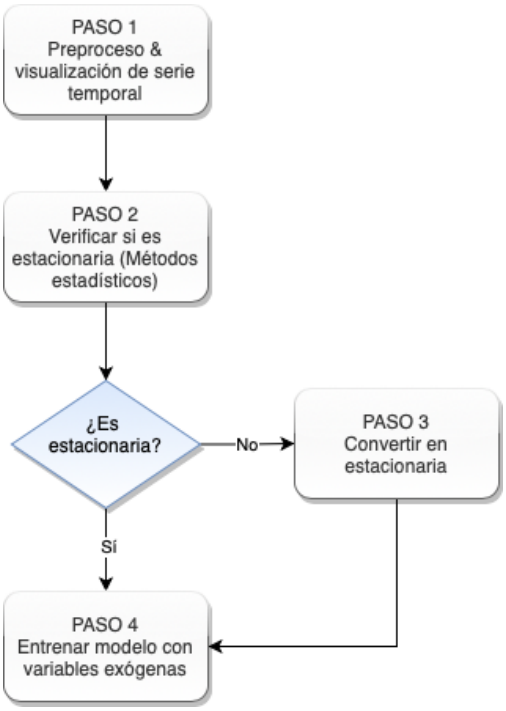


***Figura* 8:** Metodología genérica para análisis de series temporales

Variable exógena: variables que afectan un modelo sin ser afectadas por este.

2.3.1 Serie temporal financiera

Una serie de tiempo $x_1, x_2, ...x_t, x_{t+1}, ...x_n$ es una secuencia de observaciones indexada al tiempo, en la cual cada término corresponde a un valor numérico, como podrían ser series de precios OHLC ya sea que se trate de acciones o criptomonedas. La estructura interna de una serie temporal podría exhibir uno o mas de los siguientes efectos:

- Tendencia (f)
- Estacionalidad (s)
- Variaciones cíclicas (c)
- Variaciones inesperadas o aleatorias (e)

Considerando lo anterior, una serie temporal podría expresarse ya sea de manera aditiva como la suma de dichos componentes internos: $x_t = f_t + s_t + c_t + e_t$ o de manera clásica como un método multiplicativo: $x_t = f_t \times s_t \times c_t \times e_t$, el cual a su vez se puede convertir en un modelo aditivo de logaritmos de manera que: $\log(x_t) = \log(f_t) + \log(s_t) + \log(c_t) + \log(\epsilon_t)$, con t una variable indexada al tiempo de manera que $t = 1, 2, 3 \ldots n$ representa observaciones sucesivas e igualmente espaciadas en el tiempo. De esta manera el objetivo del análisis es descomponer la serie temporal para desarrollar modelos matemáticos específicos para cada componente.

2.3.2 Estacionariedad

En el análisis de series temporales es crucial, por una parte verificar estadísticamente si una serie temporal es o no estacionaria y convertirla en estacionaria mediante operaciones matemáticas especificas, en caso de no serlo.

Aunque la mayoría de las series interesantes no son estacionarias, ARIMA es eficaz siempre y cuando la serie se pueda convertir en estacionaria mediante la aplicación de transformaciones tales como el logaritmo natural, la diferenciación o la diferenciación estacional.

Un proceso estocástico Y_t puede ser estrictamente estacionario o débilmente estacionario. La estacionariedad estricta indica que todos los momentos de dicho proceso son independientes del tiempo, lo cual resulta ser extremadamente riguroso. En la estacionariedad débil sólo los dos primeros momentos son independientes del tiempo, es decir es constante en media y varianza, por lo cual es más sencillo encontrar modelos que expliquen su autocorrelación, en particular a través de la rentabilidad cuando se trata de series de precios.

En este sentido, se considera que una serie es estacionaria si la media y la varianza se mantienen constantes a lo largo del tiempo. Por el contrario una serie es no-estacionaria si la media y la varianza cambian a lo largo del tiempo. En este caso, la serie puede:

- Mostrar cambios de varianza
- Mostrar una tendencia, entendiendo por ello que la media aumenta o disminuye a lo largo del tiempo
- Presentar efectos estacionales, en los cuales en tiempos periódicos el comportamiento de la serie es parecido a lo largo del tiempo, es decir presenta movimientos de oscilación.

En un sentido amplio, si X_t es una serie estacionaria (en media y varianza) se puede decir que se cumplen las siguientes condiciones:

$$E\left[X_t\right] = \mu, \forall t \tag{1}$$

$$E\left[(X_t - \mu)^2\right] = \gamma_0, \forall t \tag{2}$$

$$Var(X_t) = \sigma^2, \forall t \tag{3}$$

$$Cov(X_t, X_{t+k}) = \gamma_k, \forall t, k \tag{4}$$

Si una serie temporal es estacionaria (en media y en varianza) encontrar un modelo que explique su autocorrelación es mucho más fácil. Es por esto que muchas veces se trabajan series de precios mediante su rentabilidad.

2.3.3 Recursive multi-step forecasting

Cuando se trabaja con series temporales, raramente se quiere predecir solo el siguiente elemento de la serie (t_{+1}), sino todo un intervalo futuro o un punto alejado en el tiempo (t_{+n}). Cada paso de predicción se conoce como step.

Dado que, para predecir el momento t_n se necesita el valor de t_{n-1}, pero t_{n-1} se desconoce, será necesario realizar predicciones recursivas en las que cada nueva predicción se basa en la predicción anterior. Este proceso se conoce como recursive forecasting o recursive multi-step forecasting.

2.3.4 Medias móviles

El efecto de suavizamiento de la media móvil (MA) sobre la serie de tiempo original proviene de la eliminación del ruido aleatorio [49]. En este caso cada punto de la serie temporal corresponde a la media aritmética de un número de puntos consecutivos de la serie original, en la cual se han seleccionado los puntos de manera que sean eliminados los efectos estacionales y/o irregularidades.

En el promedio móvil de orden m el ciclo estimado de tendencia en el tiempo t se obtiene promediando los valores de la serie temporal en k períodos de t, lo cual se expresa de la siguiente manera:

$$\hat{x}_t = \frac{1}{m} \sum_{j=-k}^{k} x_{t+j} \tag{5}$$

donde $m = 2k+1$. Con esto se elimina parte del ruido aleatorio y se obtiene un componente de ciclo de tendencia suavizado [32].

El método MA opera asumiendo que los cambios estacionales son constantes sobre períodos consecutivos según caso de uso [49]. Este supuesto puede ser válido para diversas aplicaciones.

2.3.5 Suavizado exponencial

Los pronósticos realizados a través de este método corresponden a promedios ponderados de las observaciones pasadas, en las cuales los pesos disminuyen exponencialmente según la antigüedad de las observaciones. Es decir las observaciones recientes recibirán ponderaciones mayores. Su principal ventaja es que genera pronósticos confiables para una gran gama de series de tiempo [32].

En su forma simple el pronóstico para el tiempo $t+1$ corresponde al promedio ponderado entre la observación más reciente x_t y el pronóstico previo $\hat{x}_{t|t-1}$. En el método exponencial de suavizado único, los valores pronosticados se generan de la siguiente manera [49]:

$$x_{t+1} = \alpha x_t + \alpha(1-\alpha)x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + .. \tag{6}$$

Con $\alpha \in [0, 1]$. A partir de esto, los valores ajustados se expresan como:

$$\hat{x}_{t+1|t} = \alpha x_t + (1-\alpha)\hat{x}_{t|t-1} \tag{7}$$

2.3.6 Modelos regresión para series de tiempo

2.3.7 Modelos de predicción en series temporales

Entre los supuestos para la aplicación del modelo de regresión lineal está la homocedasticidad, es decir se requiere que sea homocedástico, lo cual significa que la varianza debe ser constante a lo largo de los datos considerados. Sin embargo, en una serie temporal de tipo financiera es altamente probable que esto no se cumpla, es decir que la serie exhiba heterocedasticidad.

2.3.8 Modelos Autorregresivos

En un modelo de autorregresión el pronóstico de la variable de interés se realiza utilizando una combinación lineal de los valores pasados de dicha variable, es decir es una regresión de la variable contra sí mismo [4]. Al utilizar estos modelos se asume que la variable objetivo depende linealmente de sus valores pasados. Normalmente se restringen los modelos de auto regresión a data estacionaria [32].

El modelo autorregresivo de orden p $AR(p)$ se define en la siguiente ecuación:

$$X_t = \sum_{j=1}^{p} \varphi_j X_{t-j} + w_t \tag{8}$$

En esta ecuación X_t es el precio al día t, $\varphi_1..\varphi_p$ son coeficientes del modelo y el término para el ruido blanco es $w_t \sim N(0, \sigma^2)$. El modelo autorregresivo establece que el valor de la serie temporal en t es una combinación lineal de los p valores anteriores con la adición del término de ruido.

2.3.9 Modelos ARIMA

Es uno de los modelos más populares para predicción en series temporales. El modelo autorregresivo de media móvil (ARIMA) se utiliza para predecir valores futuros de una variable basado en valores pasados. En un modelo ARIMA(p, d, q) hay p términos autorregresivos y q términos de media móvil que en su diferencia d es estacionario.

En el modelo autorregresivo general de media móvil de órdenes p y q (ARMA(p, q)) se combinan los modelos de media móvil y autorregresivo en un nuevo modelo como se define en la siguiente ecuación:

$$X_t = \sum_{j=1}^{p} \varphi_j X_{t-j} + \sum_{j=1}^{q} \theta_j w_{t-j} + w_t \tag{9}$$

Por su parte ARIMA es la generalización del modelo ARMA en cuya ecuación se incluye el grado de diferenciación d que representa el número de veces que se han restado los valores pasados de los datos. B es el operador de retardo que se usa para acceder a observaciones

previas usando la fórmula $B^k X_t = X_t - k$. Se expresa en la siguiente ecuación:

$$\left(1 - \sum_{j=1}^{p} \varphi_j X_{t-j}\right)(1-B)^d X_t = \left(1 + \sum_{j=1}^{q} \theta_j w_{t-j}\right) w_t \tag{10}$$

Seleccionar los valores adecuados para p, d y q puede resultar bastante complejo, sin embargo, la tarea se puede simplificar con el apoyo de funciones como *auto.arima*() en R o en python con *auto_arima*() del paquete pmdarima o con *optimize.brute*() del paquete scipy que permite implementar búsqueda en grilla en el espacio de parámetros especificado.

2.4 Red Neuronal Artificial (ANN)

En lo que sigue y como paso previo a la presentación de las redes neuronales LSTM, se presentarán los aspectos conceptuales básicos de una red neuronal artificial (ANN), a partir del modelo matemático de una neurona artificial y del perceptrón multicapa.

El modelo matemático de una neurona simple, mostrado en la figura 9, facilita la comprensión del planteamiento conceptual que subyace a una red neuronal.

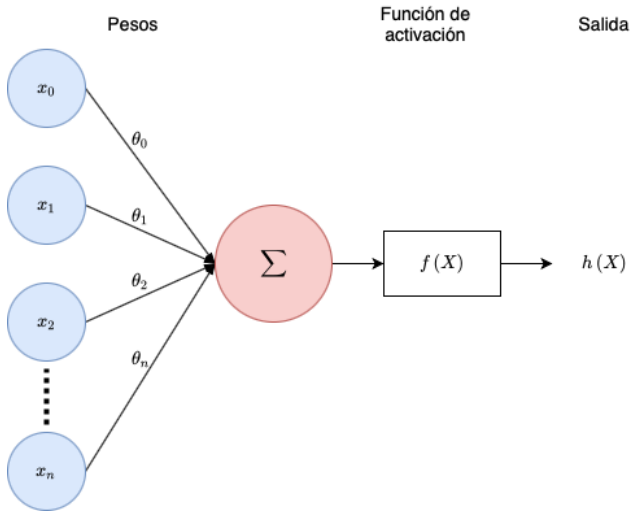


***Figura* 9:** Modelo matemático de una neurona

Este sencillo modelo presenta una serie de inputs, los cuales tienen pesos asociados que en conjunto con una función de activación, generan una salida. Las siguientes ecuaciones (relaciones), relacionan las entradas, pesos y salida del modelo mediante la función de activación, de manera que:

$$Input : X = (x_1 \dots x_n)^T \tag{11}$$

$$Pesos : \Theta = (\theta_1 \dots \theta_n)^T \tag{12}$$

$$Sesgo\ (constante\ de\ activación) : \theta_0 \tag{13}$$

$$Output : h(X) = f\left(\sum_{k=1}^{n} x_k\theta_k + \theta_0\right) = f(\Theta^T X + \theta_0) \tag{14}$$

En esta representación, θ_0 es el sesgo o constante de activación del modelo. Para el entrenamiento del modelo, los pesos (Θ) se definen inicialmente de manera aleatoria con valores cercanos a 0. De esta manera cuando el vector de valores de entrada es enviado a través del modelo, sus componentes son multiplicadas por los pesos, que a su turno han sido sintonizados a través de un algoritmo de aprendizaje. Como se aprecia en la ecuación (14), la salida es calculada a partir de esta operación entre valores de entrada y pesos. Cuando la suma ponderada de los valores de entrada excede un valor de umbral predefinido, la neurona genera un valor que pasará a la función de activación ($f(\cdot)$) para mapear el valor a la salida final de la neurona. Este proceso se repite para todas las neuronas del modelo, hasta que se calcula la salida final en la neurona de salida.

Los modelos basados en redes neuronales son conocidos por su capacidad para encontrar patrones complejos en los datasets de entrenamiento [4], sin embargo una desventaja importante de estos modelos es su falta de interpretabilidad y su necesidad de entrenarse con una gran cantidad de datos para encontrar patrones y hacer predicciones precisas.

2.4.1 Perceptrón

Fueron desarrollados como un modelo computacional simplificado de las neuronas. Una neurona típicamente recibe múltiples entradas (dendritas), tiene un cuerpo celular y genera una salida única (axón). Corresponde a uno de los primeros esquemas de aprendizaje de máquina para clasificación binaria, que consiste de una neurona de umbral binario y una regla de aprendizaje.

De manera más formal, se dice que un perceptrón consiste en un vector de pesos $\mathbf{w} = [w_1 \dots w_n]^\intercal$, uno para cada valor de entrada y un ponderador adicional b comúnmente denominado sesgo. Tanto $\mathbf{w}$ como b son parámetros del perceptrón. Generalmente se utiliza Φ para representar los parámetros, en cuyo caso $\phi_i \in \Phi$ representa el i-ésimo parámetro del perceptrón definido por $\Phi = \{\mathbf{w} \cup b\}$ que calcula la siguiente función:

$$f_\Phi(\mathbf{z}) = \begin{cases} 1 & \text{si } z = b + \mathbf{w} \cdot \mathbf{x} > 0 \\ 0 & \text{en otro caso} \end{cases}$$

La función heaviside o función escalón unitario (H) y la función signo (sgn) son funciones de paso comúnmente utilizadas en perceptrones. La función heaviside (H) puede ser definida como:

$$\mathrm{H}(z) = \begin{cases} 1 & \text{si } z > 0 \\ 0 & \text{en otro caso} \end{cases} \tag{15}$$

También suele representarse como $H(z) = 1_{z>0}$ (función indicator). Por último, la función

signo se define de la siguiente manera:

$$\text{sgn}(z) = \begin{cases} 1 & \text{si } z > 0 \\ 0 & \text{si } z = 0 \\ \text{-1} & \text{si } z < 0 \end{cases} \tag{16}$$

El sesgo suele presentarse como un peso más (por absorción), de manera que sólo se requiere una regla para la actualización de los pesos. En tal caso se requiere un input adicional (x_0) con valor 1 en cuyo caso el sesgo corresponde a su peso. De esta manera también podría encontrarse alternativamente que:

$$z = b + \mathbf{w} \cdot \mathbf{x} = b + \sum_{i=1}^{l} w_i x_i = \sum_{i=0}^{l} w_i x_i \tag{17}$$

En la ecuación w_0 representará el sesgo (b) pues lo que se desea es ajustar (actualizar) el sesgo como parte del proceso de aprendizaje, de esta manera x_0 tendrá valor 1. Se espera que el proceso de clasificación aprendido, sea capás de asignar los puntos a una u otra clase, por lo cual algunos recibirán la etiqueta 1 y otros 0 indicando pertenencia a la clase o no.

Un clásico ejemplo en el cual el perceptrón no puede aprender a clasificar y asignar correctamente las etiquetas, corresponde a la función **XOR**, la cual toma una variable binaria $\mathbf{x} = (x_1, x_2)^T \in \{0, 1\}^2$ como entrada y produce una de las salidas $y = f^*(x)$ siguientes: $y = 1,\ si\ x_1 = 1\ o\ x_2 = 1$ o bien $y = 0,\ si\ x_1 = x_2 = 0\ o\ x_1 = x_2 = 1$.

En este caso es necesario determinar los parámetros Φ, es decir pesos w_1, w_2 y sesgos considerando los siguientes casos; $(0, 0) \mapsto 0$, $(0, 1) \mapsto 1$, $(1, 0) \mapsto 1$, $(1, 1) \mapsto 0$ para encontrar la función $\hat{y} = f_\Phi(\mathbf{X})$ que permita aproximar f^*, donde $\mathbf{X} = \{[0, 0]^\intercal, [0, 1]^\intercal, [1, 0]^\intercal, [1, 1]^\intercal\}$ e $\mathbf{y} = [0, 1, 1, 0]^\intercal$ (targets) representan el conjunto de entrenamiento para ajustar el modelo, puesto que la red debe actuar correctamente en estos cuatro puntos.

Dado que el perceptrón utiliza una combinación lineal, $g : \mathbf{x} \mapsto \mathbf{w}^\intercal\mathbf{x} + b$ seguida por una función de umbral como la definida en 15 y que los parámetros del modelo son $\Phi = (w, b)$, el modelo queda completamente definido como:

$$f_\Phi(\mathbf{x}; \mathbf{w}, b) = \mathbf{x}^\intercal\mathbf{w} + b \tag{18}$$

Aplicando estas reglas al conjunto de entrenamiento obtenemos las siguientes ecuaciones:

$$w_1 + w_2 + b \leq 0 \iff b \leq -w_1 - w_2 \tag{19}$$

$$0w_1 + 0w_2 + b \leq 0 \iff b \leq 0 \tag{20}$$

$$w_1 + 0w_2 + b > 0 \iff b > -w_1 \tag{21}$$

$$0w_1 + w_2 + b > 0 \iff b > -w_2 \tag{22}$$

Sumando las ecuaciones (19) y (20) y las ecuaciones (21) y (22) obtenemos:

$$-w_1 - w_2 \geq 2b$$

$$-w_1 - w_2 < 2b$$

Resulta evidente que el sistema de inecuaciones no tiene solución. Se concluye que un modelo lineal como este no es capaz de representar la función **XOR** ya que no es linealmente separable. En la siguiente figura se aprecia con claridad esta situación, lo cual contrasta con las funciones **AND** y **OR** que son linealmente separables y por lo tanto pueden ser aprendidas por un perceptrón.

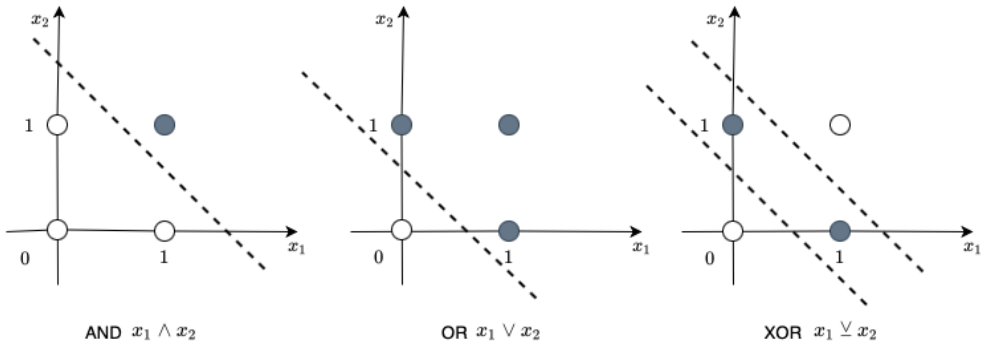


***Figura* 10:** Funciones linealmente separables v/s XOR

Expresando matricialmente el proceso de entrenamiento, podemos comprobar ambos casos (**OR** y **AND**) de manera simple. La configuración del perceptrón difiere según el caso en los valores a aproximar $\mathbf{y}$, es decir $\mathbf{y} = f^*(\mathbf{X})$, ya que lo que se desea encontrar es la función que los aproxima, es decir $\hat{\mathbf{y}} = f_\Phi(\mathbf{X})$. Los parámetros Φ para aproximar la función en cada caso son el arreglo de pesos $\mathbf{W}$ y el escalar del sesgo b. De esta manera tenemos que el conjunto $\mathbf{X}$ de entrenamiento, que es el conjunto de datos en su totalidad, corresponde a:

$$\mathbf{X} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

El arreglo de pesos para ambos casos es $\mathbf{W} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ y, también, en ambos casos la función de activación $g(\cdot)$ es la función umbral, ecuación(15), o cualquier equivalente.

Para el caso de la función **OR**, tenemos que el conjunto $\mathbf{y}$ de objetivos de entrenamiento es $\mathbf{y} = \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix}$, se selecciona $b = 0$ como valor de sesgo. Luego, al aplicar la transformación con estas configuraciones tenemos que $g(\mathbf{W}^\intercal \mathbf{X} + b) = \hat{\mathbf{y}} = \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix}$.

De manera similar, para la función **AND**, el conjunto $\mathbf{y}$ de objetivos de entrenamiento es $\mathbf{y} = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}$, se selecciona $b = -1$ como valor de sesgo. Luego, al aplicar la transformación con estas configuraciones tenemos que $g(\mathbf{W}^\intercal \mathbf{X} + b) = \hat{\mathbf{y}} = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}$.

Validar estos resultados es trivial. Por otra parte, los parámetros pueden variarse con la sola condición de cumplir con la parametrización de las inecuaciones resultantes, para ello y según sea el caso bastaría seguir un razonamiento similar al ya descrito en la función **XOR**.

2.4.2 Red neuronal feed-forward

Algunas de las limitaciones del perceptrón pueden eliminarse apilando múltiples perceptrones, en este caso la ANN resultante se conoce con el nombre de Perceptrón multicapa (MLP), sin embargo se requiere otro método de aprendizaje para ajustar los pesos. Esto conduce a la siguiente forma básica de red denominada red neuronal feed-forward y a una técnica de aprendizaje relativamente simple denominada descenso de gradiente, en el cual los datos que ingresan a la red neuronal, inicialmente avanzan de manera progresiva a través de las capas de la red en un esquema denominado propagación hacia adelante [50]. Se trata de un sistema no lineal cuyo objetivo es aproximar una función $y = f^*(x)$, para lo cual es necesario encontrar una aproximación f_Φ de f^* de manera que $\hat{y} = f_\Phi(\mathbf{x})$, lo cual requiere aprender los parámetros Φ para obtener la mejor aproximación $\hat{y}$ de la variable objetivo y (target). En este caso no hay ciclos de retroalimentación (feedback) en los cuales las salidas son retroalimentadas a la red.

Una estructura de este tipo tiene a lo menos tres capas de nodos, en las cuales cada nodo salvo los de entrada corresponde a una neurona artificial. La figura 11 esquematiza una red neuronal simple con dos unidades ocultas.

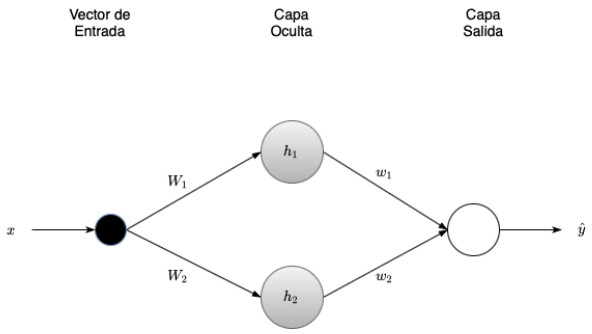


***Figura* 11:** Red neuronal simple

Esta representación muestra una red neuronal feed-forward simple con una capa de entrada, una capa oculta y una capa de salida. La capa de entrada representa un vector de regresores o características, como podrían ser p puntos anteriores en el tiempo (por ejemplo p pasos de tiempo de una serie temporal), es decir las observaciones $\mathbf{x} = [x_{t-1}, x_{t-2}, \dots x_{t-p}]^\intercal$, lo cual también se aprecia en las sigiuentes figuras 12 y 12 que presentan perceptrones con arquitecturas más complejas. Las features son consumidas o enviadas a una capa oculta con n neuronas, cada una de las cuales aplica transformaciones lineales y activaciones no lineales a los valores de entrada. El vector de unidades ocultas $h = (h_1, \dots h_n)$ implementa la función $f^{(1)} : \mathbf{x} \mapsto f^{(1)}(\mathbf{x}; \mathbf{W}, b^{(1)})$. La capa de salida $\hat{y}$ se obtiene aplicando la función $f^{(2)} : \mathbf{x} \mapsto f^{(2)}(\mathbf{x}; \mathbf{w}, b^{(2)})$ a la salida de la capa oculta h. La salida de cada neurona está representada por $h = g(\mathbf{w}^\intercal \mathbf{x} + b)$, en donde $\mathbf{w}$ son los pesos ($\mathbf{W}$ y $\mathbf{w}$ según corresponda en el ejemplo) y b el sesgo ($b^{(1)}$ y $b^{(2)}$ según corresponda en el ejemplo) de la transformación lineal y g es una función de activación no lineal. Por lo tanto en la red de la figura, la salida

$\hat{y}$ es obtenida aplicando:

$$\hat{y} = f^{(2)}\big(f^{(1)}(\mathbf{x})\big) \tag{23}$$

en donde $f^{(1)}$ es la capa oculta y $f^{(2)}$ es la capa de salida. La cantidad de capas ocultas indica la profundidad de la red [27].

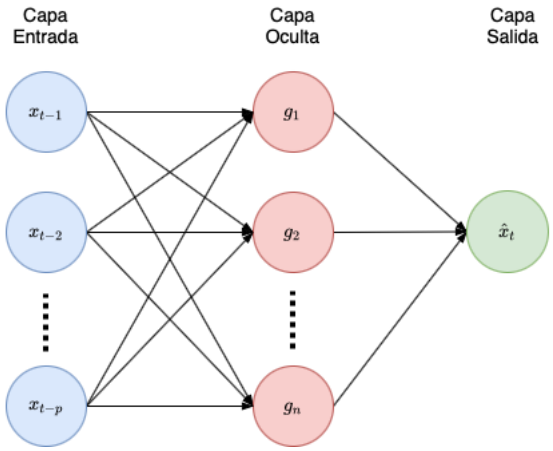


***Figura* 12:** Perceptrón de capa única

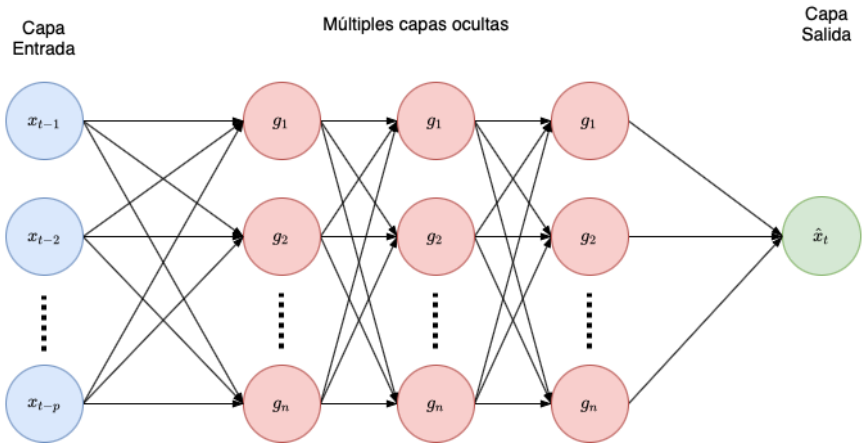


***Figura* 13:** Perceptrón multicapa

Para generalizar este caso particular, se asumirá una red con l capas de profundidad, cada una de las cuales tendrá cierta cantidad de nodos. En ella un vector de entrada $\mathbf{x}$ avanza progresivamente, de manera que $\mathbf{x}^{(l)}$ representa el vector de datos en la capa l, de este modo $\mathbf{x}^{(0)}$ es el vector de datos en la capa de entrada y x_k^{l-1} es el valor en el nodo k y es un iterable en la capa $l-1$, asimismo w_{jk}^l corresponde al ponderador (peso) entre el nodo k del nivel $l-1$ y el nodo j del nivel l. Por último b_j^l es el sesgo asociado al nodo j de la capa l. Con estas convenciones la transformación lineal en cada nodo queda definida por:

$$z_j^l = \sum_k w_{jk}^l x_k^{l-1} + b_j^l \tag{24}$$

luego z_j^l es el valor ponderado en el nodo i de la capa l, el cual a su vez es argumento de la función de activación g, que genera la salida x_j^l en la neurona j de la capa l, es decir:

$$x_j^l = g(z_j^l) \tag{25}$$

con ello el valor "activado" se propaga al siguiente nivel de la red neuronal, lo cual lleva a entender estas redes como máquinas de aproximación de funciones, diseñadas para lograr generalización estadística [27].

Luego, la arquitectura de una máquina de aproximación con profundidad arbitraria d niveles con $d \geq 1$, es decir perceptrón múlticapa (MLP), queda definida por la siguiente ecuación:

$$\boldsymbol{h}^{(l)} = g^{(l)}(\mathbf{W}^{(l)\intercal}\boldsymbol{h}^{(l-1)} + \mathbf{b}^{(l)}),\ l = 1, \ldots, d \tag{26}$$

donde $\boldsymbol{h}^{(0)}$ representa el vector inicial de regresores ($\mathbf{x}$), $g^{(\cdot)}$, $b^{(\cdot)}$ y $h^{(\cdot)}$ la función de activación, los pesos y la salida en el nivel respectivo.

El valor generado por la función de activación dependerá de la transformación no lineal utilizada. La selección de esta función de activación no es obvia y puede requerir cierto número de ensayos para optar por aquella que produce mejores resultados [50].

2.4.2.1 Función de activación

La función de activación permite modelar las complejidades no lineales subyacentes en las relaciones entre regresores y variable objetivo. Algunas de las funciones de activación más comunes son:

- Función sigmoide (σ):

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{27}$$

  en la cual los valores altos tienden de forma asintótica a 1, en tanto que valores muy bajos tienden de manera asintótica a 0. Es ampliamente utilizada en modelos de clasificación y, en términos prácticos "mapea" cualquier número real al intervalo $[0, 1]$

- Tangente hiperbólica ($tanh$):

$$g(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{28}$$

  el dominio de esta función es $(-\infty, +\infty)$ y su codominio es $(-1, 1)$. En términos prácticos, se considera que esta función de activación tiene la propiedad de mapear cualquier número real al intervalo $[-1, 1]$. Es interesante notar que las funciones $\sigma(z)$ y $tanh(z)$ están relacionadas a través de $tanh(z) = 2\sigma(2z) - 1$. Esta característica hace que ambas sufran de saturación sigmoidal, lo cual puede hacer muy difícil el aprendizaje basado en gradiente. Por esta razón, actualmente se desaconseja su utilización como unidades ocultas en redes feed-forward [27].

- Función de activación ReLU (Rectified Linear Unit):

$$g(z) = \rho(z) = \max(0, z) \tag{29}$$

esta función es frecuentemente utilizada [62] dado que puede mapear valores en el rango $[0, +\infty)$, lo cual resulta crítico en el proceso de retropropagación (backward pass) ya que las funciones que generan un limitado rango de valores contribuyen al desvanecimiento del gradiente. Esta función es prácticamente lineal ya que está segmentada en dos partes lineales, por ello conserva muchas de las propiedades que hacen fácil optimizar los modelos lineales con métodos basados en gradiente [27]. Alternativamente, la función *lRelU* (Leaky ReLU) es frecuentemente utilizada [10] ya que puede mapear valores en un rango aún más amplio.

- Función de activación lReLU:

$$g(z) = \begin{cases} \alpha z & para\ z < 0 \\ z & para\ z \geq 0 \end{cases} \tag{30}$$

el coeficiente de rectificación usualmente asume el valor $\alpha = 0.01$. Esta función puede mapear cualquier numero real al rango $(-\infty, +\infty)$.

- Función softmax:

$$g(z) = \sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad j = 1, \ldots, K \tag{31}$$

es frecuentemente utilizada en la capa final de un clasificador basado en redes neuronales, tiene la ventaja de retornar una distribución de probabilidades, por lo que mapea cualquier número real en el rango $[0, 1]$ y además la suma total resultante es 1.

2.4.2.2 Aproximación de la función XOR

El problema planteado por la incapacidad del perceptrón para aproximar la función **XOR**, ahora se puede resolver con un perceprtrón multicapa (MLP) [25], es decir con una red neuronal feed-forward para ser más exactos. El diseño de esta máquina se aprecia en la siguiente imagen.

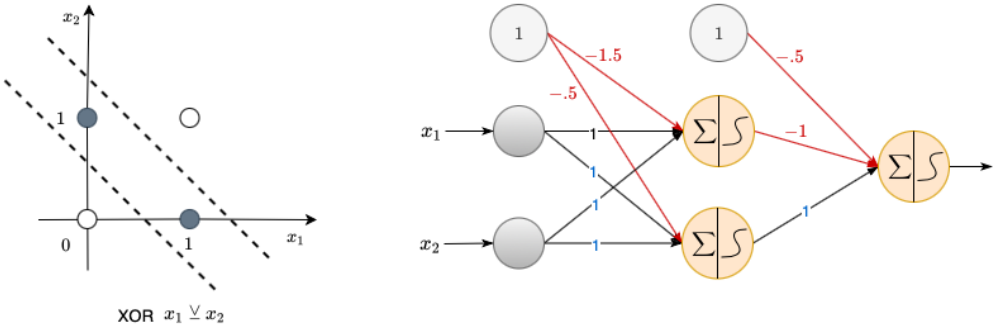


***Figura* 14:** MLP que resuelve problema de clasificación **XOR**

En este ejemplo, la función de activación es provista por la función umbral presentada en la ecuación (15) asimismo, cualquier variante de la función escalón unitario (step function) es una función de activación plausible. Por último, no está demás mencionar que las aplicaciones modernas por defecto utilizan **ReLU**, definida por la función de activación $g(z) = max(0, z),\ z \in \mathbb{R}$, presentada en la ecuación (29). Una máquina de este tipo (red neural) con un diseño provisto sólo de una unidad oculta **ReLU** y un perceptrón lineal como capa de salida resuelve el problema planteado por **XOR**, en este caso la red completa se puede especificar como $f(\mathbf{x}; \mathbf{W}, \mathbf{c}, \mathbf{w}, b) = \mathbf{w}^\intercal \max(0, \mathbf{W}^\intercal \mathbf{x} + \mathbf{c}) + b$, lo cual además representaremos con el siguiente grafo.

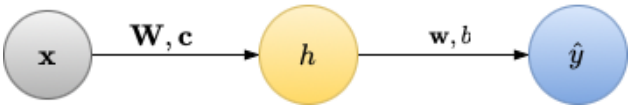


***Figura* 15:** Grafo resumido de la red **XOR**

Para la representación matricial de este MLP, el conjunto de datos de entrenamiento es definido por: $\mathbf{X} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$, $\mathbf{y} = \begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}$

Aplicando $f(\mathbf{x}; \mathbf{W}, \mathbf{c}, \mathbf{w}, b) = \mathbf{w}^\intercal \max(0, \mathbf{W}^\intercal \mathbf{x} + \mathbf{c}) + b$ con la siguiente configuración de pesos y sesgos:

Pesos: $\mathbf{W} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, $\mathbf{w} = \begin{pmatrix} 1 \\ -2 \end{pmatrix}$. Sesgo: $\mathbf{c} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$ y $b = 0$.

Entonces $\hat{\mathbf{y}} = \mathbf{w}^\intercal \max(0, \mathbf{W}^\intercal \mathbf{X} + \mathbf{c}) + b$ efectivamente produce : $\hat{\mathbf{y}} = \begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}$.

En el caso del ejemplo presentado en la figura 14, al considerar la siguiente configuración de pesos y sesgos:

Pesos: $\mathbf{W} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, $\mathbf{w} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$. Sesgo: $\mathbf{c} = \begin{pmatrix} -1.5 \\ -0.5 \end{pmatrix}$ y $b = -0.5$.

Entonces $\hat{\mathbf{y}} = \mathbf{w}^\intercal 1_{\mathbf{W}^\intercal \mathbf{X} + \mathbf{c} > 0} + b$, con $1_{z>0}$ definida en ecuación (15), efectivamente produce: $\hat{\mathbf{y}} = \begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}$. Una nota de cautela en cuanto a las funciones de paso se refiere a que sólo contienen segmentos planos, por lo cual el algoritmo de descenso de gradiente no es aplicable en tales casos.

2.4.2.3 Retropropagación en el MLP

En las ecuaciones planteadas hasta el momento tanto el vector de pesos como los sesgos son parámetros que deben ser estimados y ajustados como parte del proceso de aprendizaje. Los valores son definidos arbitrariamente al inicio, son ajustados a medida que progresa el proceso de entrenamiento. De esta manera los datos son propagados a través de la red hasta obtener el vector final de salida ($\hat{y}$). Este proceso queda reducido a la ecuación vectorizada presentada en (26): $\boldsymbol{h}^{(l)} = g^{(l)}(\mathbf{w}^{(l)\intercal} \boldsymbol{h}^{(l-1)} + \mathbf{b}^{(l)})$.

El proceso de retropropagación es un método de entrenamiento basado en minimizar la pérdida calculada por la función de costo, a través de un algoritmo de optimización como

por ejemplo el algoritmo de descenso de gradiente. La función de costo indica qué tan lejos está la aproximación f_Φ de f^*, es decir la predicción $\hat{y}$ del resultado esperado y (target) durante el proceso de entrenamiento. Existe una amplia variedad de funciones de costo, una de ellas es el error cuadrático medio (MSE), $\text{MSE} = \frac{1}{n}\sum_{x\in\mathbb{X}}(f^*(\mathbf{x}) - f_\Phi(\mathbf{x}))^2$ presentada en la siguiente ecuación:

$$\mathcal{C} = J(\Phi) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{32}$$

Con frecuencia se encontrará esta la función de costo expresada como: $\mathcal{C} = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, dado que multiplicar la función costo por un escalar no afecta la ubicación de su mínimo y simplifica el tratamiento de la derivada.

La selección de la función de costo es un aspecto relevante en el diseño de una red neuronal profunda, por lo cual será necesario considerar diversas opciones. Entre ellas se encuentra la suma de errores al cuadrado (SSE) donde $\text{SSE} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ o también $\text{SSE} = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ para simplificar el tratamiento de la derivada:

$$\mathcal{C} = J(\Phi) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{33}$$

Sin embargo no se recomienda su utilización salvo en las configuraciones más básicas, en su lugar se debe utilizar MSE, puesto que es necesario reescribir la función de costo como el promedio de las funciones de costo SSE_x para muestras individuales de entrenamiemto x, por lo cual se opta por definir $\text{MSE} := \frac{1}{n}\sum_x \text{SSE}_x$ [62].

Una buena opción en caso de problemas de clasificación binaria y multiclase es la función de costo entropía cruzada [62]. Comúnmente la entropía cruzada entre dos distribuciones de probabilidad p_j y q_j se define como $\sum_j p_j \ln q_j$. Se trata de una opción adecuada en conjunto con la función de activación softmax en la capa de salida [57]. En su planteamiento se considera un modelo dado que predice C clases, con $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_c$ las probabilidades hipotéticas (es decir, equivalentes a q_j). Si las instancias observadas para cada clase son $k_1, \ldots, k_c$, entonces $y_i = \frac{k_i}{N}$, *con* $N = \sum_{n=1}^{C} k_n$ son las probabilidades empíricas (es decir p_j) y por log-verosimilitud se tiene que:

$$\mathcal{C} = J(\mathbf{Y}, \hat{\mathbf{Y}}) = -\sum_{i}^{C} y_i \ln \hat{y}_i \tag{34}$$

En un problema de clasificación binaria, las probabilidades hipotéticas $\hat{y}_1$ e $\hat{y}_2$ corresponden a la salida sigmoide (softmax en este caso) de una única neurona que genera distribuciones de probabilidad, es decir una es el complemento de la otra, por consiguiente $\hat{y}_2 = (1 - \hat{y}_1)$, que equivale a decir que $\hat{y}_2 = (1 - \hat{y})$. Similar razonamiento se aplica a las probabilidades empíricas y_1 e y_2, luego $y_2 = (1 - y_1) = (1 - y)$. Luego, realizando las operaciones y reemplazos en la ecuación (34) tenemos que la entropía cruzada para el caso binario es:

$$\mathcal{C} = J(y, \hat{y}) = -(y \ln(\hat{y}) + (1 - y)\ln(1 - \hat{y})) \tag{35}$$

En el caso de un problema multiclase, tenemos:

$$\mathcal{C} = J(\mathbf{Y}, \hat{\mathbf{Y}}) = \sum_{i=1}^{n} \xi(y_i, \hat{y}_i) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=i}^{C} y_c^{(i)} \ln \hat{y}_c^{(i)} \tag{36}$$

Notar que no existe riesgo al aplicar ln a las predicciones ($\hat{y}$), ya que para valores bajos softmax tiende asintóticamente a 0, el mismo argumento es valido para no utilizar una función como log1p en lugar de ln.

En cualquier caso se considera que la entropía cruzada es una generalización de la pérdida logística en problemas de clasificación de clases múltiples. Y como ya se mencionó, cuantifica la diferencia entre dos distribuciones de probabilidad.

Como ha quedado de manifiesto, la selección de la función de costo es un aspecto relevante en el diseño de una red neuronal profunda, por lo cual es necesario analizar en detalle las opciones disponibles para determinar la que mejor se ajusta al problema a resolver. En este sentido conviene tener presente que usualmente MSE no es la función de costo apropiada para modelamiento de datos binarios, por otra parte su utilización se asocia a bajo rendimiento en los modelos, por lo cual se privilegian funciones de costo que mejoran el rendimiento de los mismos, además a menudo es ineficaz cuando se usan métodos de optimización de gradiente. Tanto MSE como el error absoluto medio (MAE= $\frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$) son frecuentemente utilizadas en problemas de regresión, mientras que la pérdida logística (log loss) binaria y multicategórica son funciones de costo comúnmente utilizadas en problemas de clasificación [33] que mejoran significativamente la eficacia del proceso de entrenamiento comparado con MSE. En todo caso, para el pronóstico en series de tiempo, tanto MSE como MAE se consideran aptos [49] para entrenar modelos neuronales.

Con la función de costo definida, el algoritmo de optimización debe encontrar los parámetros para los cuales la aproximación f_Φ de f^* es suficientemente buena, es decir minimizan la función de costo:

$$\min_{\Phi} J(\Phi) \tag{37}$$

El objetivo es optimizar la función de costo minimizando la pérdida $\min_\Phi J(\Phi)$, generalmente utilizado alguna variante del algoritmo de optimización de descenso de gradiente. En el proceso, los errores se retropropagan para ajustar cada peso en la red. Durante el entrenamiento, el proceso de propagación avanza hasta obtener el escalar de costo $J(\Phi)$. A partir de este momento el algoritmo de retropropagación permite que el costo fluya en reversa a través de la red a fin de calcular el gradiente, en un proceso cuya evaluación numérica puede tornarse computacionalmente costosa. La retropropagación se refiere solo al método para calcular el gradiente, mientras que otro algoritmo, como el descenso de gradiente estocástico, se utiliza para realizar el aprendizaje utilizando dicho gradiente [27]. En este caso buscamos el gradiente de la función de costo respecto de los parámetros ($\nabla_\Phi J(\Phi)$), aunque muchos algoritmos calculan otras derivadas ya sea como parte del proceso de aprendizaje o para analizar el modelo aprendido.

El algoritmo iterativamente ajusta los parámetros utilizando el gradiente con signo negativo buscando descender hasta el mínimo, de esta manera para un parámetro dado ϕ

y para una tasa de aprendizaje η que en realidad es un escalar positivo que representa el tamaño del paso, la regla básica de descenso de gradiente resta al valor actual del parámetro el gradiente de la función de costo respecto de dicho parámetro multiplicado por la tasa de aprendizaje ($\phi \leftarrow \phi - \eta \nabla_\phi J(\Phi)$). Luego, desde una perspectiva matemática, la retropropagación se expresa como:

$$\Delta\phi = -\eta \nabla_\phi J(\Phi) = -\eta \frac{\partial J(\Phi)}{\partial \phi} \tag{38}$$

El gradiente puede ser evaluado para todo el conjunto de datos de entrenamiento (lote completo) o bien por lotes de un tamaño inferior al conjunto de entrenamiento y conformados por selección aleatoria de observaciones. Esto da origen al descenso de gradiente estocástico (SGD) que puede llevar a una convergencia acelerada bajo ciertas condiciones, como por ejemplo que las instancias de entrenamiento deben ser independientes e idénticamente distribuidas (IID), para asegurar que los parámetros serán empujados hacia el óptimo global en promedio [25]. Sin embargo este algoritmos es mucho menos regular que el descenso de gradiente del lote completo y, dado que selecciona los lotes de entrenamiento aleatoriamente, podría darse que algunos sean seleccionado en más de una oportunidad por época en tanto que otros podrían no serlo, motivo por el cual es necesario prestar detenida atención a los aspectos técnicos del diseño para controlar estas eventuales situaciones. Un tamaño típico de lote sería 20, sin embargo puede llegar a centenares. Debe considerarse que a más pequeño el tamaño de lote más pequeña debiera ser la tasa de aprendizaje seleccionada [10]. Un enfoque que goza de popularidad es asignar una constante pequeña a η, en el entendido que es menos dañino un tamaño de paso muy pequeño pero que converja aunque lentamente a uno demasiado grande que finalmente no converja. Los valores de η más frecuentemente utilizados son del orden de 0.1, 0.01, 0.001 y así sucesivamente [62]. También se puede optar por un η adaptable aplicando parámetros para decaer o impulsar el proceso de búsqueda del óptimo. Otro enfoque es evaluar la función objetivo $f\big(x - \eta \nabla_x f(x)\big)$ con múltiples valores de η y seleccionar el valor que resulte en el valor más pequeño de la función objetivo [27]. La siguiente imagen ejemplifica la obtención de las derivadas parciales como parte del flujo de propagación y retropropagación en un modelo muy simple con el error cuadrático como función de costo.

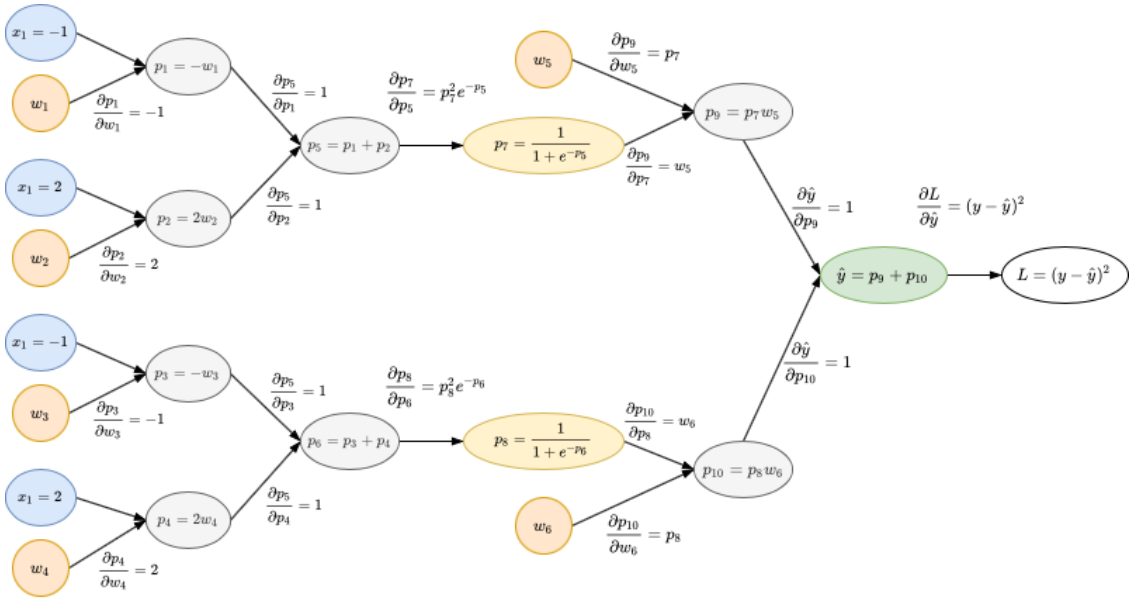


***Figura* 16:** Derivadas parciales

Este ejemplo una red simplificada de un perceptrón con una capa oculta que cuenta con dos unidades ocultas, cada una con activación sigmoidal. La unidad de salida entrega la transformación lineal de las activaciones recibidas. En el ejemplo no están representados los sesgos, sin embargo en un esfuerzo de simplificación pueden ser tratados como un peso más, si se asocian a un input adicional cuyo valor por defecto es 1 (conocido como absorción del sesgo, es decir $z_j^{(l)} = b_j^{(l)} + \sum_{i=1}^{n} w_{ij}^{(l)} x_i^{(l)} = \sum_{i=0}^{n} w_{ij} x_i$ con $w_{0j}^{(l)} = b_j^{(l)}$ y $x_0^{(l)} = 1$, en cualquier caso la ecuación (38) con $\phi = b_j^{(l)}$ nos dice que $\Delta b_j^{(l)} = -\eta \frac{\partial J(\Phi)}{\partial b_j^{(l)}}$). Es decir, sólo se requiere una regla de actualización de pesos.

En el ejemplo se aprecia la evolución de los procesos de propagación y retropropagación. Como culminación de la propagación a través de la red (forward propagation) se obtiene la pérdida L entre el valor observado y la predicción como resultado de las transformaciones no lineales y lineales. Luego, durante la retropropagación (backward propagation o backprop) se obtiene el vector gradiente, con el cual se realiza el ajuste a las estimaciones de parámetros en función de la tasa de aprendizaje y los métodos que se apliquen en su sintonización. Para ello es necesario calcular las derivadas parciales de la función de costo respecto de cada parámetro $\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial w_6}$. Como se aprecia a continuación, el desarrollo presentado en [49] resulta muy didáctico y realmente no requiere más detalle.

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial p_9} \frac{\partial p_9}{\partial p_7} \frac{\partial p_7}{\partial p_5} \frac{\partial p_5}{\partial p_1} \frac{\partial p_1}{\partial w_1}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial p_9} \frac{\partial p_9}{\partial p_7} \frac{\partial p_7}{\partial p_5} \frac{\partial p_5}{\partial p_2} \frac{\partial p_2}{\partial w_2}$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial p_{10}} \frac{\partial p_{10}}{\partial p_8} \frac{\partial p_8}{\partial p_6} \frac{\partial p_6}{\partial p_3} \frac{\partial p_3}{\partial w_3}$$

$$\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial p_{10}} \frac{\partial p_{10}}{\partial p_8} \frac{\partial p_8}{\partial p_6} \frac{\partial p_6}{\partial p_4} \frac{\partial p_4}{\partial w_4}$$

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial p_9} \frac{\partial p_9}{\partial w_5}$$

$$\frac{\partial L}{\partial w_6} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial p_{10}} \frac{\partial p_{10}}{\partial w_6}$$

Comunmente, al iniciar del entrenamiento los parámetros (Φ) son inicializados en valores aleatorios muestreados desde una distribución uniforme en el intervalo $[-1, 1]$ o una distribución normal de media 0 y varianza unitaria. Una vez obtenidos los gradientes, el siguiente paso es actualizar los parámetros (pesos) mediante el algoritmo de descenso de gradiente, aplicando la regla de actualización y la tasa de aprendizaje, ecuación (38). Se trata de un proceso iterativo, en el cual la actualización del modelo se realiza al final de cada época, las cuales corresponden a una iteración completa sobre el dataset de entrenamiento. Esta es la razón por la cual el algoritmo se denomina descenso de gradiente por lotes [25] puesto que utiliza el lote completo de entrenamiento a cada paso (η fijo o variable), como resultado puede llegar a ser extremadamente lento cuando se trata de conjuntos de datos de entrenamiento demasiado grandes.

Se ha demostrado que que un MLP con sólo una capa oculta puede modelar incluso las funciones más complejas en la medida que tenga suficientes neuronas.*****

2.4.2.4 Tipos de descenso de gradiente

Son comunes tres tipos de algoritmos de descenso de gradiente, los cuales difieren en la cantidad de datos utilizados por el algoritmo del caso.

Descenso de gradiente por lotes

El descenso de gradiente por lotes, también llamado descenso de gradiente de vainilla, calcula el error para cada ejemplo dentro del conjunto de datos de entrenamiento, pero solo después de que se hayan evaluado todos los ejemplos de entrenamiento, el modelo se actualiza. Todo este proceso es como un ciclo y se llama época de entrenamiento.

Algunas de las ventajas del descenso de gradiente por lotes son su eficiencia computacional, produce un gradiente de error estable y una convergencia estable. Algunas desventajas son que el gradiente de error estable a veces puede resultar en un estado de convergencia que no es el mejor que puede lograr el modelo. También requiere que todo el conjunto de datos de entrenamiento esté en la memoria y disponible para el algoritmo.

Descenso de gradiente estocástico (SGD)

Por el contrario, el descenso de gradiente estocástico (SGD) hace esto para cada ejemplo de entrenamiento dentro del conjunto de datos, lo que significa que actualiza los parámetros para cada ejemplo de entrenamiento uno por uno. Dependiendo del problema, esto puede hacer que SGD sea más rápido que el descenso de gradiente por lotes. Una ventaja es que las actualizaciones frecuentes permiten tener una tasa de mejora bastante detallada.

Sin embargo, las actualizaciones frecuentes son más costosas computacionalmente que el enfoque de descenso de gradiente por lotes. Además, la frecuencia de esas actualizaciones

puede resultar en gradientes ruidosos, lo que puede hacer que la tasa de error salte en lugar de disminuir lentamente. Luego, debido a su naturaleza estocástica este algoritmo es mucho menos regular que el descenso de gradiente por lotes, lo cual se traduce en que los parámetros obtenidos serán buenos pero no óptimo puesto que una vez se aproxime al mínimo, la función de costo fluctuará pero no alcanzará el mínimo.

En conclusión, la aleatoriedad es buen para escapar de los óptimos locales, pero mala porque significa que el algoritmo nunca puede establecerse en el mínimo.

Descenso de gradiente por mini-lotes (Mini-Batch GD)

Este es el método de referencia, ya que es una combinación de los conceptos de SGD y el descenso de gradientes por lotes. Simplemente divide el conjunto de datos de entrenamiento en pequeños lotes y realiza una actualización para cada uno de esos lotes. Esto crea un equilibrio entre la solidez del descenso de gradiente estocástico y la eficiencia del descenso de gradiente por lotes.

Los tamaños de mini lotes comunes oscilan entre 50 y 256, pero como cualquier otra técnica de aprendizaje automático, no existe una regla clara porque varía para diferentes aplicaciones. Este es el algoritmo de referencia cuando se entrena una red neuronal y es el tipo más común de descenso de gradiente para el aprendizaje profundo, puesto que exhibe un progreso menos errático en el espacio de parámetros en comparación a SGD, sin embargo puede resultar difícil escapar de mínimos locales.

2.4.2.5 El algoritmo

Como ya se ha mencionado, la idea general del algoritmo de descenso de gradiente es ajustar los parámetros de forma iterativa para minimizar una función de costo. En resumen, el algoritmo realiza los siguientes procesos [25]:

- Maneja un mini-lote a la vez (por ejemplo, contiene 50 instancias cada uno) y pasa por el conjunto de entrenamiento completo varias veces. Cada pasada se llama época.

- Cada mini-lote se pasa a la capa de entrada de la red, que simplemente lo envía a la primera capa oculta. El algoritmo luego calcula la salida de todas las neuronas en esta capa (para cada instancia en el mini-lote). El resultado se pasa a la siguiente capa, su salida se calcula y se pasa a la siguiente capa, y así sucesivamente hasta obtener la salida de la última capa, la capa de salida. Este es el proceso de propagación: es exactamente como hacer predicciones, excepto que todos los resultados intermedios se conservan ya que son necesarios para el pase hacia atrás.

- A continuación, el algoritmo mide el error de salida de la red (es decir, utiliza una función de pérdida que compara la salida deseada y la salida real de la red, y devuelve alguna medida del error).

- Luego calcula cuánto contribuyó cada conexión de salida al error. Esto se hace analíticamente simplemente aplicando la regla de la cadena, lo que hace que este paso sea rápido y preciso.

- Luego, el algoritmo mide cuántas de estas contribuciones de error provienen de cada conexión en la capa inferior, nuevamente usando la regla de la cadena, y así sucesivamente hasta que el algoritmo alcanza la capa de entrada. Este paso inverso mide de manera eficiente el gradiente de error en todos los pesos de conexión en la red al propagar el gradiente de error hacia atrás a través de la red (de ahí el nombre del algoritmo).

- Finalmente, el algoritmo realiza un paso de descenso de gradiente para ajustar todos los pesos de conexión en la red, utilizando los gradientes de error que acaba de calcular.

2.4.3 Redes neuronales recurrentes (RNN)

Una red neuronal recurrente (RNN) es una red en la que la salida contribuye a su propia entrada. En terminología de grafos, es un grafo cíclico dirigido, a diferencia del gráfico acíclico dirigido de las redes neuronales feed-forward. Realizar la misma tarea para cada elemento de una secuencia de entradas hace que la tarea sea recurrente, en ellas la salida depende de los cálculos previos realizados por la red y por lo tanto es importante tener el equivalente a una función de memoria que capture esa información. Por lo tanto es necesario que la red que recuerde las entradas anteriores, ya que se utilizarán en los pasos posteriores del entrenamiento.

Generalmente aumentan la arquitectura de red multicapa convencional con la adición de ciclos que conectan nodos adyacentes o pasos de tiempo. Estos ciclos constituyen la memoria interna de la red que se utiliza para evaluar las propiedades del punto de datos actual en cuestión con respecto a los puntos de datos del pasado inmediato. También es importante tener en cuenta que la mayoría de las redes neuronales feedforward convencionales se limitan a un mapeo uno a uno para la entrada y la salida [27]. Sin embargo, estas pueden realizar asignaciones de uno a muchos, de muchos a muchos (por ejemplo, traducción de voz) y de muchos a uno (por ejemplo, identificación de voz). Se utiliza un grafo computacional para representar las asignaciones entre las entradas y las salidas y la pérdida. Desplegar el grafo en una cadena de eventos proporciona una imagen clara del uso compartido de parámetros dentro de la red.

Los siguientes imágenes muestran la arquitectura de alto nivel de una RNN, en la primera de las imágenes se observa una unidad con una relación recurrente que recibe un valor de entrada x_t en el tiempo t del proceso, el cual genera la salida a y_t. La imagen de la derecha es una representación desde un punto de vista vectorial de las interacciones entre vectores de entrada $\mathbf{I}$, unidades ocultas $\mathbf{H}_n$ y salidas $\mathbf{O}$.

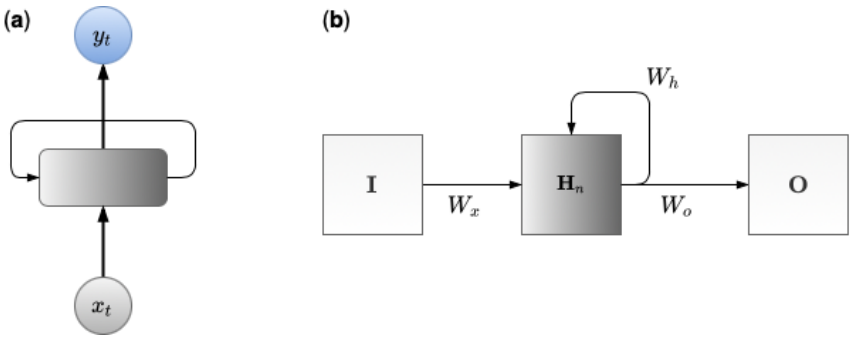


***Figura* 17:** Arquitectura genérica de una RNN

La figura (b) es una imagen simplificada para efectos de esta explicación, en la cual **I** representa las unidades de entrada (vector de entrada), $\mathbf{H}_n$ representa la capa oculta compuesta por eventualmente n unidades ocultas (neuronas), en tanto que **O** es la unidad de salida ya sea como vector o escalar, según corresponda. Las matrices $\mathbf{W}_x$, $\mathbf{W}_o$ y $\mathbf{W}_h$ son los pesos desde capa de entrada a capa oculta, desde capa oculta a capa de salida y de las conexiones entre unidades ocultas respectivamente. Por ejemplo, se podría desarrollar un modelo de pronóstico en donde el vector de entrada $[x_{t-1}, \ldots, x_{t-p}]$ de los p pasos de tiempo se propaga en la red y la salida en el último paso de tiempo es la predicción de x_t, es decir $\hat{y} = x_t$, dados $x_{t-p}, \ldots, x_{t-1}$. Indudablemente la RNN puede ser diseñada para generar una salida a cada paso de tiempo, lo cual es útil en modelos que deben generar series de palabras correspondiente a la traducción de un idioma a otro, por ejemplo.

La siguiente imagen muestra el despliegue de una RNN en la cual se observan cuatro pasos de tiempo, en esta representación h_0 porta la memoria de la serie.

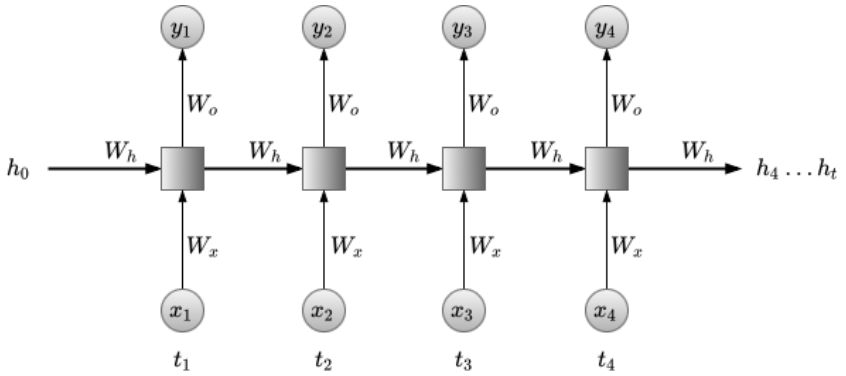


***Figura* 18:** Despliegue de una RNN

En una configuración clásica, sólo se utilizaría x_1 para proporcionar el vector de entrada y en la salida y_4 se obtendría la predicción buscada, pero para los efectos de esta explicación utilizaremos todas las entradas y salidas es decir x_i e y_i. En la figura h_t representa los inputs de la conexión recurrente. Dado que h_0 porta la memoria de la serie, al inicio se puede generar poniendo todas sus componentes en 0. La propagación en la red comienza

con el estado inicial h_0, luego para cada paso de tiempo tenemos las siguientes ecuaciones de actualización:

$$\mathbf{h}_0 = \mathbf{0} \tag{39}$$

$$\mathbf{h}_t = \mathrm{g}_h(\mathbf{b} + \mathbf{w}_h\mathbf{h}_{t-1} + \mathbf{w}_x\mathbf{x}_t) \tag{40}$$

$$\mathbf{y}_t = \mathbf{c} + \mathbf{w}_o\mathbf{h}_t \tag{41}$$

La retropropagación en una RNN es denominada retropropagacion en el tiempo (BPTT: backpropagation through time) y, como ya se ha mencionado, el propósito del proceso de retropropagación es calcular los gradientes de error (E o L, etc.) respecto de $\mathbf{W}_x$, $\mathbf{W}_h$ y $\mathbf{W}_o$. De igual manera que con las funciones de costo, se puede recurrir a sumar los gradientes para cada muestra de entrenamiento en un momento dado del tiempo, con lo cual tenemos la siguiente ecuación:

$$\frac{\partial L}{\partial w_i} = \sum_t \frac{\partial L_t}{\partial w_i} \tag{42}$$

De esta manera, para calcular el gradiente de $L - 2$ tenemos que:

$$\frac{\partial L_2}{\partial \mathbf{w}_o} = \frac{\partial L_2}{\partial \mathbf{y}_2}\frac{\partial \mathbf{y}_2}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{w}_o} \tag{43}$$

Para el caso de $\mathbf{w}_h$ ($\mathbf{w}_x$ es similar) tenemos:

$$\frac{\partial L_2}{\partial \mathbf{w}_h} = \frac{\partial L_2}{\partial \mathbf{y}_2}\frac{\partial \mathbf{y}_2}{\partial \mathbf{h}_2}\frac{\partial \mathbf{h}_2}{\partial \mathbf{w}_h} = \frac{\partial L_2}{\partial \mathbf{y}_2}\frac{\partial \mathbf{y}_2}{\partial \mathbf{h}_2}\sum_{i=0}^{2}\frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_i}\frac{\partial \mathbf{h}_1}{\partial \mathbf{w}_h} \tag{44}$$

la expresión completa depende de $\mathbf{h}_1$ ya que $\mathbf{h}_2 = \mathrm{g}_h(\mathbf{b} + \mathbf{w}_h\mathbf{h}_1 + \mathbf{w}_x\mathbf{x}_2)$. Es fácil concluir a partir de este ejemplo que esta cadena de multiplicación de gradientes será inquietantemente extensa para modelar dependencias de largo alcance y, aquí es donde surge el problema de la desaparición del gradiente. Esto dado que las funciones de activación internas serán *tanh* o *sigmoide*, lo cual conducirá a gradientes fraccionarios y para intervalos de tiempo de largo alcance, la multiplicación de estos gradientes fraccionarios reduce el producto final a cero. Por el contrario, es posible experimentar problemas de gradiente explosivo, en los que los valores de gradiente se vuelven cada vez más grandes y asignan una importancia indebida a ciertos pesos.

2.4.4 Redes LSTM

Las redes neuronales LSTM son extremadamente potentes para modelar series temporales, pues con ellas es posible predecir una cantidad arbitraria de pasos de tiempo hacia el futuro. Otros métodos, como por ejemplo, medias móviles y EMA no son apropiados en ese escenario. Por otra parte, como ya se ha visto, la utilización de métodos autorregresivos, está supeditada al cumplimiento de estrictas condiciones para otorgar validez a sus pronósticos.

Como ya se ha mencionado, todas las redes neuronales recurrentes tienen forma de cadena repetida de módulos, en éstas en general el módulo repetido tiene sólo una capa neuronal, a diferencia de una LSTM que cuenta con cuatro capas neuronales [39]. El estado de la celda principal del módulo es la línea horizontal que atraviesa la parte superior de la figura 19. La arquitectura de la red se basa en la adición o eliminación de información a este estado de celda, a través de las puertas de forget (f_t), input (i_t) y output (o_t) respectivamente. La puerta input (ecuación 46) en conjunto de la capa tanh (ecuación 47) en realidad conforman una una puerta de actualización [33].

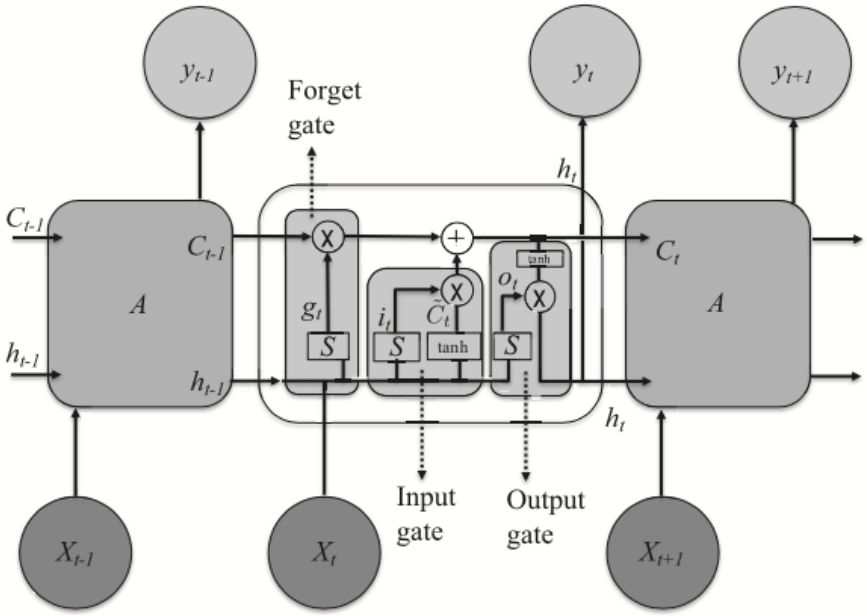


***Figura* 19:** Estructura interna red neuronal LSTM [57]

Un módulo LSTM (celda) tiene 5 componentes esenciales que le permiten modelar datos tanto a largo como a corto plazo, según se aprecia en las siguientes ecuaciones [57] que describen su funcionamiento:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{45}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{46}$$

$$\widetilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{47}$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{48}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{49}$$

$$h_t = o_t \tanh(C_t) \tag{50}$$

De acuerdo a estas ecuaciones, dado un vector de entrada x_t, en el módulo repetido de la LSTM se verifica lo siguiente:

Forget gate, ecuación 45:

El primer paso en la LSTM es definir la información a eliminar del estado de la celda. Esta decisión se realiza en la capa sigmoide, para lo cual actúa sobre h_{t-1} y x_t y genera una salida entre 0 (olvidar) y 1 (mantener), para cada componente en el estado de celda C_{t-1}.

Input gate, ecuación 46:

Esta capa sigmoide decide qué valores actualizar. Es el primer paso para decidir la nueva información que se almacenará en el estado de la celda.

Capa tanh, ecuación 47:

A continuación la capa tanh determina un vector de nuevos valores candidatos $\widetilde{C}_t$ para agregar al estado.

Cell state, ecuación 48:

Este es el momento de actualizar el antiguo estado de la celda C_{t-1} al nuevo estado C_t El nuevo estado de la celda se obtiene filtrando el antiguo estado de celda a través de forget gate (f_t) y sumando el resultado de la combinación entre input gate (i_t) y la capa tanh.

Output gate, ecuación 49:

A través de la ejecución de la capa sigmoide se toma la decisión de qué partes del estado de la celda pasarán a la salida.

Hidden state, ecuación 50:

La salida final en el módulo LSTM resulta de la multiplicación entre output gate (o_t) y la capa tanh del nuevo estado de celda, de manera que filtra el estado al que se dará salida.

3 Materiales y Métodos

3.1 Materiales

Todos los procesos y análisis que se presentan a continuación, tienen su origen en las transacciones diarias de un conjunto de 68 criptomonedas, las cuales se entiende corresponden al mercado sobre el cual se opera. De esta manera y para los efectos de este trabajo, cada criptomoneda tiene asociado un archivo plano que contiene tres meses de transacciones OHLCc diarias.

A partir del proceso de estos archivos planos, se obtienen series temporales que conforman datasets iniciales para cada criptomoneda. Este es el inicio de un laborioso proceso, en el cual con los datasets iniciales se entrenan modelos, cuyas predicciones permiten estructurar una nueva colección de datasets, correspondientes a series temporales que incluyen dichas predicciones. Con estas nuevas series temporales, se procede a estructurar lo que se ha denominado *datasets de atributos*, los cuales no son más que datasets de 3-tuplas compuestos por todas las combinaciones posibles, sin repetición, de tres de dichas series temporales con y sin preproceso, lo que en definitiva constituye el insumo para el proceso de decisión lingüístico con enfoque DLDM en esta implementación.

3.1.1 Archivos planos criptomonedas

Cada una de las 68 criptomonedas (el mercado) cuenta con un archivo plano con datos de transacciones OHLCc diarias. Cada archivo contiene tres meses de transacciones diarias que abarca el mismo período considerado en el benchmark. La tabla 1 muestra las columnas de datos que incluye cada archivo plano junto a una breve descripción de las mismas.

#Columna	Columna	Descripción
1	Date	Fecha de la transacción
2	Open	Precio de apertura del día
3	High	Valor máximo en el día de operación
4	Low	Valor mínimo en el día de operación
5	Close	Precio de cierre del día
6	Volume	Volumen de mercado
7	Cap	Capitalización de mercado

Tabla 1: Estructura de los archivos planos

Con estas columnas de datos se procede a configurar series temporales, compuestas por las correspondientes transacciones diarias que, posteriomente, se utilizarán en el proceso de entrenamiento de modelos.

3.1.2 Datasets en entrenamiento de modelos

La tabla 2 muestra que los datasets iniciales están compuestos por las columnas 2 a 5 y 7 de los archivos planos, descritos en la tabla 1. Es decir, cada criptomoneda inicialmente

cuenta con 5 series temporales de transacciones OHLCc diarias, a partir de las cuales se realizará el proceso de entrenamiento experimental de los modelos.

#Columna	Serie	Descripción de la serie
2	Open	Precio de apertura del día
3	High	Precio máximo en el día operación
4	Low	Precio mínimo en el día operación
5	Close	Precio de cierre del día
7	Log Cap	Transformación de potencia aplicando logaritmo a la capitalización de mercado (Cap).

Tabla 2: Series temporales entrenamiento y validación

Como se aprecia en la tabla 2, por razones de escala, la columna 7 correspondiente a capitalización de mercado (Cap) ha sido transformada aplicando logaritmo. Otra ventaja bien conocida es que esta transformación convierte distribuciones sesgadas a una distribución más cercana a la normal o menos sesgada.

3.1.3 Datasets de atributos para modelo lingüístico

Las series temporales complementadas con predicciones para 10 pasos de tiempo finales, constituyen los atributos con los cuales se estructuran datasets compuestos por tres atributos de beneficio (3-tuplas), los cuales son el insumo del modelo lingüístico. La tabla 3 describe estos atributos base, todos ellos complementados con las respectivas predicciones para los 10 pasos de tiempo finales.

#Atr.	Atributo	Definición de atributo
1	Day Profitability	Variación calculada como: $Close/Open$
2	Day Variability	Variación calculada como: $High/Low$
3	Open	Serie temporal con precio de apertura diario.
4	Low	Serie temporal con menor precio diario para las criptomonedas.
5	High	Serie temporal con el mayor precio diario para las criptomonedas
6	Close	Serie temporal con precio de cierre diario en el período.
7	Log Cap	Serie temporal formada por el logaritmo de la capitalización de mercado por día de operación y criptomoneda.

Tabla 3: Atributos para modelo DLDM

La implementación del modelo de decisión lingüístico con enfoque DLDM realizada en el benchmark, descansa sobre un dataset predefinido, que para todas las criptomonedas está compuesto por tres atributos de beneficio preestablecidos.

La implementación propuesta en este trabajo, considera las series temporales, que a partir de este punto son denominadas atributos, presentadas en la tabla 3, para crear todos los datasets posibles de tres atributos, es decir 3-tuplas de series temporales. Para los efectos

de presentar estos datasets, es innecesario hacerlo por extensión, pues basta decir que son $C_3^7 = 35$ 3-tuplas posibles, es decir datasets de atributos. En un exceso de detalle y sólo a modo de enumeración por comprensión, utilizando el número de atributo (#Atr.) para ello, éstas 3-tuplas de atributos (datasets de atributos) son: $(1, 2, 3), (1, 2, 4), (1, 2, 5) \ldots (5, 6, 7)$.

3.2 Método

El contexto de este proyecto es el mercado de criptomonedas como una alternativa ciertamente riesgosa de inversión pero muy interesante desde el punto de vista de la toma de decisiones, en este caso aplicada a la selección de un objetivo de inversión a partir de un conjunto de alternativas posibles. La figura 20 presenta la estructura conceptual prevista para el desarrollo de este trabajo. En dicha conceptualización, se explicitan a alto nivel los componentes y procesos que será necesario diseñar y desarrollar.



***Figura* 20:** Diseño conceptual

Como es posible apreciar en la figura, el origen del proceso se sitúa en el mercado de criptomonedas como fuente de los datos que es preciso analizar. Para ello se asume disponibilidad de un dataset inicial que da viabilidad a este desarrollo experimental, pero al mismo tiempo se plantea que la fuente podría orginarse en bases de datos de series temporales OHLC provistas por diversos servicios en línea, tanto de pago como gratuitos, cuyo consumo se implementa mediante diversas API's disponibilizadas por plataformas de transacción en línea. A partir de este contexto base, emergen cuatro grandes grupos de actividades de manifiesta complejidad, estos son: 1) El proceso de diseñar, implementar y entrenar modelos RNN de tipo LSTM cuyo propósito es generar predicciónes para cada serie temporal OHLC y critomoneda, 2) Definir configuraciones de atributos y criterios de escalamiento para dichos atributos, cuyo propósito es evaluar el impacto tanto en el ranking de criptomonedas como en el proceso de inversión simulado, 3) Diseñar, implementar y configurar un modelo lingüístico DLDM cuyo objetivo es generar ranking de criptomonedas a partir de series OHLC, que en este caso corresponderán tanto a los datos aportados directamente por el mercado como a las predicciones generadas por las redes neuronales LSTM entrenadas para el efecto y 4) Diseñar, implementar y ejecutar experimentos para evaluar comparativamente las estrategias bajo este proceso de generación del ranking.

La estructura conceptual antes descrita se ha organizado en tres grandes grupos de actividades de acuerdo a las prioridades de implementación, es así como en la figura 21 se aprecian los siguientes grupos de actividades.

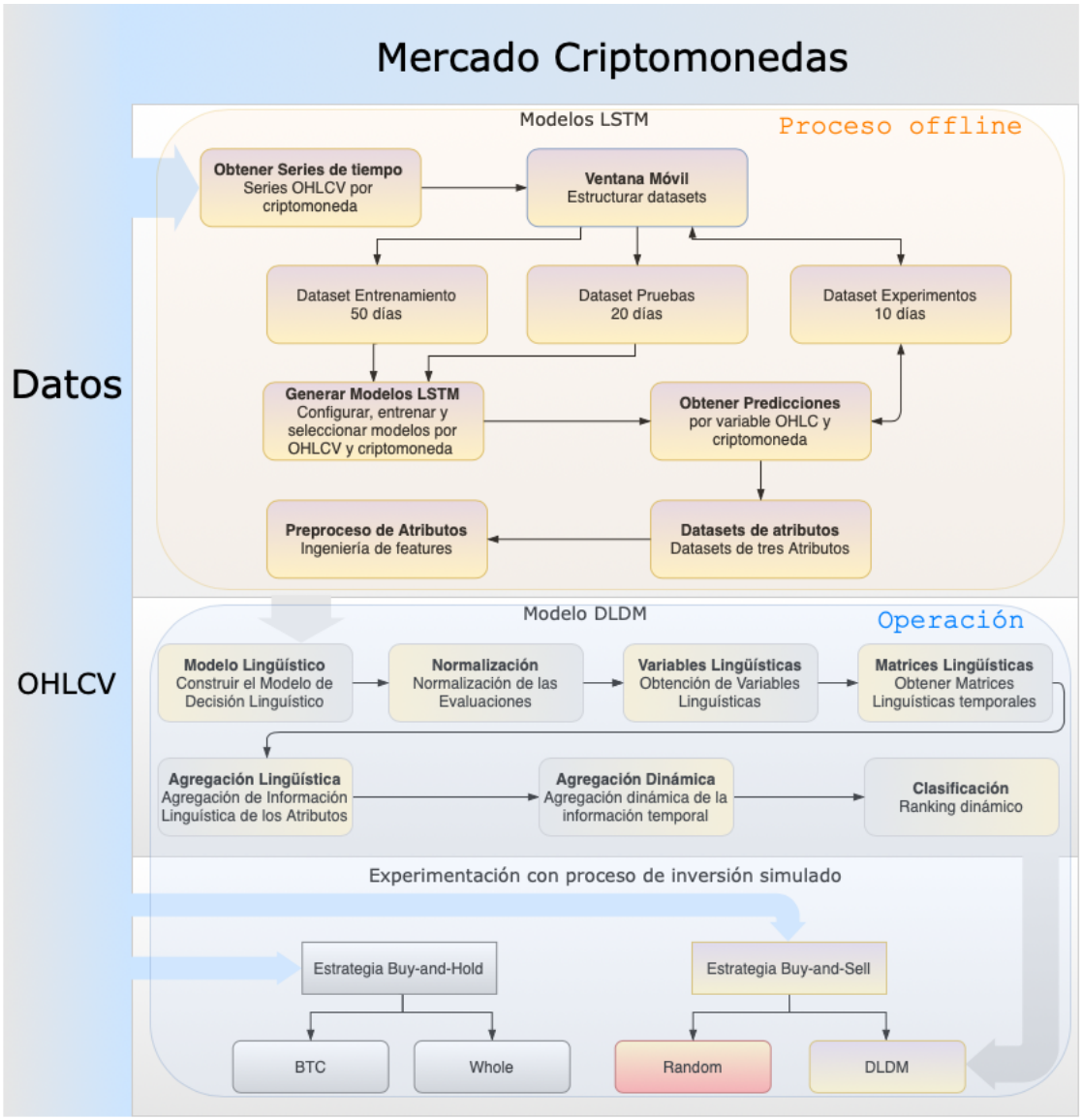


***Figura* 21:** Proceso general

En la figura 21, se aprecia que la implementación de este trabajo considera las siguientes actividades generales: 1) Actividades necesarias para el diseño, implementación, evaluación y selección de las redes neuronales LSTM con menor MAE, destinadas a generar las series temporales con las predicciones de pasos de tiempo out-of-sample a partir de tales modelos, para finalmente obtener datasets de atributos a partir de las combinaciones posibles y preproceso de estas nuevas series temporales, 2) Adaptación e implementación del enfoque DLDM, incluyendo el proceso de clasificación (ranking) dinámica para la selección de criptomonedas con mayor potencial de rendimiento, 3) Experimentación en proceso de inversión simulado, para verificar el comportamiento de la cartera objetivo (DLDM), respecto al benchmark y a las carteras alternativas, cuyo propósito es determinar las combinaciones

de atributos-preproceso que permiten el mejor rendimiento comparativo en este proceso.

A pesar que los aspectos de implementación escapan al alcance de esta tesis, en la misma figura 21 es posible apreciar dos grandes bloques denominados: Proceso offline y Operación, los cuales ofrecen una aproximación de implementación de este método. El **proceso offline** se refiere a la aplicación de los modelos LSTM seleccionados y entrenados, aplicando la metodología que se expone en este trabajo, para la obtención de las predicciones. El proceso denominado **operación**, implica aplicar el modelo LDM a los datasets de atributos complementados con las predicciones para generar el ranking dinámico y su posterior aplicación en el proceso de inversión. Adicionalmente y a medida que transcurre el proceso de inversión, considera por una parte complementar los datasets con los cierres reales, es decir reemplazar la predicción previa con el valor de cierre real y, obtener métricas que permitan posteriormente implementar procesos de mejora y optimización de las redes neuronales en operación.

Los siguientes puntos presentan en detalle cada una de las etapas que son parte de esta tesis.

3.2.1 Etapa 1: Modelos LSTM

En esta etapa se realiza el desarrollo de todos los procesos necesarios para implementar, entrenar y evaluar en función del MAE diversas configuraciones de redes neuronales LSTM. Mediante este proceso, se procederá a seleccionar el o los modelos que se utilizarán en la obtención de predicciones de series de precios OHLCc para cada criptomoneda en el período experimental correspondiente a los últimos diez pasos de tiempo de dicho período. Estas evaluaciones se realizaron en los siguientes escenarios:

- Arquitecturas de modelos con una y dos capas ocultas LSTM.
- La primera capa oculta puede contener 128, 64 o 32 unidades ocultas.
- En el caso de las arquitecturas con dos capas ocultas, la segunda contendrá la mitad de unidades ocultas presentes en la primera capa oculta.
- En el caso de una capa oculta, el proceso de entrenamiento consideró detención anticipada en: 50 y 100 épocas respectivamente.
- En el caso de dos capas ocultas, el proceso de entrenamiento consideró detención anticipada en: 5, 10, 20 50 y 100 épocas respectivamente.

Los modelos fueron entrenados con un número potencial de 500 épocas para cada criptomoneda y serie temporal, con monitoreo y tabulación del MAE en validación y selección del mejor modelo para cada caso.

Para construir el dataset para cada criptomoneda y serie temporal OHLCc se utilizó una ventana móvil de 10 pasos de tiempo (días). La predicción corresponde al siguiente paso de tiempo en la serie, es decir el onceavo valor en cada serie y así sucesivamente.

De acuerdo a este criterio, se estructuraron tres datasets: (1) Conjunto de datos de entrenamiento, (2) Conjunto de datos de validación y (3) Conjunto de datos para experimentos. El dataset de experimentos corresponde a los últimos 10 días de datos en cada serie y cada criptomoneda, que para el propósito de este trabajo serán considerados no vistos y se utilizarán para simular el proceso de inversión como si fueran datos 'out-of-sample'. De los días restantes, luego de excluir los pasos de tiempo para experimentos, se reservaron los 20 últimos pasos de tiempo, en torno a los cuales se estructuró el dataset de validación, con el cual deben obtenerse 20 predicciones como parte del proceso de validación durante el entrenamiento de cada modelo. El conjunto de entrenamiento se construyó con los pasos de tiempo restantes, luego de excluir datos de experimentación y validación.

3.2.1.1 Paso 1: Series temporales OHLC y Log Market Cap.

El proceso inicia con las series temporales asociadas a las variables OHLC y log de Capitalización de Mercado (Mkt. Cap.), que denotaremos como OHLCc, de cada una de las m criptomonedas incluidas en la muestra. Estos conjuntos iniciales de datos son representados matricialmente, de manera que $\mathbf{K}^{(k)} = \left(\mathbf{k}_t^k(v)\right)_{t,v}$, corresponde a la matriz con las observaciones en el período de análisis $t = 1, \dots T$ para cada feature $v = 1, \dots V$ para la criptomoneda k con $k = 1, \dots K$. Luego, por ejemplo $\mathbf{k}_t^1(v), \mathbf{k}_t^2(v), \dots, \mathbf{k}_t^m(v)$ representa el valor de cierre de la variable v en el paso de tiempo t para cada una de las K criptomonedas de la muestra ($1 \leq k \leq K,\ k \in \mathbb{N}$). Los pasos de tiempo t representan observaciones referidas a cierres diarios por cada criptomoneda y cada variable OHLCc, de esta manera $t = 1 \dots T$ son los T períodos con datos de cierre para cada critomoneda. Finalmente, $v = 1 \dots V$ representa las V variables observadas.

3.2.1.2 Paso 2: Escalamiento de características

Dado que los valores observados para las características v contenidas en la matriz $\mathbf{K}^{(k)}$ muestran acentuadas diferencias en sus magnitudes, dándose el caso en el cual una criptomoneda puede exhibir precios de cierre inferiores a una decena de dolares, mientras que otras podrían superar las decenas de miles de dolares. Adicionalmente, dado que este es un mercado bastante volátil y afectado por violentas fluctuaciones, no es extraño encontrarse con valores que podrían calzar en la categoría de atípicos. Estas situaciones tienen el potencial de degradar el rendimiento y poder predictivo de los modelos, ya que los estimadores están generalmente diseñados bajo el supuesto que las características varían en escalas comparables. También es necesario recordar que a menudo se asume datos aproximadamente estándarizados ($\sim \mathcal{N}(\mu = 0,\ \sigma^2 = 1)$), en particular en el caso de estimadores basados en métricas y gradientes. Estas consideraciones hacen aconsejable la aplicación de un procedimiento estandarizado para escalar las características (v). En este caso el método de escalamiento aplicado es la normalización por rangos, que para cada criptomoneda k tiene el potencial de escalar cada feature v en el rango $[0, 1]$ en función de

sus valores máximos y mínimos de acuerdo a la siguiente expresión:

$$\mathbf{X}^{(k)} = \left(x_i^k(v)\right)_{i,v} = \frac{\mathrm{k}_i^k(v) - \min_{t=1\ldots T} \mathbf{k}_t^k(v)}{\max_{t=1\ldots T} \mathbf{k}_t^k(v) - \min_{t=1\ldots T} \mathbf{k}_t^k(v)} \qquad \text{para } i = 1, \ldots T \tag{51}$$

El procedimiento recomendado es utilizar los métodos de la clase $MinMaxScaler(\ldots)$ del paquete sklearn.preprocessing que es parte de la biblioteca Scikit-learn, ya que son robustas y eficientes implementaciones extensamente validadas por una numerosa comunidad de usuarios. Además, a través de un hiper-parámetro específico es posible especificar un rango distinto de $[0, 1]$, lo cual puede resultar útil ya que los algoritmos de descenso de gradiente mejoran su rendimiento si las variables están en el rango $[-1, 1]$, lo cual tendrá impacto en la convergencia del algoritmo.

3.2.1.3 Paso 3: Estructurar dataset

Para cada criptomoneda se generan arreglos de regresores ($\breve{\mathbf{X}}^{(k)}$) y valores objetivo ($\breve{\mathbf{y}}^{(k)}$), a partir de las series temporales OHLCc previamente escaladas ($\mathbf{X}^{(k)}$). De esta manera, el conjunto de datos de entrenamiento contará con n_{train} muestras, el conjunto de datos de pruebas contará con n_{test} muestras y el conjunto de datos para experimentos contará con n_{e} muestras respectivamente. Los arreglos de regresores se obtendrán aplicando una ventana móvil con p pasos de tiempo, con lo cual $(t, t+1, \ldots, t+p-1)$ son los correspondientes pasos de tiempo para cada marca de tiempo t, de manera que el dataset de entrenamiento $\breve{\mathbf{X}}_v^{(k)}$ corresponde a un arreglo de dos dimensiones $(n_{\text{train}} \times p)$, en la que cada muestra tendrá p observaciones para la variable v y criptomoneda k en las correspondientes marcas de tiempo, esto es para los valores escalados $(x_t^k(v), x_{t+1}^k(v), \ldots, x_{t+p-1}^k(v))$ obtenidos desde el arreglo bidimensional $\mathbf{X}^{(k)}$. Asimismo a cada fila del dataset de entrenamiento corresponderá un valor objetivo, los cuales son representados por el vector $\breve{\mathbf{y}}_v^{(k)}$ con n_{train} componentes para cada variable v y criptomoneda k en la muestra, de acuerdo a la siguiente regla:

$$\begin{aligned} \breve{\mathbf{X}}_v^{(k)} &= \left(x_t^k(v), \ldots, x_{t+p-1}^k(v)\right)_t, \ \forall t \leq n_{\text{train}} \\ \breve{\mathbf{y}}_v^{(k)} &= \left(x_{t+p}^k(v)\right)_t, \ \forall t \leq n_{\text{train}} \end{aligned} \tag{52}$$

En resumen, dada una criptomoneda k, cada fila t del arreglo bidimensional $\breve{\mathbf{X}}_v^{(k)}$ tiene p valores, uno por cada paso de tiempo de la variable v (serie temporal). De igual manera, cada componente t del vector $\breve{\mathbf{y}}_v^{(k)}$ tendrá el valor de la feature (variable) v correspondiente a la marca de tiempo $(t+p)$-ésima, $\forall t \leq n_{\text{train}}$.

Aplicando el mismo criterio, el conjunto de datos de validación para la criptomoneda k se construye con los correspondientes valores de la serie temporal v para las marcas de tiempo $t = n_{\text{train}} + 1, \ldots, n_{\text{train}} + n_{\text{test}} - 1)$, de acuerdo a:

$$\begin{aligned} &\bar{\mathbf{X}}_v^{(k)} = \left(x_t^k(v), \ldots, v_{t+p-1}^k(v)\right)_t \\ &\bar{\mathbf{y}}_v^{(k)} = \left(x_{t+p}^k(v)\right)_t \\ &\forall t, \ t = n_{\text{train}} + 1, \ldots, n_{\text{train}} + n_{\text{test}} - 1 \end{aligned} \tag{53}$$

con lo cual cada arreglo $\bar{\mathbf{X}}_v^{(k)}$ es una arreglo bidimensional de $(n_{\text{test}} \times p)$, mientras que los arreglos de valores objetivo $\bar{\mathbf{y}}_v^{(k)}$ tienen dimensión $(n_{\text{test}} \times 1)$.

De igual forma, el conjunto de datos de experimentos para la criptomoneda k se construye con los valores de la serie temporal v para las correspondientes marcas de tiempo $(n_{\text{train}} + n_{\text{test}}, \ldots, n_{\text{train}} + n_{\text{e}} - 1)$, de acuerdo a la siguiente regla:

$$\begin{aligned} &\widetilde{\mathbf{X}}_v^{(k)} = \left(x_t^k(v), \ldots, x_{t+p-1}^k(v)\right)_t \\ &\forall t,\ t = n_{\text{train}} + n_{\text{test}}, \ldots, n_{\text{train}} + n_{\text{e}} - 1 \end{aligned} \tag{54}$$

con lo cual cada arreglo $\widetilde{\mathbf{X}}_v^{(k)}$ es un arreglo bidimensional de $(n_{\text{e}} \times p)$.

Como último comentario, es necesario aclarar que en estricto rigor todos los arreglos bidimensionales de regresores son para cada serie temporal, en realidad casos especiales de un arreglo tridimensional de orden $(K \times \mathit{Muestras} \times p)$, donde $\mathit{Muestras} = [n_{\text{train}} \mid n_{\text{test}} \mid n_{\text{e}}]$ según sea el caso. De igual manera, los arreglos unidimensionales de valores objetivo, pueden considerarse un caso especial de un arreglo bidimensional de orden $(K \times \mathit{Muestras})$ para cada serie temporal, donde $\mathit{Muestras} = [n_{\text{train}} \mid n_{\text{test}}]$ según sea el caso.

3.2.1.4 Paso 4: Definir, configurar y entrenar modelos LSTM

Para la definición de los modelos, se optó por considerar dos arquitecturas, una compleja de dos capas LSTM ocultas y otra más simple compuesto sólo por una capa LSTM oculta. En cada caso se optó por considerar diversas configuraciones en función del número de posibles unidades ocultas disponibles en cada de ellas. Las siguientes tablas presentan la especificación de estos modelos.

La tabla 4 muestra la especificación de las capas ocultas de los modelos genéricos de dos capas ocultas. Todos estos modelos están provistos de una capa de entrada con las unidades necesarias para el recibir diez pasos de tiempo por vez. Adicionalmente se agrega una capa de descarte definido en 20% y una capa de salida "Dense" y función de activación "linear". En todos los modelos se monitorea la pérdida en función del "mean_absolute_error" y se aplica optimizador "adam".

Todo el monitoreo y evaluaciones utilizó la métrica MAE, procediendo a seleccionar los modelos que arrojaron mínimo valor en dicho indicador en validación durante el proceso de entrenamiento. Los modelos fueron entrenados con un número potencial de 500 épocas para cada criptomoneda y serie temporal. Se implementó monitoreo y tabulación del MAE en validación, con selección del mejor modelo para cada caso, con sensibilización de detención temprana para valores establecidos en 5, 10, 20, 50 y 100 respectivamente para hiper parámetro épocas de espera.

La tabla 5 muestra la especificación de la capa oculta de los modelos genéricos de una capa oculta. Todas las especificaciones ya presentadas para los modelos con dos capas ocultas son igualmente válidas en este caso. Es decir, estos modelos también están provistos de una capa de entrada con las unidades necesarias para el recibir diez pasos de tiempo por vez. Igualmente incorpora una capa de descarte definido en 20% y una capa de salida "Dense" con función de activación "linear". Asimismo, en todos los modelos se monitorea la pérdida en función del "mean_absolute_error" y se aplica optimizador "adam".

Modelo	Capa Oculta 1		Capa Oculta 2	
Id.	**Capa**	**Unidades**	**Capa**	**Unidades**
Modelo 128-64	LSTM	128	LSTM	64
Modelo 64-32	LSTM	64	LSTM	32
Modelo 32-16	LSTM	32	LSTM	16

Tabla 4: Modelos genéricos de dos capas ocultas

De igual manera, todo el monitoreo y evaluaciones utilizó la métrica MAE, procediendo a seleccionar los modelos que arrojaron mínimo valor en dicho indicador en validación durante el proceso de entrenamiento. Los modelos fueron entrenados con un número potencial de 500 épocas para cada criptomoneda y serie temporal. Se implementó monitoreo y tabulación del MAE en validación, con selección del mejor modelo para cada caso, con sensibilización de detención temprana con valores establecidos en 50 y 100 respectivamente para hiper parámetro épocas de espera.

Modelo	Capa Oculta	
Id.	**Capa**	**Unidades**
Modelo 128	LSTM	128
Modelo 64	LSTM	64
Modelo 32	LSTM	32

Tabla 5: Modelos genéricos de una capa oculta

La estructura de los modelos instanciados, difiere tanto en cantidad de capas y unidades ocultas así como en volumen de parámetros. En cada caso ambos parámetros, total y entrenables, presentan el mismo valor.

El resumen de la estructura de los modelos instanciados con dos capas ocultas, presentado en la tabla 6, muestra con claridad las capas constitutivas de cada modelo, las unidades ocultas en cada caso, la estructura de los flujos de datos entre capas y los parámetros y los parámetros totales en cada modelo, cuyo valor coincide con el total de parámetros entrenables.

Modelo	Input	LSTM 1		LSTM 2		Dense		Parám.
Unidades	**Est.**	**Est.**	**Parám.**	**Est.**	**Parám.**	**Est.**	**Parám.**	**Totales**
(32, 16)	(, 10, 1)	(, 10, 32)	4.352	(, 16)	3.136	(, 1)	17	7.505
(64, 32)	(, 10, 1)	(, 10, 64)	16.896	(, 32)	12.416	(, 1)	33	29.345
(128, 64)	(, 10, 1)	(, 10, 128)	66.560	(, 64)	49.408	(, 1)	65	116.033

Tabla 6: Resumen modelos de dos capas ocultas

Asimismo, el resumen de la estructura de los modelos instanciados con una capa oculta, ver tabla 7, muestra con claridad las capas constitutivas de cada modelo, las unidades ocultas

en cada caso, la estructura de los flujos de datos entre capas y los parámetros y los parámetros totales en cada modelo.

Modelo	**Input**	**LSTM**		**Dense**		**Parám.**
Unidades	**Est.**	**Est.**	**Parám.**	**Est.**	**Parám.**	**Totales**
(32)	(, 10, 1)	(, 32)	4.352	(, 1)	33	4.385
(64)	(, 10, 1)	(, 64)	16.896	(, 1)	65	16.961
(128)	(, 10, 1)	(, 128)	66.560	(, 1)	129	66.689

Tabla 7: Resumen modelos de una capa oculta

Con estos modelos instanciados y las diversas configuraciones de evaluación, producto de la sensibilización del hiper parámetro de detención anticipada, se procede al entrenamiento de los modelos utilizando el dataset descrito en la regla/ecuación 52, en tanto que la validación se realiza con el dataset definido en la regla 53. Es necesario recordar que como ya se definió en el preámbulo de estas reglas, cada muestra tendrá p observaciones (10 pasos de tiempo) para cada variable v (serie temporal OHLCc) y cada criptomoneda k.

3.2.1.5 Paso 5: Obtener predicciones

Como resultado del proceso de entrenamiento, se obtiene un elevado número de modelos posibles de aplicar, uno para cada criptomoneda y serie temporal, es decir como mínimo 340 modelos a validar para cada configuración en evaluación, si se consideran un universo de 68 criptomonedas y un mínimo de 5 series temporales OHLCc para cada una. Considerando las 21 configuraciones en evaluación, se tiene que el número potencial de modelos a considerar asciende a 7.140 como mínimo, a partir de los cuales se optó por seleccionar las cinco configuraciones que minimizan el costo en evaluación.

La principal desventaja de agregar capas ocultas es que la retropropagación pierde efectividad [39] por otro lado, en concordancia con el principio de resolución de problemas conocido como "la navaja de Occam", la arquitectura de red más simple que no incrementa el costo es la mejor, puesto que se entrenará más rápido y demandará menos recursos computacionales. Por otro lado, si al incrementar capas se reduce el costo entonces es necesario incrementarlas para mejorar el rendimiento de la red. Estos son los criterios que se han tenido en mente para la selección de los cinco modelos a partir de los cuales se implementa el resto del proceso.

Es por ello que en atención a lo expuesto y para simplificar el proceso, se procede a seleccionar sólo un modelo general en cada caso, seleccionando los cinco modelos con menor costo promedio. Con esto se evita la complejidad asociada a administrar un elevado número de modelos específicos en las siguientes etapas de la evaluación.

A partir de los modelos seleccionados que han sido previamente entrenados, la generación de predicciones es un proceso bastante directo. Es por ello que las predicciones serán obtenidas mediante aplicación de los cinco modelos previamente seleccionados, de esta manera se obtendrán cinco grupos de predicciones para cada criptomoneda y serie temporal

OHLCc. Estas predicciones corresponden al período de experimentación, el cual abarca los 10 pasos de tiempo finales de la serie temporal que constituye cada atributo.

Como se aprecia en la figura 21, el proceso de experimentación considera por una parte la generación del dataset sobre el cual se aplicará el modelo para obtener la predicción. En el caso de la experimentación, estas predicciones abarcan diez pasos de tiempo. En una aplicación real en línea, se trata de la predicción out-of-sample del día siguiente $t + 1$ para cada criptomoneda y serie temporal en el tiempo t, posteriormente en el próximo paso de tiempo, es decir en $t + 1$, se tiene la oportunidad de ajustar la predicción anteriormente obtenida, en el tiempo t, con el valor real para el cierre actual y obtener la predicción del próximo paso de tiempo y así sucesivamente.

Para cada criptomoneda, este proceso arroja como resultado un conjunto de series temporales que incluyen las predicciones para el período de experimentación. En lo que sigue, estas series temporales reciben el nombre de atributos. La siguiente tabla muestra esto atributos:

#	**Atributo**	**Tipo**	**Definición de atributo**
1	Open	Básico	Valor de apertura del día
2	Low	Básico	Valor menor en el día operación
3	High	Básico	Valor mayor en el día operación
4	Close	Básico	Valor de cierre del día
5	Log Cap	Básico	Logaritmo de la capitalización de mercado
6	Day Profitability	Generado	$Close/Open$
7	Day Variability	Generado	$High/Low$

Tabla 8: Atributos en proceso de predicción experimentos

Como se aprecia en la tabla 8, los atributos 6 y 7, se obtienen a partir de atributos base que ya cuentan con las predicciones para el período experimental.

3.2.1.6 Paso 6: Generar datasets de atributos

Las tareas a desarrollar en este punto dicen relación con generar los datasets base de tres atributos sin repetición, a partir de los atributos presentes en la tabla 8 y que se constituirán en insumo del proceso lingüístico con enfoque DLDM.

Este desarrollo contempla la evaluación de conjuntos de datos compuestos por una combinación de atributos, en este sentido, el alcance definido implica la generación, en particular, de todos los datasets posibles formados por combinación de tres atributos sin repetición. Los atributos base, corresponden a las series temporales obtenidas producto del proceso predictivo presentadas previamente en la tabla 8. Como resultado de este proceso se obtienen inicialmente $C_3^7 = 35$ datasets de tres atributos, en adelante datasets de atributos.

Una vez estructurados estos conjuntos de datos de tres atributos, se procede a preprocesar esto datasets hasta obtener los conjuntos de datos definitivos que luego serán utilizados como insumo en el modelo de decisión lingüística con enfoque DLDM.

3.2.1.7 Paso 7: Preproceso de atributos

El escalamiento de features es una buena práctica, recomendada para sacar el máximo provecho posible de la mayoría de los algoritmos de aprendizaje automático. Entre otras acciones recomendadas, llevar las features a una escala comparable, en particular si estas finalmente fluctúan ya sea en $[-1, 1]$, $[0, 1]$ e incluso $[-3, 3]$ mejora la eficacia de estos algoritmos, incrementando la validez de las predicciones obtenidas. En este sentido, una de las interrogantes que se desea resolver en este trabajo es, si esta clase de preprocesamiento tiene algún impacto en mejorar el ranking dinámico producto de aplicar el modelo de decisión lingüístico a atributos preprocesados v/s el procedimiento implementado en el benchmark [65], que no hace uso de estas técnicas. Otra interrogante es, si variaciones en la combinación de atributos y/o intervenciones especificas no consideradas en la implementación del benchmark ([65]) generan o no impactos positivos sobre el modelo de decisión lingüístico y, consecuentemente en el ranking dinámico.

Conviene mencionar que se ha observado que el método de normalización utilizado en el proceso de decisión lingüístico con enfoque DLDM, tiende a verse afectado por la magnitud de valores específicos presentes en los atributos, que en sucesivas aplicaciones del método muestra que esto puede afectar la selección de los factores de beneficio o costo, según sea el caso, incrementando la posibilidad de seleccionar valores extremos como factor de normalización.

En resumen, el objetivo es transformar los atributos de manera que sus magnitudes puntuales no afecten al proceso de decisión lingüístico y, que consecuentemente el ranking dinámico se vea afectado lo menos posible por atributos con mayor peso producto de la magnitud de su valor.

A través de la experimentación que se realiza en este trabajo, se determinará el efecto en el rendimiento final de la cartera de inversión, producto de diversos métodos estándares de escalamiento y preprocesamiento (ingeniería de features) de los atributos generados por el proceso predictivo.

Los escenarios considerados en esta etapa de la experimentación son: a) Experimentación con los datasets de atributos sin aplicar preprocesamiento alguno, b) Experimentación con datasets compuestos únicamente por series temporales de atributos generados (variaciones), c) Experimentación con los datasets de atributos preprocesados.

a) Experimentación sin preprocesamiento de atributos

En este escenario de experimentación, el proceso de decisión lingüístico recibe como insumo los datasets de atributos sin transformación. Es decir, sin aplicar preprocesamiento alguno a las series temporales que los componen. Este escenario es equivalente a la implementación realizada en el benchmark, aunque extendida a todas las combinaciones de atributos.

b) Experimentación sólo con atributos de variación

En este caso, la experimentación se realiza alimentado al proceso de decisión lingüístico con datasets compuestos únicamente por las series temporales de los atributos generados. Es decir, las variaciones Day Profitability y Day Variability, incluyendo una tercera componente

constante. En general, los valores que exhiben estas series temporales fluctúan en torno a 1, ver definición de los mismos en tabla 8. Por lo mismo se desea validar el efecto que estos atributos por sí solos pueden tener sobre la generación del ranking dinámico, considerando para ello un tercer atributo de beneficio constante.

c) Experimentación con escalamiento de atributos

Este escenario de experimentación contempla que el proceso de decisión lingüístico consuma todos los datasets de atributos, pero esta vez transformados mediante algoritmos de escalamiento aplicados a las series temporales individuales que componen cada dataset.

Para la implementación de estos experimentos se optó por utilizar algunas de las clases del paquete preprocessing, disponible en la biblioteca de código abierto Scikit-learn o sklearn, por tratarse de implementaciones robustas y ampliamente validadas por una comunidad de usuarios, científicos y especialistas de alcance mundial. Tienen la gran ventaja que una vez instanciados y entrenados, la misma instancia de transformación se puede aplicar a datos no vistos (out-of-sample). En lo que se refiere a este experimento, se optó por utilizar sólo algunas de las clases disponibles en el paquete. La tabla 9 presenta las clases seleccionadas junto a una breve descripción de las mismas proveniente de la documentación disponible en el proyecto.

A partir de estos casos de experimentación se desprende que, para cada criptomoneda se realizarán 211 experimentos, ya que el caso (a) genera 35 experimentos, el caso (b) genera 1 experimento y, el caso (c) genera 175 experimentos.

3.2.2 Etapa 2: Modelo de Decisión Lingüístico

En Dynamic Linguistic Decision Making [65] se propone extender el método MP-MADM a través de los siguientes pasos: (1) Construir un modelo de decisión lingüística, (2) Normalización de evaluaciones para atributos de costo y beneficio, (3) Cálcular de las variables lingüísticas aplicando función de pertenencia, (4) Obtener matrices lingüísticas temporales, dado el proceso estocástico de precios de las criptomonedas, (5) Agregación de información lingüística de los atributos, utiliza vector de pesos que representa la importancia de los atributos y aplica el operador LWA (Linguistic Weighted Averaging) para agregar información de las filas de las matrices temporales de decisión, (6) Agregación dinámica de la información temporal, para obtener puntuaciones finales a través de la aplicación del operador DLWA (Dynamic Linguistic Weighted Averaging), (7) Clasificación de alternativas de mejor a peor de acuerdo a la puntuación obtenida en el proceso de agregación lingüística. A continuación se describe en detalle el método propuesto.

3.2.2.1 Paso 1: Construir el Modelo de Decisión Lingüístico

El objetivo es apoyar a a los tomadores de decisión en la selección de la mejor alternativa $\mathcal{A}_i$ desde un conjunto de n alternativas posibles $\mathcal{A}_1, \ldots, \mathcal{A}_i, \ldots, \mathcal{A}_n$. Cada alternativa $\mathcal{A}_i$, $i = 1, \ldots, n$, es evaluada de acuerdo al criterio o atributo G^j desde un conjunto de m atributos $G^1, \ldots, G^j, \ldots, G^m$ en diferentes momentos del tiempo (pasos de tiempo). El

Scaler	Descripción
MinMaxScaler	Escala y transforma cada feature individualmente a un rango determinado, por defecto $[0, 1]$. Permite especificar el rango, indicando para ello el mínimo (min, por defecto 0) y máximo (max, por defecto 1). Este escalador funciona mejor para los casos en que la distribución no es Gaussiana o la desviación estándar es muy pequeña. Luego, es sensible a outliers. En una serie esta transformación es: $X_std = \big(X - \min(X)\big)/\big(\max(X) - \min(X)\big)$ $X_scaled = X_std \times (max - min) + min$
MaxAbsScaler	Escalada individualmente cada feature en función de su valor máximo absoluto. Luego, automáticamente escala las features al rango $[-1, 1]$. En datos positivos se comporta igual que $MinMaxScaler$. Está diseñado para datos que ya están centrados en 0 o bien para datos de tipo sparce sin afectar dicha condición. También es sensible a la presencia de outliers. En una serie tenemos que: $X_scaled = X/\big(\max\big(\text{abs}(X)\big)\big)$
StandardScaler	Estandariza individualmente las features restando la media (u) y escalando a varianza (s) unitaria. Generalmente se considera cuando los valores de las features tienen grandes diferencias entre sus rangos. Al igual que $MinMaxScaler$, no es robusta en presencia de outliers. Está definido por: $z = \big(x - u\big)/s$
PowerTransformer	Se encuentra en la categoría de transformaciones no lineales. Se trata de una familia de transformaciones paramétricas y monótonas, se aplican para hacer que los datos sean más gaussianos, pues resulta útil para modelar problemas relacionados con la heterocedasticidad u otras situaciones en las que se desea normalidad. Por defecto aplica normalización con media 0 y varianza unitaria.
RobustScaler	Escala individualmente las features, de manera que elimina la mediana y escala los datos de acuerdo con el rango de cuartiles de las muestras, aplica por defecto rango intercuartil. Es robusto ante outliers y arroja mejores resultados que estandarizar eliminando media y escalando a varianza unitaria.

Tabla 9: Scalers seleccionados para experimentos

modelo considera una ventana de tiempo $\mathbf{t} = \{t_1, t_2, \ldots, t_p, \ldots, t_P\}$ con un total de P períodos para la evaluación, donde t_p es la marca de tiempo en el periodo p, $p = 1, 2, \ldots, P$.

3.2.2.2 Paso 2: Normalización de las Evaluaciones

Sea $x_i^j(t)$ el puntaje obtenido en la evaluación de la alternativa $\mathcal{A}_i$, respecto al criterio G^j en el tiempo t. Se asume que $x_i^j(t)$ es un valor positivo. Con el propósito de utilizar el enfoque guiado por datos, es necesario normalizar las evaluaciones desde $x_i^j(t)$ a $\hat{x}_i^j(t)$ en

función a si se trata de un atributo de beneficio o costo. Para el caso de atributos de beneficio G^j en el tiempo t, se normalizan todas las alternativas $\mathcal{A}_i$ como se muestra a continuación:

$$\hat{x}_i^j(t) = \frac{x_i^j(t)}{\max_{i=1\ldots n} x_i^j(t)} \qquad \text{para } i = 1, \ldots, n \tag{55}$$

Por el contrario, en el caso de atributos de *costo* G_j en el tiempo t, la normalización de todas las alternativas $\mathcal{A}_i$se calcula de la siguiente manera:

$$\hat{x}_i^j(t) = \frac{\min_{i=1\ldots n} x_i^j(t)}{x_i^j(t)} \qquad \text{para } i = 1, \ldots, n \tag{56}$$

3.2.2.3 Paso 3: Obtención de Variables Lingüísticas

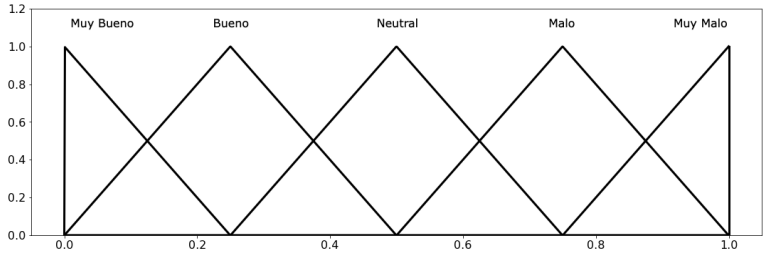
Las variables lingüísticas son aquellas cuyos valores corresponden a palabras o sentencias de un lenguaje natural o artificial. Por lo cual, los atributos $G^1, \ldots, G^j, \ldots, G^m$ son modelados como variables lingüísticas. Cada variable lingüística está compuesta por un conjunto de términos lingüísticos s_q, donde se utiliza una escala $\overline{S}_Q$ de evaluación lingüística aditiva [74], es decir,

$$\overline{S}_Q = \{s_{-Q}, \ldots, s_0 = neutral, \ldots, s_Q\} \qquad Q \in \mathbb{N} \tag{57}$$

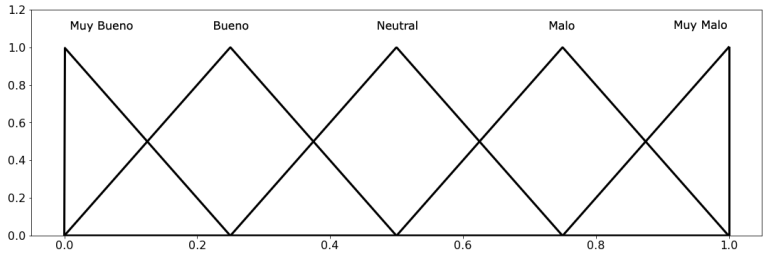
Existen una variedad de funciones para representar términos lingüísticos, la forma triangular, forma trapezoidal y forma gaussiana se encuentran entre las más comunes. Esta funciones son definidas paramétricamente y su valores pueden obtenidos directamente de los datos o a partir de grupos de expertos. Si el término lingüístico es modelado con una forma triangular, la función de pertenencia queda definida como sigue:

$$\mu(x) = \begin{cases} \frac{x-a}{b-a} & \text{si } a \leq x < b \\ \frac{x-c}{b-c} & \text{si } b \leq x < c \\ 0 & \text{en otro caso} \end{cases} \tag{58}$$

En este diseño se ha optado por utilizar cuantiles par obtener los valores de los parámetros de los términos triangulares. Los cuantiles son puntos de corte que dividen el rango en segmentos con igual cantidad de observaciones para cada uno. El rango está restringido al intervalo $[0, 1]$, ya que los atributos fueron normalizados con las ecuaciones presentadas en (55) y (56). Si las formas triangulares son seleccionadas para los K términos lingüísticos, entonces las cúspides (picos) de los triángulos (valores b en la ecuación (58)) se ubicaron en los siguientes puntos de corte: el mínimo, los percentiles definidos por $\frac{k}{K-1} \cdot 100$, $k = 1, \ldots, K-1$ y, el máximo. Los puntos extremos de los triángulos (valores a y c en la ecuación (58)) se ubicaron en la cúspide del siguiente triángulo (triángulo de la izquierda y derecha respectivamente). La figura 22 muestra las variables lingüísticas de tipo beneficio (figura 22a) y las variables de tipo costo (figura 22b), ambas con 5 términos lingüísticos: *Muy Malo*, *Malo*, *Neutral*, *Bueno* y *Muy Bueno*.



(a) Variables lingüísticas para atributo de *beneficio*



(b) Variables lingüísticas para atributo de *costo*

***Figura* 22:** Términos lingüísticos triangulares para variables lingüísticas de beneficio (22a) y costo (22b)

3.2.2.4 Paso 4: Obtener Matrices Lingüísticas temporales

Dado que el comportamiento de las criptomonedas es variable en el tiempo, entonces es necesario calcular las variables lingüísticas G^j e cada instante de tiempo t. El puntaje normalizado $\hat{x}_i^j(t)$ para cada alternativa $\mathcal{A}_i$ en el tiempo t se utiliza para obtener el vector de evaluaciones de pertenencia $\mathbf{z}_i^j(t) = [z_1, z_2, \ldots, z_K]$, donde $z_k = \mu_k^j(\hat{x}_i^j(t))$ es la función de pertenencia asociada al k-ésimo término lingüístico de la varible lingüística G^j. El valor lingüístico s_q relacionado con la evaluación $\hat{x}_i^j(t)$ se obtiene como el producto punto entre el vector de evaluación de pertenencia $\mathbf{z}_i^j(t)$ y el vector $[-Q, \ldots, 0, \ldots, Q]$ o $[Q, \ldots, 0, \ldots, -Q]$ dependiendo si se trata de un atributo de beneficio o de costo, es decir:

$$\begin{aligned} \text{Beneficio:} \quad q &= \mathbf{z}_i^j(t) \cdot [-Q, \ldots, 0, \ldots., Q] \\ \text{Costo:} \quad q &= \mathbf{z}_i^j(t) \cdot [Q, \ldots, 0, \ldots, -Q] \end{aligned} \tag{59}$$

Las matrices de decisión lingüística $\mathbf{S}_t$, $t = 1 \ldots P$, son calculadas a cada instante de tiempo t, donde el elemento s_{ij}^t ubicado en la i-ésima fila y las j-ésima columna de la matriz de decisión $\mathbf{S}_t$, $t = 1 \ldots P$, corresponde a la evaluación lingüística obtenida con la ecuación (59) para el puntaje $\mathbf{z}_i^j(t)$.

3.2.2.5 Paso 5: Agregación de Información Lingüística de los Atributos

El proceso de agregación lingüística se realiza en dos etapas. En la primera etapa, a cada instante de tiempo se agrega la información lingüística en la dimensión de cada atributo. En este sentido, un grupo de expertos define el vector de pesos $\mathbf{W} = [W_1, W_2, \ldots, W_m]$ que

permite cuantificar la importancia de los atributos. Para obtener la matriz única de decisión $\mathbf{R}$, se aplica el operador promedio ponderado lingüístico (LWA) para agregar la información de cada fila de las matrices temporales de decisión $\mathbf{S}_t$. El elemento r_{it} de la i-ésima fila y t-ésima columna de la matriz de decisión $\mathbf{R}$ se calcula como sigue:

$$\begin{aligned} r_{it} &= \text{LWA}(s_{i1}^t, s_{i2}^t, \ldots, s_{im}^t) \\ &= W_1 s_{i1}^t \oplus W_2 s_{i2}^t \oplus \ldots W_m s_{im}^t \end{aligned} \tag{60}$$

donde $\oplus$ representa el operador promediador lingüístico [79] tal que satisfaga:

$$\begin{aligned} s_{q_1} \oplus s_{q_2} &= s_{q_1+q_2}, \\ &= s_{q_2+q_1}, \\ &= s_{q_2} \oplus s_{q_1}. \end{aligned}$$

3.2.2.6 Paso 6: Agregación dinámica de la información temporal

Luego, en la segunda etapa, se utiliza el operador promediador lingüístico ponderado (DLWA) para agregar la información de la matriz lingüística de decisión $\mathbf{R}$ para obtener el puntaje final A_i de las alternativas $\mathcal{A}_i$, $i = 1..n$. El operador *DLWA* se define como sigue:

$$\begin{aligned} A_i &= \text{DLWA}(r_{i1}, r_{i2}, \ldots, r_{iP}) \\ &= \omega_1 r_{i1} \oplus \omega_2 r_{i2} \oplus \cdots \oplus \omega_P r_{iP}, \end{aligned} \tag{61}$$

La clave del dinamismo del operador DLWA operator [76] es la determinación del vector de pesos $\omega = [\omega_1, \omega_2, \ldots, \omega_P]$, donde se ha propuesto [80] la función básica monotónica de intervalo unitario (BUM). Sea $f : [0,1] \to [0,1]$ una función BUM (con $f(0) = 0, f(1) = 1, f(x) \geq f(y)$ si $x > y$). El vector de pesos ω es definido como:

$$\omega_p = f\left(\frac{p}{P}\right) - f\left(\frac{p-1}{P}\right), p = 1, 2, \ldots, P \tag{62}$$

donde la secuencia $\{\omega_p\}$ es una secuencia monotónica creciente (o decreciente), $\omega_{p+1} > \omega_p$ ($\omega_{p+1} < \omega_p$) con $p = 1, 2, \ldots, P-1$. Por ejemplo, Xu [76] ha propuesto la función $f(x) = \frac{e^{\alpha x}-1}{e^{\alpha}-1}$, $\alpha > 0$, con la cual se obtiene el siguiente vector de pesos (ponderadores):

$$\omega_p = \frac{e^{\frac{\alpha p}{P}} - 1}{e^{\alpha} - 1} - \frac{e^{\frac{\alpha(p-1)}{P}} - 1}{e^{\alpha} - 1} = \frac{e^{\frac{\alpha p}{P}} - e^{\frac{\alpha(p-1)}{P}}}{e^{\alpha} - 1} \tag{63}$$

3.2.2.7 Paso 7: Ranking de las Alternativas

Por último, las alternativas son ordenadas de mejor a peor de acuerdo al puntaje obtenido en la agregación lingüística, esto es, $\mathcal{A}_{(1)} \succeq \mathcal{A}_{(2)} \succeq \cdots \succeq \mathcal{A}_{(n)}$ de manera que los puntajes están ordenados como sigue $A_{(1)} \geq A_{(2)} \geq \cdots \geq A_{(n)}$ (Notar que el sub-índice (i) hace referencia a la i-ésima posición).

3.2.3 Etapa 3: Experimentación con estrategias simuladas

En esta etapa se utiliza el modelo de decisión lingüístico para implementar un proceso dinámico de clasificación de criptomonedas, que en cada iteración hace posible seleccionar para el próximo período de inversión, el instrumento ubicado en la primera posición del ranking.

3.2.3.1 Estrategias de inversión simuladas

En esta etapa de la experimentación, según se aprecia en la figura 20, se implementan dos métodos principales de inversión, cada uno de los cuales se compone de dos estrategias experimentales. Estas métodos de inversión son 'buy-and-hold' y 'buy-and-sell' respectivamente.

El método de 'buy-and-hold' corresponde a un estilo de inversión más bien ingenuo, en el cual se toman posiciones al inicio del proceso de inversión simulado y sólo se vende al finalizar el período de inversión, como símil del período de maduración de la inversión. En esta simulación, las estrategias que implementan este método se denominan **BTC** que invierte sólo en Bitcoin y **Whole** que invierte en todos los instrumentos del mercado.

Por el contrario, el método 'buy-and-sell' consiste en un estilo de inversión dinámico, que va realizando actualización diaria de la selección de criptomoneda a medida que progresa el proceso de inversión simulado, lo cual ciertamente requiere la venta total de la criptomoneda en existencia, para tomar una nueva posición a través de la compra de la nueva criptomoneda seleccionada. En esta simulación, las estrategias implementadas se denominan **Random** que invierte con selección aleatoria de criptomoneda y **DLDM** que sólo invierte en la criptomoneda ubicado en la primera posición del ranking dinámico.

En estos experimentos interesa comparar el rendimiento de la estrategia DLDM respecto a un benchamark interno de estrategias compuesto por las estrategas Whole, BTC y Random, para determinar en qué condiciones de preproceso y con qué datasets de 3-tuplas se obtiene el mejor resultado posible. Ciertamente, se espera que el proceso cuantitativo supere el rendimiento del benchmark de estrategias. La siguiente es una descripción mas detallada de estas carteras:

3.2.3.2 Proceso de inversión

A continuación, se encuentra una presentación estructurada y de nivel agregado del conjunto de procesos y acciones que implementan el proceso de inversión simulado.

Inicio: `Para cada Tupla en` *`datasets de atributos`* `:`

1. **Aplicar** modelo lingüístico
2. **Obtener** ranking criptomonedas para el período de inversión
3. **Asignar** monto inicial de us$100:

   $saldo_dldm \leftarrow 100$

Estrategia	Descripción
BTC	Estrategia que considera sólo inversión en Bitcoin, por tratarse de la criptomoneda más conocida y de mayor liquidez del mercado. Aplica estrategia 'buy-and-hold'
Whole	Estrategia que consiste en invertir proporcionalmente en todos los instrumentos del mercado, es decir aplica a toda la muestra de criptomonedas, con inversión equitativa en los activos que componen dicho mercado. Aplica un enfoque del tipo 'buy-and-hold'
Random	Inversión con selección aleatoria del activo. Invierte todo el capital disponible en el un activo seleccionado al azar. Aplica enfoque del tipo 'buy-and-sell'.
DLDM	Estrategia consistente en invertir en el activo mejor posicionado en el ranking dinámico, en cada iteración del proceso de decisión lingüístico. Invierte todo el capital disponible en dicho instrumento, luego implementa un enfoque del tipo 'buy-and-sell'.

Tabla 10: Carteras en experimentación

$saldo_random \leftarrow 100$

$saldo_btc \leftarrow 100$

$saldo_whole \leftarrow 100$

4. Al inicio del período de inversión:

   Cartera BTC: **Comprar** *saldo_btc* a precio de apertura

   Cartera Whole: **Comprar** *saldo_whole* a precio de apertura

5. Para cada día durante el *período de inversión*:

   5.1. En la apertura del mercado:

      5.1.1. En cartera Random:

         Seleccionar aleatoriamente una criptomoneda

         Comprar *saldo_random* en dicha criptomoneda a precio de apertura

      5.1.2. En cartera DLDM:

         seleccionar criptomoneda en primer lugar en el ranking del día

         Comprar *saldo_dldm* en dicha criptomoneda a precio de apertura

   5.2. Al cierre del mercado:

      5.2.1. Carteras 'buy-and-sell':

         Vender instrumento en cartera Random a precio de cierre

         Vender instrumento en cartera DLDM a precio de cierre

         Actualizar *saldo_random* y **tabular** rendimiento del día

         Actualizar *saldo_dldm* y **tabular** rendimiento del día

      5.2.2. Carteras 'buy-and-hold':

Cartera BTC: **Valorar y tabular** rendimiento del día
Cartera Whole: **Valorar y tabular** rendimiento del día

6. Al finalizar período de inversión:
   6.1. Cartera BTC: **Vender** a precio de cierre
   6.2. Cartera Whole: **Vender** a precio de cierre
7. Para todas las carteras:
   7.1. **Actualizar** saldos
   7.2. **Tabular** rendimiento
8. Mientras queden tuplas volver a **Inicio**

Como se aprecia, la ejecución del proceso de inversión simulado, implica realizar experimentos con todas las tuplas (datasets de atributos) cuya estructuración y composición ya fue presentada. Con estos experimentos se desea determinar el rendimiento que exhibirán las carteras presentadas en el punto previo, e identificar las tuplas y preproceso de las mismas que permiten obtener mejor rendimiento en la cartera DLDM respecto de las restantes carteras y también respecto al benchmark.

En cada iteración del proceso, mientras queden 3-tuplas, se realiza un nuevo experimento y se tabula el rendimiento de cada cartera. El análisis de estos datos, finalmente permitirá identificar los experimentos, carteras y tuplas con mejor rendimiento comparado.

4 Resultados

4.1 Entrenamiento de modelos LSTM

Los resultados que se presentan a continuación dicen relación con el proceso de entrenamiento de los modelos LSTM para cada serie temporal OHLC de cada una de las **68** criptomonedas incluidas en la muestra. La presentación de estas series temporales se encuentra en el punto y tabla 2 (Series temporales para entrenamiento y validación), por lo cual no es necesario extenderse más en este punto, salvo mencionar los nombres de las **5** columnas del caso: Open, High, Low, Close y Log Cap. las series temporales 'Day Profitability' y 'Day Variability' se calculan a partir de las series OHLC, por lo cual no se consideran directamente en el proceso.

Adicionalmente, la descripción detallada que se encuentra en el punto 3.2, detalla el proceso general y los modelos considerados en el proceso de entrenamiento. En la misma sección se encuentra el detalle de los modelos, según la arquitectura que se trate.

Para el caso de los modelos secuenciales compuestos por dos capas ocultas LSTM, los 3 modelos opcionales considerados en el proceso de entrenamiento se detallan en la tabla 4. El proceso de entrenamiento, a su vez incluye sensibilización del hiperparámetro de detención temprana para 5 valores opcionales predefinidos, los cuales corresponden a: 5, 10, 20, 50 y 100 respectivamente. En resumen, se tiene: **68** criptomonedas, **5** series temporales OHLC para cada una de ellas en entrenamiento, **3** opciones de modelos con dos capas ocultas LSTM a entrenar y **5** valores opcionales de detención anticipada a aplicar durante el entrenamiento de cada modelo, todo lo cual resulta en un total de **5100** modelos entrenados, de los cuales sólo se preservó el mejor en cada caso.

Aplican criterios similares al caso de los modelos secuenciales compuestos por una capa oculta LSTM, las 3 opciones de modelos consideradas en este caso se detallan en la tabla 5. En este caso sólo se consideraron 2 posibles valores para detención temprana, correspondientes a 50 y 100 respectivamente. Aplicando un criterio similar al caso previo, tenemos en resumen: **68** criptomonedas, **5** series temporales OHLC para cada una de ellas en entrenamiento, **3** opciones de modelos con una capa oculta LSTM a entrenar y **2** valores opcionales de detención anticipada a aplicar durante el entrenamiento de cada modelo, todo lo cual resulta en un total de **2040** modelos entrenados, de los cuales sólo se preservó el mejor en cada caso.

Luego se tiene que en total se entrenaron **7140** modelos en este proceso, tabulando en cada caso el MAE obtenido en validación de cada modelo LSTM. La tabla resultante tiene 1428 filas y las columnas identifican a la criptomoneda, el modelo entrenado y el valor del índice MAE para el modelo de cada serie temporal. A continuación se presenta una vista parcial de dicha tabla ya que no es factible presentar la tabla completa.

La tabla 11 muestra una vista parcial del índice MAE tabulado para cada modelo. La columna *Id* representa el identificador numérico asociado a la criptomoneda en particular. La columna *Cripto* muestra la criptomoneda a que corresponden las series temporales para las cuales se entrenaron y seleccionaron los mejores modelos. La columna *Tipo modelo* identifica el modelo al que corresponde el valor de MAE, en ella el primer número indica la

#	Id	Cripto	Tipo modelo	Open	High	Low	Close	Log Cap
1	1	BTC	128_100	191.7253	150.4380	144.5059	182.4822	0.0205
2	1	BTC	128_50	193.5554	146.6888	147.0234	184.1636	0.0202
3	1	BTC	128_25	214.7705	150.8955	214.6077	239.6542	0.0213
4	1	BTC	128_10	233.6113	231.2320	241.8357	283.9120	0.0282
5	1	BTC	128_5	256.1699	253.5456	239.0136	263.5831	0.0295
6	1	BTC	64_100	187.7053	150.0633	145.7320	194.8335	0.0210
7	1	BTC	64_50	193.4643	152.9857	144.3209	244.2948	0.0209
8	1	BTC	64_25	215.4572	153.0637	235.3222	253.1786	0.0281
9	1	BTC	64_10	249.0316	242.7381	241.4712	253.9384	0.0282
10	1	BTC	64_5	400.8709	252.9661	402.0526	267.1348	0.0287
11	1	BTC	32_100	205.1020	153.8514	163.8963	247.4391	0.0227
12	1	BTC	32_50	204.2493	155.1172	217.5083	252.9825	0.0282
13	1	BTC	32_25	236.2221	230.0276	231.2053	253.7158	0.0284
14	1	BTC	32_10	241.0742	247.6969	230.5557	252.4122	0.0283
15	1	BTC	32_5	395.3809	328.7831	405.3572	396.8558	0.0467
16	1	BTC	128_1_100	192.6532	150.2848	151.4753	186.3277	0.0207
17	1	BTC	128_1_50	195.9074	150.6578	213.9000	179.6322	0.0207
18	1	BTC	64_1_100	200.0705	154.3698	158.5176	192.9953	0.0216
19	1	BTC	64_1_50	207.7265	155.8551	160.4089	220.2416	0.0215
20	1	BTC	32_1_100	201.0967	170.8740	182.2454	215.2325	0.0230
21	1	BTC	32_1_50	217.3817	177.6746	213.6247	230.4264	0.0244
...	...	...	...	...	...	...	...	...
1414	68	ZEN	64_50	1.6499	1.4247	1.5593	1.6762	0.0525
1415	68	ZEN	64_25	1.6305	1.4763	1.6083	1.7195	0.0535
1416	68	ZEN	64_10	2.8379	1.4885	1.4997	1.6736	0.0773
1417	68	ZEN	64_5	2.9303	2.6672	2.7707	2.9161	0.0896
1418	68	ZEN	32_100	1.6705	1.4819	1.6318	1.6416	0.0518
1419	68	ZEN	32_50	1.7119	1.4107	1.5249	1.6640	0.0535
1420	68	ZEN	32_25	1.5740	1.3916	1.5473	1.6266	0.0507
1421	68	ZEN	32_10	2.7456	2.9470	2.7929	2.7915	0.0753
1422	68	ZEN	32_5	2.8818	3.1009	3.2823	3.0366	0.0821
1423	68	ZEN	128_1_100	1.6787	1.0953	1.1062	1.5408	0.0465
1424	68	ZEN	128_1_50	1.6750	1.1813	1.1276	1.5409	0.0518
1425	68	ZEN	64_1_100	1.6450	1.2532	1.2718	1.6692	0.0517
1426	68	ZEN	64_1_50	1.6900	1.3371	1.2449	1.6465	0.0526
1427	68	ZEN	32_1_100	1.6436	1.2738	1.3285	1.6522	0.0512
1428	68	ZEN	32_1_50	1.6458	1.3583	1.5140	1.6830	0.0539

Tabla 11: Vista parcial MAE de los modelo LSTM

cantidad de unidades ocultas en la primera capa oculta y el último número indica el valor de detención anticipada en el entrenamiento, de existir un 1 entre ambos indica que el modelo

sólo tiene una capa oculta. Las restantes columnas corresponden a las series temporales de cada criptomoneda y el valor de MAE tabulado para cada una de ellas.

4.2 Selección de modelos LSTM

Por simplicidad se optó por seleccionar las 5 implementaciones de modelos con menor valor promedio en la métrica MAE. Como criterio adicional, se podría optar por seleccionar en primer lugar los modelos de arquitectura más simple posible, es decir los de una capa oculta con la menor cantidad de unidades ocultas. Los modelos cuyas arquitecturas consideran dos capas ocultas pueden resultar más costosos de entrenar pero también podrían exhibir mejores métricas. Con estos modelos se procedió a la implementación del proceso que incluye el modelo de decisión DLDM y el ranking dinámico de criptomonedas en función de las predicciones generadas por los modelos seleccionados, lo que finalmente conduce a la ejecución de los experimentos y a la comparación de las diferentes estrategias simuladas de inversión.

#	Tipo modelo	Open	High	Low	Close	Log Cap
1	128_100	6.7301	5.2535	5.4620	6.6542	0.0475
2	128_1_100	6.8685	5.4722	5.6222	6.6475	0.0488
3	128_50	6.8869	5.7291	5.6681	6.7510	0.0509
4	64_100	6.8888	5.6428	5.6992	6.9143	0.0515
5	128_1_50	6.9555	5.5510	6.5018	6.7998	0.0502
6	64_1_100	7.0359	5.8655	5.8244	6.9151	0.0606
7	32_1_100	7.0517	6.4039	6.3423	7.3334	0.0566
8	64_50	7.0537	6.1534	5.6802	7.8492	0.0673
9	32_100	7.3013	6.4146	6.0031	7.8218	0.0686
10	64_1_50	7.3492	6.1454	6.1457	7.3986	0.0597
11	128_25	7.3918	5.8622	7.0322	7.6029	0.0676
12	32_1_50	7.4500	6.7105	6.8600	8.0100	0.0762
13	32_50	7.5792	6.5234	7.2327	8.1607	0.0662
14	64_25	7.8030	6.2776	7.3422	8.3865	0.0771
15	128_10	8.4519	8.3061	7.8244	8.8281	0.0977
16	32_25	8.7207	8.4170	8.2267	9.0781	0.0933
17	64_10	8.8887	9.0646	8.1592	9.2572	0.1082
18	128_5	9.3201	9.0779	8.4133	9.0132	0.1322
19	32_10	9.5247	9.5637	8.5170	9.6687	0.1014
20	64_5	11.6602	10.2862	12.1433	9.5008	0.1338
21	32_5	17.1291	11.8607	15.3189	15.0407	0.1211

Tabla 12: Modelos LSTM con MAE promediado

En la tabla 12 se aprecian los valores promedio del indicador MAE para cada modelo sobre la muestra de 68 criptomonedas. De ellos se procedió a seleccionar los primeros 5

modelos en orden de complejidad estructural, es decir: 128_1_50, 128_1_100, 64_100, 128_50 y 128_100, dado que en general exhiben el menor indicador promedio tabulado. La tabla 13 presenta y describe estos modelos desde el más simple al más complejo en cuanto la estructura de capas y unidades ocultas que lo componen.

#	Modelo	Capas Ocultas	Unidades Capa 1	Unidades Capa 2	Early Stop
1	128_1_50	1	128	-	50
2	128_1_100	1	128	-	100
3	64_100	2	64	32	100
4	128_50	2	128	64	50
5	128_100	2	128	64	100

Tabla 13: Modelos seleccionados

4.3 Experimentación en proceso de inversión

Con los 5 modelos seleccionados, ver tabla 13, se procede a generar las predicciones para el período experimental correspondiente a los últimos 10 pasos de tiempo de cada serie de tiempo. Para efectos de la experimentación estos valores se suponen *out-of-sample* y, como ya se dijo, permiten complementar los últimos 10 pasos de tiempo de cada serie temporal con las predicciones generadas por estos modelos. Para ver descripción en mayor detalle consultar Paso 5: Obtener predicciones, en la sección Método.

4.3.1 Datasets de atributos

Las series temporales, básicas y generadas, combinadas de a tres, conforman los datasets de atributos. Los atributos básicos corresponden a: *Open*, *Low*, *High*, *Close* y *Log Cap*. Los atributos generados son: *Day Profitability* y *Day Variability*. Para mayor detalle consultar información presentada en la tabla 8 : Atributos en proceso de predicción experimentos. En el punto Paso 6: Generar datasets de atributos, se explica que las combinaciones de estos 7 atributos de tres en tres generan un total de $C_3^7 = 35$ datasets, que se utilizarán en la experimentación del método DLDM y en la generación de ranking dinámico, a partir del cual se evaluarán las estrategias de inversión consideradas en este trabajo.

Los 35 datasets de atributos dan lugar a un total de 211 casos de experimentación, para más detalle consultar: Paso 7: Preproceso de atributos, los cuales se originan en: los 35 datasets de atributos inicialmente obtenidos, más 175 dataset obtenidos al aplicar las 5 opciones de *scalers* presentados en la Tabla 9, más un caso adicional que corresponde al dataset compuesto por los siguientes atributos: (*Day Profitability*, *Day Variability*, *1*). En el último caso de experimentación, el último atributo de valor es constante e igual a 1, el cual fue seleccionado a partir de una extensa serie de pruebas y ensayos, en base a la observación que parece indicar que en este caso valores superiores a la unidad afectan negativamente al ranking dinámico, la cual se espera validar durante la experimentación. Para más detalle consultar Paso 7: Preproceso de atributos.

En resumen los 35 datasets de atributos son los que se detallan en la tabla 14, los cuales que, como ya se mencionó, generan los dataset definitivos con y sin preprocesamiento inicial, a los cuales se agrega el dataset (*Day Profitability*, *Day Variability*, *1*).

Dataset	**Atributo 1**	**Atributo 2**	**Atributo 3**
1	Day Profitability	Day Variability	Open
2	Day Profitability	Day Variability	Close
3	Day Profitability	Day Variability	High
4	Day Profitability	Day Variability	Low
5	Day Profitability	Day Variability	Log Cap
6	Day Profitability	Open	Close
7	Day Profitability	Open	High
8	Day Profitability	Open	Low
9	Day Profitability	Open	Log Cap
10	Day Profitability	Close	High
11	Day Profitability	Close	Low
12	Day Profitability	Close	Log Cap
13	Day Profitability	High	Low
14	Day Profitability	High	Log Cap
15	Day Profitability	Low	Log Cap
16	Day Variability	Open	Close
17	Day Variability	Open	High
18	Day Variability	Open	Low
19	Day Variability	Open	Log Cap
20	Day Variability	Close	High
21	Day Variability	Close	Low
22	Day Variability	Close	Log Cap
23	Day Variability	High	Low
24	Day Variability	High	Log Cap
25	Day Variability	Low	Log Cap
26	Open	Close	High
27	Open	Close	Low
28	Open	Close	Log Cap
29	Open	High	Low
30	Open	High	Log Cap
31	Open	Low	Log Cap
32	Close	High	Low
33	Close	High	Log Cap
34	Close	Low	Log Cap
35	High	Low	Log Cap

Tabla 14: Datasets de atributos

Los 211 datasets de atributos ya presentados son el insumo sobre el cual se aplicarán

los procesos contemplados en el modelo de decisión lingüístico dinámico (DLDM), cuya presentación detallada se encuentra en la sección **Etapa 2: Modelo de Decisión Lingüístico**. Este proceso culmina con el ranking dinámico de criptomonedas, desde el cual y en cada iteración del mismo se selecciona la criptomoneda ubicada en el primer lugar de dicha clasificación, tal como se explica en el punto **Paso 7: Ranking de las Alternativas**.

4.3.2 Resultado en experimentos con proceso de inversión simulado

Es necesario aclarar inicialmente que, para evaluar los experimentos, se han obviado los costos asociados a operar en los mercados y sólo se considera la inversión inicial para el cálculo del retorno, es decir, se calcula como: *(Ingreso - Inversión)/Inversión* para los efectos de evaluar los resultados experimentales del proceso simulado de inversión.

El proceso de inversión simulado, en el cual en cada iteración del ranking dinámico se invierte en la criptomoneda que se ubica en el primer lugar del ranking, corresponde a la estrategia (cartera) denominada DLDM y se explica con suficiente detalle en el punto **Etapa 3: Experimentación con estrategias simuladas**. La materialización de dicho proceso de inversión se detalla en el algoritmo esquematizado en la sección **Proceso de inversión**.

Los experimentos realizados con los datasets de atributo (tabla 14) complementados con las predicciones de cada uno de los modelos seleccionados (tabla 13), generan 5 tablas, una por modelo, con 211 filas cada una con los resultados del rendimiento de cada estrategia de inversión. De cada una de estas tablas sólo se seleccionan las primeras (5) filas con rendimiento igual o superior a 256% en la estrategia DLDM por modelo. Este límite corresponde al mejor rendimiento obtenido en el proceso de inversión simulado obtenido en el benchmark, correspondiente al trabajo de Torres *et al.* [65], quienes obtuvieron los resultados que muestra la tabla 15 aplicando estrategia DLDM con selección de una criptomoneda en cada periodo de inversión. La tabla muestra escenarios diferenciados por cantidad de criptomonedas incluidas en el conjunto de referencia. De esta manera distinguen escenarios con: 5, 15, 25, 35, 45, 50, 55 y 68 criptomonedas respectivamente.

Estrategia	**Cantidad de criptomonedas en conjunto referencia**							
Mejor Criptomoneda	**5**	**15**	**25**	**35**	**45**	**50**	**55**	**68**
Rendimiento DLDM-1	-22%	-10%	72%	66%	122%	256%	210%	95%

Tabla 15: Rendimiento estrategia DLDM-1, Torres *et al.* [65]

4.3.2.1 Experimento que superan máximo retorno del benchmark

La tabla 16 muestra el rendimiento obtenido en los experimentos con los dataset configurados a partir de las predicciones generadas por cada modelo LSTM para los últimos 10 pasos de tiempo de cada serie temporal. Para cada modelo se muestran todas las filas cuyo rendimiento final es igual o superior a 256% que es máximo rendimiento generado por el benchmark [65], como se aprecia en la tabla 15. Las columnas se definen como:

#	LSTM Modelo	Atributos 1	2	3	Scaler	Máximo Día	%	Final %
1	128_1_50	Day Profitability	Day Variability	1	No Escalado	70	2337	1763
2	128_1_50	Day Profitability	Day Variability	Open	No Escalado	60	875	488
3	128_1_50	Day Profitability	Day Variability	Close	No Escalado	60	875	488
4	128_1_50	Day Profitability	Day Variability	Low	No Escalado	60	875	488
5	128_1_50	Day Profitability	Day Variability	High	No Escalado	60	780	431
1	128_1_100	Day Profitability	Day Variability	1	No Escalado	70	2337	1763
2	128_1_100	Day Profitability	Day Variability	Open	No Escalado	60	875	488
3	128_1_100	Day Profitability	Day Variability	Close	No Escalado	60	875	488
4	128_1_100	Day Profitability	Day Variability	Low	No Escalado	60	875	488
5	128_1_100	Day Profitability	Day Variability	High	No Escalado	60	780	431
1	64_100	Day Profitability	Day Variability	1	No Escalado	70	2337	1763
2	64_100	Day Profitability	Day Variability	Open	No Escalado	60	875	488
3	64_100	Day Profitability	Day Variability	Close	No Escalado	60	875	488
4	64_100	Day Profitability	Day Variability	Low	No Escalado	60	875	488
5	64_100	Day Profitability	Day Variability	High	No Escalado	60	780	431
1	128_50	Day Profitability	Day Variability	1	No Escalado	70	2337	1763
2	128_50	Day Profitability	Day Variability	Open	No Escalado	60	875	488
3	128_50	Day Profitability	Day Variability	Close	No Escalado	60	875	488
4	128_50	Day Profitability	Day Variability	Low	No Escalado	60	875	488
5	128_50	Day Profitability	Day Variability	High	No Escalado	60	780	431
1	128_100	Day Profitability	Day Variability	1	No Escalado	70	2337	1372
2	128_100	Day Profitability	Day Variability	Open	No Escalado	60	875	488
3	128_100	Day Profitability	Day Variability	Close	No Escalado	60	875	488
4	128_100	Day Profitability	Day Variability	Low	No Escalado	60	875	488
5	128_100	Day Profitability	Day Variability	High	No Escalado	60	780	431

Tabla 16: Rendimiento estrategia DLDM por modelo

- Columna #:

  Indica el orden relativo al modelo respecto al cual la combinación de atributos y scaler, eventualmente, aplicado logran un retorno final superior al del benchmark.

- Columna LSTM:

  Identifica el modelo LSTM con que se complementarom las series temporales cuyo rendimimiento final superó al benchmark.

- Columna Atributos:

  Muestra la combinación de atributos con las cuales el rendimiento final supera al benchmark.

- Columna Scaler:

  Indica si el retorno final se obtuvo con atributos escalados o no. En caso de uso de escalador indica el scaler aplicado.

- Columna Máximo:

  Para cada combinación de atributos indica el día en el período de inversión en el cual se logró el retorno máximo en el proceso de inversión simulado con la estrategia DLDM y, también informa porcentualmente dicho retorno.

- Columna Final

  Es el retorno al final del periodo de inversión para el modelo y atributos correspondientes.

En resumen la tabla 17 presenta el retorno obtenido con la estrategia DLDM asociado al modelo LSTM de estructura más simple de los 5 modelos inicialmente seleccionados, que en este caso resulta ser el modelo denominado **Modelo 128_1_50**, ya que exhibe mejores o similares retornos respecto a los restantes modelos de estructura más compleja, según se desprende del análisis de la tabla 16. Para efectos de comparación, adicionalmente se presentan los retornos obtenidos por el benchmark [65] para conjuntos de referencia con 50, 55 y 68 criptomonedas, los primeros dos conjuntos de referencia son los que exhiben los retornos más altos para la estrategia DLDM-1 documentada en el benchmark [65], en tanto, el último caso presenta un rendimiento del 95% para el conjunto de referencia con 68 criptomonedas que es equivalente a la muestra de criptomonedas consideradas en este trabajo.

LSTM	**Retornos estrategia DLDM**					**Retornos estrategia DLDM-1 [65]**		
Seleccionado	1°	2°	3°	4°	5°	**50 Criptos**	**55 Criptos**	**68 Criptos**
128_1_50	1763%	488%	488%	488%	431%	256%	210%	95%

Tabla 17: Estrategia DLDM vs DLDM-1 en benchamark [65]

Estos datos muestran con claridad que los 5 retornos más altos, asociados al modelo de estructura más simple entre los modelos seleccionados, en conjunto con la estrategia DLDM implemantada en este trabajo sobre el total de 68 criptomonedas, superan a todos los retornos documentados en el benchmark [65] en la estrategia DLDM-1 para todos los conjuntos de referencia. En particular, el menor de los 5 retornos más altos seleccionados para la estrategia DLDM equivale a 431% y supera en un 354% el retorno de la estrategia DLDM-1 del benchmark para el mismo conjunto de 68 criptomonedas, adicionalmente, según se aprecia en la tabla 17 supera en un 68% el retorno del conjunto de referencia de 50 criptomonedas y supera en un 105% el retorno del conjunto de referencia de 55 criptomonedas.

Finalmente, la tabla 18 presenta los atributos con los cuales se obtienen los 5 rendimientos finales más altos con la estrategia DLDM que superan el retorno del 256% para el modelo seleccionado (**Modelo 128_1_50**). En ella también se aprecia el porcentaje en que cada uno de estos retornos finales supera al retorno del conjunto de referencia de 68 criptomonedas del benchmark [65]. El primer caso, que corresponde al más alto rendimiento obtenido con la estrategia DLDM, supera en 1755% el rendimiento de la estrategia DLDM-1 para el conjunto de referencia de 68 criptomonedas, en tanto el segudo, tercero y cuarto casos la superan en un 413% cada uno y, como ya se dijo, el último caso la supera en un 354%.

#	Atributos			Scaler	DLDM Final	DLDM-1 68 Crip	Supera en
	1	2	3				
1	Day Profitability	Day Variability	1	No Escalado	1763%	95%	1755%
2	Day Profitability	Day Variability	Open	No Escalado	488%	95%	413%
3	Day Profitability	Day Variability	Close	No Escalado	488%	95%	413%
4	Day Profitability	Day Variability	Low	No Escalado	488%	95%	413%
5	Day Profitability	Day Variability	High	No Escalado	431%	95%	354%

Tabla 18: Rendimiento estrategia DLDM con **Modelo 128_1_50**

4.3.2.2 Experimentos con retorno de 95% o superior

Para finalizar este análisis, la tabla 19 presenta los casos en los cuales, con el modelo LSTM de estructura más simple (**Modelo 128_1_50**), la estrategia DLDM genera retornos que igualan o superan el retorno del 95% obtenido con la estrategia DLDM-1 con el conjunto de referencia de 68 criptomonedas informado en el benchmark [65]

#	Atributos			Scaler	DLDM Final
	1	2	3		
1	Day Profitability	Day Variability	1	No Escalado	1763%
2	Day Profitability	Day Variability	Open	No Escalado	488%
3	Day Profitability	Day Variability	Close	No Escalado	488%
4	Day Profitability	Day Variability	Low	No Escalado	488%
5	Day Profitability	Day Variability	High	No Escalado	431%
6	Day Profitability	Day Variability	Low	MaxAbsScaler	227%
7	Day Profitability	Day Variability	Open	MaxAbsScaler	207%
8	Day Profitability	Day Variability	Log Cap	MaxAbsScaler	189%
9	Day Profitability	Day Variability	Low	MinMaxScaler	160%
10	Day Profitability	Day Variability	Close	MaxAbsScaler	157%
11	Day Profitability	Day Variability	High	MaxAbsScaler	99%
12	Day Profitability	Day Variability	Log Cap	No Escalado	95%

Tabla 19: Rendimiento estrategia DLDM de 95% o superior

Al considerar como límite el retorno del 95% informado en el benchmark [65] para el conjunto de referencia de 68 criptomonedas, tenemos que los casos de retornos iguales o superiores a dicho límite se incrementan e incluyen casos en que los scalers sí inciden en el resultado, por ejemplo en el caso 6 y 7 de la tabla 19 ambos superan el 200% aplicando *MaxAbsScaler*. Lamentablemente, estos casos resultan poco atractivos ya que están muy por debajo del rendimiento máximo del 256% informado en en benchmark.

4.3.2.3 Rendimiento del benchmark estrategias

El benchmark de estrategias de inversión o también denominado benchmark interno, considera las estrategias de inversión simplista denominada *buy-and-hold* y la estrategia

buy-and-sell, como se muestra en la figura 21. Las estrategias especificas consideradas en cada una de ella corresponden a BTC, Whole y Random, ver descripción en tabla 10.

Obviamente el retorno obtenido a partir de estas estrategias es el mismo en todos los experimentos, es decir en los $211 \times 5 = 1055$ experimentos. La tabla 20 muestra y compara dichos retornos con los obtenidos con la estrategia DLDM para todos los casos que superaron el benchmark [65] y asociados al modelo LSTM de estructura más simple entre los utilizados en los experimentos, es decir Modelo 128_1_50.

LSTM	**Estrategia DLDM**					**Benchmark de estrategias**		
Seleccionado	**1**	**2**	**3**	**4**	**5**	**Whole**	**Random**	**BTC**
128_1_50	1763%	488%	488%	488%	431%	54%	3%	-34%

Tabla 20: Estrategia DLDM vs benchamark de estrategias

Al observar la tabla 20, es posible concluir que la estrategia DLDM, en todas las opciones seleccionadas también supera ampliamente al benchmark de estrategias.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Con la información y el análisis de los resultados discutidos en el capítulo precedente, es posible concluir que:

1. Para cada uno de los modelos LSTM seleccionados de acuerdo al criterio ya explicado, sólo 5 experimentos lograron superar al benchmark externo correspondiente al trabajo ex-post presentado Torres *et al.* [65], si se asume como condición que superen el rendimiento más alto obtenido con la estrategia DLDM-1 que corresponde a un retorno de 256%.

2. Si se considera como límite inferior el rendimiento de 95% obtenido con la estrategia DLDM-1 para el conjunto de referencia de 68 criptomonedas, los casos en que el retorno logrado con la estrategia DLDM igualan o superan este limite se incrementan, sin embargo resultan poco atractivos dado el rendimiento máximo de 256% de la estrategia DLDM-1 para el conjunto de referencia de 50 criptomonedas.

3. Así mismo, todos estos experimentos superaron ampliamente al bechmark de estrategias.

4. En general con todos los modelos LSTM se obtienen retornos idénticos y para las mismas combinaciones de atributos, salvo para el primer caso asociado al Modelo 128_100 en el cual el restorno es marcadamente inferior.

5. De manera razonable, es posible concluir que bastaría con seleccionar el modelo LSTM de estructura más simple, ya que genera retornos iguales o superiores a los restantes modelos, y a la vez resulta más simple de entrenar que los otros modelos.

6. Se concluye que la aplicación de preproceso a los atributos no incide en generar retornos que superen al benchmark externo [65] con retorno máximo de 256%. En general, todos los experimentos que superaron el retorno máximo del benchmark externo [65] lo lograron con atributos no preprocesados (no escalados). Sin embargo al considerar como referencia un retorno de 95%, se incrementan los casos que lo superan o igualan e incluyen casos con aplicación de preproceso con aplicación de scalers como *MinMaxScaler* y *MaxAbsScaler*.

7. El optar por un tercer atributo de valor unitario mejora el desempeño del proceso incluido en el enfoque dinámico de decisión lingüístico y del ranking dinámico. En este caso es viable suponer que la explicación podría estar relacionada con el proceso de normalización que es parte del enfoque DLDM, dado que un atributo de valor unitario realmente no se torna preponderante en dicho proceso de normalización.

8. Sólo los atributos correspondientes a variaciones *(Day Profitability, Day Variability)* combinados con algunos de los siguientes atributos de valor: *1*, *Open*, *Close*, *Low*

o *High* logran superar al benchmark externo [65] suponiendo un retorno superior al 256% en el proceso de inversión simulado. El resto de las combinaciones no logran superarlo, aunque algunas generan retornos aceptables, como ocurre con los casos adicionales al considerar un retorno a partir del 95%, como el obtenido con la estrategia DLDM-1 con el conjunto de referencia de 68 criptomonedas, en cuyo caso se agrega a las combinaciones el atributo de valor *Log Cap* tanto escalado como no.

9. Luego de evaluar todas las combinaciones de 3 atributos, se determinó que las variaciones *Day profitability* (atributo 1) y *Day variability* (atributo 2) deben estar presentes en el proceso de decisión ya que contienen bastante información y, por lo tanto, sólo queda analizar el efecto que tiene el atributo 3 en el rendimiento de la cartera DLDM.

5.2 Trabajo futuro

Como trabajo futuro surgen variadas iniciativas y posibles líneas de investigación, entre las ellas se identifican las siguientes:

1. Validar el efecto de utilizar modelos específicos para cada criptomoneda y serie temporal OHLC en la predicción, en lugar de seleccionar un modelo de aplicación general.

2. Incrementar las observaciones en los datasets para analizar el efecto de las alzas empinadas y desplomes del mercado sobre las técnicas de entrenamiento de los modelos de redes neuronales propuestos.

3. Complementar los pronósticos de precios con pronósticos de tendencias para determinar posibles alzas o bajas en los mismos y, adaptar la estrategia de inversión para incluir esta información en la actualización diaria de la decisión de inversión. Interesa evaluar el impacto en el retorno al final del período de inversión.

4. En cuanto a la selección de hiperparámetros, parece razonable considerar, aplicar y evaluar técnicas para sintonización de los mismos, sin embargo esto por sí sólo es un proyecto complejo.

References

[1] Mohamad Ashari Alias, Siti Zaiton Mohd Hashim, and Supiah Samsudin. "Multi Criteria Decision Making and its Applications: A Literature Review". In: *Jurnal TeknoJlogi Maklumat* n/a.n/a (2008). URL: https://core.ac.uk/download/pdf/11784916.pdf.

[2] IBM Analytics. "Foundational Methodology for Data Science". In: *White Paper* (June 2015), p. 6. eprint: IMW14828-USEN-00.

[3] Andrew Ang, William N. Goetzmann, and S. Schaefer. "The Efficient Market Theory and Evidence: Implications for Active Investment Management". In: *Microeconomics: General Equilibrium & Disequilibrium Models of Financial Markets eJournal* (2010).

[4] Alireza Ashayer. "Modeling and Prediction of Cryptocurrency Prices Using Machine Learning Techniques". MA thesis. East Carolina University: Department of Computer Science, Apr. 2019.

[5] Valerie Belton and Theodor Stewart. *Multiple Criteria Decision Analysis: An Integrated Approach*. Jan. 2002. ISBN: 978-0-7923-7505-0. DOI: 10.1007/978-1-4615-1495-4.

[6] Fischer Black. "Noise". In: *The Journal of Finance* 41.3 (1986), pp. 528–543. DOI: https://doi.org/10.1111/j.1540-6261.1986.tb04513.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1986.tb04513.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1986.tb04513.x.

[7] Axel Brando et al. "Uncertainty Modelling in Deep Networks: Forecasting Short and Noisy Series". In: *ArXiv* abs/1807.09011 (2018).

[8] Martín Caride, Aurelio F. Bariviera, and Laura Lanzarini. "Stock Returns Forecast: An Examination By Means of Artificial Neural Networks". In: Jan. 2018, pp. 399–410. ISBN: 978-3-319-69988-2. DOI: 10.1007/978-3-319-69989-9_23.

[9] Nic Carter. *An Introduction to the Efficient Market Hypothesis for Bitcoiners*. Ed. by Medium.com. [Online; posted 19-February-2020]. Feb. 2020. URL: http://www.medium.com/@nic__carter/an-introduction-to-the-efficient-market-hypothesis-for-bitcoiners-ed7e90be7c0d. (accessed: 10.02.2021).

[10] Eugene Charniak. *Introduction to Deep Learning*. 2018. ISBN: 978-0-262-03951-2.

[11] Cathy Chen and Christian Hafner. "Sentiment-Induced Bubbles in the Cryptocurrency Market". In: *Journal of Risk and Financial Management* 12 (Apr. 2019), p. 53. DOI: 10.3390/jrfm12020053.

[12] Zheshi Chen, Chunhong Li, and Wenjun Sun. "Bitcoin price prediction using machine learning: An approach to sample dimension engineering". In: *Journal of Computational and Applied Mathematics* 365 (2020), p. 112395. ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2019.112395. URL: https://www.sciencedirect.com/science/article/pii/S037704271930398X.

[13] Department for Communities and Local Government. "Multi-criteria analysis: a manual". In: Jan. 2009, p. 168. ISBN: 978-1-4098-1023-0. URL: www.communities.gov.uk.

[14] Haresh Dedania, Vipul Shah, and Rajesh Sanghvi. "Portfolio Management: Stock Ranking by Multiple Attribute Decision Making Methods". In: *Technology and Investment* 6 (Nov. 2015), pp. 141–150. DOI: 10.4236/ti.2015.64016.

[15] Vasily Derbentsev et al. "Forecasting CryptocurrencyPrices Time Series Using Machine Learning". In: *Proceedings of the Selected Papers of the 8th International Conference on Monitoring, Modeling & Management of Emergent Economy (M3E2-EEMLPEED 2019)* (Aug. 2019). URL: http://ceur-ws.org/Vol-2422/paper26.pdf.

[16] Carlos Díaz-Contreras, Alejandra Aguilera-Rojas, and Nathaly Guillén-Barrientos. "Lógica difusa vs. modelo de regresión múltiple para la selección de personal". In: *Ingeniería. Revista chilena de ingeniería* 22 (Oct. 2014), pp. 547–559. DOI: 10.4067/S0718-33052014000400010.

[17] Aniruddha Dutta, Saket Kumar, and Meheli Basu. "A Gated Recurrent Unit Approach to Bitcoin Price Prediction". In: *Journal of Risk and Financial Management* 13.2 (2020). ISSN: 1911-8074. DOI: 10.3390/jrfm13020023. URL: https://www.mdpi.com/1911-8074/13/2/23.

[18] Eugene Fama. "Efficient Capital Markets: A Review of Theory and Empirical Work". In: *Journal of Finance* 25 (1970), pp. 383–417.

[19] Eugene Fama. "Efficient Capital Markets: II". In: *Journal of Finance* 46.5 (1991), pp. 1575–617. URL: https://EconPapers.repec.org/RePEc:bla:jfinan:v:46:y:1991:i:5:p:1575-617.

[20] Fan Fang et al. *Cryptocurrency Trading: A Comprehensive Survey*. 2020. arXiv: 2003.11352 [q-fin.TR].

[21] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "Knowledge Discovery and Data Mining: Towards a Unifying Framework". In: *KDD-96 Proceedings* (1996), pp. 82–88. URL: www.aaai.org.

[22] Pablo Fernandez. "CAPM: un modelo absurdo (CAPM: An Absurd Model)". In: *SSRN Electronic Journal* (Jan. 2014). DOI: 10.2139/ssrn.2499455.

[23] José Figueira et al. *Multiple Criteria Decision Support*. Jan. 2005. ISBN: 978-0-387-23081-8. DOI: 10.1007/0-387-23081-5_24.

[24] Xing Gao and Daniel Ladley. "Noise trading and market stability". In: *The European Journal of Finance* (July 2021), pp. 1–19. DOI: 10.1080/1351847X.2021.1947338.

[25] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Ed. by O'Reilly Media. June 2019, p. 196. URL: http://www.oreilly.com.

[26] Korhan Gokmenoglu and Shahram Alaghemand. "A multi-criteria decision-making model for evaluating priorities for foreign direct investment". In: *Croatian Operational Research Review* 6 (Oct. 2015), pp. 489–510. DOI: 10.17535/crorr.2015.0037.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016. DOI: 10.5772/47850.

[28] Tian Guo, Albert Bifet, and Nino Antulov-Fantulin. "Bitcoin Volatility Forecasting with a Glimpse into Buy and Sell Orders". In: *2018 IEEE International Conference on Data Mining (ICDM)* (Nov. 2018). DOI: 10.1109/icdm.2018.00123. URL: http://dx.doi.org/10.1109/ICDM.2018.00123.

[29] Martín Harding Affeld. "Eficiencia de mercado y regulación financiera: El nuevo paradigma y una aplicación al caso chileno". Facultad de Economía y Negocios, Universidad de Chile, Sept. 2011. URL: http://repositorio.uchile.cl/handle/2250/108072.

[30] Jang Huisu. "Predicting Bitcoin Prices by Using Rolling Window LSTM model". In: 2018.

[31] Ching-Lai Hwang and Kwangsun Yoon. *Methods for Multiple Attribute Decision Making. Lecture Notes in Economics and Mathematical Systems*. 1981. ISBN: 978-3-540-10558-9. DOI: 10.1007/978-3-642-48318-9.

[32] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. Ed. by Australia Monash University. Sept. 2018, p. 504.

[33] Stefan Jansen. *Machine Learning for Algorithmic Trading*. Ed. by Packt. July 2020, p. 790. ISBN: 978-1-83921-771-5. URL: https://www.packt.com.

[34] P. Jay et al. "Stochastic Neural Networks For Cryptocurrency Price Prediction". In: *IEEE Access* 8 (Apr. 2020), pp. 82804–82818. DOI: 10.1109/ACCESS.2020.2990659.

[35] P. Jay et al. "Stochastic Neural Networks For Cryptocurrency Price Prediction". In: *IEEE Access* 8 (Apr. 2020), pp. 82804–82818. DOI: 10.1109/ACCESS.2020.2990659.

[36] Suhwan Ji, Jongmin Kim, and Hyeonseung Im. "A Comparative Study of Bitcoin Price Prediction Using Deep Learning". In: *Mathematics* 7 (Sept. 2019), p. 898. DOI: 10.3390/math7100898.

[37] Cengiz Kahraman. "Multi-Criteria Decision Making Methods and Fuzzy Sets". In: vol. 16. Jan. 2008, pp. 1–18. ISBN: 978-0-387-76812-0. DOI: 10.1007/978-0-387-76813-7_1.

[38] Alan Kirman. "Is it rational to have rational expectations?" In: *Mind & Society* 13 (June 2014), pp. 29–48. DOI: 10.1007/s11299-014-0136-x.

[39] Jon Krohn and Beyleveld Grant abd Dseanlss Aglaé. *Deep Learning Illustrated, A Visual, Interactive Guide to Artificial Intelligence*. Addison-Wesley, 2020. ISBN: 9780135116692.

[40] Deepak Kumar and S. Rath. "Predicting the Trends of Price for Ethereum Using Deep Learning Techniques". In: Jan. 2020, pp. 103–114. ISBN: 978-981-15-0198-2. DOI: 10.1007/978-981-15-0199-9_9.

[41] Salim Lahmiri and Stelios Bekiros. "Cryptocurrency forecasting with deep learning chaotic neural networks". In: *Chaos, Solitons Fractals* 118 (Jan. 2019), pp. 35–40. DOI: 10.1016/j.chaos.2018.11.014.

[42] Laura Lanzarini, Juan Caride, and Aurelio F. Bariviera. "Are Technical Trading Rules Useful to Beat the Market? Evidence from the Brazilian Stock Market". In: June 2011.

[43] Laura Lanzarini et al. "THE EFFECT OF CHANGES OF THE HURST EXPONENT IN RETURN PREDICTABILITY: THE CASE OF THE DUTCH MARKET". In: July 2012, pp. 384–393. DOI: 10.1142/9789814415774_0025.

[44] Shuanglong Liu, Chao Zhang, and Jinwen Ma. "CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets". In: Oct. 2017, pp. 198–206. ISBN: 978-3-319-70095-3. DOI: 10.1007/978-3-319-70096-0_21.

[45] J. Bradford De Long et al. "The Survival of Noise Traders in Financial Markets". In: *The Journal of Business* 64.1 (1991), pp. 1–19. ISSN: 00219398, 15375374. URL: http://www.jstor.org/stable/2353070.

[46] Sean McNally, Jason Roche, and Simon Caton. "Predicting the Price of Bitcoin Using Machine Learning". In: Mar. 2018, pp. 339–343. DOI: 10.1109/PDP2018.2018.00060.

[47] Mansooreh Mollaghasemi and Julia Pet-Edwards. *Making Multiple-Objective Decisions*. Ed. by IEEE Computer Society Press. 1997, p. 91. ISBN: 0-8186-7407-5.

[48] B. Pai et al. "A Time Series Cryptocurrency Price Prediction Using LSTM". In: Jan. 2022, pp. 653–662. ISBN: 978-981-16-1341-8. DOI: 10.1007/978-981-16-1342-5_50.

[49] Avishek Pal and PKS Prakash. *Practical Time Series Analysis -Master Time Series Data Processing, Visualization, and Modelling using Python*. Ed. by Birmingham: Packt Publishing. Sept. 2017, p. 224. ISBN: 978-1-78829-022-7. URL: www.packtpub.com.

[50] Witold Pedrycz and Shyi-Ming Chen. *Deep Learning: Concepts and Architectures*. Springer, 2020. ISBN: 978-3-030-31756-0. DOI: 10.1007/978-3-030-31756-0.

[51] Kua-Hsin Peng and Gwo-Hshiung Tzeng. "A hybrid dynamic MADM model for problem-improvement in economics and business". In: *Technological and Economic Development of Economy* 19 (Dec. 2013), pp. 638–660. DOI: 10.3846/20294913.2013.837114.

[52] Thearasak Phaladisailoed and Thanisa Numnonda. "Machine Learning Models Comparison for Bitcoin Price Prediction". In: July 2018, pp. 506–511. DOI: 10.1109/ICITEED.2018.8534911.

[53] Víctor Yepes Piqueras. *Clasificación de los métodos de toma de decisión multicriterio multiatributo*. Ed. by victoryepes.blogs.upv.es. [Online; Published: 26 noviembre, 2018]. Nov. 2018. URL: https://victoryepes.blogs.upv.es/2018/11/26/clasificacion-metodos-madm/. (accessed: 10.02.2021).

[54] Tara Rawat and Vineeta Khemchandani. "Feature Engineering (FE) Tools and Techniques for Better Classification Performance". In: (May 2019). DOI: 10.21172/ijiet.82.024.

[55] Jonathan Rebane et al. "Seq2Seq RNNs and ARIMA models for Cryptocurrency Prediction: A Comparative Study". In: Aug. 2018.

[56] Lee S. Redding. "Noise Traders and Herding Behavior". In: *Series: IMF Working Papers* 1996.104 (Sept. 1996), p. 16. ISSN: 1018-5941. DOI: https://doi.org/10.5089/9781451947960.001.

[57] Jesús Rogel-Salazar. *Advanced Data Science and Analytics with Python*. Ed. by CRC Press. June 2020, p. 383. ISBN: 978-0-429-44661-0 / 978-1-138-31506-8 / 978-0-429-44664-1. URL: https://www.crcpress.com/Chapman-HallCRC-Data-Mining-and-Knowledge-Discovery-Series/book-series/CHDAMINODIS.

[58] James Royal and Kevin Voigt. *What Is Cryptocurrency? Here's What You Should Know*. Ed. by Nerdwallet.com. [Online; posted 18-January-2022]. Jan. 2022. URL: https://www.nerdwallet.com/article/investing/cryptocurrency-7-things-to-know. (accessed: 18.01.2022).

[59] Fatih Sari. "COMPARISON OF TOPSIS AND VIKOR MULTI CRITERIA DECISION ANALYSIS TECHNIQUES". In: *Selcuk University Journal of Engineering ,Science and Technology* 6 (Dec. 2018), pp. 825–831. DOI: 10.15317/Scitech.2018.172.

[60] Timo Seppäläin. *Basics of Stochastic Analysis*. Ed. by University of Wisconsin–Madison Department of Mathematics. Nov. 2014, p. 402.

[61] Takuya Shintate and Lukáš Pichl. "Trend Prediction Classification for High Frequency Bitcoin Time Series with Deep Learning". In: *Journal of Risk and Financial Management* 12 (Jan. 2019), p. 17. DOI: 10.3390/jrfm12010017.

[62] Sandro Skansi. *Introduction to Deep Learning, From Logical Calculus to Artificial Intelligence*. Ed. by Springer. 2018, p. 196. ISBN: 978-3-319-73003-5. DOI: 10.1007/978-3-319-73004-2.

[63] Donald Specht. "A General Regression Neural Network". In: *IEEE Transactions on Neural Networks* 2 (Nov. 1991), pp. 568–578. DOI: 10.1109/72.97934.

[64] Ola Surakhi et al. "Time-Lag Selection for Time-Series Forecasting Using Neural Network and Heuristic Algorithm". English. In: *Electronics* 10.20 (Oct. 2021). ISSN: 2079-9292. DOI: 10.3390/electronics10202518.

[65] Romina Torres et al. "A dynamic linguistic decision making approach for a cryptocurrency investment scenario". English. In: *IEEE Access* (2020). Publisher Copyright: CCBY Copyright: Copyright 2020 Elsevier B.V., All rights reserved. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3045923.

[66] Gwo-Hshiung Tzeng and Jih-Jeng Huang. *Multiple Attribute Decision Making*. 1981, p. 269. ISBN: 978-3-642-48318-9. DOI: 10.1007/978-3-642-48318-9.

[67] Gwo-Hshiung Tzeng and Jih-Jeng Huang. *Multiple Attribute Decision Making Methods and applications*. 2011, p. 350. ISBN: 978-1-4398-6157-8.

[68] Gómez-Espinosa A. Valencia F. and Valdés-Aguirre B. "Price Movement Prediction of Cryptocurrencies Using Sentiment Analysis and Machine Learning". In: *2018 IEEE International Conference on Data Mining (ICDM)* 21(6), 589 (2019). DOI: 10.3390/e21060589. URL: https://www.mdpi.com/1099-4300/21/6/589/htm.

[69] Pierre Venter and Eben Mare. "GARCH Generated Volatility Indices of Bitcoin and CRIX". In: *Journal of Risk and Financial Management* 13 (June 2020), p. 121. DOI: 10.3390/jrfm13060121.

[70] Arjen Vleugel, Marco Spruit, and Anton Daal. "Historical Data Analysis through Data Mining From an Outsourcing Perspective: The Three-Phases Model." In: *IJBIR* 1 (Jan. 2010), pp. 42–65. DOI: 10.4018/jbir.2010070104.

[71] Christian Walter. "The Efficient Market Hypothesis, the Gaussian Assumption, and the Investment Management Industry". In: *SSRN Electronic Journal* (Nov. 2000). DOI: 10.2139/ssrn.267443.

[72] Kai-Yin Woo et al. "Review on Efficiency and Anomalies in Stock Markets". In: *Economies* 8 (2020), pp. 1–51.

[73] Chih-Hung Wu et al. "A New Forecasting Framework for Bitcoin Price with LSTM". In: Nov. 2018, pp. 168–175. DOI: 10.1109/ICDMW.2018.00032.

[74] Zeshui Xu. "Deviation measures of linguistic preference relations in group decision making". In: *Omega* 33.3 (2005), pp. 249–254.

[75] Zeshui Xu. *Linguistic Decision Making: Theory and Methods*. Apr. 2012. ISBN: 978-3-642-29439-6. DOI: 10.1007/978-3-642-29440-2.

[76] Zeshui Xu. *Linguistic Decision Making: Theory and Methods*. Springer London, Limited, 2012. ISBN: 9783642294396.

[77] Zeshui Xu. "On multi-period multi-attribute decision making". In: *Knowledge-Based Systems* 21 (Mar. 2008), pp. 164–171. DOI: 10.1016/j.knosys.2007.05.007.

[78] Zeshui Xu. *Uncertain Multi-attribute Decision Making Methods and Applications*. May 2015. ISBN: 978-3-662-45639-2. DOI: 10.1007/978-3-662-45640-8.

[79] Zeshui Xu. *Uncertain Multi-Attribute Decision Making: Methods and Applications*. Springer Berlin Heidelberg, 2015. ISBN: 9783662456408. URL: https://books.google.cl/books?id=RUSMBgAAQBAJ.

[80] Ronald R. Yager. "OWA aggregation over a continuous interval argument with applications to decision making". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 34.5 (2004), pp. 1952–1963.

[81] Xin-She Yang. *Nature-Inspired Optimization Algorithms*. Mar. 2014.

[82] Yecheng Yao et al. "Predictive Analysis of Cryptocurrency Price Using Deep Learning". In: *International Journal of Engineering & Technology* 7 (3.27) (2018), pp. 258–264. DOI: 10.3390/e21060589. URL: www.sciencepubco.com/index.php/IJETResearch%20paper.

[83] Boris Yatsalo et al. "Decerns: A Framework for Multi-Criteria Decision Analysis". In: *International Journal of Computational Intelligence Systems* 8 (Feb. 2015). DOI: 10.1080/18756891.2015.1023586.

[84] Yao Yecheng et al. "Predictive Analysis of Cryptocurrency Price Using Deep Learning". In: *International Journal of Engineering and Technology(UAE)* 7 (Aug. 2018), pp. 258–264. DOI: 10.14419/ijet.v7i3.27.17889.

[85] Binyamin Yusoff. "Decision Analysis, Uncertainty Theories and Aggregation Operators in Financial Selection Problems". PhD in Business dissertation. Universitat de Barcelona, Apr. 2017. URL: https://www.tesisenred.net/bitstream/handle/10803/403402/BENYAMIN%5C%20YUSOFF_PhD_THESIS.pdf?sequence=1&isAllowed=y.

[86] M. Zeleny. *Multiple Criteria Decision Making*. Ed. by New York McGraw Hill. 1982, p. 563.