INSTITUTO TECNOLÓGICO DE LEÓN

Tesis

Pronóstico financiero usando modelos de inteligencia computacional

Que presenta:

Lic. Francisco Javier Pedroza Castro

Maestría en Ciencias
de la Computación

Con la dirección de:

Dr. Juan Martín Carpio Valadez

Con la co-dirección de:

Dr. Alfonso Rojas Domínguez

Revisores:

Dr. Manuel Órnelas Rodríguez

Dr. Héctor José Puga Soberanes

León, Guanajuato, México. Noviembre de 2022

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de León
División de Estudios de Posgrado e Investigación

León, Guanajuato, 11/octubre/2022
OFICIO No. DEPI-110-2022

C. FRANCISCO JAVIER PEDROZA CASTRO
PRESENTE

De acuerdo al fallo emitido por la Comisión Revisora, integrada por los: **Dr. Juan Martín Carpio Valadez, Dr. Alfonso Rojas Domínguez, Dr. Manuel Ornelas Rodríguez, Dr. Héctor José Puga Soberanes** y considerando que cubre todos los requisitos establecidos en los Lineamientos Generales para la Operación del Posgrado del Tecnológico Nacional de México, se autoriza la impresión del trabajo de tesis titulado: **"Pronóstico financiero usando modelos de inteligencia computacional".** Lo que hacemos de su conocimiento para los efectos y fines correspondientes.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Ciencia Tecnología y Libertad

PATRICIA DONATO JIMÉNEZ
SUBDIRECTORA ACADÉMICA

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LEÓN
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

C.c.p. Expediente

JMCV/CLDG

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

León, Guanajuato., a 05 de septiembre del 2022.

C. ING. LUIS ROBERTO GALLEGOS MUÑOZ
JEFE DE SERVICIOS ESCOLARES
P R E S E N T E

Por este medio hacemos de su conocimiento que la tesis titulada **"Pronóstico financiero usando modelos de inteligencia computacional"**, ha sido leída y aprobada por los miembros del Comité Tutorial para su evaluación por el jurado del acto de examen de grado al alumno (a) **C. Francisco Javier Pedroza Castro**, con número de control **M20241265** como parte de los requisitos para obtener el grado de Maestro(a) en Ciencias de la Computación (MCCOM-2011-05).

Sin otro particular por el momento, quedamos de Usted.

A T E N T A M E N T E
COMITÉ TUTORIAL

Dr. Juan Martín Carpio Valadez
DIRECTOR

Dr. Alfonso Rojas Domínguez
CODIRECTOR

Dr. Manuel Ornelas Rodríguez
REVISOR

Dr. Héctor José Puga Soberanes
REVISOR

INSTITUTO TECNOLÓGICO DE LEÓN

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

DECLARACION DE AUTENTICIDAD Y DE NO PLAGIO

Yo, **Pedroza Castro Francisco Javier** identificado con No. control **M20241265**, alumno (a) del programa de la **Maestría en Ciencias de la Computación**, autor (a) de la Tesis titulada: "**Pronóstico Financiero Usando Modelos de Inteligencia Computacional**" DECLARO QUE:

1.- El presente trabajo de investigación, tema de la tesis presentada para la obtención del título de MAESTRO (A) EN CIENCIAS DE LA COMPUTACIÓN es original, siendo resultado de mi trabajo personal, el cual no he copiado de otro trabajo de investigación, ni utilizado ideas, fórmulas, ni citas completas "stricto sensu", así como ilustraciones, fotografías u otros materiales audiovisuales, obtenidas de cualquier tesis, obra, artículo, memoria, etc. en su versión digital o impresa.

2.- Declaro que el trabajo de investigación que pongo a consideración para evaluación no ha sido presentado anteriormente para obtener algún grado académico o título, ni ha sido publicado en sitio alguno.

3.- Declaro que las pruebas o experimentos derivados de esta investigación fueron realizados bajo el consentimiento de los involucrados y con fines estrictamente académicos conforme a criterios éticos de confidencialidad.

Soy consciente de que el hecho de no respetar los derechos de autor y hacer plagio, es objeto de sanciones universitarias y/o legales por lo que asumo cualquier responsabilidad que pudiera derivarse de irregularidades den la tesis, así como de los derechos sobre la obra presentada.

Asimismo, me hago responsable ante el Tecnológico Nacional de México/Instituto Tecnológico de León o terceros, de cualquier irregularidad o daño que pudiera ocasional por el incumplimiento de lo declarado.

De identificarse falsificación, plagio, fraude, o que el trabajo de investigación haya sido publicado anteriormente; asumo las consecuencias y sanciones que de mi acción se deriven, responsabilizándome por todas las cargas pecuniarias o legales que se deriven de ello sometiéndome a las normas establecidas en los Lineamientos y Disposiciones de la Operación de Estudios de Posgrado en el Tecnológico Nacional de México.

León, Guanajuato a 13 del Octubre de 2022

Francisco Javier Pedroza Castro

Nombre y firma del autor

NOMBRE DEL TRABAJO

TESIS_MCC_PCFJ_VERSION_TURNITIN.pdf

AUTOR

Francisco Javier Pedroza Castro

RECUENTO DE PALABRAS

31091 Words

RECUENTO DE CARACTERES

159522 Characters

RECUENTO DE PÁGINAS

138 Pages

TAMAÑO DEL ARCHIVO

6.1MB

FECHA DE ENTREGA

Oct 10, 2022 10:03 AM CDT

FECHA DEL INFORME

Oct 10, 2022 10:04 AM CDT

7% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base d

- 5% Base de datos de Internet
- Base de datos de Crossref
- 4% Base de datos de trabajos entregados
- 2% Base de datos de publicaciones
- Base de datos de contenido publicado de Crossr

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

ACUERDO PARA USO DE OBRA (TESIS DE GRADO)

A QUIEN CORRESPONDA
PRESENTE

Por medio del presente escrito, **Francisco Javier Pedroza Castro**(en lo sucesivo el AUTOR) hace constar que es titular intelectual de la obra denominada: "**Pronóstico Financiero Utilizando Modelos de Inteligencia Computacional**", (en lo sucesivo la OBRA) en virtud de lo cual autoriza al Tecnológico Nacional de México/Instituto Tecnológico de León (en lo sucesivo TECNM/IT León) para que efectúe resguardo físico y/o electrónico mediante copia digital o impresa para asegurar su disponibilidad, divulgación, comunicación pública, distribución, transmisión, reproducción, así como digitalización de la misma con fines académicos y sin fines de lucro como parte del Repositorio Institucional del TECNM/ITLeón.

De igual manera, es deseo del AUTOR establecer que esta autorización es voluntaria y gratuita, y que de acuerdo a lo señalado en la Ley Federal del Derecho de Autor y la Ley de Propiedad Industrial el TECNM/IT León cuenta con mi autorización para la utilización de la información antes señalada, estableciendo que se utilizará única y exclusivamente para los fines antes señalados. El AUTOR autoriza al TECNM /IT León a utilizar la obra en los términos y condiciones aquí expresados, sin que ello implique se le conceda licencia o autorización alguna o algún tipo de derecho distinto al mencionada respecto a la "propiedad intelectual" de la misma OBRA; incluyendo todo tipo de derechos patrimoniales sobre obras y creaciones protegidas por derechos de autor y demás formas de propiedad intelectual reconocida o que lleguen a reconocer las leyes correspondientes. Al reutilizar, reproducir, transmitir y/o distribuir la OBRA se deberá reconocer y dar créditos de autoría de la obra intelectual en los términos especificados por el propio autor, y el no hacerlo implica el término de uso de esta licencia para los fines estipulados. Nada de esta licencia menoscaba o restringe los derechos patrimoniales y morales del AUTOR.

De la misma manera, se hace manifiesto que el contenido académico, literario, la edición y en general de cualquier parte de la OBRA son responsabilidad de AUTOR, por lo que se deslinda al (TECNM/ITLeón) por cualquier violación a los derechos de autor y/o propiedad intelectual, así como cualquier responsabilidad relacionada con la misma frente a terceros. Finalmente, el AUTOR manifiesta que estará depositando la versión final de su documento de Tesis, OBRA, y cuenta con los derechos morales y patrimoniales correspondientes para otorgar la presente autorización de uso.

En la ciudad de León , del estado de Guanajuato, México a los 13 días del mes de Octubre de 2022.

Atentamente,
Francisco Javier Pedroza Castro
Nombre y firma autógrada de EL AUTOR

Agradecimientos

Agradezco al Instituto Tecnológico de León (ITL) del Tecnológico Nacional de México (TeCNM), por ofertar la maestría en Ciencias de la Computación, la cual tuve la oportunidad de cursar y finalizar con la presente tesis.

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT), instituto que me otorgó una beca[1] de estudios de posgrado de tiempo completo a través del Programa Nacional de Posgrados de Calidad (PNPC), agradezco porque esta beca me permitió realizar y finalizar mis estudios de la maestría en Ciencias de la Computación.

Agradezco al Dr. Manuel Ornelas Rodríguez y al Dr. Héctor José Puga Soberanes, quienes se dan el tiempo de asistir a los seminarios de evaluación de avances de investigación y darme sus recomendaciones y sugerencias de investigación.

Agradezco a mi director de tesis, Dr. Martín Carpio Valadez quién tuvo la paciencia de explicarme dudas que tengo y ofrecerme su tiempo para darme sus recomendaciones a pesar de ser horas poco convencionales; por compartir sus experiencias y la paciencia para compartir su conocimiento y resolver dudas así como su paciencia para revisar mi trabajo de investigación.

Agradezco a mi co-director de tesis, Dr. Alfonso Rojas Domínguez, que con su basta experiencia me ha enseñado varias cosas, entre ellas: la importancia de hacer bien o lo mejor posible los trabajos desde un inicio para evitar correcciones que son difíciles de detectar en documentos extensos; por enseñarme a ser atento en los pequeños detalles, los cuales son de suma importancia a pesar de ser pequeños; por la dedicación y tiempo que da a la

[1]El número de la beca otorgada por el PNPC es el 774627.

revisión de mis trabajos y sus observaciones precisas; por enseñarme a expresar las ideas de manera clara y concisa y darme sus sugerencias de experimentación; entre otras cosas, pero principalmente por la paciencia que ha tenido conmigo a lo largo de todo el trabajo de investigación.

Agradezco al Dr. Martín Carpio Valadez y el Dr. Alfonso Rojas Domínguez, por los tips de investigación que me han compartido, desde como leer un artículo, formar el estado del arte, hasta como seleccionar un artículo de calidad.

Por último, y no menos importante, quiero agradecer a mis padres, Rita Castro Rodríguez y Francisco Javier Pedroza Pérez, por su comprensión al no molestarse conmigo por trabajar fines de semana en la investigación y demás ocasiones similares. Agradezco a mi hermana Alejandra Pedroza Castro que del mismo modo comprende que hay ocasiones en las que tuve que estar trabajando en la investigación y me dio mi espacio; de igual manera a mi hermana María Magdalena Pedroza Castro y mis sobrinos Leonel Rosales Pedroza y Kida Valeria Rosales Pedroza, que llegan realizar visitar a la casa y no se molestan que me quede haciendo la investigación.

Tabla de Contenido

Tabla de Contenido **XI**

Tablas **XV**

Figuras **XVII**

1. Introducción **9**

1.1. Antecedentes 9

1.1.1. Pronósticos financieros 10

1.1.2. Inteligencia computacional en pronósticos financieros 11

1.1.2.1. Tipos de pronósticos y de datos 12

1.1.3. Metodología en pronósticos financieros usando Inteligencia Computacional 18

1.2. Planteamiento del problema 21

1.3. Justificación 22

1.4. Objetivos 22

1.4.1. Objetivo general 22

1.4.2. Objetivos específicos 22

1.5. Hipótesis 23

1.6. Alcance y limitaciones 23

2. Marco Teórico **25**

2.1. Perceptrón 25

2.1.1. Funciones de activación 26
2.2. Perceptrón Multicapa 28
2.3. Red Neuronal Recurrente 30
2.4. Long Short - Term Memory 34
2.4.1. Hiperparámetros 37
2.5. Metaheurísticas 39
2.5.1. Metaheurísticas basadas en poblaciones 41
2.5.2. Particle Swarm Optimization 42
2.5.3. Flower Pollination Algorithm 44
2.5.4. Estimation Of Distribution Algorithm 46
2.5.4.1. Estimation of Distribution Algorithm for a Low Number of Function Evaluations 46
2.6. Automated Machine Learning 51
2.6.1. Optimización de hiperparámetros 51

3. Estado del arte **53**
3.1. Pronósticos financieros 54
3.2. HPO en pronósticos financieros 63
3.3. HPO en otros campos 69

4. Metodología **75**
4.1. Metodología de optimización de hiperparámetros 75
4.2. Metodología de comparación de metaheurísticas 82

5. Resultados Experimentales **85**
5.1. Resultados de optimización 85
5.2. Resultados de convergencia 93

6. Discusiones **101**
6.1. Discusiones de soluciones 101

6.2. Discusiones de convergencia 104

7. Conclusiones y trabajos a futuro **107**

A. Artículo COMIA **113**

B. Artículo ISCI **129**

C. Experimentos realizados con SEED **145**

D. Experimentos con red LSTM para realizar pronósticos **149**

Bibliografía **157**

Tablas

2.1. Hiperparámetros. 39

3.1. Resultados experimentales de Lin *et al.* (2021) [76]. 56
3.2. Resultados experimentales de Chen *et al.* (2022) [26]. 56
3.3. Resultados experimentales de Chen *et al.* (2021) [24] en el sector de Alimentos y Bebidas. 58
3.4. Resultados experimentales de Li *et al.* (2022) [74]. 61
3.5. Resultados experimentales de Bas *et al.* (2022) [10]. 62
3.6. Resultados experimentales de Bazrkar & Hosseini *et al.* (2022) [11]. 62
3.7. Estado del arte en optimización de hiperparámetros en pronósticos financieros. 63
3.8. Resultados experimentales de Chung & Shin (2018) [29]. 65
3.9. Resultados experimentales de Bhandari *et al.* (2022) [13]. 65
3.10. Resultados experimentales de Kumar *et al.* (2022) [68]. 66
3.11. Resultados experimentales de Deng *et al.* (2022) [33]. 67
3.12. Resultados experimentales de Zhu *et al.* (2022) [118]. 69
3.13. Resultados experimentales de Kilink & Haznedar (2022) [61]. 71
3.14. Resultados experimentales de Chen *et al.* (2019) [23] [61]. 71

4.1. Hiperparámetros y espacio de búsqueda. 79
4.2. Parámetros empleados en metaheurísticas. 80

5.1. Tiempo de ejecución de experimentos 86

5.2. Soluciones de la OHP de las redes LSTM y estadísticas de rendimiento de las soluciones. 90
5.3. Porcentajes de convergencia. 94

C.1. Soluciones encontradas por SEED así como los resultados de su rendimiento. 146

Figuras

1.1. No. de publicaciones por año que usan DL [98]. 12
1.2. Frecuencia de tipo de pronósticos realizados en el periodo 2005–2019 [98]. . . 13
1.3. Tipo de DL vs. Tipo de pronóstico [98]. 14
1.4. Porcentaje de tipo de variables de entrada en pronósticos financieros [66]. . . 15
1.5. Metodología de Oncharoen & Vateekul (2018) [87] 16
1.6. Metodología identificada por Cavalcante *et al.* (2016) [18]. 18
1.7. Metodología propuesto por Cavalcante *et al.* (2016) [18]. 19
1.8. Metodología identificada por Kumar *et al.* (2021) [66]. 20
1.9. Metodología propuesta por Kumar *et al.* (2021) [66]. 20
1.10. Metodología identificada por Bustos *et al.* (2020) [17]. 21

2.1. Sinapsis de una neurona biológica.(Imagen editada de [12]) 26
2.2. Funciones de Activación. (Imágenes tomadas de [12]) 27
2.3. MLP con tres capas ocultas. (Imagen editada de [12].) 29
2.4. Propagación de una RNN. (Imagen editada de [108].) 32
2.5. Estructura de una celda LSTM. (Elaboración propia). 34

3.1. Comparación de diferentes modelos realizando pronósticos probados por Kanwal *et al.* (2022) [59]. 55
3.2. Arquitectura propuesta por Kanwal *et al.* (2022) [59]. 57
3.3. Metodología de pronóstico de Chen *et al.* (2022) [26] 57
3.4. Sistema de comercio automático de Silva *et al.* (2020) [100] 60

3.5. Metodología propuesta por Deng *et al.* (2022) [34] 67
3.6. Metodología de optimización de hiperparámetros de Chung & Shin (2018) [29]. 68

4.1. Diseño de red LSTM usando *dropout*. (Elaboración propia) 77
4.2. Metodología de optimización de hiperparámetros. (Elaboración propia) . . . 78
4.3. Mapeo del Espacio de Números Discretos al Espacio de Búsqueda del tamaño de lote vía (4.4). (Elaboración propia) 80

5.1. *Fitness* vs. Iteración de las metaheurísticas en la optimización de hiperparámetros. (Elaboración propia) 87
5.2. Pérdida promedio y dispersión de entrenamiento de 33 experimentos. (Elaboración propia) 88
5.3. Diagrama de caja y bigote con datos de evaluación de los ER. (Elaboración propia) 91
5.4. Precio de cierre y Predicción promedio de 33 experimentos. (Elaboración propia) 92
5.5. Convergencia de No. de Celdas. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia) 94
5.6. Convergencia de No. de Épocas. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia) 95
5.7. Convergencia de No. de Estados Ocultos. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia) 96
5.8. Convergencia de Factor de Aprendizaje. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia) 97
5.9. Convergencia de Tamaño del Lote. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia) 98
5.10. Convergencia de Tamaño de la Ventana. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia) 99

C.1. Diagrama de caja y bigote de Prueba de pronóstico promedio de 33 experimentos. (Elaboración propia) 146

C.2. Diagrama de caja y bigote de Prueba de pronóstico promedio de 33 experimentos. (Elaboración propia) 147

D.1. Pérdida vs. número de estados ocultos. (Elaboración propia) 150
D.2. Acercamiento de pérdida vs. número de estados ocultos. (Elaboración propia) 151
D.3. Predicción de una red LSTM con un estado oculto. (Elaboración propia) . . 151
D.4. Predicción de una red LSTM con dos estados ocultos. (Elaboración propia) . 152
D.5. Predicción de una red LSTM con 25 estados ocultos. (Elaboración propia) . 152
D.6. Predicción de una red LSTM con 50 estados ocultos. (Elaboración propia) . 153
D.7. Predicción de una red LSTM con 75 estados ocultos. (Elaboración propia) . 153
D.8. Predicción de una red LSTM con 100 estados ocultos. (Elaboración propia) . 154
D.9. Predicción de una red LSTM con 125 estados ocultos. (Elaboración propia) . 154
D.10. Predicción de una red LSTM con 150 estados ocultos. (Elaboración propia) . 155

Algoritmos

1. Funcionamiento general de una P-metaheurística [102] 41
2. PSO, asumiendo minimización [40] 43
3. FPA, asumiendo minimización [112] 46
4. EDA general [4] 47
5. LNFE [77] 49
6. Generar nuevos Individuos de LNFE [77] 50
7. Optimización de hiperparámetros 81

Acrónimos

- $\boldsymbol{\alpha}$: Factor de aprendizaje.
- **ABC**: Artificial Bee Colony.
- **AF**: Análisis Fundamental.
- **ANN**: Redes Neuronales Artificiales.
- **ARCH**: Autoregressive Conditional Heteroskedasticity.
- **ARIMA**: Autoregressive Integrated Moving Average.
- **ARMA**: Autoregressive Moving Average.
- **AT**: Análisis Técnico.
- **AutoML**: Automated Machine Learning.
- $\mathbf{B}_s$: Tamaño del lote.
- **BM**: Búsqueda Manual.
- **CF**: Llamada a función.
- $\mathbf{Cl}_s$: Celda LSTM.
- **CNN**: Convolutional Neural Network.
- **CPT**: Convergencia Promedio Total.

- **DBN**: Deep Belief Networks.
- **DE**: Diferential Evolution.
- ***dis***: Números Discretos.
- **DL**: Deep Leaning.
- $\mathbf{E}_s$: Número de épocas.
- **EB**: Espacio de búsqueda.
- **EDALNFE** Estimation of Distribution Algorithm for a Low Number of Function Evaluations.
- **ECM**: Error Cuadrático Medio.
- **EM**: Ensemble methods.
- **ER**: Experimentos de rendimiento.
- **ESD**: Empirical Selection Distribution.
- **FPA**:Flower Pollination Algorithm.
- **FPA-GOOGLE**: Optimización de hiperaparámetros con FPA de una red LSTM para realizar un pronóstico del precio de cierre de las acciones de Google.
- **FPA-NIKE**: Optimización de hiperaparámetros con FPA de una red LSTM para realizar un pronóstico del precio de cierre de las acciones de Nike.
- **GA**: Algoritmos Genéticos.
- **GAN**: Generative Adversarial Network.
- **GARCH**: Generalized Autoregressive Conditional Heteroskedasticity.
- **GRU**: Gated Recurrent Unit.

- **GS**: Grid Search.
- $\mathbf{H}_s$: Estado oculto.
- **HFT**: High-Frequency Trading.
- **HPO**: Optimización de Hiperparámetros.
- **IC**: Inteligencia Computacional.
- **ICA**: Independent Component Analysis.
- **IT**: Indicador técnico.
- **KPCA**: Kernel-based Principal Component Analysis.
- **LDS**: Logarithmic Data Smoothing.
- **LNFE**: Estimation of Distribution Algorithm for a Low Number of Function Evaluation.
- **LNFE-GOOGLE**: Optimización de hiperaparámetros con LNFE de una red LSTM para realizar un pronóstico del precio de cierre de las acciones de Google.
- **LNFE-NIKE**: Optimización de hiperaparámetros con LNFE de una red LSTM para realizar un pronóstico del precio de cierre de las acciones de Nike.
- **LSTM**: Long Short-Term Memory.
- **MAD**: Mean Absolute Diference.
- **MAE**: Mean Absolute Error.
- **MB**: Modelos Bayesianos.
- **ME**: Modelos Estadísticos.
- **MEMD**: Multivariate Empirical Mode Decomposition.

- **MF**: Mercados Financieros.
- **MH**: Mecanismos Híbridos.
- **ML**: Machine Learning.
- **MLP**: Multilayer Perceptron o perceptrón multicapa.
- **NLICA**: Non-linear Independent Component Analysis.
- **NLP**: Procesamiento de lenguaje natural.
- **OAT**: Orthogonal Array Tuning.
- **OCLHV**: Open price, close price, Low price, High price y Volume share price.
- **OSM**: Optimización sin metaheurísticas.
- **P-metaheurística**: Metaheurística basada en poblaciones.
- **PCA**: Principal Component Analysis.
- **PF**: Pronóstico Financiero.
- **PSO**: Particle Swarm Optimization.
- **PSO-GOOGLE**: Optimización de hiperaparámetros con PSO de una red LSTM para realizar un pronóstico del precio de cierre de las acciones de Google.
- **PSO-NIKE**: Optimización de hiperaparámetros con PSO de una red LSTM para realizar un pronóstico del precio de cierre de las acciones de Nike.
- **RBM**: Restricted Boltzmann Machines.
- **RMSE**: Raíz del error cuadrático medio.
- **RNN**: Redes Neuronales Recurrentes.
- **RS**: Random Search.

- **SEED**: Symmetric-Approximation Energy-Based Estimation of Distribution.
- **SEED-GOOGLE**: Optimización de hiperaparámetros con SEED de una red LSTM para realizar un pronóstico del precio de cierre de las acciones de Google.
- **SEED-NIKE**: Optimización de hiperaparámetros con SEED de una red LSTM para realizar un pronóstico del precio de cierre de las acciones de Nike.
- **SRNN**: RNN simple o Elman Network.
- **Std**: Desviación Estándar.
- **SVM**: Máquinas de Soporte Vectorial.
- $\mathbf{W}_s$: Tamaño de la ventana.
- **WAV**: Wavelet Neural Networks.
- **WS**: Secuencia de calores también llamada ventana.
- **WT**: Wavelet Transformation.

Resumen

Pronóstico financiero usando modelos de inteligencia computacional, es una obra que busca realizar un pronóstico financiero utilizando una red neuronal artificial tipo *Long-short Term Memory* (LSTM), optimizando los hiperparámetros de la red mediante metaheurísticas.

En esta investigación se identificó que la comunidad científica, inversores y academia, pasaron de utilizar análisis técnico, análisis fundamental y métodos estadísticos clásicos para realizar pronósticos financieros a pasar a utilizar Inteligencia Computacional (IC) para realizar pronósticos financieros. Entre los métodos de IC utilizados se encuentran desde Lógica Difusa, Deep Learning, hasta algoritmos metaheurísticos.

Entre los algoritmos más utilizados para realizar pronósticos financieros son los que pertenecen al Deep Learning, las redes LSTM son las que más destacan de este campo. Sin embargo, existe una tendencia de hibridar el Deep Learning con otros algoritmos, entre las propuestas se encuentran: algoritmos metaheurísticos (para optimizar parámetros y/o hiperparámetros de una red), Lógica Difusa, Convolutional Neural Networks, entre otros algoritmos.

Los algoritmos evolutivos (pertenecientes a las metaheurísticas) son ampliamente usados en la hibridación con Deep Learning para realizar pronósticos. En la revisión tanto de antecedentes como del estado del arte se identificó que la hibridación ha mostrado resultados superiores al realizar pronósticos financieros en contraste con modelos no híbridos.

En la hibridación con algoritmos evolutivos se realizan optimizaciones de hiperparámetros de redes LSTM para que las redes realicen pronósticos financieros. Entre los algoritmos evolutivos se encuentran: *Particle Swarm Optimization* (PSO), *Flower Pollination Algo-*

rithm (FPA), Algoritmos Genéticos (GA, por sus siglas en inglés), entre otros algoritmos metaheurísticos basados en poblaciones. Mientras que en campos distintos a pronósticos financieros, la optimización de hiperparámetros se realizan con modelos de *Reinforcement Learning*, *Deep Learning* y *Estimation of Distribution Algorithm* (EDA), teniendo resultados satisfactorios.

En la presente tesis se propone una metodología para la optimización de hiperparámetros y una metodología para la comparación de metaheurísticas. La optimización de hiperparámetros consiste en: 1) generar una población de hiperparámetros, 2) los cuales son utilizados para diseñar las redes LSTM, posteriormente 3) se entren las distintas redes LSTM, 3) se evalúan, 4) si no cumple con un criterio de paro, se actualiza la población y se regresa al paso de diseñar las redes LSTM.

En este trabajo se utilizan tres metaheurísticas distintas para llevar a cabo la optimización de hiperparámetros, a saber, PSO, FPA y *Estimation of Distribution Algorithm for a Low Number of Function Evaluations* (LNFE). Las soluciones obtenidas por cada metaheurística incluyendo a una red optimizada sin utilizar metaheurísticas, fueron comparadas basándose en el rendimiento de las redes LSTM con hiperparámetros optimizados. En la comparación de las metaheurísticas se consideró inicializar las poblaciones con la misma semilla, comparar el valor *fitness* y la convergencia de las soluciones en cada variable.

Los resultados muestran que las redes LSTM cuyos hiperparámetros fueron ajustados utilizando metaheurísticas superan a la red LSTM, las cuales se ajustaron sus hiperparámetros sin utilizar metaheurísticas. De las soluciones LNFE, tiene mayor convergencia que las demás metaheurísticas, y FPA tiene menor convergencia, no obstante, el rendimiento de las soluciones fue mayor a las soluciones de las demás metaheurísticas.

Al comparar el proceso de optimización de las tres metahuerísticas (PSO, FPA y LNFE), se encuentra que LNFE converge en una menor cantidad de iteraciones con resultados satisfactorios y un menor costo computacional, mientras que FPA, presenta característica de mayor exploración que de intensificación; mientras que PSO muestra un equilibrio entre exploración y convergencia.

Capítulo 1

Introducción

En este capítulo se exponen diversas opiniones científicas, revisiones científicas, entre otros artículos de Pronósticos Financieros (PF) y PF utilizando modelos de Inteligencia Computacional (IC). En los cuales se identifica la evolución de los PF desde sus inicios, utilizando Análisis Técnico (AT) y Análisis Fundamental (AF), pasando por Modelos Estadísticos (ME), hasta modelos de IC desde Redes Neuronales Artificiales (ANN, por sus siglas en inglés) y *Deep Learning* (DL), campos del *Machine Learning* (ML). Al final del capítulo se describen conclusiones para determinar el problema de investigación, su justificación, objetivo del trabajo, hipótesis, alcances y limitaciones.

1.1. Antecedentes

Las acciones financieras son afectadas por diversos factores como: política publica, política fiscal, política monetaria, comercio internacional, fenómenos meteorológicos, noticias, guerras, entre otros factores, lo que hace a las series de tiempo financieras: no-lineales, no-paramétricas y dinámicas. De acuerdo a Fama (1995) [36], las acciones financieras que cotizan en las bolsas siguen una caminata aleatoria (*Random Walk* (RW)), impidiendo pronosticar a corto plazo el movimiento y la magnitud de las acciones financieras en el mercado de valores; él asevera que no se puede utilizar el precio pasado de las acciones y/o información pública de compañías y gobiernos, para pronosticar el precio de las acciones financieras.

No obstante, la comunidad científica e industria financiera han propuesto diferentes modelos para pronosticar las acciones financieras, así como de otros activos, con el objetivo de disminuir el riesgo en la toma de decisiones en el mercado de valores bursátiles; incluso la industria y comunidad científica diseña Sistemas de Comercio o Sistemas de Comercio Automatizado para aumentar sus ganancias y reducir los riesgos. En las siguientes secciones se habla de los métodos de pronósticos clásicos (1.1.1) y de los avances en pronósticos financieros aplicando modelos de inteligencia computacional (1.1.2).

1.1.1. Pronósticos financieros

Los métodos comúnmente utilizados para realizar un pronóstico financiero son AT [82], AF [86] y Métodos Estadísticos Clásicos (MEC) [46]. El AT es un método utilizado para analizar el precio de las acciones haciendo uso de gráficos e Indicadores Técnicos (IT), asumiendo que la información de todos los factores económicos son absorbidos por el precio de las acciones, mientras que en AF se asume que se puede pronosticar el precio de las acciones analizando factores como el nivel de productividad, posicionamiento en el mercado, competencia en el mercado, análisis de estados financieros, reportes anuales, noticias económicas, entre otros. En el caso de MEC se usan distintos modelos (estadísticos y económicos) y supuestos (estadísticos y económicos) para realizar pronósticos; entre los modelos utilizados se encuentran: *Autoregressive Moving Average* (ARMA) [92], *Autoregressive Integrated Moving Average* (ARIMA) [1], *Generalized Autoregressive Conditional Heteroskedasticity* (GARCH) [65], *Exponential Smoothing* (ES) [15], entre otros; en los supuestos se encuentran: homocedasticidad, linealidad [46], bienes homogéneos, sistema económico en equilibrio [96], entre otros.

En las últimas décadas se ha observado un creciente uso de la IC para realizar pronósticos financieros en la bolsa de valores. En la siguiente subsección se abordan diversas investigaciones que realizan pronóstico financiero utilizando IC.

1.1.2. Inteligencia computacional en pronósticos financieros

En las últimas décadas el interés de la comunidad científica en la IC para realizar pronósticos financieros va en aumento, entre las áreas de interés se encuentran: el ML [3], sistemas multi-objetivos [79,90], Lógica Difusa, Metaheurísticas (Algoritmos Genéticos [55], *Particle Swarm Optimization* (PSO) [110] y demás Algoritmos Evolutivos), entre otros.

Entre las técnicas de IC el ML ha tenido mayor auge, con técnicas como: ANN y *Deep Leaning* (DL) [35, 57, 71], Máquinas de Soporte Vectorial (SVM, por sus siglas en inglés) [22,72], Modelos Bayesianos (MB) [52,80], Mecanismos Híbridos (MH) [7], Optimización de parámetros, Optimización de Hiperparámetros (HPO, por sus singlas en inglés) [16, 35, 51, 91], *Ensemble Methods* (EM) [64, 85, 91] y demás modelos.

En el periodo de 2016 a 2018 el uso de DL y EM ha ido en aumento en comparación a años anteriores. En la Fig. 1.1, se muestra un histograma de la frecuencia de publicaciones que utilizan DL por año. El creciente uso de DL se debe al éxito que ha tenido en pronósticos financieros y que ha llegado a superar modelos *benchmark* [98].

En las investigaciones realizadas por Sezer, *et al.* (2020) [98] se identifican 140 artículos que usan DL. Entre las técnicas más utilizadas del DL en el periodo de 2018 a 2019 son: Redes Neuronales Recurrentes (RNN por sus singlas en inglés) con 52.5 %, *Multilayer Perceptron* (MLP) con 28.6 % y *Convolutional Neural Network* (CNN) con 16.9 %, entre otras. Entre las RNN, el tipo de red más usada fue alguna variante de la LSTM con 60.4 %, seguido de *Vanilla* RNN y GRU. De la misma manera Li *et al.* (2021) [73] reconocen que las redes LSTM (incluyendo LSTM híbridas) son ampliamente utilizadas (28 artículos de 37 analizados) con éxito.

A pesar del éxito del DL [1] encontrar los hiperparáemtros [2] que brinden el rendimiento esperado es un problema; la literatura refiere el uso de métodos para abordar el problema como: Búsqueda Manual (BM), *Grid Search* (GS), *Random Search* (RS), MB [98], así como

[1]Entre los distintos modelos de DL se enceuntran: MLP, LSTM, CNN, RNN, *Deep Reinforcement Learning* (DRL), *Deep Belief Networks* (DBN), *Restricted Boltzmann Machines* (RBM), entre otros.

[2]Entre los hiperaprámetros que se ajustan en el DL se encuentran: Número de capas ocultas, número de neuronas, técnica de regularización, inicialización de pesos o parámetros, función de activación, factor de aprendizaje, número de épocas, tamaño de lote, algoritmo de optimización, entre otros.

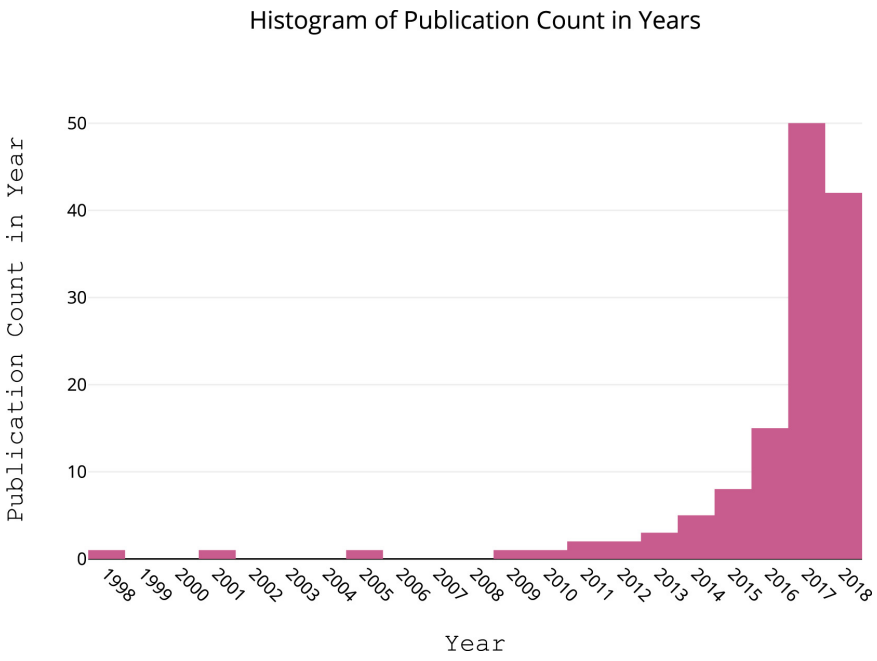


Figura 1.1: No. de publicaciones por año que usan DL [98].

distintos algoritmos evolutivos[3].

La relativa facilidad de aplicar RNNs (como LSTM y variantes) para realizar pronósticos financieros con éxito permite un amplio uso de ellas; empero, se tienen expectativas alentadoras en otro tipo de redes como: *Generative Adversarial Network* (GAN), RL y *Graph* CNN, esta última tiene potencial uso en Sistemas de Comercio Automatizado y *High-Frequency Trading* (HFT). De acuerdo a Sezer, *et al.* (2020) [98] *Graph* CNN puede representar diferentes partes de un sistema de comercio como: portafolio de inversión, redes sociales de la comunidad científica, AF, AT [98], entre otras partes. Probablemente, en el futuro se apliquen este tipo de modelos en pronósticos financieros, Algoritmos de Comercio y/o Administración de Portafolios de Inversión [98].

1.1.2.1. Tipos de pronósticos y de datos

En PF usando IC la mayoría de los trabajos realizan pronósticos de precios de acciones [4] e índices, seguido de otros pronósticos como tendencias de diferentes tipos de activos e

[3]La literatura en pronósticos financieros se refiere al ajuste de hiperaparámetros como modelos híbridos [66], en esta tesis nos referimos a ellos como métodos de HPO.

[4]Por lo genera se utiliza los datos de *Open price*, *Close price*, *Low price*, *High price* y *Volume share price* (OCLHV).

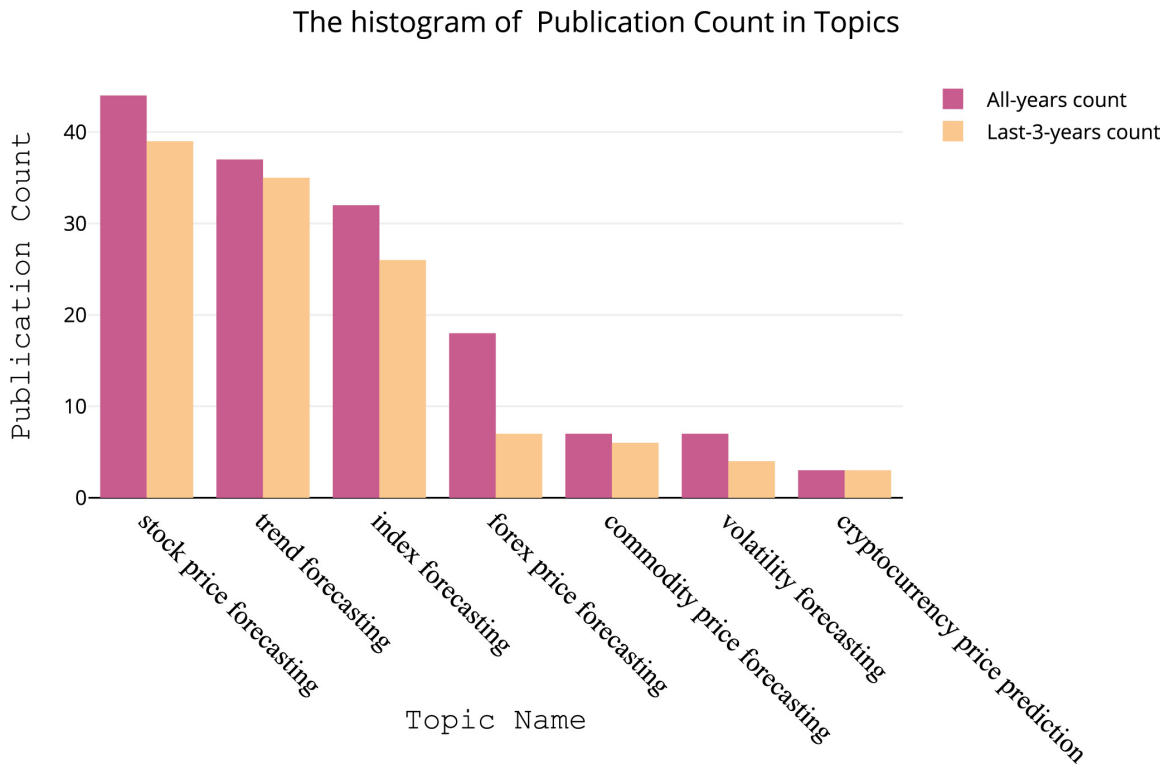


Figura 1.2: Frecuencia de tipo de pronósticos realizados en el periodo 2005–2019 [98].

índices. Entre los activos que se han intentado pronosticar tanto pare predecir valores como predecir tendencias son: Divisas (incluyendo criptomonedas), *Commodities* (p. ej. petróleo y oro), Bonos Gubernamentales, Volatilidad, entre otros (Fig. 1.2) [66, 98].

Para tener una visión amplia de los tipos de pronósticos realizados y modelos utilizados, podemos ver la Fig. 1.3, en la cual se muestra que una mayor cantidad de trabajos realizan pronósticos de precios de acciones, índices y tendencias, utilizando RNNs como modelo de predicción [98].

En la mayoría de los pronósticos el tipo de entrada que se utiliza como predictor son IT con un 39 % seguido del precio de las acciones con 31 %. Mientras que los datos no estructurados (reportes anuales, noticias económicas, noticias financieras, *blogs*, entre otros) comienzan a tener interés por porte de la comunidad científica [98], esto bajo el supuesto de que la información no estructurada revela los sentimientos de los inversionistas y que en función de ella se puede realizar un pronóstico de los mercados financieros [19, 88, 93].

Independientemente del tipo de dato de entrada (p. ej. precio de las acciones) y salida

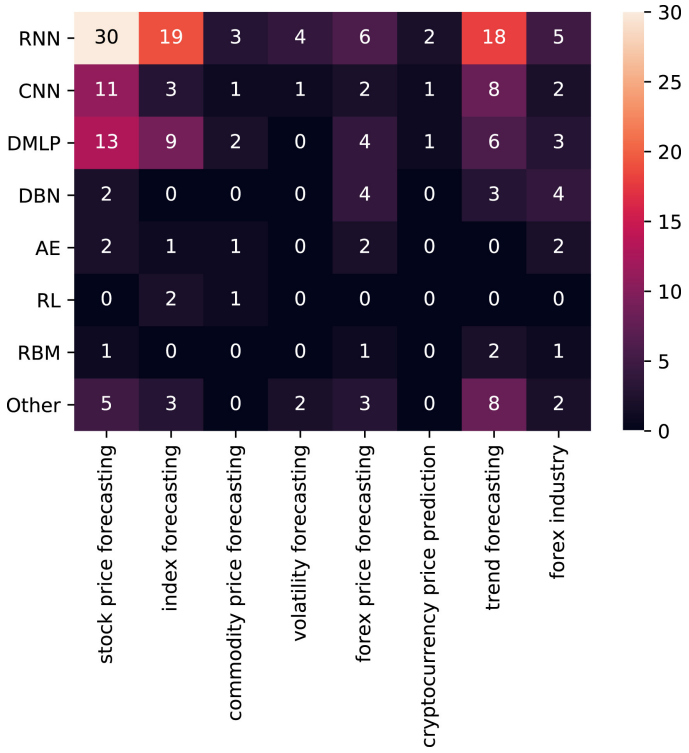


Figura 1.3: Tipo de DL vs. Tipo de pronóstico [98].

(p. ej. tendencia), la literatura sugiere realizar una normalización de los datos. El modelo más utilizado en la literatura es $min - max$ para escalar los datos de entrada/salida en el intervalo $[0, 1]$ [41, 69, 89]. Otras técnicas de escalamiento que se han utilizado son: Función Sigmoid [43, 84], *Bayesian Information Criteria* (BIC) [48], *Independent Component Analysis* (ICA) [78], $min - max$ en el intervalo $[-9, 0.9]$ [6, 47], *Logarithmic Data Smoothing* (LDS) [31], *Principal Component Analysis* (PCA) [109, 117], *Kernel-based Principal Component Analysis* (KPCA) [117], *Wavelet Transformation* (WT) [27], *Non-linear Independent Component Analysis* (NLICA) [60], entre otras.

Al generar representaciones internas, el DL no requieren de preprocesamiento de datos sofisticados o complejos para un rendimiento satisfactorio [12, 107]. Por lo que el DL permite la entrada/salida de los datos escalados en el intervalo $[0, 1]$ usando $min - max$. Esta característica del DL está en controversia en pronósticos financieros, pues algunos autores sugieren utilizar únicamente modelos de DL (p. ej. CNN y LSTM) para sustituir el preprocesamiento de datos [98, 107], mientras que otros autores proponen el uso de DL

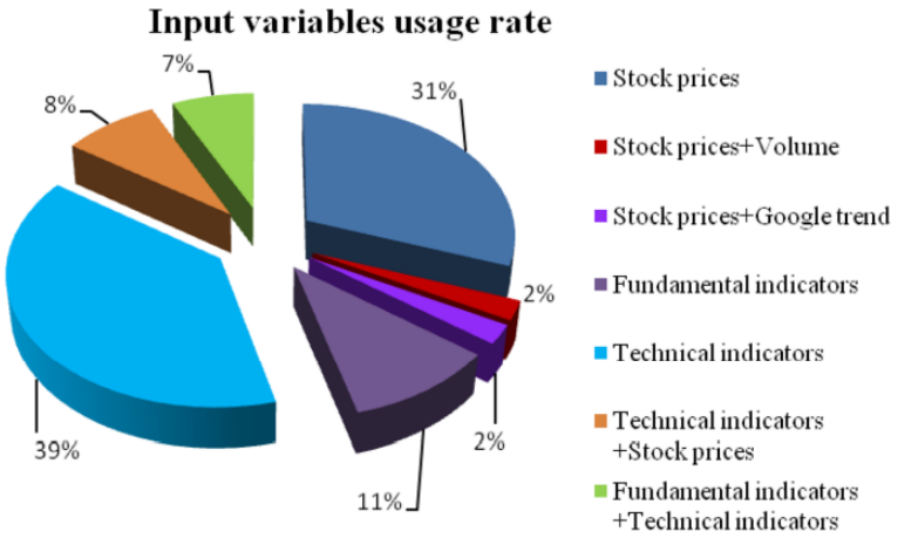


Figura 1.4: Porcentaje de tipo de variables de entrada en pronósticos financieros [66].

con IT[5] [32, 73]. Estas dos propuestas tienen resultados satisfactorios superando a modelos *benchmark*; lo cual abre el debate de si es necesario utilizar IT con DL o solo DL.

Como se puede inferir, el tipo de métrica utilizada para disminuir el error de un modelo de ML o DL depende del tipo de problema, ya sea un pronóstico de precio, pronóstico de tendencia, o análisis de sentimientos; p. ej. en regresiones se usa *Mean Absolute Error* (MAE), *Mean Absolute Diference* (MAD), *Root Mean Square Error* (RMSE), Error Cuadrático Medio (ECM), entre otras. Mientras que en clasificación se hace uso de F-*score*, *Accuracy*, *Theil coefficient* (utilizado con frecuencia en economía para medir la inequidad económica), entre otras [66, 73]. En esta tesis se utilizó ECM porque se realizó un pronóstico del precio de cierre de las acciones financieras.

Aunque se seleccione una métrica adecuada para obtener un modelo con mayor precisión no siempre se obtiene un alto rendimiento, por lo que algunos autores han abordado el problema integrando al entrenamiento tanto la volatilidad como los rendimientos, utilizando el índice Sharpe y *Cross-Entropy* (1.2), con lo cual han obtenido resultados satisfactorios [87]; mientras que otros autores cambian el enfoque, proponiendo Weighted-F-Score, integrando el error y rendimientos, dándole diferente peso a los errores (1.7) [107]. Esto da información

[5]Los IT son un tipo de preprocesamiento de datos utilizados en análisis técnico para realizar pronósticos financieros.

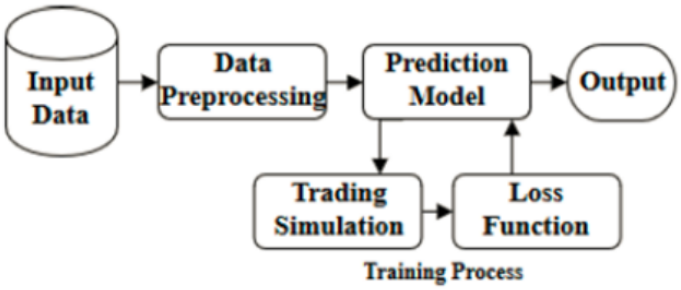


Figura 1.5: Metodología de Oncharoen & Vateekul (2018) [87]

relevante para futuras investigaciones en las que se considere el pronóstico financiero de series de tiempo en conjunto con la rentabilidad de las inversiones.

El índice Sharpe se calcula:

$$SharpeRatio = \frac{Mean(Returns)}{Std(Returns)} \tag{1.1}$$

donde $Mean(Returns)$ es la media del retorno de inversión en un periodo de tiempo determinado, y $Std(Returns)$ es la desviación estándar de los retornos de inversión en un periodo determinado.

La pérdida basada en entropía cruzada y el índice Sharpe se calcula:

$$Loss = CrossEntropy \cdot (1 - alpha) + alpha \cdot \log\left(\frac{1}{max(0.01, SharpeRatio)}\right) \tag{1.2}$$

donde $alpha$ es un parámetro que da peso a los componentes.

Oncharoen & Vateekul (2018) [87] realizan un entrenamiento de una red CNN-LSTM y simulan inversiones con la red, calculan el rendimiento y el índice Sharpe, para finalmente calcular la pérdida (Fig. 1.5).

$$N_{tp} = N_{tu} + N_{td} + \beta_3^2 N_{tu} \tag{1.3}$$

$$E_{1st} = E_{wutd} + E_{wdtu} \tag{1.4}$$

$$E_{2nd} = E_{wutf} + E_{wdtf} \tag{1.5}$$

$$E_{3rd} = E_{wftu} + E_{wftd} \tag{1.6}$$

$$F = \frac{1 + \beta_1^2 + \beta_2^2 N_{tp}}{(1 + \beta_1^2 + \beta_2^2)N_{tp} + E_{1st} + \beta_1^2 E_{2nd} + \beta_1^2 E_{3rd}} \tag{1.7}$$

donde N_{tp} es el valor de *True Positive*, E_{1st}, es el error tipo 1 que consiste en una predicción de tendencia contraria (p. ej. si la tendencia era al alza, "*Up*", y se predijo como a la baja, "*Down*"), E_{2nd} es el error tipo 2 que se produce cuando se predijo una tendencia y en realidad no hubo tendencia ("*Flat*"), E_{3rd} es el error tipo 3 el cual predice que no habrá tendencia cuando en realidad existirá una tendencia. N_{tu} es el valor de *True Up*, N_td es el valor de *True Down*, N_{tf} es el valor de *True Flat*, E_{wutd} es el valor de *Wrong UP but True Down*, E_{wdtu} es el valor de *Wrong Down but True Up*, E_wutf es el valor de *Wrong Up but True Flat*, E_{wdtf} es el valor de *Wrong Down but True Flat*, E_{wdtf} es el valor de *Wrong Down but True Flat*, E_{wftu} es el valor de *Wrong Flat but True Up*, E_{wftd} es el valor de *Wrong Flat but True Down*. Y $\beta_1 = \beta_2 = 0.5$ y $\beta_3 = 0.125$, son los parámetros de cada error respectivamente.

La idea general de la ec. (1.7) es dar un peso a los errores, porque el error de vender cuando hay una alza del precio de las acciones tiene mayor impacto que si se decide no realizar alguna acción de compra/venta cuando hay una alza del precio de las acciones. De ese modo, las betas le asignan un peso a cada uno de los errores.

Finalmente, en las investigaciones gran parte de los datos, como OCLHV de las acciones, se obtienen de fuentes libres, como: Yahoo Finance, Google Finance, entre otros [17]; no obstante, esta información por lo general se obtiene por día; también se puede obtener la información financiera en intervalos de tiempo menores, p. ej. el precio de las acciones por micro-segundo (información que se puede utilizar para diseñar sistemas de comercio

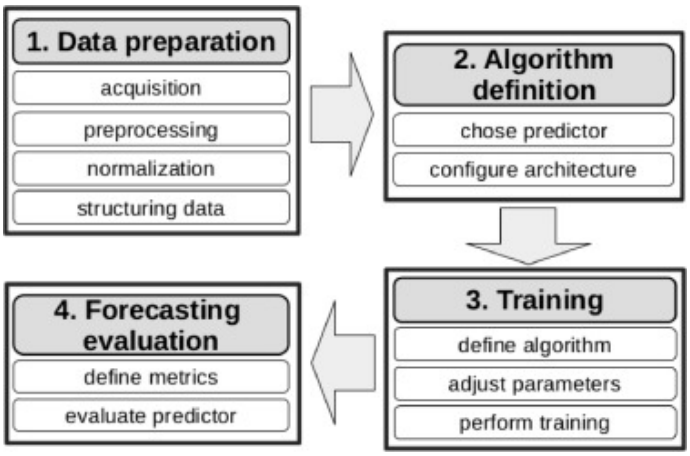


Figura 1.6: Metodología identificada por Cavalcante *et al.* (2016) [18].

automatizados para el HFT) desde otras plataformas, por ejemplo de la Bolsa Mexicana de Valores[6].

1.1.3. Metodología en pronósticos financieros usando Inteligencia Computacional

En investigaciones de Mercados Financieros (MF) y PF se identificó que la metodología de trabajo es la misma que se utiliza en ML: preprocesar datos, selección de algoritmo, entrenamiento del modelo y llevar a producción.

De acuerdo a la revisión científica realizada por Cavalcante *et al.* (2016) [18] la mayoría de los trabajos se limitan a realizar pronósticos financieros dejando de lado el sistema de comercio. Cavalcante *et al.* (2016) [18] resume la metodología en la Fig. 1.6; por lo que el propone una metodología que contemple Estrategias de Comercio (*Trading Strategies*) y Evaluación de Rendimientos (*Money Evaluation*) (Fig, 1.7), consistiendo en: 1) Sistema de reglas para decidir cuando comprar o vender; 2) sistema de control de riesgo, fijando montos de pérdida; 3) monto de transacción por operación; 4) la evaluación del rendimiento [18, 20].

Kumar *et al.* (2021) [66] en su revisión científica encuentran una metodología de trabajo similar a la de Cavalcante *et al.* (2016) 1.6 (Fig. 1.8). Sin embargo, ellos proponen su propia metodología basándose en la revisión científica realizada por ellos (Fig. 1.9). Los

[6]https://www.bmv.com.mx/.

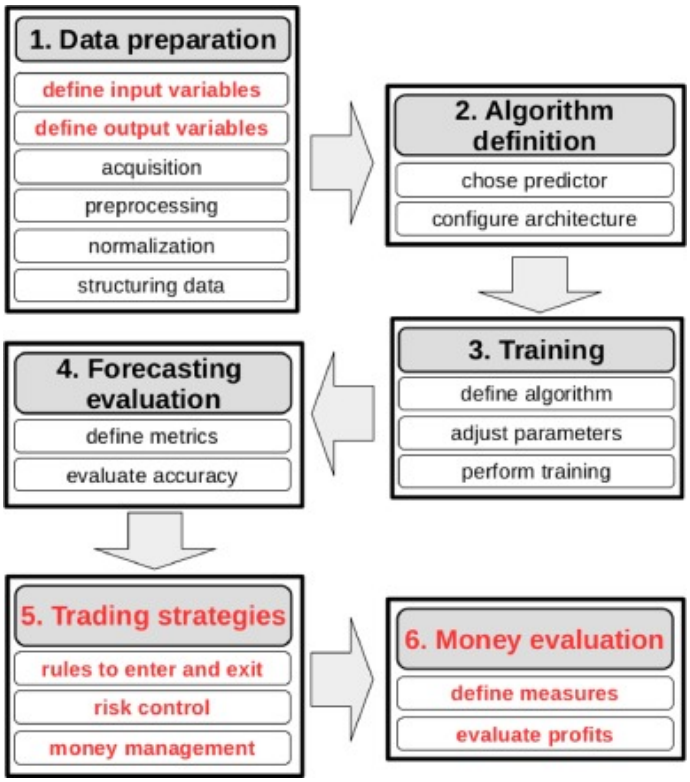


Figura 1.7: Metodología propuesto por Cavalcante *et al.* (2016) [18].

pasos son: 1) Obtener los datos históricos de las acciones; 2) calcular IT utilizando OCLHV; 3) preprocesar los datos normalizando en el rango de $[0, 1]$; 4) aplicar la extracción de características sobre los IT para determinar cuál de ellos tiene mayor influencia en el precio de las acciones y reducción de dimensiones; 5) desarrollar modelos híbridos de pronósticos financieros usando ANNs; 6) aplicar optimización evolutiva para ajustar los parámetros del modelo para proveerlo de precisión; 7) evaluar la capacidad de predicción utilizando diferentes métricas; 8) presentar la señal de comercio o el precio de la acción que se predice. Kumar *et al.* (2021) [66] consideran la hibridación de redes neuronales con algoritmos evolutivos al identificar en diferentes trabajos que son los que mejor rendimiento generan, por tal razón en la presente tesis se decide usar redes neuronales híbridas; del mismo modo Kumar *et al.* (2021) [66], encuentran que el escalamiento de los datos en el intervalo de entre 0 y 1 es de lo más usado en la literatura, por lo que se decide utilizar dicha normalización en la presente tesis.

En esta sección se identifica la existencia de preguntas abiertas, problemas y nuevos

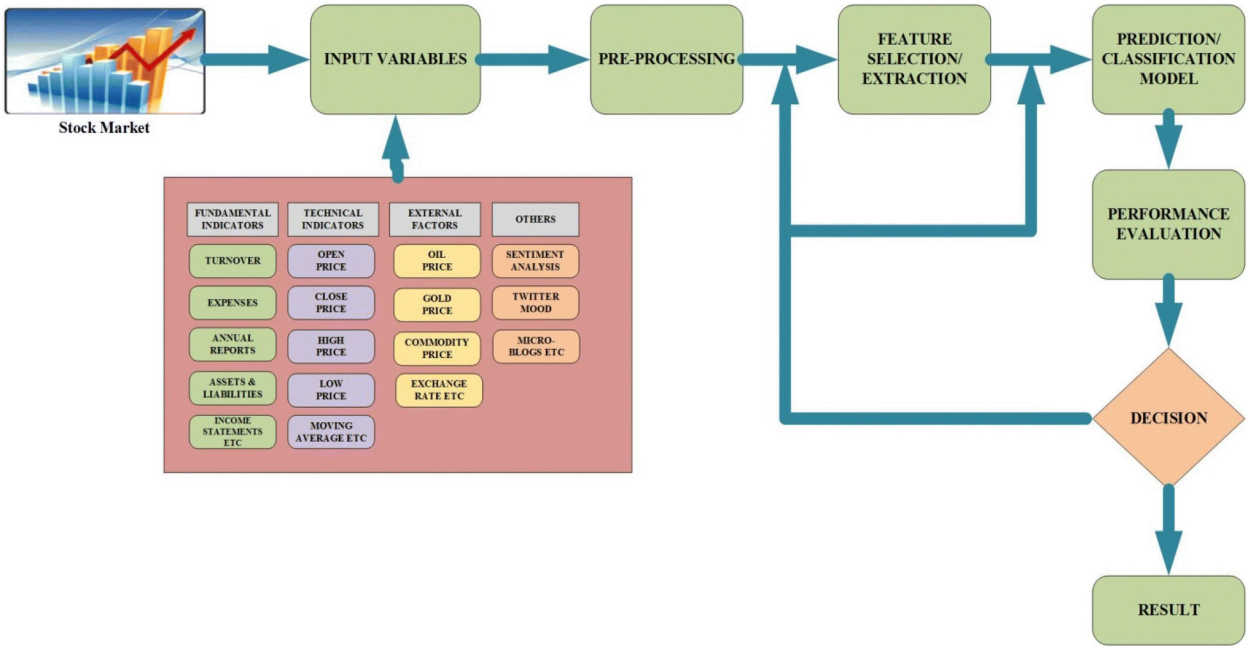


Figura 1.8: Metodología identificada por Kumar *et al.* (2021) [66].

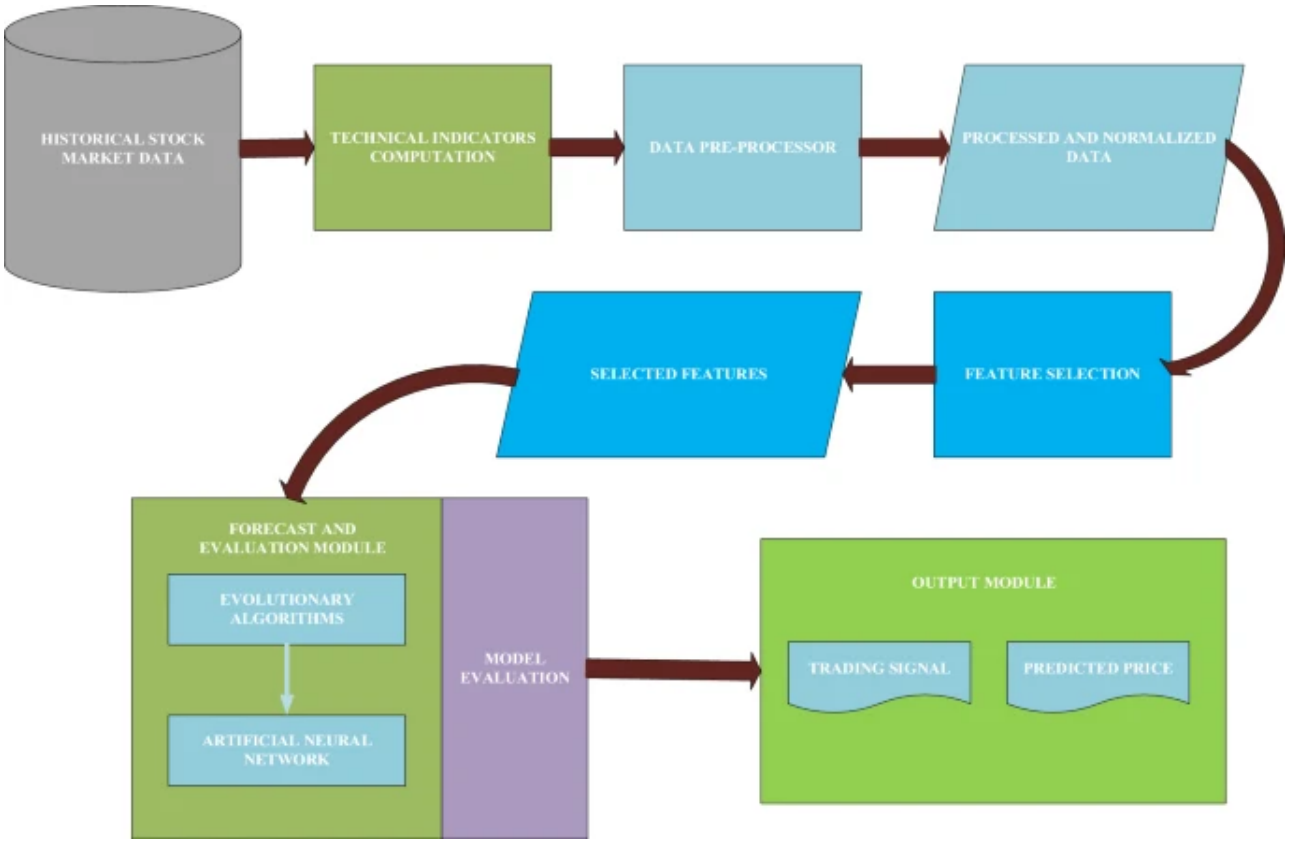


Figura 1.9: Metodología propuesta por Kumar *et al.* (2021) [66].

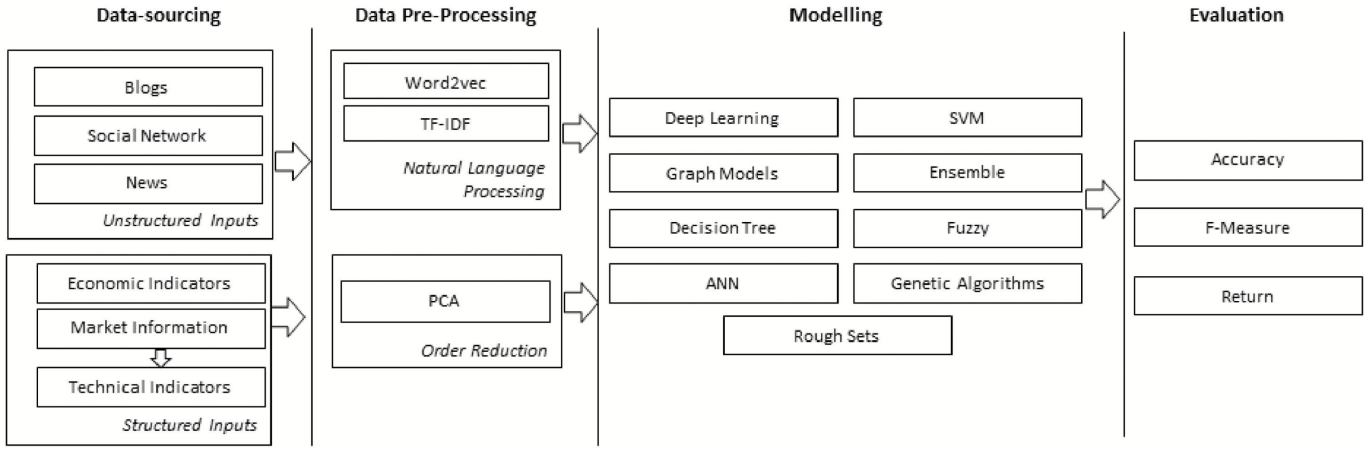


Figura 1.10: Metodología identificada por Bustos *et al.* (2020) [17].

campos de investigación. Se plantea saber si el uso de indicadores técnicos es relevante para un adecuado pronóstico financiero usando DL o usar únicamente DL; o si es mejor hacer pronóstico de precios o de tendencias; en los problemas, encontramos que no siempre un modelo con alta precisión se traduce en altos rendimientos, incluso la literatura propone desde utilizar otro tipo de funciones de pérdida hasta usar diferentes métricas de evaluación. También se plantea usar diferentes tipos de modelos DL como GAN, CNN o usar modelos de RL, DRL y modelos híbridos (desde hibridar redes con otro tipo de redes, hasta usar algoritmos evolutivos con redes para ajustar sus parámetros y/o hiperparámetros). Esto muestra que el campo de pronósticos financieros aún tiene bastante por resolver. En el presente trabajo solo nos enfocamos en un problema, el cual se expone en la siguiente sección.

1.2. Planteamiento del problema

En los antecedentes se exponen diferentes líneas de investigación a seguir en pronósticos financieros como: probar nuevas arquitecturas y/o modelos de DL, probar distintas maneras de preprocesar datos (p. ej. IT o la serie únicamente escalada) e hibridar RNNs con algoritmos evolutivos, entre otros.

Siguiendo la línea de optimización de hiperparámetros y bajo los antecedentes del éxito que han tenido las redes LSTM en pronósticos financieros, se formula la pregunta, **¿se puede**

realizar la optimización de hiperparámetros de una red tipo LSTM utilizando algoritmos metaheurísticos para que la red realice un pronóstico financiero?

1.3. Justificación

El ajuste de hiperparámetros de las ANN y DL es un trabajo que requiere experiencia, por lo general éste se lleva a cabo mediante *Grid Search* y *Random Search* [12], no obstante, campos como el *Automated Machine Learning* (AutoML) buscan automatizar el proceso tanto de preprocesamiento de datos y selección de características, como de optimización de hiperparámetros, mediante diferentes algoritmos como modelos Bayesianos [53] y Algoritmos Genéticos [53, 61], y DRL [23]. Este campo ha tenido auge en los últimos años por el uso de ML y DL en diferentes campos, donde se exige tener personal especializado.

En pronósticos financieros, el uso de DL requiere de ajuste de hiperparámetros para un adecuado rendimiento [45, 75, 98, 106]. Se han realizado pocos trabajos en la HPO, en el campo de PF, empero con éxito y con sugerencias de la literatura como campo de exploración (la literatura en finanzas le llama hibridación de redes con algoritmos evolutivos) por lo cual realizar investigaciones en este campo es relevante tanto para pronósticos financieros como para el AutoML [5, 66, 68].

1.4. Objetivos

1.4.1. Objetivo general

- Realizar pronóstico financiero usando una red neuronal tipo LSTM optimizando sus hiperparámetros con un algoritmo metaheurístico.

1.4.2. Objetivos específicos

1. Buscar qué algoritmos se han utilizado para optimizar hiperparámetros de redes neuronales artificiales.

2. Buscar qué algoritmos se han utilizado para optimizar hiperparámetros de redes neuronales artificiales tipo LSTM.

3. Generar un pronóstico financiero optimizando los hiperparámetros de una red neuronal artificial tipo LSTM usando *Particle Swarm Optimization* (PSO).

4. Generar un pronóstico financiero optimizando los hiperparámetros de una red neuronal artificial tipo LSTM usando *Flower Polination Algorithm* (FPA).

5. Generar un pronóstico financiero optimizando los hiperparámetros de una red neuronal artificial tipo LSTM usando *Estimation of Distribution Algorithm for a Low Number of Function Evaluations* (LNFE[7]).

6. Generar un pronóstico financiero optimizando los hiperparámetros de una red neuronal artificial tipo LSTM usando Symmetric-Approximation Energy-Based Estimation of Distribution (SEED).

Además de las metaheurísticas mencionadas en los objetivos 1 y 2, se contemplan otras metaheurísticas seleccionadas previamente como PSO y FPA porque se identifican que en la literatura ha tenido resultados satisfactorios en optimización de hiperparámetros [67] y se propone utilizar EDAs, por el éxito que han tenido en otros campos [77].

1.5. Hipótesis

Se puede realizar un pronóstico financiero usando una red neuronal artificial tipo LSTM optimizando los hiperparámetros usando un algoritmo metaheurístico.

1.6. Alcance y limitaciones

Si bien, existen diferentes líneas de investigación mostradas en antecedentes y propuestas novedosas en el estado del arte, en el presente trabajo solo se realizó un pronóstico de acciones

[7]En este trabajo nos referimos a EDALNFE como LNFE.

financieras de Google y Nike usando una red LSTM por su ventaja de trabajar directamente con series de tiempo y obtener rendimientos satisfactorios. El proceso de optimización de hiperparámetros se llevó a cabo con tres metaheurísticas: PSO, FPA y LNFE; excluyendo a SEED porque los tiempos de entrega de investigación no permitieron evaluar el algoritmo con diferentes funciones *benchmark* y comprobar que el algoritmo está funcionando de manera correcta.

Capítulo 2

Marco Teórico

En el presente capítulo se muestra la teoría de la red neuronal Long Short - Term Memory (LSTM), así como los algoritmos metaheurísticos utilizados: *Flower Pollination Algorithm* (FPA), *Particle Swarm Optimization* (PSO), *Estimation of Distribution Algorithm for a Low Number of Function Evaluations* y al final del capítulo se habla del *Automated Machine Learning* (AutoML) enfocandose en la optimización de hiperparámetros (HPO, por sus siglas en inglés).

2.1. Perceptrón

El percetrón es el concepto básico de una red neuronal artificial. El percetrón simula una neurona biológica. Una neurona biológica transmite la información hacia otra neurona mediante la sinapsis, que es el enlace que se encuentra entre neurona y neurona (Fig. 2.1). Esta información que recibe la neurona hace que la neurona se active o no, generando una salida o pulso (*spike*). El perceptrón, utiliza pesos o parámetros (vector W) para simular la sinapsis y la información que pasa de una neurona a otra neurona es un vector (X), que representa la información que viaja de neurona a neurona [12].

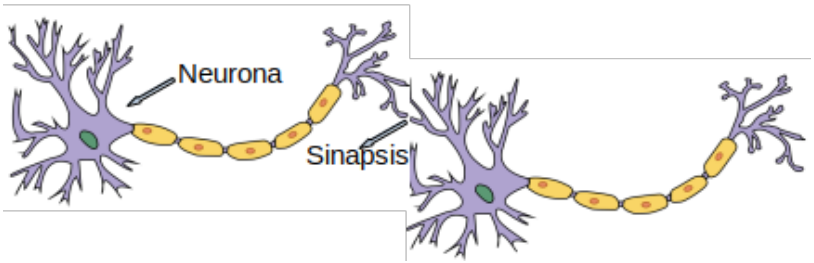


Figura 2.1: Sinapsis de una neurona biológica.(Imagen editada de [12])

El perceptrón se define como:

$$z_i = \sum_{i=1}^{|X|} x_i w_i + b = X \cdot W \tag{2.1}$$

donde w_i es el i-ésimo parámetro de un perceptrón, x_i es la i-ésima entrada, b es un sesgo (*bias*) para desplazar la recta del origen y $|X|$ es el tamaño de la entrada del vector X y en la representación vectorial $b = x_0$.

Para simular la activación, la red neuronal puede utilizar funciones matemáticas, las cuales se definen como funciones de activación.

La activación de un perceptrón se define como:

$$y = f_{activation}(z) \tag{2.2}$$

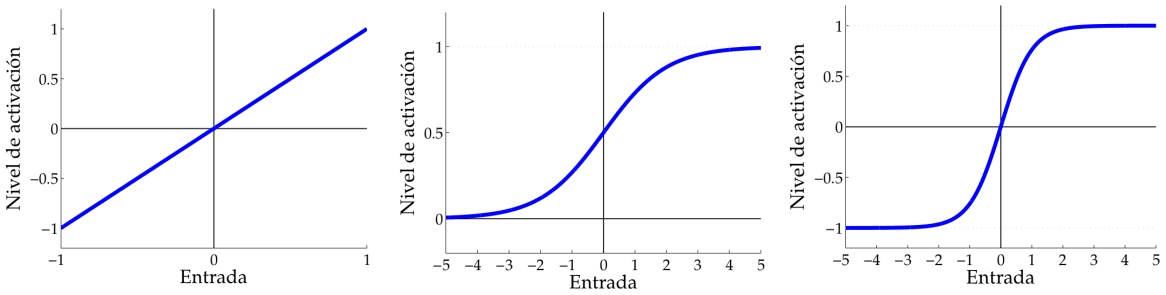
Las funciones de activación se utilizan dependiendo del problema que se desea resolver, ya sea de regresión, clasificación o generar datos, entre otros. En la siguiente sección se definen algunas funciones de activación.

2.1.1. Funciones de activación

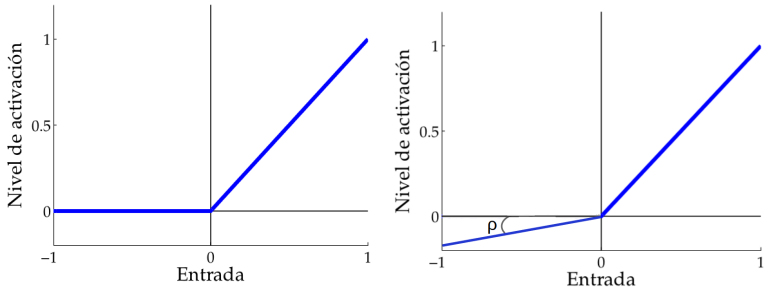
A continuación se exponen algunas funciones de activación.

Función de activación lineal (Fig. 2.2(a)):

$$y = f_{lin}(z) = z \tag{2.3}$$



(a) Lineal (b) Sigmoide (c) Tangente hiperbólica



(d) ReLu (e) Leaky ReLu

Figura 2.2: Funciones de Activación. (Imágenes tomadas de [12])

Función de activación sigmoide (Fig. 2.2(b)):

$$y = \text{sig}(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

Función de activación tangente hiperbólica (Fig. 2.2(c)):

$$y = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.5}$$

Función de activación ReLU (Fig. 2.2(d)):

$$y = f_{relu}(z) = max(0, z) \tag{2.6}$$

Función de activación Leaky ReLU (Fig. 2.2(e)):

$$y = f_{relu}(z) = max(0, z) + min(\rho z, 0) \tag{2.7}$$

donde ρ es un parámetro que por lo general es menor a 1 y se ajusta por el diseñador de la red neuronal.

Cada función de activación se utiliza dependiendo del problema o aplicación que se esté haciendo con la red neuronal artificial. Cada una de las funciones realiza diferente recorrido como se puede apreciar en las gráficas de la Fig. 2.2. P. ej. en caso de realizar una regresión se puede utilizar la función de activación lineal (2.3); o si se desea hacer clasificaciones se puede utilizar la función ReLu (2.6); todo depende del problema a resolver[1].

2.2. Perceptrón Multicapa

El perceptrón multicapa (MLP, por sus siglas en inglés), es un modelo matematico que simula una red neuronal artificial que consiste en una capa de entrada, una o varias capas ocultas y una capa de salida. Donde cada puede tener uno o múltiples perceptrones. La capa de entrada, está conectada hacia adelante y la capa o capas ocultas están conectadas hacia adelante hasta conectar con la capa de salida [12, 106].

El MLP para generar una salida o propagación hacia adelante se realiza los siguientes pasos [12, 106]:

1) Activaciones de las neuronas de entrada:

$$A = X \tag{2.8}$$

donde A es el vector de activaciones de la cada de entrada.

2) Activaciones de las neuronas de la capa oculta c:

$$A_c = f_{activation}(Z_c) \tag{2.9}$$

donde $Z_c = W_{c-1}A_{c-1}$, A_{c-1} es el vector de las activaciones de la capa $c-1$, W_{c-1} es el vector de parámetros de la capa $c-1$.

3) Activaciones de las neuronas de la capa de salida C:

[1]Para conocer más funciones de activación ver [12].

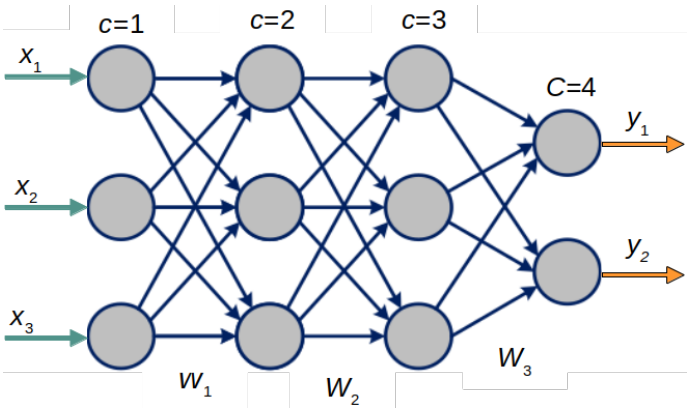


Figura 2.3: MLP con tres capas ocultas. (Imagen editada de [12].)

$$Y_C = f_{activation}(Z_c), \text{ para } i = 1, \ldots, n_C \tag{2.10}$$

donde $Z_C = W_{C-1}A_{C-1}$, A_{C-1} es el vector de activaciones de las neuronas de la última capa oculta y W_{C-1} es el vector parámetros de la última capa oculta.

En la Fig. 2.3, se muestra un MLP; en la Fig. 2.3, cada círculo es un perceptrón o neurona, las flechas verdes son las entradas, las flechas azules son los parámetros y las flechas anaranjadas son las salidas. Éste MLP tiene una capa de entrada con 3 perceptrones, 2 capas ocultas, cada una con 3 perceptrones y una capa de salida con dos perceptrones. Y para realizar la propagación hacia adelante se hace lo siguiente (2.14):

$$A_1 = X \tag{2.11}$$

donde A_1 es la activación de la capa de entrada.

$$A_2 = f_{activation}(W_1 A_1) \tag{2.12}$$

donde A_2 es la activación de la primera capa oculta.

$$A_3 = f_{activation}(W_2 A_2) \tag{2.13}$$

donde A_3 es la activación de la segunda y última capa oculta.

$$Y = f_{activation}(W_3 A_3) \tag{2.14}$$

donde Y es la salida de la red neuronal.

El modelo MLP tiene la desventaja de que no tiene memoria (almacenar entradas que recibió anteriormente), como se observa en la propagación hacia adelante, toda la información va en una sola dirección, entra la información y sale información; el segundo problema es que las entradas tienen que ser de una dimensión predefinida y no puede recibir un vector de distinta dimensión. En respuesta a los dos problemas mencionados se proponen las Redes Neuronales Recurrentes [12, 106].

2.3. Red Neuronal Recurrente

Las Redes Neuronales Recurrentes (RNNs, por sus siglas en inglés) nacen como propuesta que se da a las redes MLP, las cuales no tienen memoria y no pueden recibir entradas de diferente tamaño [12, 106].

Una RNN, es un tipo de red neuronal que a diferencia de la red neuronal MLP, puede recibir entradas de diferente dimensión, porque la RNN maneja las entradas como secuencias y hacen uso de su "memoria" para almacenar cada una de las partes de la secuencia para integrarlas con la siguiente entrada [12, 106]. Éste proceso se lleva a cabo de manera recurrente, p. ej. tenemos una secuencia de entrada $X = [x_1, x_2, \ldots, x_n]$, y queremos predecir x_{n+1}, donde n es el número de elementos del vector. Lo que hace la función recurrente es lo siguiente:

$$H_1 = f_{recurrent}(x_1, H_0) \tag{2.15}$$

donde $f_{recurrent}$ es la función recurrente, x_1 es elemento 1 del vector X y H se le llama estado oculto, en este caso $i = 0$ y $H_0 = 0$, siendo el estado oculto inicial. Después se genera el siguiente estado oculto:

$$H_2 = f_{recurrent}(x_2, H_1) \tag{2.16}$$

éste proceso se repite hasta que:

$$H_{n+1} = f_{recurrent}(x_n, H_n) \tag{2.17}$$

donde H_{n+1} se procesa con otro vector de parámetros W y pasa por una función de activación para obtener la salida $\hat{x}_{n+1}$ (2.18) que es la estimación de x_{n+1}.

De esta manera se puede procesar el estado oculto para obtener una salida:

$$y = f_{activation}(WH_{n+1}) = \hat{x}_{n+1} \tag{2.18}$$

éste proceso de recurrencia tiene similitud con el proceso de propagación hacia adelante de una red MLP, como se vio en la sección anterior en las Ecs. (2.11) a (2.14). Por lo que se puede imaginar una RNN como una MLP con un número de capas iguales al tamaño de la secuencia y que si cambia el tamaño de la secuencia cambia el número de capas del MLP; el efecto de cambiar del número de capas se debe a la recurrencia, pero en la RNN es el proceso de recurrencia (Fig. 2.4); por la razón de que las entradas en una RNN se trabajan como secuencias, la RNN puede trabajar con entradas de diferente tamaño.

Una vez entendida la idea general del funcionamiento de las RNNs, en los siguientes párrafos se explica de manera más formal la teoría de las RNNs, la RNNs simple y la red LSTM.

Para exponer la teoría de una RNN, primero definimos $\mathbf{x}_{i:j}$ para denotar una secuencia de vectores $\mathbf{x}_i, \ldots, \mathbf{x}_j$.

Una RNN es una función recurrente que toma como entrada una secuencia ordenada de tamaño arbitrario n, donde cada vector tiene d_{in} dimensiones, quedando del modo $\mathbf{x}_{1:n} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, donde $\mathbf{x}_i \in \mathbb{R}^{d_{in}}$ y retorna una señal como vector de d_{out} dimensiones, el cual es $\mathbf{y}_n \in \mathbb{R}^{d_{out}}$ [44].

$$\mathbf{y}_i = RNN(\mathbf{x}_{1:i}) \tag{2.19}$$

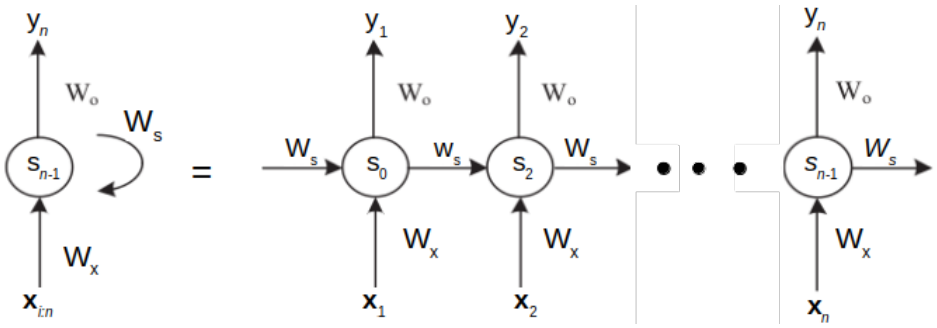


Figura 2.4: Propagación de una RNN. (Imagen editada de [108].)

donde la señal $\mathbf{y}_i$ se obtiene del vector $\mathbf{x}_{1:i}$ de la secuencia $\mathbf{x}_{1:n}$; donde $\mathbf{x}_{1:i} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i]$. La secuencia que contiene cada salida i se denota por $RNN^*(\cdot)$ y se expresa de la siguiente manera:

$$\mathbf{y}_{1:n} = RNN^*(\mathbf{x}_{1:n}) \tag{2.20}$$

Para generar la propagación hacia adelante de una RNN se toma un estado $\mathbf{s}_{i-1}$ y toma un vector de entrada $\mathbf{x}_i$ para generar un nuevo estado s_i, éste nuevo estado es mapeado a un vector y_i usando una función de activación $f_{activation}(\cdot)$ [44]. Cuando se recibe la primera entrada de la secuencia el estado inicial ($\mathbf{s}_0$) es igual a cero. En la Fig. 2.4 se ilustra el proceso de propagación; donde W_0 es el parámetro o parámetros de salida, W_s el parámetro o parámetros del estado oculto y W_x es el parámetro o parámetros de entrada.

Cuando se construye una RNN primero se definen las dimensiones de la entrada $\mathbf{x}_i$, así como la dimensión de la salida $\mathbf{y}_i$. La dimensión del estado $\mathbf{s}_i$ está en función del tamaño o dimensión de la salida.

$$RNN^*(\mathbf{x}_{1:n}; \mathbf{s}_0) = \mathbf{y}_{1:n} \tag{2.21}$$

$$\mathbf{y}_i = f_{activation}(\mathbf{s}_i) \tag{2.22}$$

$$\mathbf{s}_i = f_{activation}(\mathbf{s}_{i-1}, \mathbf{x}_i) \tag{2.23}$$

Esta es la idea básica de una RNN, a continuación se dan detalles de una RNN simple

o Elman Network (SRNN, por sus siglas en inglés) en honor a Elman, quien la propuso en 1990.

La SRNN se formula de la siguiente manera:

$$\mathbf{s}_i = R_{SRNN}(\mathbf{x}_i, \mathbf{s}_{i-1}) = f_{activation}(\mathbf{s}_i W_{\mathbf{s}} + \mathbf{x}_i W_{\mathbf{x}} + b) \tag{2.24}$$

$$\mathbf{y}_i = F_{SRNN}(\mathbf{s}_i) = f_{activation}(\mathbf{s}_i W_o) \tag{2.25}$$

donde $\mathbf{s}_i$ e $\mathbf{y}_i \in \mathbb{R}^{d_x}$, $W_{\mathbf{x}} \in \mathbb{R}^{d_x \times d_s}$ son los parámetros (pesos o "conexiones") de $\mathbf{x}$, $W_{\mathbf{s}} \in \mathbb{R}^{d_s \times d_s}$ son los parámetros (pesos o "conexiones") de $\mathbf{s}$, donde d_x, es la dimensión del vector de entrada, d_s es la dimensión del estado oculto, $W_o \in \mathbb{R}^{d_s \times d_o}$ son los parámetros de salida, donde d_o es la dimensión de la salida y $b \in \mathbb{R}^{d_s}$ es el sesgo o *bias*.

Cada uno de los estados $\mathbf{s}_{i-1}$ y entradas $\mathbf{x}_i$ son linealmente transformados al multiplicarse matricialmente por los pesos o "conexiones" de la red. Estos pesos sirven para generar la información llamada estado el cual mantiene la información de la RNN que se concatena con la siguiente entrada o genera la salida final.

Los estados por lo general se pasan por una función de activación, la cual puede ser lineal o no lineal, p. ej. si se esta trabajando en clasificación de texto, se puede usar una función SoftMax, la cual produce un vector de probabilidad, usado para predecir la probabilidad de que un texto pertenezca a una determinada clase; en otros casos donde se realiza predicciones de números (problema de regresión), se puede utilizar una función sigmoide, bajo la idea de que la salida esta escalda en el intervalo $[0, 1]$, o trabajar con una función lineal [12,44,106]. Todo depende del problema a resolver.

Una de las desventajas de las SRNN es que en el entrenamiento entre más larga sea la secuencia puede generar un desvanecimiento del gradiente (*vanishing gradient*); para ejemplificar, se supone que se tiene una secuencia con 10 vectores, el primer gradiente es igual a 0.1, el segundo gradiente es de $0.1 \cdot 0.1 = 10^{-2}$, si esto se repite en los siguientes gradientes tendremos un gradiente de 10^{-10}, un número cercano a 0, por lo que al actualizar los pesos de la SRNN será mínimo, impidiendo el aprendizaje de la SRNN. En el entrenamiento

puede surgir lo contrario, una explosión del gradiente (*exploding gradient*); para ejemplificar, tenemos una secuencia con 10 vectores, el primer gradiente es de 10, el segundo gradiente es de $10 \cdot 10 = 10^2$, si se repite el proceso tendremos 10^{10}, por lo que al actualizar los pesos de la SRNN será excesivo, impidiendo una convergencia y aprendizaje de la SRNN. Por esta razón, las SRNN tienen problemas al trabajar con secuencias largas. En la siguiente sección presentamos una red neuronal que logra resolver el problema del *vanishing gradient* y *exploding gradient* [45].

2.4. Long Short - Term Memory

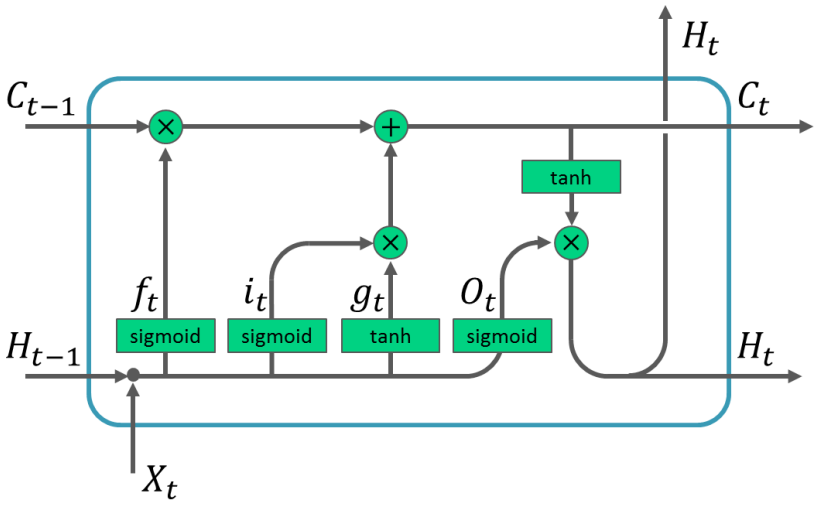


Figura 2.5: Estructura de una celda LSTM. (Elaboración propia).

Long Short - Term Memory (LSTM) se propone en 1997 por los autores Hochreiter & Schmidhuber, con el objetivo de evitar los problemas de *vanishing gradient* y *exploding gradient* en el entrenamiento que se presenta en RNNs más simples p. ej. la SRNN. En los estudios realizados por Hochreiter & Schmidhuber (1997) [50], encuentran que la LSTM supera a la SRNN y Neural Sequence Chunking al trabajar con secuencias de entrada más largas. Por ejemplo en el problema de *Reber Grammar*, el procentaje de éxito de una red LSTM está por encima del 97 %, mientras que una SRNN tiene resultados menores a 1 % [50].

Para evitar el *vanishing gradient* y *exploding gradient*, Hochreiter & Schmidhuber (1997) [50] usan compuertas lógicas y un modelo de gradiente que trunca el error para algunos

parámetros, esto permite a la red LSTM aprender de secuencias (retrasos) más largas que las redes SRNN, llegando hasta secuencias de entrada de tamaño igual a 1000. El uso de compuertas lógicas permite a la red LSTM recordar secuencias de entrada más largas, con una complejidad de O(1) al ser un modelo local en espacio y tiempo [50].

La red LSTM tiene un vector de estados que funciona como memoria para guardar información de la secuencia a lo largo del tiempo. En la Fig. 2.5 se presenta una celda LSTM, en la parte superior de la celda LSTM se muestra una serie de flechas, las cuales representan la *cell state* de la Ec. (2.30); la *cell state* funciona de memoria a largo plazo y corto plazo de la red LSTM.

La *cell state* se encarga de transmitir información relévente del estado C_t al estado C_{t+1}. Para mantener la información relevante, se añade o se elimina información conforme la red recibe los datos de una secuencia $X_{1:T}$. Para añadir información a la *cell state*, se utiliza una compuerta lógica *forgate gate* (2.27), la *forgate gate* utiliza una función sigmoide (2.4) para procesar la información entre $[0, 1]$, de esta manera, al realizar la multiplicación $f_t \otimes C_{t-1}$ se elimina información de la *cell state* [2] [44].

Para añadir información a la *cell state* la entrada se procesa utilizando la compuerta lógica *input gate* (2.26) que procesa la información entre $[0, 1]$ con una función sigmoide (2.4) indicando que tanta información nueva se añade, la información candidata que se añadirá se genera con la ecuación *candidate gate* (2.29), esta ec. procesa la información entre $[-1, 1]$ utilizando una función tanh (2.5) y cuando se realiza la multiplicación $i_t \otimes g_t$ se obtiene la nueva información que se añade a la *cell state* [44].

Con lo explicado en los dos párrafos anteriores, tenemos qué cantidad de información se elimina de la *cell state* ($f_t \otimes C_{t-1}$) y que información se añade a la *cell state* ($i_t \otimes g_t$), de esta maenera se obtiene la *cell state* ($C_t = f_t \otimes C_{t-1} + i_t \otimes g_t$) actualizada [44].

Para generar el estado oculto (*hidden state*), se utiliza la *cell state* (2.30) en combinación con la compuerta lógica *output gate* (2.28). Primero se escala la *cell state* entre $[-1, 1]$ utilizando una función tanh (2.5) y con la *output gate* se decide que cantidad de información

[2] En el presente trabajo sig$(\cdot)$, se refiere a la función sigmoide; tanh$(\cdot)$, es la función tangente hipérbolica; y finalmente $\otimes$ se refiere al producto elemento por elemento o producto Hadamard.

será parte del siguiente estado oculto, porque *output gate* provee de un valor entre $[0, 1]$ utilizando una función sigmoide (2.4). De esta manera, al multiplicar $O_t \otimes \tanh(C_t)$, se genera el nuevo estado oculto H_t (*hidden state*) [44].

$$\textbf{\textit{Input gate}}: \quad i_t = \text{sig}(\mathbf{x}_t U_i + H_{t-1} W_i + b_i), \quad i_t \in \mathbb{R}^{d_H} \tag{2.26}$$

$$\textbf{\textit{Forget gate}}: \quad f_t = \text{sig}(\mathbf{x}_t U_f + H_{t-1} W_f + b_f), \quad f_t \in \mathbb{R}^{d_H} \tag{2.27}$$

$$\textbf{\textit{Output gate}}: \quad O_t = \text{sig}(\mathbf{x}_t U_o + H_{t-1} W_o + b_o), \quad O_t \in \mathbb{R}^{d_H} \tag{2.28}$$

$$\textbf{\textit{Candidate gate}}: \quad g_t = \tanh(\mathbf{x}_t U_g + H_{t-1} W_g + b_g), \quad g_t \in \mathbb{R}^{d_H} \tag{2.29}$$

$$\textbf{\textit{Cell state}}: \quad C_t = f_t \otimes C_{t-1} + i_t \otimes g_t, \quad C_t \in \mathbb{R}^{d_H} \tag{2.30}$$

$$\textbf{\textit{Hidden state}}: \quad H_t = O_t \otimes \tanh(C_t) \tag{2.31}$$

$$\textbf{\textit{State}}: \quad \mathbf{s}_t = R_{LSTM}(\mathbf{s}_{t-1}, \mathbf{x}_t) = [C_t; H_t], \quad \mathbf{s}_t \in \mathbb{R}^{2d_H} \tag{2.32}$$

donde U y $W \in \mathbb{R}^{d_{\boldsymbol{x}} \times d_H}$ son los pesos, donde $d_{\boldsymbol{x}}$ es la dimensión de la entrada y d_H es la dimensión del estado oculto, y $\mathbf{x}_t \in \mathbb{R}$. En caso de utilizar tamaños de lote mayor a 1 entonces: $\mathbf{s}_t \in \mathbb{R}^{2d_H \times d_b}$, C_t, H_t, i_t, f_t, O_t, $g_t \in \mathbb{R}^{d_H \times d_b}$, y $b \in \mathbb{R}^{d_b \times d_H}$, donde d_b el tamaño del lote[3].

Para generar el siguiente estado oculto (H_{t+1}) y la siguiente *cell state* (C_{t+1}) se hace el proceso de manera recurrente utilizando el estado oculto y la *cell state* de t (H_t y C_t)

Una vez explicada la red LSTM se puede entender que al realizar el entrenamiento el algoritmo *backpropagation* propaga el error a través de la *cell state* evitando el *vanishing gradient* y el *exploding gradient*.

Una red neuronal LSTM puede tener múltiples celdas LSTM, a éste tipo de red se llama LSTM Múltiple; sin embargo, en la presente tesis se le llama red LSTM a pesar de tener múltiples celdas LSTM, porque al final una red neuronal puede tener múltiples capas conectadas de diferente manera.

[3]En esta tesis también se refiere al tamaño del lote como B_s.

2.4.1. Hiperparámetros

La presente tesis busca realizar la optimización de hiperparámetros de una red LSTM, por lo que en esta sección se definen los hiperparámetros de una red LSTM. Los hiperparámetros que se definen, son los que se optimizaron en esta tesis.

Los hiperparámetros que se optimizaron en la presente tesis se presentan en la Tabla 2.1. Cada uno juega un papel en el funcionamiento de la red LSTM.

Para entender que es un hiperparámetro, primero se define qué es un parámetro[4]. Un parámetro de una red neuronal artificial son las conexiones de perceptrón a parceptrón, estás conexiones llamadas parámetros permiten a la red neuronal procesar información de entrada para generar una salida que resuelve un problema para la cual red fue entrenada.

Si una persona desea realizar el ajuste de los parámetros de manera manual para que la red neuronal realize determinada función, será casi imposible encontrar los parámetros adecuados de la red, porque una red neuronal: 1) puede llegar a tener miles de parámetros y 2) los valores de los parámetros se encuentra en el espacio de los números continuos. Esto hace imposible ajustar los datos de manera manual; por esta razón, el ajuste de parámetros suele llevarse a cabo con algún algoritmo basado en *backpropagation*[5] [12, 45, 106].

Mientras los parámetros son elementos de una red que son ajustados mediante un algoritmo basado en *backpropagation*, los hiperparámetros son los elementos de una red que pueden ser ajustados de manera manual; por lo general, para realizar una búsqueda de hiperparámetros se utiliza *Grid Search* (GS) o *Random Search* (RS) [38].

El número de celdas y el número de estados ocultos son hiperparámetros que definen la arquitectura de una red LSTM. La arquitectura de una red neuronal determina el rendimiento de la red; puede existir casos en los que la red sea "demasiado" profunda y generar rendimientos adecuados o por el contrario, rendimientos paupérrimos. Otro caso puede ser que la red no sea "muy" profunda y genere un rendimiento esperado o por otra parte un

[4]Los hiperparámetros y los parámetros no son elementos exclusivos de las redes LSTM, estos elementos también se presentan en distintos algoritmos de ML y para cada algoritmo de ML los hiperparámetros pueden ser distintos.

[5]Para saber más del algoritmo de entrenamiento *backpropagation* revisar [94].

rendimiento pésimo; la razón de ello es que los hiperparámetros deben estár sintonizados al momento de realizar el entrenamiento para que la red tenga un funcionamiento esperado, de lo contrario su rendimiento será pésimo [12, 45, 106].

El número de estados ocultos es un hierparámetro de la arquitectura de la red LSMT, el cual determina el número de neuronas que tendrá cada una de las celdas, afectando en la profundidad de la red y en su rendimiento, una red con "demasiados" estados ocultos corre el riesgo de sobre ajustarse en el entrenamiento; por el contrario, con un número "reducido" de estados ocultos se corre el riesgo de que la red LSTM no aprenda de los ejemplos en el entrenamiento [12, 45, 106].

El número de épocas es se refiere al número de veces que se le presentan los ejemplos (conjunto de entrenamiento) a una red neuronal artificial para que la red aprenda a solucionar un problema. El número de épocas puede ser elevado, promedio o bajo. Bajo el supuesto que los demás hiperparámetros están sintonizados y no cambian, si el número de épocas es "elevado", existirá una alta probabilidad de que el algoritmo se sobre ajuste en el entrenamiento, mientras que si el número es "demasiado bajo", la red neuronal no aprenderá en el entrenamiento. [12, 45, 106].

El factor de aprendizaje (α) al ser un valor que afecta en la velocidad de aprendizaje, un factor de aprendizaje que es elevado, no permite que el error de entrenamiento converja, mientras que un valor pequeño deviene entrenamiento lento, porque necesita un elevado número de épocas para lograr el aprendizaje, esto se traducen en un alto costo computacional. Un adecuado ajuste de α produce un entrenamiento adecuado con un costo computacional razonable [12, 45, 106].

El tamaño del lote es un hierparámetro que juega un papel al calcular el error en el proceso de entrenamiento, porque en el entrenamiento se puede calcular el error sobre un ejemplo o sobre un conjunto de ejemplos, en ese caso se calcula un error promedio. Cuando se utiliza más de un ejemplo para calcular el error se dice que se está utilizando un lote. Y el tamaño de lote define el número de ejemplos que se utiliza para calcular el error en el proceso de entrenamiento de la red neuronal artificial [12, 45, 106].

Tabla 2.1: Hiperparámetros.

Hiperparámetro	**Símbolo**
Celda LSTM	Cl_s
Estado oculto	H_s
Tamaño de la ventana	W_s
Factor de aprendizaje	α
Tamaño de lote	B_s
Número de épocas	E_s

El tamaño de la ventana (W_s) es un concepto que se utiliza en finanzas, para referirse a la secuencia de entradas que recibe una red recurrente; esta secuencia tiene p. ej. el precio de las acciones al cierre del día en un periodo de tiempo; o puede contener indicadores técnicos en un periodo de tiempo determinado. Bajo los supuestos que: la secuencia esta formada por el precio de las acciones al cierre del día, si el tamaño de la secuencia es igual a 10; entonces se expresa que "el tamaño de la ventana es de 10 o igual a 10" [67].

Otro hiperparámetro que se utilizó en la presente tesis es el *dropout*; éste hiperparámetro no se optimizó, pero se utilizó en el proceso de entrenamiento. El *dropout* es una técnica de regularización que evita el sobre ajuste de los parámetros de una red neuronal artificial. El efecto que provoca en el entrenamiento es un porcentaje de parámetros seleccionados de manera aleatoria, no son ajustados en una época en el entrenamiento [12].

2.5. Metaheurísticas

Las metaheurísticas son una familia de las técnicas de optimización aproximada. Las metaheurísticas proveen soluciones "aceptables" en un tiempo razonable a problemas complejos en ciencia e ingeniería. Las metaheurísticas no garantizan la solución óptima sin embargo la solución es lo suficientemente aceptable para poder tomar decisiones. No obstante, a diferencia de los algoritmos de aproximación, las metaheurísticas no definen qué tan cerca está una solución del óptimo [102].

El término metaheurística está compuesto por dos palabras: *heurística* del antiguo Griego *heuriskein* el cual significa, el arte de descubrir nuevas estrategias o reglas para resolver

problemas; y *meta*, también del antiguo Griego, que significa, metodología de nivel superior [102].

Las metaheurísticas son un tipo de algoritmo que tiene origen en los algoritmos de búsqueda informada; los algoritmos de búsqueda informada usan heurísticas para guiar la búsqueda. Por ejemplo, se tiene un laberinto y se desea encontrar la ruta de salida de éste. Si el lector imagina ver el laberinto desde lo alto, estaría viendo un plano como el diseñado por un arquitecto. Si el lector dibuja una línea recta desde el inicio del laberinto hasta su salida, podrá encontrar la distancia más corta, en caso de no existir algún laberinto. Ahora bien, si esta información se le da a un individuo para solucionar el laberinto, el individuo sabrá que entre más se aleje de esa distancia, más lejano estará de encontrar la ruta para salir del laberinto, por lo que el individuo intentará buscar una ruta que se aleje lo menos posible a la distancia que se le dio para guiar su búsqueda. En este caso la distancia juega el papel de función heurística; una función heurística provee a los algoritmos de búsqueda información adicional del problema, lo cual permite evaluar que tan buena es la solución encontrada [95]; en el caso de las metaheurísticas, éstas utilizan otro tipo de heurísticas para solucionar problemas, p. ej., los algoritmos genéticos imitan la evolución genética como heurística para obtener mejores soluciones para determinado problema; otro ej. es el caso *Particle Swarm Optimization*, el cual usa la información de la mejor solución encontrada para guiar la búsqueda (esto se explica en el apartado 2.5.2), también está el algoritmo *Simulated Annealing* (SA), el cual se inspira en el proceso de recocido y enfriamiento del metal para cambiar sus propiedades, de la misma manera existe una infinidad de metaheurísticas inspiradas en diferentes fenómenos y especies naturales [40, 102].

Las metaheurísticas se utilizan por lo general en problemas que no pueden ser solucionados por métodos exactos (p. ej. usando cálculo diferencial y método simplex); las metaheurísticas también se aplican para resolver problemas con funciones no continuas o a los problemas llamados NP-Completos. Se recomienda utilizar métodos exactos siempre que sea posible [102]; de no ser posible aplicar un método exacto, se opta por utilizar metaheurísticas. En la presente tesis, el problema de optimización de hiperaparámetros tiene variables

mixtas, por lo que utilizar metaheurísticas para encontrar una solución aproximadamente óptima es viable.

2.5.1. Metaheurísticas basadas en poblaciones

Las metaheurísticas basadas en poblaciones (P-metaheurísticas) tienen características similares entre sí. Las P-metaheurísticas proveen mejoras en una población de manera iterativa. Las P-metaheurísticas inician con una población; después generan una nueva población y se integra a la población actual mediante un proceso de selección, el proceso de selección consiste en seleccionar las mejores soluciones o mejor solución de la nueva población; la idea central es que conforme se generan nuevas poblaciones estás sean mejores que las anteriores. Este proceso se repite hasta que se cumple un criterio de paro [102]. En el Algoritmo 1 se muestra el funcionamiento general de una P-metaheurística.

Algoritmo 1 Funcionamiento general de una P-metaheurística [102]

1: $P \leftarrow P_0$. ▷ Generar población inicial.
2: $t \leftarrow 0$
3: **while** *Criterio de paro* no se cumpla, **do**
4: $P_t^{'} \leftarrow$ Generar$(P_t^{'})$ ▷ Generar una nueva población.
5: $P_{t+1} \leftarrow$Seleccionar-Población$(P_t \cup P_t^{'})$ ▷ Seleccionar la nueva población.
6: $t \leftarrow t + 1$
7: Identificar la mejor solución $\mathbf{g}^*$.
8: **end while**
9: **return** La mejor solución $\mathbf{g}^*$ encontrada.

Diferentes algoritmos como los Algoritmos Evolutivos, *Scatter Search*, *Artificial Immune Systems*, *Bee Colony*, *Estimation of Distribution Algorithms*, *Particle Swarm Optimization* [102] y *Flower Pollination Algorithm*, pertenecen a las P-metaheurísticas. En los siguientes apartados se muestran las P-metaheurísticas utilizadas en la presente tesis.

Si bien, RS es un algoritmo utilizado para ajustar los hiperparámetros en algoritmos de ML, las P-metaheurísticas tienen ventajas sobre RS, porque RS es un algoritmo de búsqueda no informada y las P-metaheurísticas son algoritmos de búsqueda informada, lo que es una ventaja sobre RS al realizar la búsqueda en una menor cantidad de tiempo, teóricamente.

Por esta razón las P-metaheurísticas son una opción viable para realizar la búsqueda de hipearparámetros.

2.5.2. Particle Swarm Optimization

Particle Swarm Optimization (PSO) canónico fue propuesto por J. Kennedy y R.C. Eberhart en el año de 1996. Éste algoritmo metaheurístico se inspira en el comportamiento social de animales, insectos y humanos; considerado como algoritmo bio-inspirado [40].

El algoritmo PSO trabaja de la siguiente manera: inicia con una población de individuos (partículas) en una posición ($\mathbf{x}_i$) determinada de manera aleatoria, la posición es una solución. El algoritmo va moviendo las partículas a una velocidad ($\mathbf{v}_i$) a través del espacio del problema, buscando mejorar su posición mediante perturbaciones, aprovechando la información de sí mismo (la mejor posición en la que alguna vez estuvo la i-ésima partícula, la cual es la posición $\mathbf{p}_i$) y sus vecinos (mediante la mejor solución $\mathbf{g}^*$ de la población).

La actualización de la velocidad de las partículas se da mediante las siguientes ecuaciones:

$$\mathbf{v}_i^{New} \leftarrow \mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{g}^* - \mathbf{x}_i) \tag{2.33}$$

la ecuación que actualiza $\mathbf{v}_i$, a la que llamamos "*update* $\mathbf{v}_i$", donde $\varphi_1 = c_1 R_1$, $\varphi_2 = c_2 R_2$, R_1 y R_2 son valores aleatorios uniformes en el intervalo $[0, 1]$, c_1 es el coeficiente cognitivo y c_2 el coeficiente social. La Ec. (2.33) muestra los tres componentes que representan el comportamiento de las partículas, el primero v_i es el momentum, el cual es la velocidad actual; el siguiente componente cognitivo $\varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i)$; y por último, el componente social $\varphi_2 \otimes (\mathbf{g}^* - \mathbf{x}_i)$. Estos tres componentes son utilizados para crear una nueva velocidad $\mathbf{v}_i^{New}$ que desplaza a la partícula (Ec. (2.34). El pseudocódigo del PSO se muestra en el Algoritmo 2.

$$\mathbf{x}_i^{New} \leftarrow \mathbf{x}_i + \mathbf{v}_i^{New} \tag{2.34}$$

ésta es la ecuación que actualiza $\mathbf{x}_i$, a la que llamamos *update* $\mathbf{x}_i$.

Se puede modificar el algoritmo para hacerlo más general, añadiendo inercia y restrin-

Algoritmo 2 PSO, asumiendo minimización [40]

Generación de población inicial de manera aleatoria $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$.
Encontrar la mejor solución $\mathbf{g}^*$ en X.
while *Criterio de paro* no se cumpla, **do**
    for each partícula $\mathbf{x}_i$ $(i = 1$ to $n)$ **do**
        if $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ **then** ▷ $f(\cdot)$ es la función *fitness* o función objetivo.
            $\mathbf{p}_i \leftarrow \mathbf{x}_i$; actualizar la mejor posición de la partícula i.
        end if
        Identificar la mejor posición, $\mathbf{g}^*$, de la iteración.
        Actualizar velocidad $\mathbf{v}_i$ vía (2.33).
        Actualizar posición $\mathbf{x}_i$ vía (2.34).
    end for
end while
return La mejor solución encontrada $\mathbf{g}^*$.

giendo el movimiento de las partículas. La primera manera es añadiendo un peso o coeficiente de inercia w, modificando la ecuación (2.33) de la siguiente manera [40]:

$$\mathbf{v}_i^{New} \leftarrow w\mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{g}^* - \mathbf{x}_i) \tag{2.35}$$

El coeficiente de inercia w disminuye la velocidad en cada iteración, haciendo que el algoritmo intensifique la búsqueda, generando la convergencia en una menor cantidad de iteraciones [40].

La segunda modificación del PSO sirve para evitar que las partículas salgan del espacio de búsqueda; esto se logra añadiendo el coeficiente de constricción (χ) a la ecuación (2.35), realizando la modificación se obtiene [40]:

$$\mathbf{v}_i^{New} \leftarrow \chi(w\mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{g}^* - \mathbf{x}_i)) \tag{2.36}$$

$$\chi = \frac{2}{|2 - \varphi' - \sqrt{\varphi'^2 - 4\varphi'}|} \tag{2.37}$$

donde $\varphi' = c_1 + c_2$, $\varphi' > 4$, siendo φ' un escalar, la literatura recomienda $c_1 = c_2 = 2.05$ [40, 102].

2.5.3. Flower Pollination Algorithm

Flower Pollination Algorithm (FPA) fue desarrollado por Xin-She Yang en 2012 inspirado en el proceso de polinización de plantas; considerando a la polinización como un proceso de optimización [112].

FPA usa cuatro reglas polinización:

1. **Biotic y cross-pollination**, se consideran como los procesos de polinización global.

   - **Biotic**: polinización mediante animales o insectos.
   - **Cross-pollination**: proceso de polinización que se da solo a través de otras plantas.

   Las partículas de polen/flores son movidas por los polinizadores, los cuales obedecen el vuelo de Lévy.

2. En la polinización local se utiliza **abiotic pollination** y **self-pollination**.

   - **Abiotic pollination**: el proceso de polinización se lleva a cabo sin polinizadores sin necesitar de otra especie.
   - **Self-pollination**: polinización que se da entre la misma especie.

3. Los polinizadores pueden desarrollar **Flower Constancy**.

   - **Flower Constancy**: los polinizadores visitan solo cierta especie de flores, pasando por alto otras especies. "*Que es equivalente a una probabilidad de reproducción que es proporcional a la similitud de dos flores involucradas*" [112].

4. El cambio entre la polinización local y la polinización global puede ser controlada por el *switch probability*, $p \in [0, 1]$; los estudios muestran que un valor adecuado es 0.8, aunque en la experimentación el valor puede iniciar en 0.5 [112].

$$\mathbf{x}_i^{New} \leftarrow \mathbf{x}_i + \gamma L(\lambda_{fpa})(\mathbf{g}^* - \mathbf{x}_i) \tag{2.38}$$

donde $\mathbf{x}_i$ es la flor o el vector solución de la iteración actual, $\mathbf{x}_i^{New}$, es la nueva solución creada; $\mathbf{g}^*$ es la mejor solución de la iteración actual; γ es un factor escalar que controla el tamaño de pasos del vuelo de Lévy; $L(\lambda_{fpa})$ es el tamaño de paso basado en el vuelo de Lévy, para mover el polen y $L > 0$, se obtiene de una distribución de Lévy como se muestra en la ec. (2.39):

$$L(\lambda_{fpa}) \sim \frac{\lambda_{fpa}\Gamma(\lambda_{fpa})\sin(\pi\lambda_{fpa}/2)}{\pi} \cdot \frac{1}{s^{1+\lambda_{fpa}}}, (s \gg s_0 > 0) \tag{2.39}$$

donde $\Gamma(\lambda_{fpa})$ es la función Gamma, una distribución válida para pasos largos s; λ_{fpa}, es un parámetro con el que se generan los números aleatorios del vuelo de Lévy, se suele utilizar con un valor de 1.5. En teoría se requiere que $|s| \gg 0$, pero en la práctica puede ser tan pequeño como 0.1 [112].

El Algoritmo de Mantegna es una manera de generar números aleatorios bajo una distribución de Lévy, el cual genera el tamaño de los pesos s usando dos distribuciones Gausianas, U y V; estas dos distribuciones crean números aleatorios que generan s, mediante la siguiente transformación [112]:

$$s = \frac{U}{|V|^{1/\lambda_{fpa}}} \tag{2.40}$$

donde $U \sim \mathcal{N}(0, \sigma^2)$, y $V \sim \mathcal{N}(0, 1)$. La varianza puede ser calculada mediante [112]:

$$\sigma^2 = \left[\frac{\Gamma(1+\lambda_{fpa})}{\lambda_{fpa}\Gamma((1+\lambda_{fpa})/2)} \cdot \frac{\sin(\pi\lambda_{fpa}/2)}{2^{(\lambda_{fpa}-1)/2}}\right]^{1/\lambda_{fpa}} \tag{2.41}$$

Para realizar la polinización local se utilizan las reglas de polinización 2 y 3 para obtener la siguiente ecuación [112]:

$$\mathbf{x}_i^{New} \leftarrow \mathbf{x}_i + \epsilon(\mathbf{x}_j - \mathbf{x}_k) \tag{2.42}$$

donde $\mathbf{x}_j$ y $\mathbf{x}_k$ son dos flores que se muestrean de la misma población actual; ϵ es un número aleatorio uniforme en el intervalo $[0, 1]$. La Ec. (2.42) equivale a una caminata aleatoria.

El pseudocódigo de FPA se muestra en el Algoritmo 3.

Algoritmo 3 FPA, asumiendo minimización [112]

Crear población inicial de manera aleatoria $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$.
Encontrar la mejor solución $\mathbf{g}^*$.
Definir el sesgo de polinización $\rho \in [0, 1]$. ▷ Se sugiere que $\rho = 0.8$.
while *Criterio de paro* no se cumpla, **do**
    for each flor $\mathbf{x}_i$ ($i = 1$ to n) **do**
        if $rand < \rho$ **then** ▷ Donde $rand \in [0, 1]$ es un número aleatorio.
            Crear números aleatorios por cada variable de la solución usando el vuelo de Lévy.
            Crea solución temporal $\mathbf{x}'$ con la polinización global Ec. (2.38).
        else
            Crear un número aleatorio ϵ de manera uniforme ($\mathcal{U} \in [0, 1]$) por cada variable.
            Crear solución temporal $\mathbf{x}'$ con la polinización local Ec. (2.42).
        end if
        if $f(\mathbf{x}') < f(\mathbf{x}_i)$ **then** ▷ $f(\cdot)$ es la función *fitness* o función objetivo.
            $\mathbf{x}_i \leftarrow \mathbf{x}'$
        end if
    end for
    Identifica la mejor solución $\mathbf{g}^*$ de la iteración.
end while
return La mejor solución encontrada $\mathbf{g}^*$.

2.5.4. Estimation Of Distribution Algorithm

Los EDAs, a diferencia de otros algoritmos evolutivos que usan operadores genéticos para generar una nueva población, los EDAs crean una población a partir de una distribución de probabilidad definida de manera explícita, la cual es estimada usando un conjunto de individuos (soluciones) pertenecientes a la generación anterior [70]. El pseudocódigo del algoritmo general de un EDA se muestra en el Algoritmo 4 [4].

2.5.4.1. Estimation of Distribution Algorithm for a Low Number of Function Evaluations

LNFE es un algoritmo que realiza una búsqueda en el espacio entre $[0, 1]$; esto permite separar la búsqueda del algoritmo del problema en sí, lo que permite despreocuparse por los límites de búsqueda, mediante un escalamiento de los valores, usando por ejemplo el

Algoritmo 4 EDA general [4]

1: $t \leftarrow 0$: iteración.
2: Crear población inicial $P^{(0)}$.
3: **while** *Criterio de paro* no se cumpla, **do**
4: Seleccionar mejor solución $S^{(t)}$ de $P^{(t)}$.
5: Construir el modelo probabilístico $M^{(t)}$ desde $S^{(t)}$.
6: Muestrear de $M^{(t)}$ para generar nuevas soluciones candidatas $O^{(t)}$.
7: Incorporar $O^{(t)}$ en $P^{(t)}$.
8: $t \leftarrow t + 1$.
9: **end while**
10: **return** La mejor solución encontrada P_{best}.

escalamiento $min - max$ [77].

El algoritmo inicia creando una población aleatoria bajo una distribución uniforme, con tamaño $N_p = 8D + 2$, donde D es el número de variables a optimizar. Luego, se evalúa la población inicial mediante una función *fitness*. A continuación, se usa un algoritmo de selección, *Empirical Selection Distribution* (ESD), para asignar un valor de probabilidad a cada solución de la población de acuerdo a su valor *fitness*. La ESD tiene el objetivo de sesgar la solución hacia una región prometedora en el espacio de búsquedas. Los índices de las soluciones ordenados se almacenan en la variable *ibest* y se calcula la media y varianza usando las siguientes ecuaciones [104].

$$\mu \mathrm{x} = \frac{\sum_{i=1}^{|X|} freq_i \cdot \mathbf{x}_i}{|\widehat{S}|} \tag{2.43}$$

donde $\widehat{S}$ es una muestra de la población, y $|\widehat{S}|$ es el tamaño de la población. $freq_i$ es el número de veces en la que aparece el valor $\mathbf{x}_i$ en la muestra, X es la población y $|X|$ es el tamaño de la población.

$$\sigma \mathrm{x} = \frac{\sum_{i=1}^{|X|} \left[e^{\Delta\beta f(\mathbf{x}_i)(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T} \right]}{\sum_{i=1}^{|X|} e^{\Delta\beta f(\mathbf{x}_i)}} \tag{2.44}$$

donde σx es la covarianza, $\Delta\beta$ es un parámetro que requiere análisis, cuando se usa la Distribución Boltzmann $\Delta\beta = 1/T_b$, donde T_b es la temeperatura, como lo hacen Yunpeng *et al.* (2006) [113].

La nueva población se crea usando dos distribuciones. La primera se calcula con Distribución Normal Multivariada, por lo que se calcula con la diagonal de la matriz de covarianza; la matriz de covarianza y la varianza de la distribución se escalan siguiendo la regla: si f_{elite} mejora, la varianza aumenta, de lo contrario, disminuye.

Si el valor *fitness* del mejor individuo no mejora después de cierto número de iteraciones, la población se reinicia, con una distribución uniforme al rededor del individuo elite (mejor solución/individuo), preservando las mejor soluciones C_{resets}.

La segunda población se calcula sobre un vector de proyección, que apunta en una dirección de búsqueda prometedora. Éste vector es modificado mediante perturbaciones aleatorias. El algoritmo comprueba si el vector de proyección actual U, es mejor que el anterior mediante la correlación de Spearman entre la población actual y sus correspondientes valores *fitness*. Cuanto mayor es la correlación, mejor es el vector de proyección. Si la correlación es perfecta, el vector de proyección está en perfecto crecimiento/decrecimiento hacia una dirección de la función *fitness* (o función objetivo). Por lo tanto, lo que se hace es un muestreo en todas las dimensiones usando una Distribución Multivariada, para luego desplazarlo en dirección del vector proyección. El proceso de muestreo se realiza con el Algoritmo 6.

Entre las ventajas de LNFE se encuentra que LNFE realiza una menor cantidad de llamadas a función [6] (CF, por sus siglas en inglés) y presenta una convergencia más rápida comparado con *Grid Search* y *Differential Evolution* (DE). Otra característica de LNFE comparado con otros algoritmos evolutivos es el uso de la convergencia como criterio de paro como único parámetro ajustable, de acuerdo a Lopez-Farias *et al.* (2021) [77].

Tanto la característica de convergencia en una menor cantidad de iteraciones como una menor cantidad de CF de LNFE, hacen que éste algoritmo sea prometedor para la HPO de una red LSTM. Esto se debe a que en la presente tesis una CF implica diseñar una red LSTM y realizar un entrenamiento de una red LSTM, por lo que la búsqueda de LNFE se traduce en menos entrenamientos de reds LSTM y menor costo computacional.

[6] Una llamada a función es la avaluación de una función en un valor dado.

Algoritmo 5 LNFE [77]

$N_p \leftarrow 8D + 2$; $C_{elite} \leftarrow 0; C_{resets} \leftarrow 0$
$X \leftarrow \text{initialize}(d, N_p)$ ▷ Inicia población de manera aleatoria X.
$F \leftarrow \text{evaluate}(X, f(\cdot))$ ▷ Donde $f(\cdot)$ es la función *fitness*.
$[ibest, ps] \leftarrow \text{selection}(F)$
$[X_{elite}, F_{elite}] \leftarrow [X_{ibest(1)}, F_{ibest(1)}]$
$f_{reset} \leftarrow F_{elite(1)}$ ▷ Guardar la mejor solución.
while *Criterio de paro* ($var_c > var_{min}$) no se cumple, **do**
    $\left[\hat{X}, U\right] \leftarrow \text{GenNewInd}(X, F, X_{elite}, ps, ibest, U \leftarrow rand(D))$ ▷ Via Algoritmo 6.
    $\hat{F} \leftarrow \text{evaluate}(\hat{X}, f(\cdot))$ ▷ Evaluar la nueva población.
    if $|F_{elite} - F_{ibest(1)}| > |10^{-4} F_1|$ **then** ▷ Si no mejora
        $C_{elite} \leftarrow 0$ ▷ Reiniciar población.
        $[X_{elite}, F_{elite}] \leftarrow \left[X_{ibest(1)}, F_{ibest(1)}\right]$ ▷ Actualizar la mejor solución elite.
    else
        $C_{elite} \leftarrow C_{elite} + 1$ ▷ Incrementar el contador de elite.
    end if
    if $C_{elite} = MaxC_{elite}$ **then** ▷ Si la petición de la mejor solución se repite en un limite, reiniciar.
        $C_{resets} \leftarrow C_{resets} + 1$ ▷ Incrementar el contador de reinicio.
        $X_{elite} \leftarrow X_{ibest(1,\ldots,C_{resets})}$
        $F_{elite} \leftarrow F_{ibest(1,\ldots,C_{resets})}$
        $X \leftarrow \text{reinitialize}(D, Np - 1, C_{resets}, X_{elite})$ ▷ Reiniciar población.
        $F \leftarrow \text{evaluate}(X, f(\cdot))$ ▷ Evaluar la nueva población.
        $[X, F] \leftarrow [[X, X_{elite}], [F, F_{elite}]]$ ▷ Agregar la mejor solución elite a la nueva población.
        $[ibest, F_{elite}] \leftarrow \text{selection}(F)$
        $[X_{ibest}, ps] \leftarrow \left[X_{ibset(1)}, F_{ibest(1)}\right]$
        $C_{elite} \leftarrow 0$
        if $F_{elite(1)} = f_{reset}$ or $C_{resets} = MaxC_{resets}$ **then** ▷ Si no mejora, detener el algoritmo.
            break
        else
            $f_{reset} \leftarrow F_{elite(1)}$ ▷ Si la solución mejora actualizarla.
        end if
    end if
    $[X, F] \leftarrow [X(ibest), F(ibest)]$ ▷ Ordenar las soluciones de acuerdo a su valor *fitness*.
    $[X, F] \leftarrow [X(1, \ldots, Np/2), F(1, \ldots, Np/2)]$ ▷ Mantener la mitad de la población con mejor *fitness*.
    $[X, F] \leftarrow \left[\left[X, \hat{X}\right], \left[F, \hat{F}\right]\right]$ ▷ Añadir nuevas soluciones a la población.
end while
return La mejor solución encontrada, $[x_{elite}, f_{elite}]$

Algoritmo 6 Generar nuevos Individuos de LNFE [77]

procedure GENNEWIND($X, F, x_{elite}, ps, ibest$)
  if first-call= $True$ **then**
    $cnt \leftarrow 0$; $\sigma_1 \leftarrow -1$; $\sigma_2 \leftarrow -1$; $f_{elite}^{local} \leftarrow 1e100$
  end if
  $success \leftarrow False$
  if $f_{elite}^{local} = F_{ibest(1)}$ and $\sigma_1 = -1$ **then**
    $success \leftarrow True$
  end if
  $f_{elite}^{local} \leftarrow F_{ibest(1)}$
  μx $\leftarrow$ weightedColumnMean(X, p) ▷ Via ec. (2.43).
  σx $\leftarrow$ weightedColumnSD(X, μx$, p$) ▷ Via ec. (2.44).
  if success = $False$ **then**
    $cnt \leftarrow cnt + 1$
  end if
  increment = $True$
  if $cnt \geq 6$ or $cnt = 0$ **then**
    σx $\leftarrow 1.05\sigma$x
    $Scale_1 \leftarrow max(\sigma_1/|\sigma\text{X}|, 1)$
    if $cnt = 10$ **then**
      $cnt \leftarrow 0$
    end if
  else
    σx $\leftarrow 0.6\sigma$x
    $Scale_1 \leftarrow min(\sigma_1/|\sigma\text{X}|, 1)$
    increment $\leftarrow False$
  end if
  $\sigma_1 \leftarrow Scale_1|\sigma\text{x}|$
  $U_e \leftarrow U$
  $corr_{max} \leftarrow$ SpearmanCorr(XU_t, F)
  for $trial = 1$ to 2000 **do**
    $U_t \leftarrow U_e + randn(D)/(4\sqrt{D})$
    $U_t \leftarrow U_t/|U_t|$
    $corr_{trial} \leftarrow$ SpearmanCorr(XU_t, F)
    if $|corr_{trial}| > |corr_{max}|$ **then**
      $corr_{max} \leftarrow sign(corr_{trial}) \cdot corr_{trial}$
      $U_e \leftarrow sign(corr_{trial}) \cdot U_t$
    end if
  end for
  for $i = 1$ to D **do**
    $\hat{X}_i \leftarrow X_{ibest(1)} + Scale_1 \cdot randn(D) \cdot \sigma$x
  end for
  $D \leftarrow (0.05U_e + 0.95U)/|0.05U_e + 0.95U|$
  $\sigma_t \leftarrow \text{dot}(X_{ibest(1)} - X_{ibest(Np/2)}, U_e)$
  if increment = $True$ **then**
    $Scale_2 \leftarrow max(\sigma_2/\sigma_t, 1)$
  else
    $Scale_2 \leftarrow min(\sigma_2/\sigma_t, 1)$
  end if
  $\sigma_2 \leftarrow Scale_2\sigma_t$
  for $i = 1$ to D **do**
    $\hat{X}_i \leftarrow \hat{X}_i + (factor_s randn(\cdot) + factor_s) \cdot \sigma_t \cdot U$
    boundToLimits($\hat{X}_i$)
  end for
  return $[\hat{X}, U]$
end procedure

2.6. Automated Machine Learning

En las últimas décadas ha habido un aumento de las investigaciones en ML y aplicaciones, especialmente del DL; sin embargo, los rendimiento de los modelos de ML son muy sensibles a diferentes decisiones de diseño, p. ej. en el DL, es necesario encontrar una arquitectura "correcta", así como encontrar el tipo de regularizadores y demás hiperparámetros; hasta lograr un rendimiento deseado. Ese proceso de selección de hiperparámetros por lo general se lleva a cabo mediante prueba y error, usando RS y GS hasta encontrar un conjunto de opciones que briden al modelo de ML el rendimiento deseado [53, 67].

El campo del Automated Machine Learning (AutoML) tiene el objetivo de tomar las decisiones de selección de las técnicas de preprocesamiento de datos, selección de características y selección de hiperparámetros de manera sistemática y automatizada, aproximándose a una solución óptima [53]. De manera general, el AutoML se divide en preprocesamiento de datos y optimización de hiperparámetros.

El AutoML, al automatizar el proceso ya mencionado, permite acceder a modelos de ML que se encuentran en el estado del arte. Incluso las soluciones que provee el AutoML llegan a superar a expertos en el campo del ML; además de que el AutoML reduce los tiempos y costos al desarrollar una aplicación de ML. Por estas razones, múltiples empresas comienzan a desarrollar sus propios sistemas de AutoML [53].

Las ventajas de usar HPO son: reducir el esfuerzo humano al aplicar modelos de ML, proveer de rendimientos que compiten con el estado del arte, mejorar la reproducibilidad y proveen de imparcialidad en los estudios científicos [53].

2.6.1. Optimización de hiperparámetros

La optimización de hiperparámetros consiste en seleccionar una configuración de hiperparámetros que brinden a un modelo de ML el mejor rendimiento posible con la menor pérdida posible. En esta sección se define el problema de la optimización de hiperparámetros.

Sea $\mathcal{A}$ un algoritmo de ML con N hiperparámetros. Se denota el dominio del n-ésimo hiperparámetro como Λ_n y el espacio de configuraciones de hiperparámetros como $\mathbf{\Lambda} = \Lambda_1 \times \Lambda_2 \times \ldots \Lambda_N$. Un vector de hiperparámetros está denotado por $\lambda_{hpo} \in \Lambda$ y $\mathcal{A}$ con sus hiperparámetros instanciados λ_{hpo} se denota por $\mathcal{A}_{\lambda_{hpo}}$ [53].

El dominio de los hiperparámetros puede encontrarse en: el espacio de los continuos, el espacio de los números discretos, números binarios o categóricos, entre otros. Por esta razón se considera un problema de variables mixtas [39]. Estos espacios pueden ser condicionales, un hiperparámetro puede ser relevante cuando otro hiperparámetro toma cierto valor. Por ejemplo, al seleccionar un optimizador que utiliza o no momento, se convierte en relevante o irrelevante el parámetro del momento [53].

El problema de optimización de hiperparámetros se define como:

$$\lambda^*_{hpo} = \arg \min_{\lambda_{hpo} \in \Lambda} \mathbb{E}_{(D_{train}, D_{valid}) \sim \mathcal{D}} \mathbf{V}(\mathcal{L}, \mathcal{A}_{\lambda_{hpo}}, D_{train}, D_{valid}) \tag{2.45}$$

donde $\mathbf{V}(\mathcal{L}, \mathcal{A}_{\lambda_{hpo}}, D_{train}, D_{valid})$ mide la pérdida ($\mathcal{L}$) de un modelo $\mathcal{A}$ con los hiperparámetros λ_{hpo} entrenado con los datos D_{train} y evaluado con los datos D_{valid}. En la práctica se suele tener un conjunto de datos finito $D \sim \mathcal{D}$, por lo que se hace una aproximación con los datos en la Ec. (2.45) [53].

En la presente tesis se propone realizar la optimización de hiperparámetros utilizando P-metaheurísticas porque en el proceso de optimización, la P-metaheurística realiza la búsqueda de los hiperparámetros λ_{hpo} que minimicen la pérdida del modelo LSTM. En el proceso de optimización se utiliza un conjunto de entrenamiento D_{train} el cual es la serie de tiempo de los precios de las acciones financieras. La idea es que los hiperparámetros que encuentre la P-metaheurística, haga que la red LSTM realice un pronóstico financiero. Estas ideas se utilizan para realizar la metodología de optimización de hiperparámetros.

Capítulo 3

Estado del arte

En los últimos años se han utilizado diferentes propuesta de IC para realizar pronósticos financieros; entre las técnicas que han tomado mayor relevancia en pronósticos financieros es el ML, no obstante su uso aún tiene preguntas abiertas por resolver y líneas de investigación prometedoras, desde la hibridación de DL con Algoritmos Evolutivos para optimizar hiperparámetros [2, 29, 37, 55], generar sistemas de comercio automatizado utilizando redes neuronales y algoritmos genéticos (GA, por sus siglas en inglés) [35] hasta optimización de parámetros [10, 14, 21, 99, 111, 114] entre otros.

Si bien, existen diferentes preguntas y líneas de investigación en pronósticos financieros usando IC y ML, la presente tesis se enfoca en la HPO de una red LSTM para realizar un pronóstico financiero, porque la HPO es un campo que ha tomado relevancia por la emergencia del ML y DL, por sus aplicaciones satisfactorias en diferentes campos como: medicina [5], visión por computadora [105], problemas de física [25] entre otros campos, como en PF [67].

En el presente capítulo se analizan trabajos recientes en los que se realizan pronósticos financieros y pronósticos financieros usando DL con HPO, se analizan trabajos que usan diferentes tipos de redes neuronales artificiales y modelos de ML; también se analizan trabajos que realizan HPO en campos distintos a pronósticos financieros.

En este capítulo primero se presentan los avances más recientes en pronósticos finan-

cieros, después se presentan trabajos de optimización de hiperparámetros de modelos de inteligencia computacional en pronósticos financieros y termina con la optimización de hiperparámetros en otros campos.

3.1. Pronósticos financieros

Ante la necesidad de tener modelos de predicción precisos en pronósticos financieros, distintos autores han propuesto diferentes vías para tratar de llegar al objetivo; en esta sección se presentan distintos trabajos que buscan dicho fin.

Kanwal *et al.* (2022) [59] proponen un modelo híbrido entre Bidirectional Cuda Deep Neural Network (CuDNN[1]) [8], Long Short-Term Memory y una Convolutional Neural Network de una dimensión (BiCuDNNLSTM-1dCNN). En su trabajo compararon el modelo con 4 modelos distintos: LSTM-DNN, LSTM–CNN, CuDNNLSTM y LSTM. Los resultados experimentales muestran que BiCuDNNLSTM-1dCNN supera los algoritmos mencionados al predecir *Crude Oil*, *Global X DAX*, *Germany ETF*, *DAX Performance-Index* y *Hang Seng Index*; los errores en evaluación usando ECM se muestran en la Fig. 3.1.

La propuesta de Kanwal *et al.* (2022) [59] tiene ciertas diferencias a otros trabajos que utilizan las CNNs como filtros al inicio de la red neuronal artificial [24,107], con el objeto de generar representaciones internas en las siguientes capas, mientras Kanwal *et al.* (2022) [59] utilizan la CNN después de una capa BiCuDNNLSTM (Fig. 3.2), aunque no es habitual utilizar una capa CNN como ellos lo hacen, los resultados son satisfactorios.

Lin *et al.* (2021) [76], realizan un modelo LSTM con *Complete Ensemble Empirical Mode Decomposition with Adaptive Noise* (LSTM-CEEMDAN), para pronosticar los índices S&P500 y *China Securities* 300 (CSI300). La red que proponen la diseñan con múltiples celdas LSTM; sin embargo no especifican la arquitectura, solo especifican la hibridación con CEEMDAN. La propuesta se compara con SVM, SRNN, *Wavelet Neural Networks* (WAV), entre otros modelos. Los resultados experimentales muestra que su propuesta supera

[1]CuDNN es un modelo de DL acelerado con GPU.

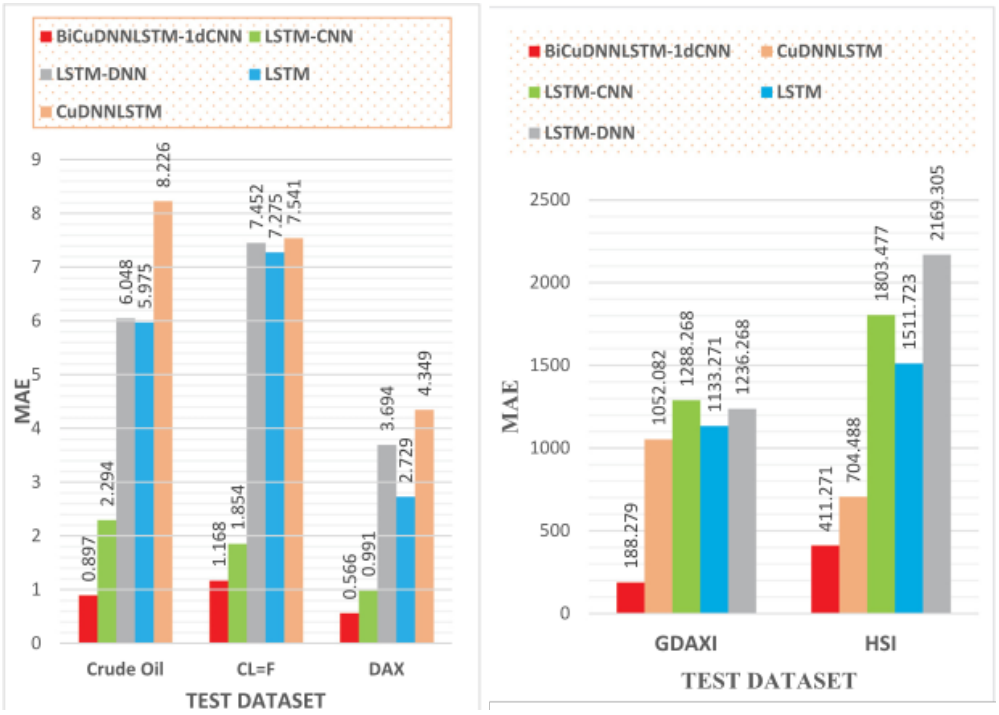


Figura 3.1: Comparación de diferentes modelos realizando pronósticos probados por Kanwal *et al.* (2022) [59].

a modelos *benchmark* como se aprecia en los ECM's de la Tabla 3.1.

Chen *et al.* (2022) [26] realizan un pronóstico financiero de las acciones de 16 bancos de China. Primero realizaron una clusterización para agrupar los bancos con tendencias de precios similares. La información de los precios de las acciones clusterizados fue normalizada para entrenar una red LSTM, la cual realizará un pronóstico *many - to - many* (Fig. 3.3). La propuesta del modelo le llaman KD-LSTM. Los autores toman un banco de cada clúster obtenido y utilizan la información de su clúster para realizar el pronóstico. Su propuesta la comparan con un modelo LSTM que solo utiliza el histórico del banco que predice. Los resultados de la comparación se muestran en la Tabla 3.2. De acuerdo a sus resultados experimentales, su propuesta KD-LSTM tiene mayor precisión que otros modelos *benchmark*, lo que permite tener decisiones más rentables.

Un punto relevante del trabajo de Chen *et al.* (2022) [26] es que agrupan a los bancos que están correlacionados; de esta manera la red neuronal puede pronosticar no solo el comportamiento de un banco, sino de otros bancos correlacionados. Esto puede extrapolarse a las acciones financieras, agruparlas y no solo predecir una sola, sino varias utilizando

Tabla 3.1: Resultados experimentales de Lin *et al.* (2021) [76].

Índice	**Modelo**	**ECM**
S&P500	LSTM	4554.9380
	SVM	1386.4169
	SRNN	1337.2046
	WAV	2025.7638
	CEEMDAN-LSTM	436.0208
CSI300	LSTM	3646.5100
	SVM	2375.7305
	SRNN	2327.1569
	WAV	5500.6258
	CEEMDAN-LSTM	615.0082

el mismo modelo; esto también se puede trabajar con los índices porque son indicadores generales de las economías; p. ej. el índice Standard & Poor's 500 (S&P500) pondera las acciones de las compañías que cotizan en Standard & Poor's, por lo que entrenar una red para pronosticar el índice S&P500 podría tener la capacidad de pronosticar diferentes acciones que se operan en Standard & Poor's.

Tabla 3.2: Resultados experimentales de Chen *et al.* (2022) [26].

	ECM	
Banco	**Modelo KD-LSTM**	**LSTM**
BCM	0.0017	0.0031
CEB	0.0014	0.0056
BOC	0.0015	0.0070
CNCB	0.0109	0.0228

BCM significa Bank of Communications, CEB significa China Everbright Bank, BOC significa Bank of China y CNCB significa China CITIC Bank.

Chen *et al.* (2021) [24] realizan un pronóstico de las tendencias de las acciones financieras. Los autores utilizan una *Graph* CNN (GC–CNN) para analizar el comportamiento individual de las acciones y la información del mercado, como volatilidad y acciones correlacionadas. El correlacionar activos financieros para realizar pronósticos se repite en el trabajo de Chen *et al.* (2022) [26]; la ventaja de Chen *et al.* (2021) [24] es que analiza todo la información de manera simultánea con GC-CNN. La propuesta de Chen *et al.* (2021) [24] es puesta a prueba simulando inversiones. En la simulación, tiene mayor rendimiento que

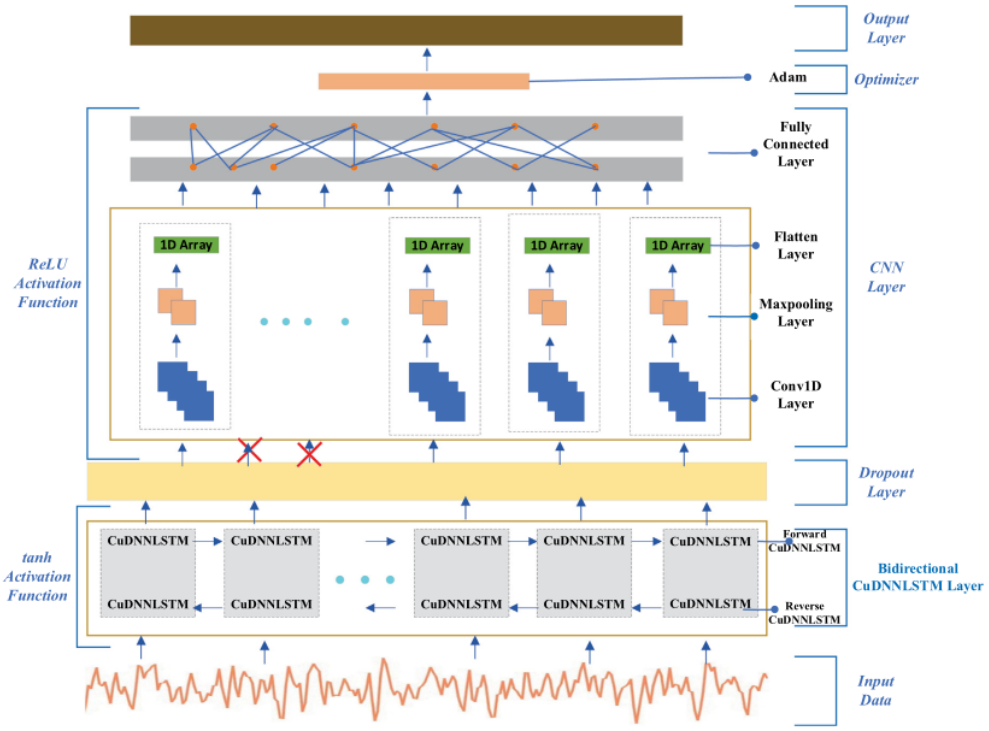


Figura 3.2: Arquitectura propuesta por Kanwal *et al.* (2022) [59].

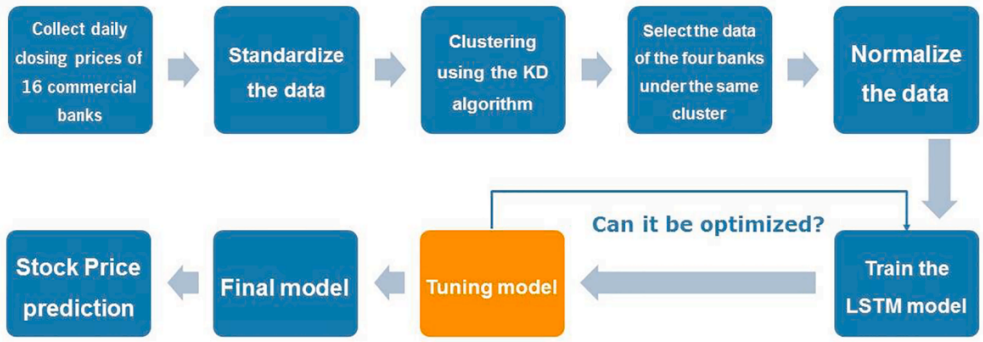


Figura 3.3: Metodología de pronóstico de Chen *et al.* (2022) [26]

usando las señales: SRI, SMA short-term y MACD [2] entre otros modelos. La simulación se realizó en diferentes sectores económicos, en cada uno de los sectores el porcentaje del retorno anual sobre la inversión fue mayor, en la Tabla 3.3 se muestran los resultados del sector de Alimentos y Bebidas.

Tabla 3.3: Resultados experimentales de Chen *et al.* (2021) [24] en el sector de Alimentos y Bebidas.

Modelo	**Retorno promedio Anual**
GC–CNN	33.42 %
Señal RSI	8.30 %
Señal SMA short-term	7.05 %
Señal MACD	9.25 %
CNN-LSTM	6.65 %

Retorno promedio anual de inversión utilizando la propuesta GC-CNN de Chen *et al.* (2021) [24] en el sector de Alimentos y Bebidas.

Para calcular el retorno promedio anual se utiliza la siguiente Ec.:

$$ARR = \left[\left(\frac{TMoney}{StartMoney} \right)^{\frac{1}{\text{No. Años}}} - 1 \right] \cdot 100\,\% \tag{3.1}$$

donde ARR es el retorno promedio anual, $TMoney$ = capital disponible+(precio de cierre)(No. acciones) y $StartMoney$ es el monto de dinero con en el que se inicia la inversión.

Los trabajos de Chen *et al.* (2021) [24] y Chen *et al.* (2022) [26], dejan ver dos cosas: 1) el encontrar la correlación entre activos financieros permite utilizar la información para realizar pronósticos más precisos y 2) usar *Graph* CNN[3] para analizar información financiera de manera simultánea y con resultados satisfactorios.

Colasanto *et al.* (2022) [30] proponen mejorar el modelo Black y Litterman's[4], para realizar pronósticos de distintas acciones financieras que cotizan en el *NASDAQ-100*, Amazon, PepsiCo, Apple, Microsoft, Broadcom, Adobe, Mondelez, AstraZeneca, T-Mobile, Netflix, entre otras. Para llevar a cabo el pronóstico hicieron análisis de sentimientos de noticias

[2]Las señales SRI, SMA short-term y MACD son estrategias de inversión utilizadas comúnmente en finanzas.

[3]Las *Graph* CNN se identificaban como modelos propmetedores en el 2020 por Sezer *et al.* [98]

[4]Black y Litterman's es un modelo de finanzas para el diseño de portafolios de inversión de acciones financieras.

financieras (p. ej., del Financial Times) con la ayuda de procesamiento de lenguaje natural (NLP, por sus siglas en inglés) del modelo BERT [83]. Los autores diseñan un sistema dinámico de portafolio de inversión; consistiendo en actualizar los porcentajes de inversión de las diferentes acciones conforme la información de las noticias fluye, de ese modo aumentan los beneficios. Aunque su propuesta solo tuvo rendimientos positivos en 4, Microsoft Inc. con 6.63 %, Apple con 5.67, EssilorLuxottica con S.A 4.7 % y UnipolSai Assicurazioni con 9.18 %. Los autores concluyen que su propuesta puede ayudar a inversores con poca experiencia a orientar sus decisiones cuando alguna noticia impacta en el precio de las acciones financieras.

Illa *et al.* (2022) [54] con el objetivo de generar mejores estrategias de comercio plantean predecir si el precio de la compra del índice Dow Jones Industrial Average (DJIA) será mayor al de su costo. Para realizar la predicción utilizan SVM y *Random Forest* [5]. Los autores logran generar mejores estrategias de comercio. Las predicciones que hicieron fueron de tendencias. En sus resultados muestran que *Random Forest* tuvo 81.6 % de precisión y SVM tuvo hasta 89.4 % de precisión.

Silva *et al.* (2020) [100], crean un sistema de comercio automatizado con un agente basado en una red LSTM. El sistema de comercio recibe los datos, realiza un preprocesamiento de ellos y genera los IT (Indicadores Técnicos); para generar la entrada de la red LSTM, la información entra a la red LSTM, la salida es enviada al sistema de comercio para generar una de las siguientes señales: compra, venta o de mantener el activo financiero y finalmente se realiza una evaluación de los rendimientos generados (Fig. 3.4). Este sistema aprende de los datos temporales, IT, administración de riesgo y considera los costos de transacción. Los autores probaron diferentes arquitecturas de redes LSTM, combinadas con diferentes estrategias de inversión. El diseño de la red LSTM que proponen finalmente tiene 4 celdas LSTM; para el entrenamiento usaron *dropout* entre celda y celda para evitar el sobre ajuste de la red. Los resultados de su investigación son satisfactorios, generando rendimientos de hasta 228.94 % a pesar de tener baja precisión en sus predicciones.

[5] SVM y *Random Forest* son modelos de ML.

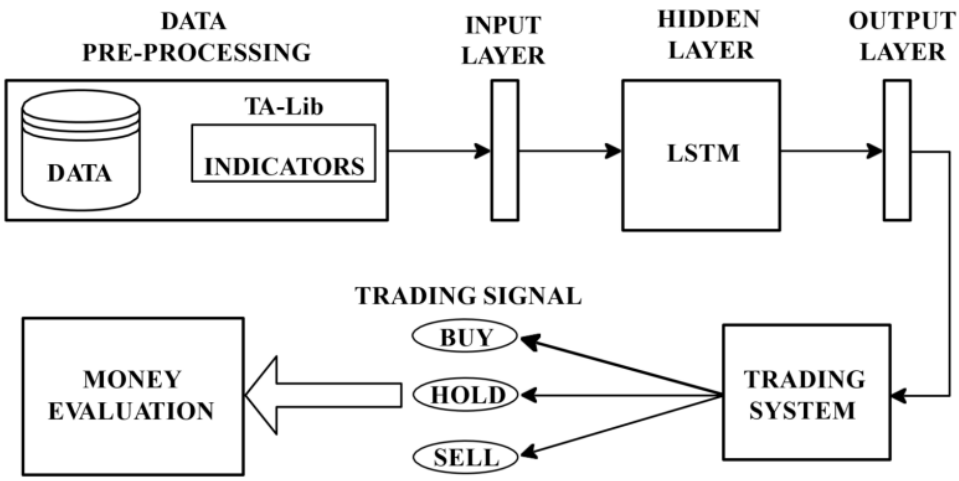


Figura 3.4: Sistema de comercio automático de Silva *et al.* (2020) [100]

Banik *et al.* (2022) [9] crean un sistema de soporte para los inversionistas en bolsa, este sistema utiliza una red LSTM, la cual genera un pronóstico para los próximos 30 días. Las pruebas del sistema generan los siguientes resultados, para el Error Cuadrático Medio, 4.13 %, en el Error Absoluto Medio, 3.24 % y la media del error absoluto porcentual 1.21 %. De acuerdo a los autores, su propuesta supera a: ARIMA, regresión lineal, SNaive, entre otros modelos. A pesar de que se superó al modelo ARIMA, otros trabajos han utilizado una red LSTM con ARIMA resultando en un rendimiento superior a una red LSTM [74].

Aunque en antecedentes (Sec. 1.1) se planteó el debate de si es necesario usar o no indicadores técnicos, en el estado del arte comienza a aparecer una preferencia por el uso de indicadores técnicos con DL, como lo hace Banik *et al.* (2022) [9] y Chen *et al.* (2021) [24]. Empero, el debate sigue, porque para definir si necesario usar o no IT se debe realizar una comparación para poder cerciorarse del impacto que tienen los IT en los pronósticos financieros.

La información de Colasanto *et al.* (2022) [30], Banik *et al.* (2022) [9], Silva *et al.* (2020) [100], Illa *et al.* (2022) [54] y Chen *et al.* (2021) [24], coincide con lo visto en antecedentes, que se identifica el uso de pronósticos financieros con sistemas de inversión y diseño de portafolios de inversión; lo cual es una tendencia.

Li *et al.* (2022) [74] realizan un pronóstico financiero proponiendo un modelo multiescala

basado en ML y un modelo econométrico. Hacen uso de una SVM de regresión para generar las características no lineales de la serie temporal del cierre de precio. Las acciones que utilizan en el trabajo son de las compañías: *Gree Electric Appliances Inc.*, *Inner Mongolia Yili Industrial Group Limited* y *East Money Information*. Li *et al.* (2022) [74] proponen utilizar la información que provee la SVM como entrada del ARIMA (SVM-ARIMA) para realizar el pronóstico. El modelo propuesto es evaluado con el Error Absoluto Medio, Error Cuadrático Medio y la media del error porcentual. Los autores concluyen que la estrategia multiescala juega un rol importante para mejorar el pronóstico y que su metodología permite un modelo práctico y preciso. Este trabajo deja entrever que no siempre los modelos DL son los mejores, y que otros modelos de ML híbridos pueden generar rendimientos satisfactorios. La propuesta de Li *et al.* (2022) [74] fue comparada con ANN, una MSV de regresión, entre otros modelos. Para realizar la experimentación utilizaron tres conjuntos de datos distintos, *Gree electric appliances, Inc.* de Zhuhai, *Inner Mongolia Yili Industrial Group Limited* de Share y *East Money Information Co., Ltd.* Los resultados del ECC se muestran en la Tabla 3.4.

Tabla 3.4: Resultados experimentales de Li *et al.* (2022) [74].

Acciones financieras	**Modelo**	**ECM**
Gree electric appliances, Inc.	ANN	1.8336
	SVM de regresión	1.555
	SVM-ARIMA	0.2413
Inner Mongolia Yili Industrial Group Limited	ANN	0.7774
	SVM de regresión	0.7251
	SVM-ARIMA	0.1695
East Money Information Co., Ltd	ANN	0.693
	SVM de regresión	0.6589
	SVM-ARIMA	0.1137

Bas *et al.* (2022) [10] proponen usar un modelo de RNN para pronosticar el índice DJIA y las acciones de Nike, entrenando a la RNN con PSO para evitar el *vanishing gradient* y el *exploding gradient*. El proceso de entrenamiento consiste en generar una población de parámetros, los cuales son evaluados con la pérdida de la SRNN con datos de entrenamiento; la idea es minimizar la pérdida en cada iteración del PSO. La propuesta (SRNN-PSO) es

comparada con LSTM y Pi-Sigma artificial neural network (PSGM), concluyendo que la RNN supera a los modelos *benchmark* mencionados. Este modelo deja de manifiesto que una RNN simple pude tener rendimientos superiores a una LSTM, a pesar de que la propuesta de la LSTM se diseñó para superar las dificultades que tienen las SRNNs simples en el aprendizaje; lo cual deja entrever que el problema de aprendizaje de las SRNNs simples no está en el diseño de las SRNNs, sino en el algoritmo de aprendizaje. Los resultados de sus experimentos están en la Tabla 3.5, la cual muestra las estadísticas de los rendimientos de 30 experimentos para el periodo de 2018. Las estadísticas se calculan sobre la raíz del error cuadrático medio.

Tabla 3.5: Resultados experimentales de Bas *et al.* (2022) [10].

Modelo	**Media**	**Desviación estándar**	**Valor mínimo**	**Valor máximo**
LSTM	454.50	9.48	425.71	467.42
PSGM	453.05	19.75	427.29	506.89
SRNN-PSO	438.38	0.91	437.15	441.94

Promedios de las raíces de los errores cuadráticos medios de los experimentos de Bas *et al.* (2022) [10].

Bazrkar & Hosseini *et al.* (2022) [11] plantean un modelo de ML basado en PSO y SVM (SVM-PSO) para pronosticar las acciones de Amazon, Oracle e IBM. La hibridación busca ajustar los parámetros de SVM con PSO. Su método lo contrastan con otros algoritmos. SVM, ANN y LSTM. Los resultados que obtienen superan a los modelos mencionados y en cada una de las acciones pronosticadas tienen precisiones arriba del 90 %. En la Tabla 3.6 se muestra la precisión de las predicciones de las acciones.

Tabla 3.6: Resultados experimentales de Bazrkar & Hosseini *et al.* (2022) [11].

Modelo	**Amazon**	**IBM**	**Oracle**
SVM-PSO	0.9871	0.9895	0.9770
SVM	0.7423	0.8764	0.9250
ANN	0.2047	0.9361	0.9387
LSTM	0.9385	0.9774	0.9701

Precisión de la predicción de las acciones de los experimentos realizados por Bazrkar & Hosseini *et al.* (2022).

Las propuestas de Colasanto *et al.* (2022) [30], Lin *et al.* (2021) [76], Li *et al.* (2022) [74], Bazrkar & Hosseini *et al.* (2022) [11], Kanwal *et al.* (2022) [59] y Bas *et al.* (2022) cae en

una de la tendencia de hibridar modelos para realizar pronósticos financieros; lo cual se identificó en antecedentes (Sec. 1.1).

Las investigaciones de Bas *et al.* (2022) y Bazrkar & Hosseini *et al.* (2022), presentan que el uso de metaheurísticas logran ajustar los parámetros de algoritmos de ML con resultados satisfactorios. Incluso supera a las redes LSTM, las cuales se identificaron en antecedentes (Sec. 1.1) como las preferidas por los investigadores, por su rendimiento y la facilidad con la que se aplican [98]. Por otro lado, los modelos híbridos de Kanwal *et al.* (2022) [59] y Lin *et al.* (2021) [76] consiste en combinar diferentes modelos de DL, los cuales se encuentran en otra de las tendencias identificadas en antecedentes (Sec. 1.1).

En pronósticos financieros, si bien las redes neuronales recurrentes tipo LSTM se usan con mayor frecuencia, otros modelos estadísticos como ARIMA se siguen aplicando pero hibridados con modelos de ML. Sin embargo, el empleo de otras redes como RNN híbridas con algoritmos metaheurísticos o SVM-PSO logran tener rendimientos que superan a las redes LSTM. En otros casos, redes LSTM hibridas con otros algoritmos de DL, superan a las redes LSTM simples. Se puede decir que en el estado del arte se encuentra que los modelos de pronóstico híbridos son los que mejor rendimiento generan.

En la siguiente sección se muestran trabajos donde se realiza HPO, los cuales también son considerados híbridos en la literatura de pronósticos financieros.

3.2. HPO en pronósticos financieros

En la presente sección se muestran investigaciones en los que se optimizan hiperparámetros de distintos modelos de ML para realizar pronósticos financieros o sistemas de comercio automatizado.

Tabla 3.7: Estado del arte en optimización de hiperparámetros en pronósticos financieros.

Autores	**Modelo**	**Hiperparámetros**	**Método de optimización**
Chung & Shin (2018) [29]	LSTM	Cl_s, H_s y W_s	GA
Bhandari *et al.* (2022) [13]	LSTM	Cl_s, H_s y E_s, Optimizador, α y W_s	GA
Kumar *et al.* (2022) [68]	Bi-LSTM-ARIMA	H_s, α, W_s, *dropout*, E_s, B_s y selección de características	ABC y DE
Deng *et al.* (2022) [33]	LSTM-MEMD	H_s, W_s, α, B_s y E_s	OAT
Kumar *et al.* (2021) [67]	LSTM	Cl_s, H_s, W_s, E_s, B_s y selección de características	PSO y FPA
Deng *et al.* (2022) [34]	LightGBM-NSGA-II-SW	α, tamaño máximo y mínimo de hojas	NSGA-II

En la investigación realizada se identifica que el modelo LSTM es ampliamente utilizado en pronósticos financieros para pronosticar el precio de las acciones o tendencias, como se describe en la Sec. 3.1; el modelo LSTM se usa ya sea solo o híbrido con otros modelos. HPO en pronósticos financieros no es la excepción; en la Tabla 3.7, se muestra que el modelo LSTM es ampliamente utilizado. Se identifica que la arquitectura de la red LSTM se optimiza en cada uno de los trabajos, al igual que el tamaño de la ventana. Únicamente en dos trabajos se considera el factor de aprendizaje, en tres se considera el tamaño de lote, en cuatro el tamaño de la ventana, en cuatro el número de épocas y en dos la selección de características[6]. La selección de características no se considera dentro del campo de HPO, pero si se considera dentro del campo del AutoML, como preprocesamiento de datos.

Cada uno de los autores en la Tabla 3.7, valoran que esos hiperparámetros son los más relevantes. Para fines de esta tesis, se consideró utilizar Cl_s, H_s, α, W_s, B_s y E_s; fijando como optimizador el Optimizador Adam, el cual ha demostrado resultados satisfactorios [62]; y no se consideró la selección de características al ser parte del preprocesamiento de datos en el AutoML.

En la Tabla 3.7 se muestra que la mayoría de los trabajos presentados en esta sección utilizan P-metaheurísticas (GA, DE, ABC, NSGA-II, PSO y FPA); la razón de que los trabajos utilice P-metaheurísticas puede encontrarse en que las P-metaheurísticas realizan una búsqueda informada para encontrar una solución paroximadamente óptima. El único trabajo que no utilizó P-metaheurísticas, es el de Deng *et al.* (2022) [33]; quienes utilizan OAT (*Orthogonal Array Tuning*) [115].

En los siguientes párrafos se describen las investigaciones de la Tabla 3.7. Se describe el proceso que llevaron a cabo y resultados experimentales de sus investigaciones.

Chung & Shin (2018) [29], realizan un pronóstico financiero utilizando una red LSTM optimizando los hiperparámetros (tamaño de la ventana y arquitectura) usando un GA. Su metodología de trabajo consiste en obtener los datos, dividirlos en datos de entrenamiento y datos de prueba, correr el algoritmo genético que provee de los hiperparámetros de la red

[6]La selección de características en pronósticos financieros considera la selección de indicadores técnicos y fundamentales.

LSTN, entrenar la red, evaluar el *fitness*, y repetir el proceso hasta cumplir un criterio de paro, para al final evaluar a la red LSTM obtenida con los datos de evaluación (Fig. 3.6). Ellos obtuvieron una red con tamaño de ventana igual a 10, 2 celdas LSTM con 15 y 17 estados ocultos respectivamente. Ellos concluyen GA encuentra los hiperparámetros, que permiten configurar una red LSTM que supera a los modelos *benchmark*. Los autores no precisan que modelo *benchmar* solo describen que es un modelo simple. En la Tabla 3.8 se muestran los errores cuadráticos medios que tuvieron los modelos en la experimentación.

Tabla 3.8: Resultados experimentales de Chung & Shin (2018) [29].

Modelo	**ECM**
Benchmark	209.45
GA–LSTM	181.99

Bhandari *et al.* (2022) [13] proponen un modelo LSTM para pronosticar el precio de cierre del índice S&P500. En su modelo integran nueve predictores distintos provenientes de datos fundamentales del mercado, datos macroeconómicos e indicadores técnicos. En su trabajo optimizan los hiperparámetros probando diferentes configuraciones por fuerza bruta, probando: número de celdas, estados ocultos, número de épocas, optimizador, factor de aprendizaje y tamaño de la ventana; ellos concluyendo que una red LSTM con una sola celda tiene un rendimiento superior que una red LSTM con múltiples capas, y que el óptimo del número de estados ocultos es 150; en la Tabla 3.9 se muestran las estadísticas de las raíces de los errores cuadráticos medios (RMSE, por sus siglas en inglés) de la red LSTM con diferentes estados ocultos; las estadísticas se obtienen de 30 experimentos realizados.

Tabla 3.9: Resultados experimentales de Bhandari *et al.* (2022) [13].

Métrica	**Estados ocultos** $\rightarrow$	10	30	50	100	150	200
RMSE	Valor mínimo	34.7359	43.8253	38.5586	37.2795	37.9416	62.1324
	Valor máximo	77.4861	72.1660	60.7464	49.4979	43.4026	88.8964
	Media	49.9564	57.0731	47.1908	42.7093	40.4574	73.1992
	Std	9.7758	8.0805	4.9642	2.9514	1.3957	5.3066

Kumar *et al.* (2022) [68] proponen un modelo para pronosticar las acciones de Nike y de los índices S&P 500, Dow Jones Industrial Average y NASDAQ GS. Ellos plantean analizar

los sentimientos de los inversionistas y usar dos modelos, ARIMA y Bi-LSTM, optimizando los hiperparámetros y seleccionado las características como los indicadores técnicos. Para realizar la optimización, los autores utilizan *Artificial Bee Colony* (ABC) y *differential evolution* (DE); proponiendo tres modelos diferentes combinando el orden y la manera en que se usa cada algoritmo p. ej. DE-ABC-Bi-LSTMARIMA. Los hiperaparámetros que optimizan son: tamaño de la ventana, número de celdas LSTM, probabilidad del *dropout*, número de épocas, tamaño del lote y factor de aprendizaje. Los autores indican que los resultados de sus propuestas superan a modelos *benchmark* (*stacked* LSTM y LSTM). En la Tabla 3.10 se muestra el rendimiento de los modelos comparados.

Tabla 3.10: Resultados experimentales de Kumar *et al.* (2022) [68].

Acción financiera	**Métrica**	**LSTM**	***Stacked* LSTM**	**Bi-LSTM**
DJIA	RMSE	185.95	179.921	175.048
NASDAQ GS		3.708	2.668	2.509
S&P500		19.563	18.896	18.534
NIKKI 225		162.771	162.03	160.869

Deng *et al.* (2022) [33] realizan un pronóstico de varios pasos en el tiempo hacia el futuro del índice S&P500 (entre otros activos financieros) desarrollando un modelo híbrido LSTM *many - to - many*, con *Multivariate Empirical Mode Decomposition* (MEMD). Los hiperparámetros de la red LSTM fueron optimizados con el modelo *Orthogonal Array Tuning* (OAT) [115]. MEMD, realiza una descomposición del precio del índice; despúes, la información obtenida de la descomposición es usada como entrada de la red LSTM para realizar el pronóstico. Los hiperparámetros que optimizaron fueron: número de épocas, número de estados ocultos, factor de aprendizaje, tamaño de la ventana, y tamaño del lote. Los resultados experimentales muestran que su propuesta supera a modelos *benchmark* como ARIMA y LSTM simple. En la Tabla 3.11 se muestra el rendimiento al pronosticar el precio que tendrá el índice S&P500 a los 10 días.

Deng *et al.* (2022) [34] proponen un modelo híbrido que pronostica un índice bursátil y realiza una simulación de comercio. El modelo propuesto usa *Light Gradient Boosting Machine* (LightGBM), *Non dominated Sorting Genetic Algorithm-II* (NSGA-II) y *Sliding*

Tabla 3.11: Resultados experimentales de Deng *et al.* (2022) [33].

Modelo	RMSE
ARIMA	186.5447
LSTM	79.7980
MEMD-LSTM	42.0254

RMSE que Deng *et al.* (2022) [33] obtuvieron al pronosticar el precio que tendrá índice S&P 500 al los 10 días.

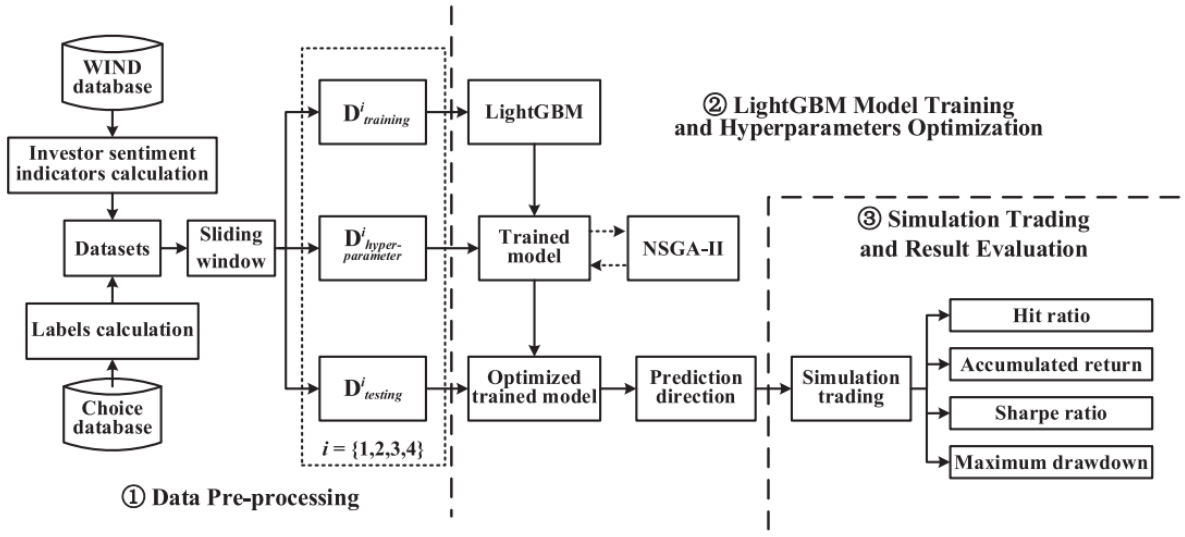


Figura 3.5: Metodología propuesta por Deng *et al.* (2022) [34]

Window (SW), nombrándolo LightGBM-NSGA-II-SW. Este modelo hace un análisis de sentimientos para pronosticar la tendencia del índice Shanghai Stock Exchange (SSE); la tendencia obtenida se usa para realizar la simulación del comercio con LightGBM. Mientras que NSGA-II se utiliza para realizar la optimización de hiperparámetros. La optimización de hiperparámetros tieme múltiples objetivos: maximizar la tasa de aciertos y el retorno acumulado. Al final de los experimentos se obtuvo 60.34 % de tasa de aciertos, 28.43 % de retorno acumulado y 8.35 % de reducción máxima, y el valor de 3.24 del índice Sharpe. La metodología descrita se puede ver en la Fig. 3.5. Los autores concluyen que su propuesta puede utilizarse para monitorear los mercados y generar políticas de regulación del mercado de valores Chino. Esta propuesta puede ser significativa para el diseño de políticas como en otras investigaciones que comienzan a utilizar modelos de IC para el diseño de políticas públicas [58, 63, 116].

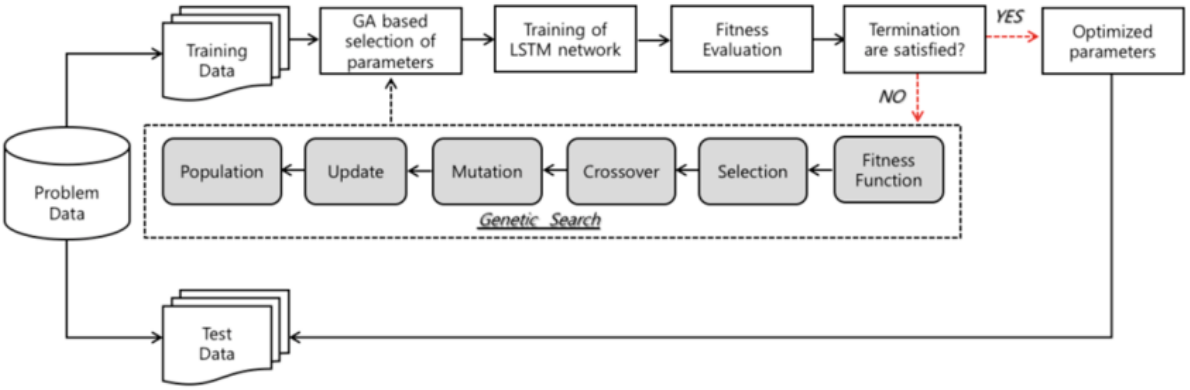


Figura 3.6: Metodología de optimización de hiperparámetros de Chung & Shin (2018) [29].

Kumar *et al.* (2021) [67] utilizan una red LSTM para realizar pronósticos financieros de tendencias, optimizando sus hiperparámetros utilizando PSO y FPA. Los hiperparámetros que optimizaron fueron: el tamaño de la ventana, tamaño del lote, número de épocas, número de celdas y número de estados ocultos, incluyendo selección de indicadores técnicos; los experimentos que realizaron se hicieron con seis activos diferentes, obteniendo diferentes diseños de redes LSTM. Entre las redes que obtuvieron el mejor rendimiento se encuentra la siguiente: tamaño de la ventana de 12, 2 celdas LSTM, 25 estados ocultos, tamaño del lote de 12, y 155 épocas. Su propuesta supera a redes optimizadas sin metaheurísticas; y que PSO converge más rápido que FPA, pero FPA tiene mejores resultados, con una tasa de error de hasta 10 % menos, con precisiones, recuerdos y F1-*score* de hasta 80 %, mientras que en las mismás metricas PSO se encuentra por encima del 60 %. En la presente tesis se decide basarse en este trabajo pero optimizando un hiperparámetro más, el factor de aprendizaje, sin considera la selección de características, porque eso entra dentro del preprocesamiento del AutoML.

En pronósticos financieros la HPO, muestra tener resultados que superan a los modelos que no utilizan HPO; y la red LSTM son de las más usadas en la literatura, La hibridación de redes LSTM y realizar HPO genera resultados que superan a los no híbridos y que no realiza HPO. Entre los algoritmos más utilizados para realizar HPO son las P-metaheurísticas.

En antecedentes (Sec. 1.1), Cavalcate *et al.* (2016) [18] sugieren utilizar los pronósticos

financieros en conjunto con los sistemas de comercio automatizado y Sezer *et al.* (2020) [98] menciona que en el futuro se realizaría el uso de pronósticos financieros con sistemas de comercio automatizado; en la presente sección se identifica que ya se está haciendo uso de los pronósticos financieros en comercio automatizado y se apoyan diseñando los sistemas con HPO.

3.3. HPO en otros campos

En esta sección se exponen diferentes trabajos de HPO en campos distintos a pronósticos financieros, desde investigaciones que buscan aumentar los beneficios por el tiempo de uso en páginas web, hasta investigaciones en simulación de crecimiento urbano. Al final se muestran conclusiones de los trabajos.

Zhu *et al.* (2022) [118], desarrollan una red LSTM con un mecanismo de atención para realizar un pronóstico de radiación solar, optimizando la selección de características e hiperparámetros usando un GA, el modelo fue llamado AGA-LSTM. Los hiperparámetros que seleccionaron fue número de celdas LSTM, número de estados ocultos, tamaño de la ventana y número de retrasos. Su propuesta superó GA-LSTM y LSTM; los resultados de nRMSE (3.2) se muestran en la Tabla 3.12 utilizando intervalos de 5 minutos en la predicción.

$$nRMSE = \frac{1}{\overline{y}}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \cdot 100\,\% \tag{3.2}$$

donde N es el número de ejemplos, y es el valor real, $\hat{y}$ es la predicción y $\overline{y}$ es la media de todos los valores medidos.

Tabla 3.12: Resultados experimentales de Zhu *et al.* (2022) [118].

Modelo	**nRMSE**
LSTM	14.34 %
GA-LSTM	8.83 %
AGA-LSTM	6.35 %

Thakur *et al.* (2022) [103], realizaron un pronóstico de series temporales univariantes,

optimizando los hiperparámetros (tamaño de la ventana y estados ocultos) de una red LSTM con un GA (GA-OLSTM). La metodología que ellos usan es la misma de Chung & Shin (2018) [29] (Fig. 3.6). Los hiperparámetros óptimos que encontraron fueron $W_s = 49$ con $H_s = 9$, $W_s = 12$ con $H_s = 8$, $W_s = 40$ con $H_s = 8$ y $W_s = 36$ con $H_s = 2$. El ECM de una LSTM sin optimizar hiperparámetros con GA fue de 0.006, mientras que el ECM del modelo GA-OLSTM fue de 0.004. Los autores mencionan que GA-OLSTM es ligeramente superior a LSTM.

Los trabajos de Zhu *et al.* (2022) [118] y Thakur *et al.* (2022) [103] muestran que el uso de GA para optimizar hiperparámetros de una red LSTM tiene resultados satisfactorios no solo en pronósticos financieros, sino también en otro campo como el de predicción de la radiación solar, porque tanto la radiación solar como los precios de las acciones son series de tiempo. Incluso Thakur *et al.* (2022) [103] utiliza la metodología propuesta por Chung & Shin (2018) [29], la cual se utiliza para realizar pronósticos financieros.

Souissi & Ghorbel (2022) [101], realizaron un pronóstico de Click-through rate (CTR), un indicador que se utiliza en publicidad digital para aumentar las ganancias y mejorar la experiencia de usuario. Para llevar a cabo el pronóstico utilizan una red LSTM, optimizando sus hiperparámetros y selección de características (realizan todo el proceso de AutoML), con un GA, PSO y *Upper Confidence Bound* (UCB); teniendo tres propuestas LSTM-GA, LSTM-PSO, y UCB-LSTM-GA. Los hiperparámetros que optimizaron fueron tamaño de la ventana y el número de estados ocultos. Los rendimientos sobre la inversión de sus propuestas fueron de: 87 % para LSTM-GA, 89 % para LSTM-PSO y 92 % para UCB-LSTM-GA.

Kilink & Haznedar (2022) [61], realizan un pronóstico de la corriente de ríos usando una red LSTM optimizando hiperparámetros con un GA (GA-LSTM); ellos optimizan: tamaño de la ventana y número de estados ocultos. Los resultados superan a LSTM y regresión lineal; en la Tabla 3.13 se muestran los RMSE para el conjunto de datos (Kilayak) creado por los autores.

Lopez-Farias *et al.* (2021) [77] realizan una simulación de crecimiento urbano con el algoritmo: *The FUTure Urban-Regional Environment Simulation* (FUTURES); este algoritmo

Tabla 3.13: Resultados experimentales de Kilink & Haznedar (2022) [61].

Modelo	**RMSE**
Regresión lineal	1.3755
LSTM	1.2668
GA-LSTM	0.9302

requiere ajuste de parámetros para realizar la simulación, por lo general el proceso se lleva mediante *Grid Search* y se ha llegado a hacer con *Differential Evolution* (DE) o probando diferentes valores hasta encontrar una configuración que provea de resultados satisfactorios. En su trabajo proponen una metaheurística EDALNFE para ajustar los parámetros del algoritmo FUTURES. Los autores contrastaron su algoritmo con *Grid Search* y DE. Para comparar los algoritmos realizan 5 experimentos de cada algoritmo; de las cuales DF realizó 512 llamadas a función en cada experimento y LNFE realizó aproximadamente en promedio 191 llamadas a función; y el mejor valores *fitness* que encontró DF fue de 18.64, en el caso de LNFE el mejor valor *fitness* fue de 28.95; en el caso de *Grid Search* el número de llamadas a función fue de 512 y el mejor valor *fitness* encontrado fue de 22.64; por lo que los autores llegan a la conclusión de LNFE es mejor que DE y *Grid Search* al optimizar los parámetros de FUTURES.

Chen *et al.* (2019) [23] proponen un modelo de DRL basado en LSTM (RPR-BP-Agent) para optimizar hiperparámetros de *Random Forest* y XGBoost para clasificar números dibujados a mano. Sus resultados muestran que la propuesta supera a *Random Search*, Optimización Gausiana Bayesiana (OGB) y *Treestructured Parzen Estimato* (TPE), en términos de precisión, eficiencia y estabilidad. De acuerdo a los autores, estos modelos superan al estado del arte; los resultados de la precisión se presentan en la Tabla 3.14. Ellos sugieren probar su propuesta en la optimización de hiperparámetros de redes neuronales artificiales.

Tabla 3.14: Resultados experimentales de Chen *et al.* (2019) [23] [61].

Modelo	**Precisión**
OGB	0.9394
RS	0.9331
RPR-BP-Agent	0.9654

Li *et al.* [75] proponen un modelo LSTM para optimización de hiperparámetros, el cual llaman Hyperparameter Exploration LSTM-Predictor (HELP). Ellos realizan el entrenamiento de HELP con métodos de estrategias evolutivas. El modelo HELP recibe una serie de hiperparámetros como entrada y pronostica su valor *fitness*. Las regiones con mayor *fitness* tienen mayor probabilidad de ser elegidas en la siguiente iteración. Ellos realizan la HPO para diferentes tipos de redes neuronales, incluyendo clasificación y generación de datos. En clasificación utilizaron CNN, para clasificar el conjunto de datos *MNIST*[7]; la arquitectura que utilizaron fue fija, ajustando el factor de aprendizaje, el porcentaje de *dropout* y la tasa de la función LeakyReLU (Fig. 2.2(e)). Para generación de datos usan *Generative Adversarial Net* (GAN), utilizando el conjunto de datos *CIFAR-10*[8]. La arquitectura de la red GAN fue fija, ajustando únicamente el factor de aprendizaje de manera independiente para las dos partes de la red GAN. De acuerdo a los autores, su algoritmo reduce el tiempo de búsqueda de los mejores hiperparámetros. Para entrenar el modelo utilizar *backpropagation* con estrategias evolutivas [56] y PSO. En el caso de la optimización de hiperparámetros para clasificación utilizando CNN, se requirió realizar 215 iteraciones para obtener 95 % de precisión; en el caso de generación de imágenes con GAN, para tener un *Inception Score*[9] de 7.0 se requirió 198, 600 iteraciones. Al final sus resultados superan al estado del arte.

El algoritmo más utilizado en la literatura para realizar HPO es el GA, sin embargo; este algoritmo tiene ciertas desventajas, porque requiere sintonizar múltiples parámetros para realizar la optimización y necesita una representación específica para aplicar operadores genéticos sobre las soluciones candidatas. Otras alternativas pueden ser ABC, PSO, DE, NSHA-II y FPA, las cuales han tenido resultados satisfactorios [34, 67, 68], la ventaja de usar este tipo de algoritmos es que sus representaciones son más sencillas que las que se usan en los GA. Otros algoritmos parecen prometedores, p. ej. HELP [75] sin embargo, este algoritmo solo se ha probado con dos variables.

[7] *MNIST* es una base de datos de números escritos a mano. Esta base de datos se puede obtener de http://yann.lecun.com/exdb/mnist/.

[8] CIFAR-10 es una base de datos de imágenes de distintas clases como: avión, perro, automovil, entre otras. Esta base de datos se puede obtener de https://www.cs.toronto.edu/~kriz/cifar.html.

[9] *Inception Score* es un algoritmo utilizado para medir la calidad de una imagen.

En esta sección también se identifica que se está experimentando con otro tipo de algoritmos para realizar HPO, entre los algoritmos se encuentra *Estimation Of Distribution Algorithm* (EDA), *Deep Reinforcement Learning* (DRL) y DL. En el caso de DL encontramos HELP, un modelo de agentes basado en redes LSTM, el cual optimiza hiperaprámetros de redes tipo DL, no obstante el número de hiperparámetros que optimizan es reducido, lo que deja la investigación abierta para probar con más hiperparámetros. En DRL el modelo propuesto basado en LSTM solo es propuesto en modelos de ML como *Random Search*, siendo modelos con menor cantidad de hiperparámetros, empero de acuerdo a los autores se sugiere optimizar hiperparámetros de modelos de DL. En el caso de la EDA, se utiliza una nueva propuesta, LNFE, la cual de acuerdo a Scopus [10] no ha sido citada, indicando que es un modelo no probado en otros problemas de optimización; sin embargo, tiene la ventaja de disminuir el número de llamadas a función. LNFE puede disminuir el tiempo de ejecución y el costo computacional de HPO en ANN y DL al utilizar menos llamadas; por esta razón y que es una P-metaheurística, se decide utilizar LNFE en la presente tesis.

De estos trabajos se identifica que de redes LSTM son ampliamente utilizados para pronosticar series de tiempo porque demuestran tener rendimientos superiores a modelos *benchmark* y que incluso usar algoritmos P-metaheurísticos para la optimización de hiperparámetros provee modelos que superan a modelos LSTM sin optimización de hiperparámetros con metaheurísticas.

Conclusiones

En pronósticos financieros, las redes LSTM son algoritmos utilizados ampliamente en pronósticos financieros y que LSTM se híbrida (incluyendo la HPO) con otro tipo de algoritmos. En el estado del arte se identifica que LSTM se utiliza como parte de los sistemas de comercio automatizado.

Los algoritmos recientemente utilizados para HPO son metaheurísticas (PSO, FPA y

[10]https://www.scopus.com/

GA) para optimizar los hiperparámetros de redes LSTM. También se han explorado otro tipo de modelos para HPO como OAT. Mientras que la optimización de hiperparámetros en otros campos, se ha hecho con EDA, DL y DRL. Cada uno de estos algoritmos muestran ciertas ventajas de acuerdo a la investigación, no obstante, destacando LNFE en las EDA y HELP en DL, porque en la búsqueda de hiperparámetros, los algoritmos asignan probabilidades altas a regiones de soluciones prometedoras y convergen en una menor cantidad de tiempo.

Al encontrar en la literatura que PSO y FPA tienen rendimientos esperados en HPO de redes LSTM en pronósticos [67] y que LNFE es un algoritmo que reduce el número de llamadas a función con una convergencia en menor cantidad de tiempo [77] se decide utilizar estos algoritmos para la HPO de una red LSTM para realizar pronósticos financieros.

Capítulo 4

Metodología

En el presente capítulo se expone la metodología utilizada para realizar el pronóstico del precio de cierre de las acciones de Google y Nike. Para realizar el pronóstico se utilizó una red neurona LSTM y para ajustar los hiperparámetros de la red LSTM se utilizaron metaheurísticas. En este capítulo también se describe la metodología para comparar las metaheurísticas.

4.1. Metodología de optimización de hiperparámetros

Para realizar la optimización de hiperparámetros, primero se define la función *fitness*, la cual consiste en el entrenamiento de una red LSTM y retornar su pérdida. La función *fitness* recibe un vector de una solución candidata (vector solución), el cual contiene los hiperparámetros de la red LSTM, la función *fitness* toma los hiperparámetros para diseñar la red LSTM, después entrena la red y finalmente retorna el error (pérdida) como valor *fitness*.

La idea de la optimización de hiperaprámetros consiste en encontrar una configuración de hiperparámetros que minimicen la pérdida de la red LSTM cuando ésta busca realizar el pronóstico del precio de cierre de las acciones financieras ya sea de las acciones financieras de Google o de las acciones financieras de Nike.

Para realizar el pronóstico del precio de cierre de las acciones financieras de Google y

de las acciones de Nike primero se recabó la base de datos de las series de tiempo de las acciones financieras; las cuales se obtuvieron de la página web Yahoo Finance[1]. El periodo obtenido para las acciones de Google fue del 24 de Agosto de 2004 al 21 de Enero de 2022, en el caso de las acciones de Nike, fue del 2 de Diciembre de 1980 al 24 de Enero de 2022.

Para realizar el experimento, los datos de la serie de tiempo fueron normalizados usando el escalamiento $min-max$ considerando los valores del Espacio de Búsqueda (EB) para cada variable. La normalización de los datos se consideró de acuerdo a Kumar *et al.* (2019) [69], Senapati *et al.* (2018) [97] y Chopra *et al.* (2019) [28], entre otros.

Para que la red LSTM realice el pronóstico, ésta recibe una secuencia del precio de cierre de las acciones como se muestra a continuación:

$$WS = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{W_s}] \tag{4.1}$$

donde WS es la secuencia de entrada llamada ventana, $\mathbf{x}_i$ es el i-ésimo precio de cierre de la acción financiera y W_s es el tamaño de la ventana.

La red LSTM realiza el pronóstico del vector $\mathbf{x}_{W_s+1}$. La red LSTM recibirá WS y procesará el precio de cierre a precio de cierre, como se muestra a continuación:

$$s_1 = R_{LSTM}(s_0, \mathbf{x}_1)$$

$$s_2 = R_{LSTM}(s_1, \mathbf{x}_2)$$

hasta obtener:

$$s_{W_s} = R_{LSTM}(s_{W_s-1}, \mathbf{x}_{W_s})$$

donde $s_0 = 0$, $s_{W_s} = [C_{W_s}, H_{W_s}]$, H_{W_s} es el estado oculto al final de procesar toda la secuencia, este valor se pasa por una capa de un perceptrón utilizando una función lineal para generar la predicción del valor $\hat{\mathbf{x}}_{W_s+1}$.

Con el propósito de obtener la pérdida de entrenamiento se utilizó el ECM, definido a

[1]https://finance.yahoo.com/.

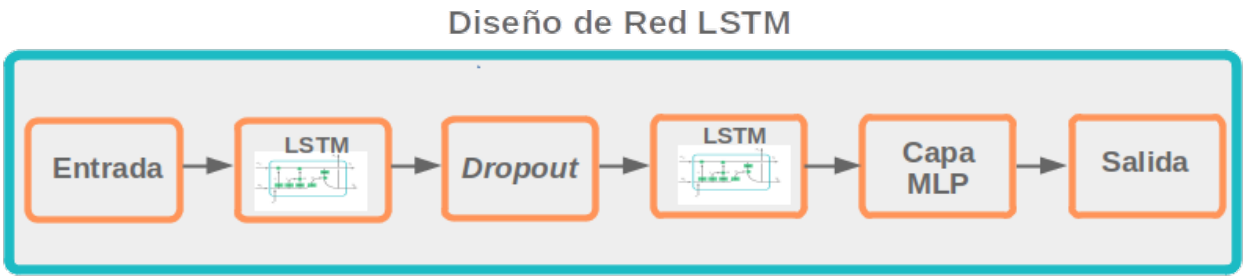


Figura 4.1: Diseño de red LSTM usando *dropout*. (Elaboración propia)

continuación:

$$\text{ECM} = \frac{1}{B_s}\sum_{i=1}^{B_s}(\hat{\mathbf{x}}_i - \mathbf{x}_i) \tag{4.2}$$

donde B_s es el tamaño del lote.

Para entrenar la red LSTM se seleccionó el 65 % de los datos de la serie de tiempo del precio de cierre de las acciones y 35 % para evaluación. El optimizador que se utilizó es el Optimizador Adam[2] para entrenar la red LSTM; el factor de aprendizaje se fijó como se utiliza generalmente en la búsqueda de hiperparámetros.

La función *fitness* consiste en diseñar y entrenar una red LSTM y devolver el valor de la pérdida de entrenamiento, la cual se evalúa usando el ECM de entrenamiento ($F_{fitness}$ ≡ECM). Ésta función *fitness* es la que se busca minimizar en el proceso de optimización de hiperparámetros utilizando metaheurísticas.

En el diseño de la red se utiliza el mismo número de estados ocultos para cada celda LSTM, en el caso de tener una red LSTM con una configuración con más de una celda LSTM. En las configuraciones de más de una celda LSTM se utilizó el regularizador *dropout*[3] entre celda y celda LSTM; la probabilidad del *dropout* fue de 50 % (Fig. 4.1). La aplicación del *dropout* se realizó de acuerdo a Silva *et al.* (2020) [100].

Con el propósito de reducir el costo computacional al realizar cada llamada a función, se decidió añadir una condición de paro de entrenamiento, la cual consiste en detener el entrenamiento en caso de que la pérdida no mejora después de 5 épocas; también se programó

[2]Para más detalles del Optimizador Adam revisar [62]

[3]El regularizador *dropout* evita el sobre ajuste de una red neuronal, simulando el apagado de determinado porcentaje de neuronas en una capa. Este proceso se aplica en cada actualización de pesos de la red neuronal artificial en el entrenamiento.

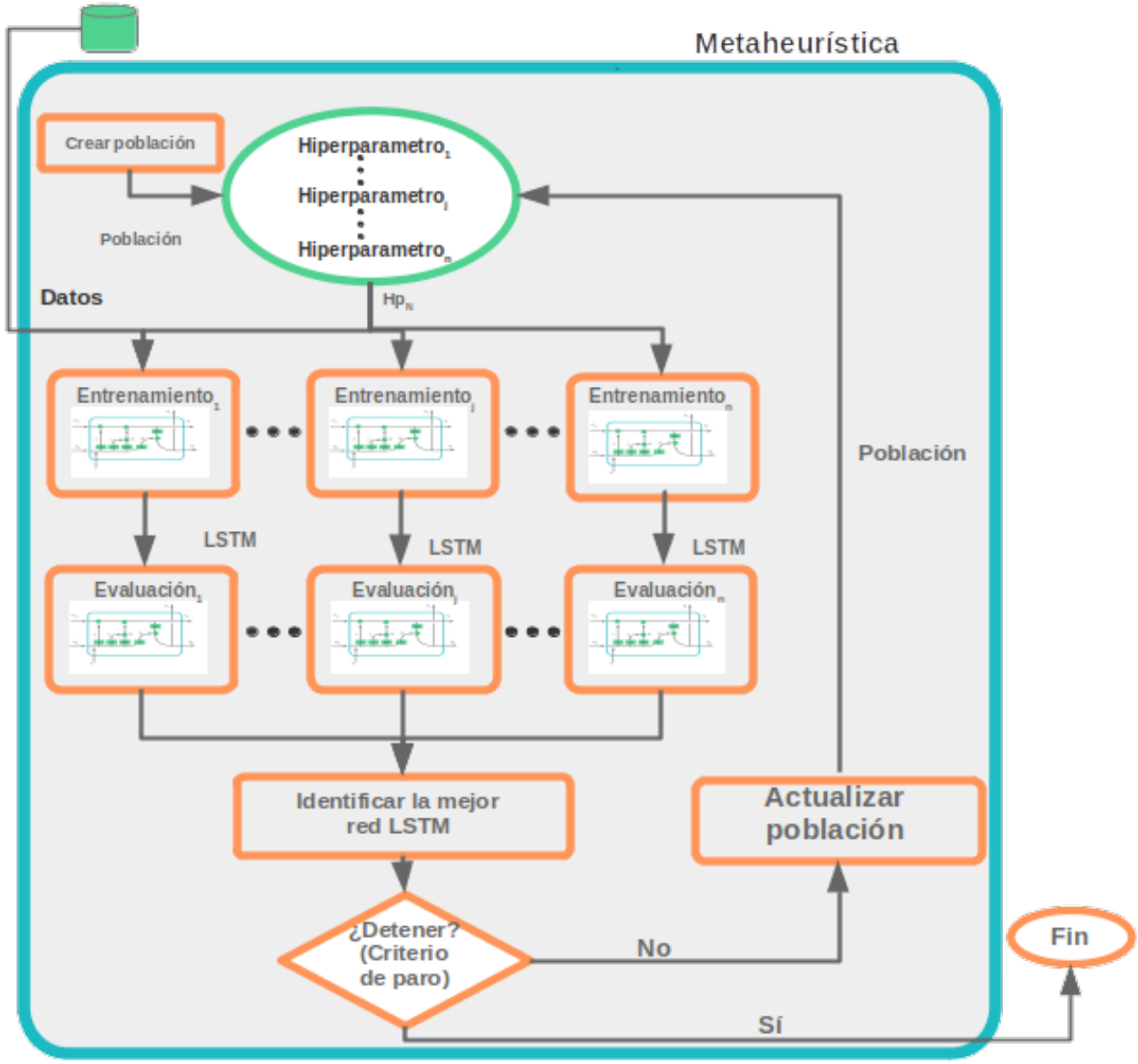


Figura 4.2: Metodología de optimización de hiperparámetros. (Elaboración propia)

el algoritmo de tal manera que en cada llamada a función se guardara el valor *fitness* y su vector solución, para que al encontrar la misma solución en la optimización, no se realice un reentrenamiento de la red LSTM y solo se retorne el valor *fitness*.

Cada metaheurística necesita definir ciertos parámetros, incluyendo el EB; a continuación se mencionan los valores fijados para cada metaheurística. Los parámetros del PSO se ajustaron de acuerdo a Gendreau & Potvin (2019) [40], mientras que los parámetros de FPA se ajustaron de acuerdo a Yang (2020) [112]. En ambos casos se utilizan 25 soluciones, como se hace en la investigación de Kumar *et al.* (2021) [67]. Los parámetros se muestran en la

Tabla 4.1: Hiperparámetros y espacio de búsqueda.

Hiperparámetro	**Símbolo**	**Espacio de búsqueda**	**Límites**
Celda LSTM	Cl_s	$\{ 1, 2, 3, 4, 5, 6, 7, 8\}$	$[1, 8]$
Estado oculto	H_s	$\{ 1, 20, 50, 80, 96, 110, 150\}$	$[1, 7]$
Tamaño de la ventana	W_s	$\{ 20, 40, 60, 80, 90, 100\}$	$[1, 6]$
Factor de aprendizaje	α	$\{ 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$	$[1, 5]$
Tamaño de lote	B_s	$\{ 8, 16, 32, 64\}$	$[1, 4]$
Número de épocas	E_s	$\{ 12, 25, 50, 100, 150\}$	$[1, 5]$

Tabla 4.2. En el caso de LNFE, el diseño del algoritmo únicamente permite fijar el número de soluciones en 26 y no en 25.

Los hiperparámetros optimizados fueron: Celdas LSTM (Cl_s), número de estados ocultos (H_s), tamaño de la ventana (W_s), factor de aprendizaje (α), tamaño del lote (B_s) y número de épocas (E_s). En la Tabla 4.1 se muestran los hiperparámetros con su símbolo, espacio de búsqueda y los límites de búsqueda de cada metaheurística con los que se realizó la optimización. El EB del tamaño de la ventana se fijó de acuerdo a Sezer *et al.* (2020) [98]. El EB del número de estados ocultos se seleccionó de acuerdo a experimentos previos (ver Apéndice D) para reducir el EB. El número de celdas LSTM se seleccionó de acuerdo a Kumar et al. (2021) [67].

Las poblaciones iniciales de PSO y FPA se generaron con la misma semilla generadora de números aleatorios; en el caso de LNFE se tomó la población inicial que se generó en los algoritmos anteriores y se mapeó en el espacio entre $[0, 1]$, utilizando el escalamiento $min - max$. Para poder comparar los algoritmos bajo condiciones similares, las metaheurísticas se ajustaron para realizar aproximadamente 425 llamadas a función.

PSO y FPA son algoritmos que realizan su búsqueda en el espacio de los números continuos. Para evaluar la función *fitness* se consideró el EB como un conjunto finito ordenado. Cada vector solución se mapea al espacio de los Números Discretos (dis) vía (4.3), los valores obtenidos (g) de cada vector se mapean al EB vía (4.4), para obtener los Valores Válidos (VV) (Fig. 4.3); estos valores forman el vector solución de hiperparámetros que se usan para diseñar y entrenar la red LSTM y así obtener el valor *fitness* (ECM). Éste proceso se diseñó siguiendo las ideas de Coelho *et al.* (2015) [39], quienes proponen una metodología para

Tabla 4.2: Parámetros empleados en metaheurísticas.

Parámetros	**Valores**
PSO	
Coeficiente cognitivo, c_1	2.05
Coeficiente social, c_2	2.05
Inercia, w	0.8
Número de soluciones	25
FPA	
Sesgo de polinización, ρ	0.8
Factor escalar del vuelo de Lévy, γ	0.1
Número de soluciones	25
LNFE	
Número de soluciones	26

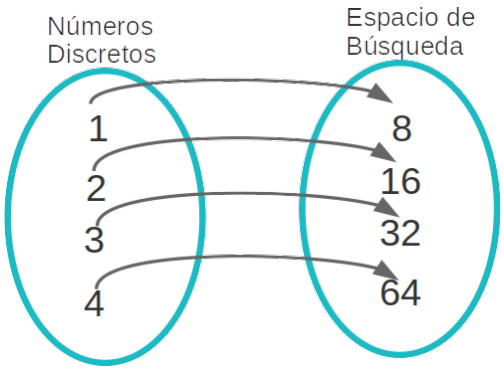


Figura 4.3: Mapeo del Espacio de Números Discretos al Espacio de Búsqueda del tamaño de lote vía (4.4). (Elaboración propia)

optimización de problemas con variables mixtas.

$$dis(a.b) = \begin{cases} \lceil a.b \rceil, \text{ si } b \geq 5 \\ \lfloor a.b \rfloor, \text{ en otros casos} \end{cases} \tag{4.3}$$

$$VV : dis \longrightarrow \text{EB} \tag{4.4}$$

En el caso de LNFE que trabaja en el espacio de valores entre $[0, 1]$, para obtener los VV se hace el proceso inverso de normalización que se realizó para obtener la primera población de LNFE usando el escalamiento $min - max$, esto retorna valores en el espacio de los números continuos, valores similares a los de PSO y FPA. Éstos valores se mapearon vía (4.3) y (4.4) para obtener el vector solución con los VV para diseñar, entrenar la red

LSTM y obtener el valor *fitness* (ECM).

Con objeto de realizar una comparación de las soluciones obtenidas por cada metaheurística se tomó y entrenó la mejor red LSTM encontrada por Moghar & Hamiche (2020) [81]. Estos autores ajustaron los hiperparámetros de la red LSTM de manera manual; ellos probaron 4 configuraciones diferentes, cambiando el número de épocas de: 12, 25, 50 y 100. En la presente tesis, la red LSTM se entrenó utilizando la base de datos de Google y Nike, dividiendo la base de datos en conjunto de entrenamiento y conjunto de prueba en proporción $65 - 35\,\%$, respectivamente. Los hiperparámetros de la red LSTM optimizada sin metaheurísticas son: 4 Cl_s, 96 estados ocultos, tamaño de la ventana de 50, factor de aprendizaje de 10^{-2}, tamaño del lote de 16 y 100 épocas. Estos hiperparámetros se usaron tanto para los experimentos realizados con las acciones de Google como las acciones de Nike. Se hicieron 33 experimentos de la réplica para obtener evidencia estadística que pueda ser compara con cada una de las soluciones obtenidas por las metaheurísticas.

Se inició el proceso de optimización (Fig. 4.2), como se muestra en el Algoritmo 7, una vez que: se ajustaron los parámetros y se fijaron los EB de cada metaheurística, se programó cada metaheurística para que realice aproximadamente 425 llamadas a función, se normalizaron los datos de la serie de tiempo de los precios de cierre y las poblaciones iniciales de las metaheuristicas y las redes LSTM se generaran con la misma semilla de los generadores de números aleatorios.

Algoritmo 7 Optimización de hiperparámetros

```
1: Fijar parámetros de metaheurística, Espacios de Búsqueda y criterio/s de paro.
2: Crear población inicial de las soluciones/hiperparámetros.
3: while Criterio de paro no se cumpla, do
4:     for each solución do
5:         Evaluar solución con la función fitness.
6:     end for
7:     Seleccionar la solución que diseña la red LSTM con el mejor valor fitness.
8: end while
9: return Retornar el vector solución que contiene los mejores hiperparámetros.
```

Finalizado el proceso de optimización de hiperparámetros, las soluciones obtenidas por cada metaheurística se utilizaron para diseñar la red LSTM y entrenar (experimento) 33

veces cada red neuronal, con el objetivo de tener evidencia estadística y comparar las soluciones; con los entrenamientos se obtuvo la media de la pérdida y la desviación estándar de cada época y la medida y la desviación estándar de la evaluación[4] de las redes LSTM. Con cada media y desviación estándar de la pérdida de cada época se realizó una gráfica donde se muestran las curvas de entrenamiento de cada solución, incluyendo a la red LSTM propuesta por Moghar & Hamiche (2020) [81]; con el objetivo de hacer una comparación visual de cada curva de entrenamiento. Con la media y desviación estándar de la evaluación, se hizo una gráfica donde se muestran los diagramas de caja y bigote para cada una de las soluciones con el objetivo de realizar una comparación del rendimiento de las soluciones.

En la Tabla 5.2 se muestran las soluciones y la evidencia estadística. Se muestra el mejor valor *fitness*, así como los valores estadísticos que indican el mayor rendimiento de la solución, marcados en verde; la solución media y rendimientos medios se muestran en amarillo; mientras que el peor valor *fitness*, así como los valores estadísticos que indican el menor rendimiento de la solución, se marcaron en rojo. Este proceso se aplicó para el caso de las acciones financieras de Google y luego para el caso de las acciones financieras de Nike.

4.2. Metodología de comparación de metaheurísticas

En esta sección se expone la metodología propuesta para realizar la comparación del rendimiento de la metaheurísticas. Los criterios para evaluar las metaheurísticas fueron: el valor *fitness* y la convergencia.

Para evaluar la convergencia de las metaheurísticas se realizó una comparación de la dispersión en la población de la primera iteración (primera población) contra la dispersión de la población en la última iteración (última población).

Antes de realizar la comparación de las dispersiones se normalizaron con el escalamiento $min - max$ considerando los límites del EB para cada una de las variables. Una vez normalizadas las soluciones se calculan las diferencias porcentuales entre la dispersión de la última

[4]En la presente tesis se refiere a prueba como evaluación, con el objetivo de evitar confusiones en el lector.

población contra la primera población de cada una de las variables (4.5).

$$D(P_v^{first}, P_v^{last}) = \frac{Std(P_v^{first}) - Std(P_v^{last})}{Std(P_v^{first})} \cdot 100 \tag{4.5}$$

donde $D(\cdot)$ es la diferencia porcentual de la variable v, P_v^{first} es la primera población de la variable v, P_v^{last} es la última población de la variable v y $Std(\cdot)$ es la desviación estándar.

$D(\cdot)$ es un valor que si es 100 % la variable que se está optimizando con la metaheurística ha convergido, indicando que todos los valores de la población de la variable son el mismo. Mientras $D(\cdot)$ se aleje más del 100 %, los valores de la población de la variable van siendo distintos, en el caso de que $D(\cdot) = 0\,\%$, indicará que todos los valores de la población de la variable no han convergido.

Para tener una idea de la convergencia de todas las variables de la solución se realizó el promedio de las diferencias porcentuales (4.6).

$$CPT(P^{first}, P^{last}) = \frac{1}{V} \cdot \sum_{v=1}^{V} D(P_v^{first}, P_v^{last}) \tag{4.6}$$

donde CPT es la convergencia promedio total de las diferencias porcentuales de las variables de la solución y V es el número de variables.

La información obtenida por las diferencias porcentuales de cada población inicial contra las poblaciones finales se muestran en la Tabla 5.3, marcando en morado aquellas diferencias mayores al 66.66 %; en verde las diferencias que se encuentran entre 66.66 % y 33.33 % y en amarillo las diferencias menores a 33.33 %; indicando alta convergencia, convergencia promedio y baja convergencia, respectivamente. Esto se hace con la idea de comparar el grado de convergencia dividiendo los porcentajes en 3 categorías.

Posteriormente, se realizaron gráficas de convergencia. Para ello, primero se normalizaron cada una de las variables de cada población obtenida por iteración utilizando el escalamiento $min - max$, considerando el EB. Una vez normalizados los datos, se hicieron gráficas de convergencia (Figs. 5.5 a 5.10), obteniendo 6 gráficas, una por cada variable optimizada tanto para el caso de las acciones de Google como de las acciones de Nike. Estas

gráficas están compuestas de la media y la dispersión de cada variable por cada iteración, mostrándose del modo: Media y Dispersión vs. Iteración.

Las gráficas de Media y Dispersión vs. Iteración, permiten tener un entendimiento del comportamiento cada metaheurística; entender el proceso de búsqueda por cada iteración. Una gráfica de Media y Dispersión vs. Iteración nos puede decir si alguna metaheurístca converge en una menor cantidad de iteraciones o proveer información de si la metaheurística está realizando una búsqueda intensiva o exploratoria.

Capítulo 5

Resultados Experimentales

En este capítulo se presentan los resultados experimentales de la optimización de los hiper-aparámetros de la red LSTM con mtaheurísticas; la evidencia estadística del rendimiento de las redes LSTM para pronosticar las acciones financieras de Google y de Nike; así como los resultados del rendimiento del proceso de optimización de las metaheurísticas.

5.1. Resultados de optimización

En esta sección se presentan los resultados de las soluciones/hiperparámetros obtenidas por cada metaheurística. Se muestran los resultados de las optimizaciones y los resultados de los experimentos de rendimiento.

En este trabajo, un experimento de optimización consiste en utilizar una metaheurística para que realice la optimización de hiperparámetros de una red LSTM; la optimización de los hiperparámetros tiene el objetivo de encontrar una configuración de hiperparámetros que haga o permita a una red LSTM realizar un pronóstico del precio que tendrá una acción financiera al final del día siguiente; en esta tesis solo se dice: "pronóstico del precio de cierre"; con el objetivo de dar simplicidad a la redacción.

A continuación se exponen las características del equipo de cómputo, el lenguaje de programación y *frameworks* que fueron utilizados para llevar a cabo la experimentación; los tiempos (en minutos) de ejecución experimental se muestran en la Tabla 5.1.

Tabla 5.1: Tiempo de ejecución de experimentos

	PSO	FPA	LNFE
Google	54.07	81.74	51.2
Nike	111.27	275.3	144.61

Tiempo de ejecución de cada experimento por cada metaheurística medido en minutos.

Los experimentos se realizaron en una computadora con procesador RYZEN 5 5600x, una tarjeta gráfica GPU-GTX-1050TI y RAM de 16GB. Se utilizó el lenguaje de programación Python con diferentes librerías y módulos, principalmente: NumPy, Matplotlib, Pandas, Jupyter-lab, SciPy y Keras de TensorFlow. Como editor se utilizó Visual Studio Code.

A pesar de que los experimentos de LNFE se diseñaron para que el algoritmo LNFE realice aproximadamente 425 llamadas a función, al final el número de llamadas a función fue distinto en el experimento LNFE-GOOGLE [1], en tal experimento se realizaron 428 llamadas a función; en el caso de LNFE-NIKE, el número de llamadas a función fue de 425. En los experimentos de PSO y FPA, el número de llamadas a función fue de 425.

El algoritmo LNFE se inició con una población de 26 soluciones, tanto para el caso de los experimentos con las acciones de Google como las acciones de Nike; el algoritmo LNFE, después de la primera iteración redujo la población a 16 soluciones.

Comenzar cada población con la misma semilla permite que los valores de la función *fitness* en la primera iteración sean iguales. En el caso de las acciones de Google, los valores de la función *fitness* fueron 7.43E-5, y en el caso de las acciones de Nike fueron 1.23E-5. Esto se puede apreciar en la gráfica de convergencia (Fig. 5.1).

En la gráfica de la Fig. 5.1 se muestran las curvas de convergencia de la optimización de hiperparámetros utilizando las metaheurísticas.

A continuación se exponen los experimentos de rendimiento (ER), los cuales consistieron en realizar 33 entrenamientos de cada una de las 6 soluciones encontradas por cada experimento de optimización (PSO-GOOGLE, PSO-NIKE, FPA-GOOGLE, FPA-NIKE,

[1] En la presente tesis LNFE-GOOGLE, se refiere al experimento de optimizar los hiperparámetros de una red LSTM utilizando LNFE, para que la red realice un pronóstico del precio de cierre de las acciones de GOOGLE; esta notación tiene un sentido similar en otros casos, p. ej. PSO-NIKE, se refiere a la optimización de hiperparámetros de la red LSTM, con PSO y para pronosticar el precio de cierre de las acciones de Nike.

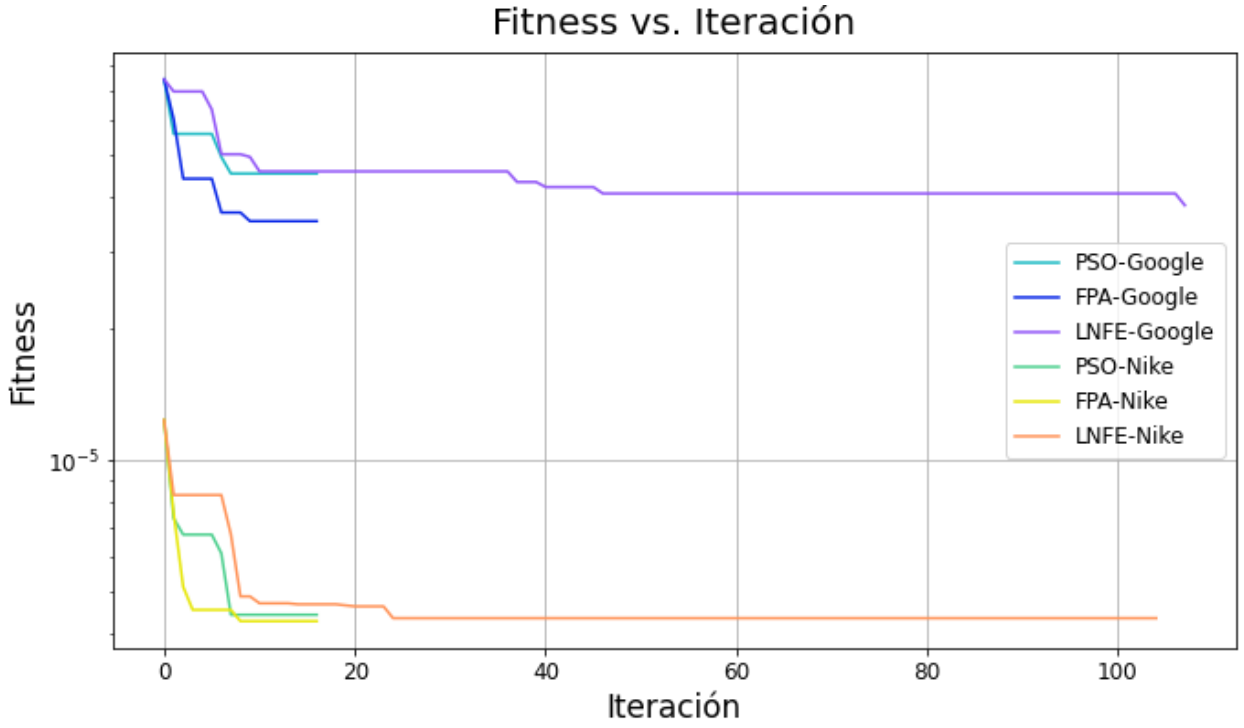


Figura 5.1: *Fitness* vs. Iteración de las metaheurísticas en la optimización de hiperparámetros. (Elaboración propia)

LNFE-GOOGLE y LNFE-NIKE), incluyendo la solución que fue optimizada sin usar metaheurísticas (OSM).

En la Fig. 5.2 se muestran las medias y dispersiones de los ER en el entrenamiento de cada una de las soluciones. Las configuraciones para las soluciones de las acciones de Google, muestran curvas de entrenamiento cercanas; este fenómeno se repite para el caso de las acciones de Nike; mientras que en caso de las configuraciones de OSM, la curva de perdida queda por encima de las configuraciones obtenidas por las metaheurísticas. Este comportamiento se discute en el Capítulo 6.

En la Fig. 5.2 las configuraciones PSO-NIKE, FPA-NIKE y LNFE-NIKE, las curvas de pérdida se encuentran por debajo de PSO-GOOGLE, FPA-GOOGLE y LNFE-GOOGLE. Este patrón se repite en la gráfica de *Fitness* vs. Iteración (Fig. 5.1), en la gráfica las curvas del valor *fitness* del caso de las acciones de Nike se encuentran por debajo de las curvas de los valores *fitness* de las acciones de Google.

En la Tabla 5.2 se muestran los resultados experimentales de la optimización de hiper-

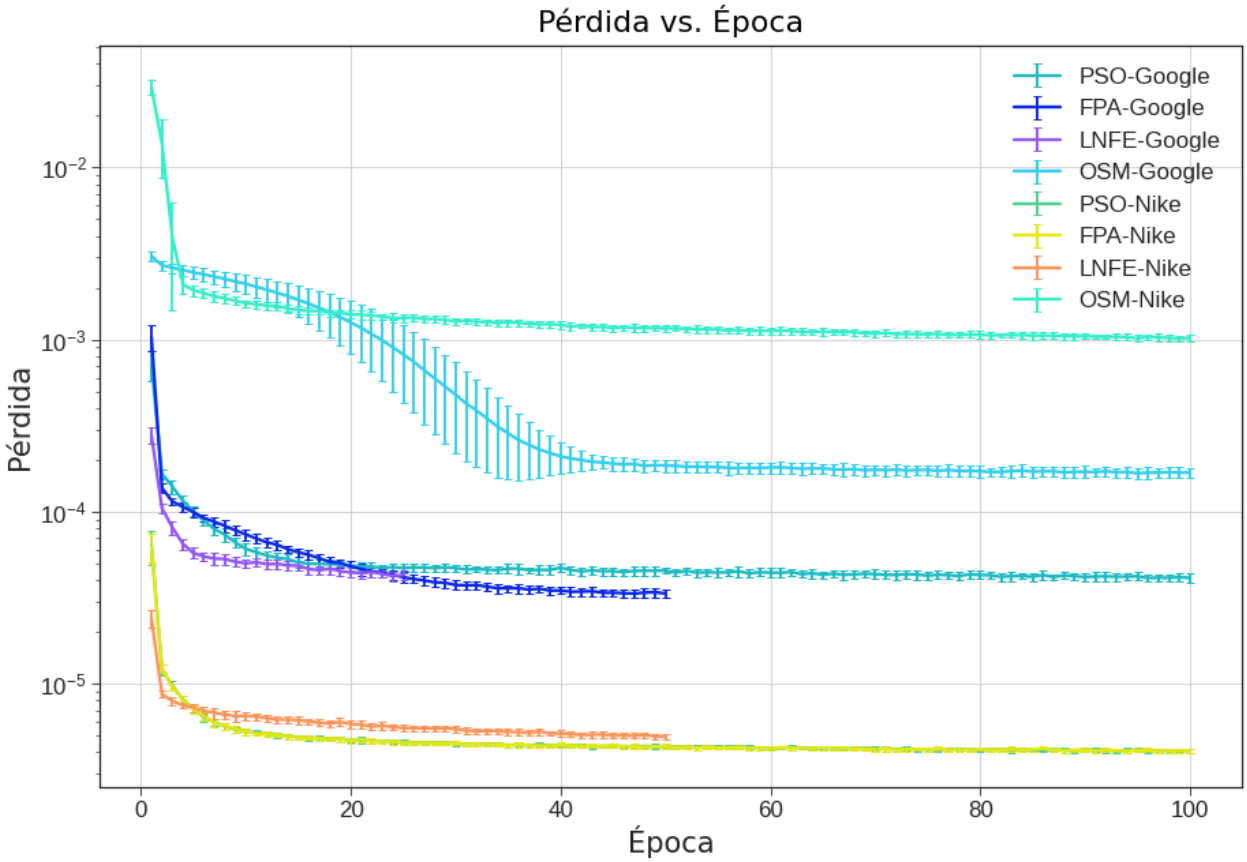


Figura 5.2: Pérdida promedio y dispersión de entrenamiento de 33 experimentos. (Elaboración propia)

parámetros y los ER de cada una de las soluciones. Los mejores valores *fitness* tanto del caso de las acciones de Google como de las acciones de Nike se muestran en verde, los peores valores *fitness* se muestran en rojo y los valores en amarillo son los de rendimiento medio.

Las soluciones obtenidas por cada metaheurística nos dan información de que las redes no necesitan más de una celda LSTM para realizar un pronóstico con un rendimiento satisfactorio. Esto concuerda con los experimentos hechos previamente (Apéndice D). De hecho la mayoría de las soluciones obtenidas por las metaheurísticas tienen 1 celda LSTM y 150 estados ocultos, a excepción de la solución PSO-GOOGLE la cual difiere con 96 estados ocultos.

Los valores encontrados de α por cada metaheurística para el caso de las acciones de Google son iguales ($\alpha = 10^{-3}$); mientras que para el caso de Nike en las soluciones de PSO y FPA, fue de 10^{-4} y en LNFE fue de 10^{-3}. Lo cual muestra cierto grado de consistencia

(esto se discute con más detalle en el Capítulo 6).

En el caso del tamaño de la ventana, todos los valores fueron distintos tanto para el caso de las acciones de Google como las acciones de Nike. En el caso del tamaño del lote para las acciones de Google, ningún valor fue igual, mientras que en el caso de las acciones de Nike, todos los valores fueron iguales, lo cual muestra inconsistencia.

En el caso de las soluciones de las acciones de Google, el número de épocas difirió en la mitad de las soluciones, con la salvedad del caso de las soluciones de las acciones de Nike con PSO y FPA, y las acciones de Google con PSO, los cuales coincidieron.

Cada una de las soluciones encontradas por PSO, FPA y LNFE en el caso de las acciones de Google, tienen valores de la función *fitness* cercanos, en el orden de 10^{-5}. Los experimentos de rendimiento también tienen resultados similares, los promedios de pérdida se encuentran en el orden de 10^{-5}, en la desviación estándar de pérdida en 10^{-6}, en el promedio de evaluación en 10^{-4} y en la desviación estándar en 10^{-4}.

En el caso de las soluciones de Nike de PSO, FPA y LNFE, los valores de la función *fitness* son cercanos, en el orden de 10^{-6}. Los experimentos de rendimiento también tienen resultados similares, los promedios de pérdida se encuentra en el orden de 10^{-6}, en la desviación estándar de pérdida en 10^{-7}, en el promedio de evaluación en 10^{-4} y en la desviación estándar en 10^{-5}.

A pesar de la similitud en las estadísticas para el caso de las acciones de Google, se encuentra que las curvas de entrenamiento (Fig. 5.2) son diferentes para cada una de las soluciones (de PSO, FPA y LNFE) sin embargo, se encuentran en una región cercana del orden de 10^{-5}. En el caso de las acciones de Nike las curvas de la solución de PSO y FPA se traslapan y mientras que en el caso de LNFE-NIKE difiere en su curva de entrenamiento; no obstante, las curvas se encuentran en el orden de 10^{-6}. El mismo efecto se ve en el diagrama de caja y bigote para datos de evaluación (Fig. 5.3).

En los ER de las acciones de Google, PSO tiene el peor rendimiento con datos de entrenamiento; LNFE presenta el rendimiento medio; y FPA el mejor rendimiento. Con datos de evaluación, LNFE tiene el peor rendimiento, PSO el rendimiento medio y FPA el mejor

Tabla 5.2: Soluciones de la OHP de las redes LSTM y estadísticas de rendimiento de las soluciones.

	Acciones de Google				Acciones de Nike			
Hiperpárametros	**PSO**	**FPA**	**LNFE**	**OSM**	**PSO**	**FPA**	**LNFE**	**OSM**
Cl_s	1	1	1	4	1	1	1	4
H_s	96	150	150	96	150	150	150	96
W_s	80	40	60	50	80	90	100	50
α	10^{-3}	10^{-3}	10^{-3}	10^{-2}	10^{-4}	10^{-4}	10^{-3}	10^{-2}
B_s	32	64	8	16	16	16	16	16
E_s	100	50	25	100	100	100	50	100
No. de parámetros:	37,729	91,351	91,351	260,065	91,351	91,351	91,351	260,065
Información de solución	**PSO**	**FPA**	**LNFE**	**OSM**	**PSO**	**FPA**	**LNFE**	**OSM**
No. de Iteraciones:	16	16	109	-	16	16	106	-
Llamadas a función:	425	425	428	-	425	425	425	-
Fitness:	4.52E-5	3.52E-5	3.83E-5	-	4.43E-6	4.28E-6	4.35E-6	-
Promedio de pérdida:	4.10E-5	3.33E-5	4.27E-5	1.68E-4	4.07E-6	4.06E-6	4.91E-6	1.02E-3
Desviación estándar de pérdida:	2.72E-6	1.75E-6	2.41E-6	1.14E-5	1.16E-7	1.14E-7	1.99E-7	4.81E-5
Promedio de evaluación:	3.25E-4	2.07E-4	7.12E-4	3.86E-3	1.08E-4	1.17E-4	5.56E-4	2.99E-3
Desviación estándar de evaluación:	2.54E-4	1.20E-4	6.14E-4	1.09E-3	4.53E-5	3.72E-5	3.84E-4	5.33E-4

rendimiento. Considerando únicamente las soluciones obtenidas por las metaheurísticas. En estos casos solo se comparó las soluciones de las metaheurísticas porque el rendimiento de la solución OSM no compite con las soluciones obtenidas por metaheurísticas.

En los ER con las acciones de Nike, utilizando datos de entrenamiento, FPA tiene el mejor rendimiento, PSO el rendimiento medio y LNFE el peor rendimiento. Con datos de evaluación, PSO tiene el mejor rendimiento, con varianza media; FPA el rendimiento medio con la menor varianza; y LNFE con el peor rendimiento. En cada caso de la comparación se descartó los ER de OSM porque los rendimientos de las redes LSTM de OSM son inferiores a todas las soluciones.

Para apreciar de manera más precisa los resultados experimentales con datos de evaluación se presentan los diagramas de Caja y Bigote en la Fig. 5.3. Las soluciones para el caso de las acciones de Google tienen mayor error de evaluación y mayor dispersión que las soluciones para el caso de las acciones de Nike. En el caso de las acciones de Nike, los errores de evaluación se encuentran cercanos. A pesar de que las soluciones de OSM tienen mayor error, su dispersión es menor.

Los errores de evaluación se reflejan en las gráficas de pronósticos que están en la Fig. 5.4. Mostrando que las soluciones de PSO, FPA y LNFE compiten entre sí, dejando por detrás a las soluciones de OSM.

En los pronósticos de OSM las predicciones tienden a empeorar conforme van aumen-

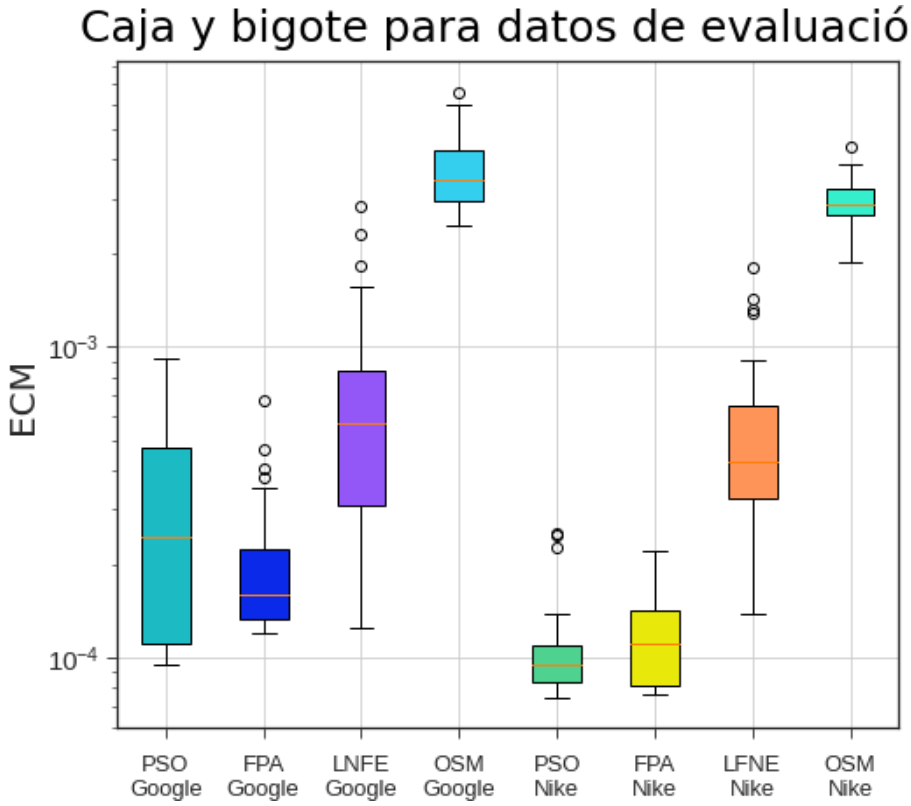


Figura 5.3: Diagrama de caja y bigote con datos de evaluación de los ER. (Elaboración propia)

tando los días desde 2018 a 2022. Este fenómeno se repite para las predicciones de las soluciones de LNFE, aunque se pronuncia más en OSM. Incluso en las predicciones de las soluciones de OSM y LNFE en las partes donde aumenta el error de evaluación, parece que las predicciones es una copia del precio de las acciones de días anteriores.

En la Fig. 5.4 se observa en la solución LNFE-GOOGLE en el periodo de predicción de 2021 a 2022, que el valor de la predicción se encuentra por debajo de la serie original; repitiéndose el patrón en LNFE-NIKE en periodo de 2018 a 2022. Mientras que en los casos de FPA y PSO, tanto para Google como Nike, las curvas de las predicciones son cercanas al valor real.

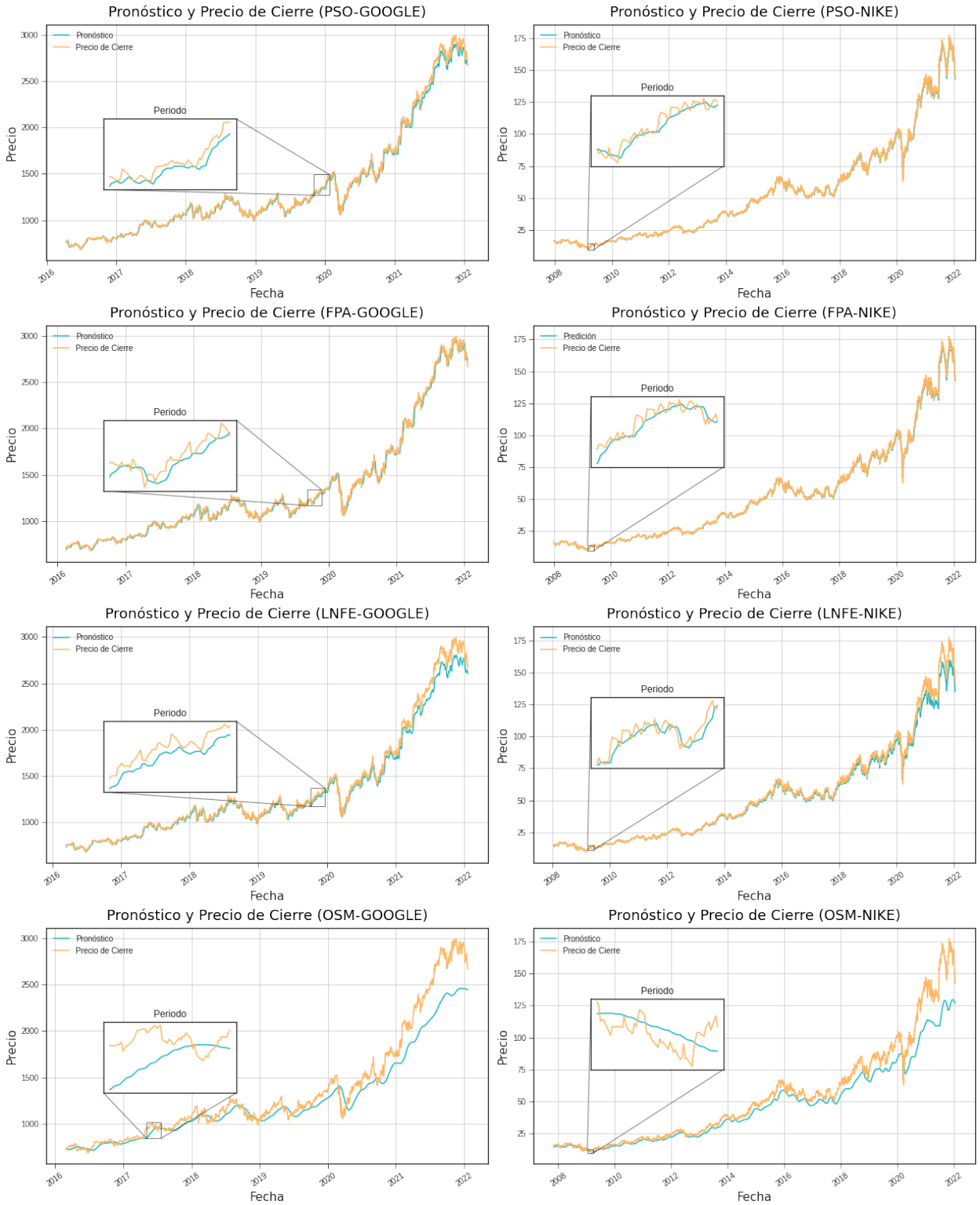


Figura 5.4: Precio de cierre y Predicción promedio de 33 experimentos. (Elaboración propia)

5.2. Resultados de convergencia

En esta sección se muestran los resultados y análisis de la convergencia de cada una de las metaheurísticas al optimizar los hiperparámetros de las redes LSTM para realizar el pronóstico de las acciones de Google y de las acciones de Nike.

En la sección anterior ya se mencionó que las poblaciones iniciales de cada metaheurística son iguales; por lo que cada solución tendrá el mismo promedio y dispersión por cada variable en un inicio. Esto se puede ver en las gráficas de las Figs. 5.5 a 5.10.

La comparación se llevó a cabo usando las Ecs. (4.5) y (4.6), formando la Tabla 5.3. Para más detalles de cómo se formó la Tabla, ver el Capítulo 4.

En la Tabla 5.3, se aprecia que el algoritmo con mayor convergencia promedio es PSO, tanto para PSO-GOOGLE como PSO-NIKE, con 86.0 % y 92.8 %, respectivamente. Y todos los porcentajes de convergencia individuales están por encima de 66.66 % para cada una de las variables optimizadas por PSO.

En el caso de convergencia promedio del algoritmo LNFE, los valores de convergencia promedio son de 71.5 % y 85.1 %, para el caso de las acciones de Google y el caso de las acciones de Nike, respectivamente

En la optimización con el algoritmo FPA los porcentajes de convergencia promedio fueron de 27.1 % y 27.4 %, para el caso de las acciones de Google y el caso de las acciones de Nike, respectivamente.

En primera instancia, se puede apreciar que PSO tiene la mejor convergencia, seguido de LNFE y FPA, no obstante la dispersión de la última población no describe el comportamiento de la dispersión en cada iteración del proceso de optimización, para poder analizar el historial de las dispersiones en cada iteración se hace uso de gráficas de convergencia, que se muestran más adelante.

En el caso de la las celdas LSTM se puede observar en las gráficas de convergencia (Fig. 5.5) que la búsqueda de este hiperparámetro converge a un mismo punto. En la búsqueda del número de celdas LSTM, PSO y LNFE tiene porcentajes de convergencia por encima del 66.66 %, en el caso las soluciones de FPA, tiene porcentajes de convergencia entre 33.33 %

Tabla 5.3: Porcentajes de convergencia.

		PSO		FPA		LNFE	
Hiperparámetro	**DEI**	**Google**	**Nike**	**Google**	**Nike**	**Google**	**Nike**
Cl_s	0.27	89.1 %	96.3 %	48.2 %	65.3 %	96.4 %	94.9 %
H_s	0.30	87.5 %	93.9 %	76.1 %	96.3 %	75.1 %	89.0 %
W_s	0.31	91.7 %	94.5 %	−11.4 %	−19.8 %	71.2 %	94.3 %
α	0.34	80.4 %	92.8 %	66.1 %	37.9 %	67.1 %	83.7 %
B_s	0.26	89.8 %	91.9 %	−18.1 %	−28.2 %	49.6 %	72.1 %
E_s	0.30	77.9 %	87.7 %	2.1 %	13.1 %	69.4 %	76.9 %
CPT	-	86.0 %	92.8 %	27.1 %	27.4 %	71.5 %	85.1 %

DEI, significa Desviación Estándar Inicial. CPT, significa convergencia promedio total. Los valores marcados en morado son los que tienen un porcentaje mayor a 66.66 %; los valores marcados en verde son aquellos que se encuentran en el intervalo de entre 66.66 % y 33.33 %; y los valores marcados en amarillo son aquellos menores a 33.33 %.

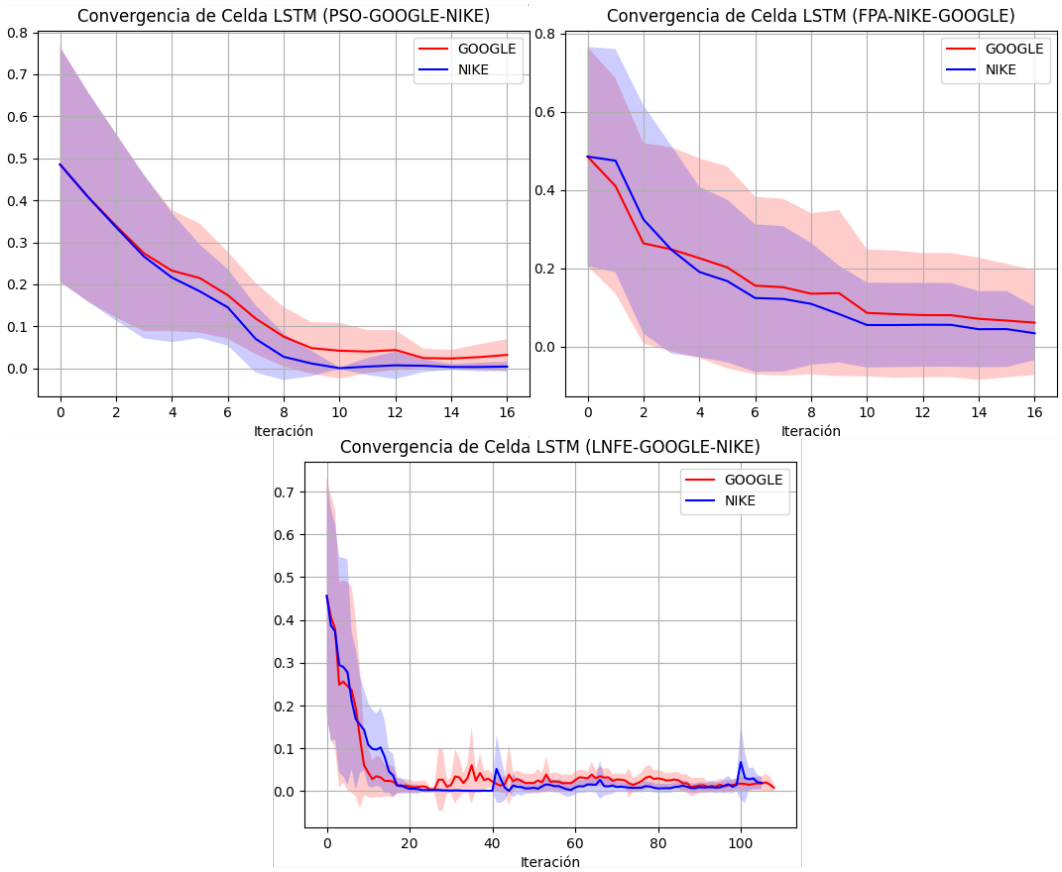


Figura 5.5: Convergencia de No. de Celdas. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia)

y 66.66 %.

En la sección anterior se describe que el número de épocas fue diferente para la mayoría de las soluciones; este efecto también se observa en las gráficas de convergencia de la Fig.

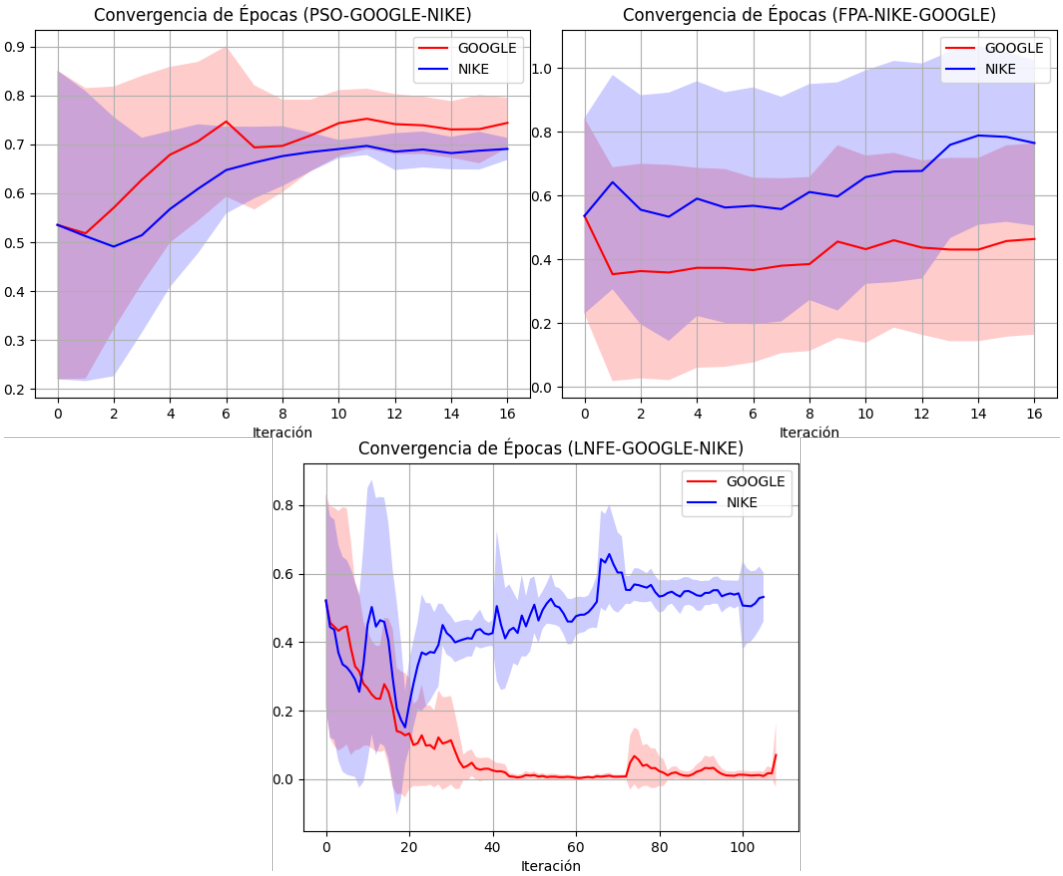


Figura 5.6: Convergencia de No. de Épocas. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia)

5.6, teniendo diferentes valores al final de la búsqueda en cada metaheurística. En el caso de PSO y FPA, se nota una tendencia de converger a cierto de valores; en el caso de FPA, la búsqueda se mantuvo con una dispersión en cada iteración, disminuyendo únicamente en un 27.1 % para el caso de FPA-GOOGLE y 27.4 % para el caso de FPA-NIKE.

El número de estados ocultos tuvo una convergencia más pronunciada comparando en cada una de las metaheurísticas (Fig. 5.7) si se compara con los demás hiperparámetros. Comparándolo con las gráficas de convergencia del número de celdas LSTM (Fig. 5.5), el comportamiento fue similar, con la única salvedad que en el caso de PSO-GOOGLE convergió en otro valor, no obstante el estado oculto se encuentra cercano al límite superior del espacio de búsqueda.

En el caso FPA-NIKE, el número de estados ocultos convergió en la iteración número 8, con 200 llamadas a función. En el caso de LNFE, el algoritmo convergió con menos llamadas

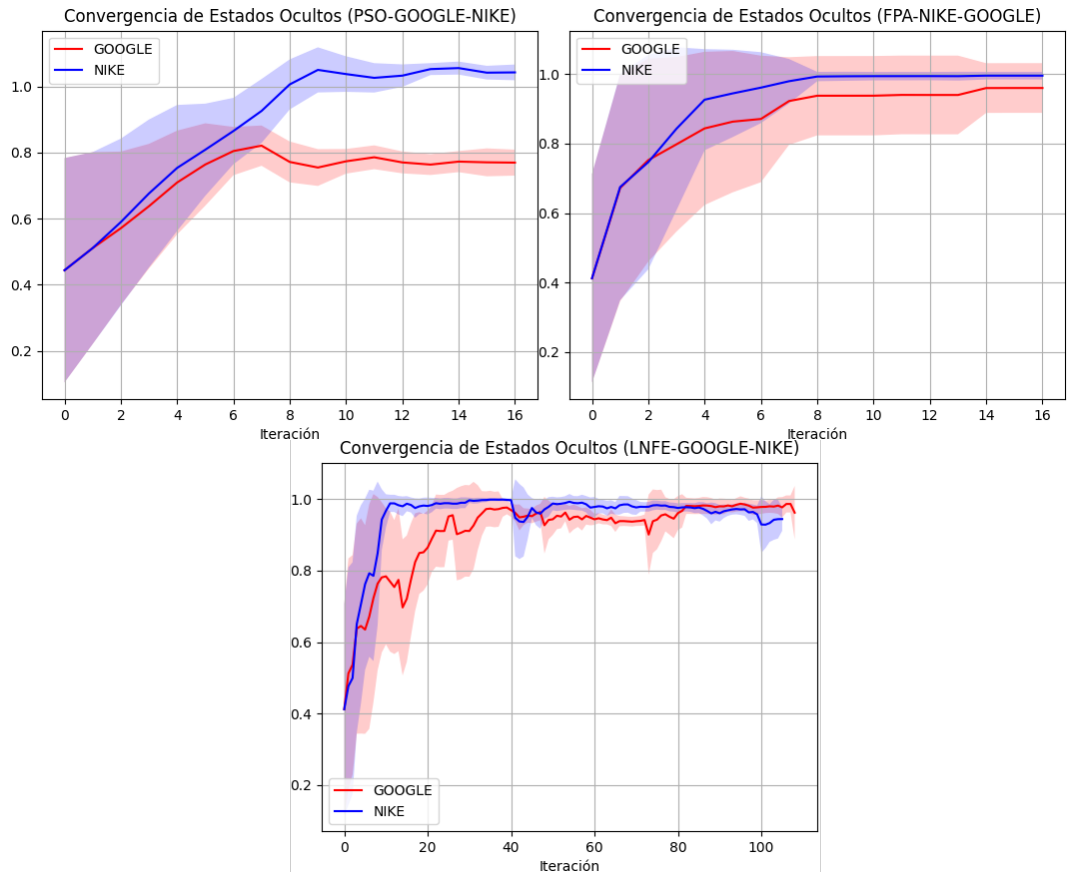


Figura 5.7: Convergencia de No. de Estados Ocultos. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia)

a función. En el caso de las acciones de Google, fueron 168 llamadas a función con un valor de la función *fitness* de 4.33E−5; en el caso de las acciones de Nike la convergencia se dio en 143 llamadas a función, con un valor de la función *fitness* de 4.35E−6.

En las gráficas de convergencia del factor de aprendizaje de la Fig. 5.8 se observa un comportamiento similar a las demás gráficas de convergencia descritas. La optimización con LNFE converge en una menor cantidad de llamadas a función, PSO tiene una convergencia equilibrada, entre intensificación y exploración, porque inicia con una dispersión amplia y en cada iteración la dispersión va disminuyendo poco a poco; mientras que FPA mantiene la exploración en cada iteración; porque la dispersión de FPA se mantiene relativamente constante. Los valores del factor de aprendizaje a los que convergen PSO, FPA y LNFE, en el caso de las acciones de Google son iguales; en el caso de las acciones de Nike, solo difiere en LNFE. La búsqueda en cada iteración del factor de aprendizaje se mantiene entre el

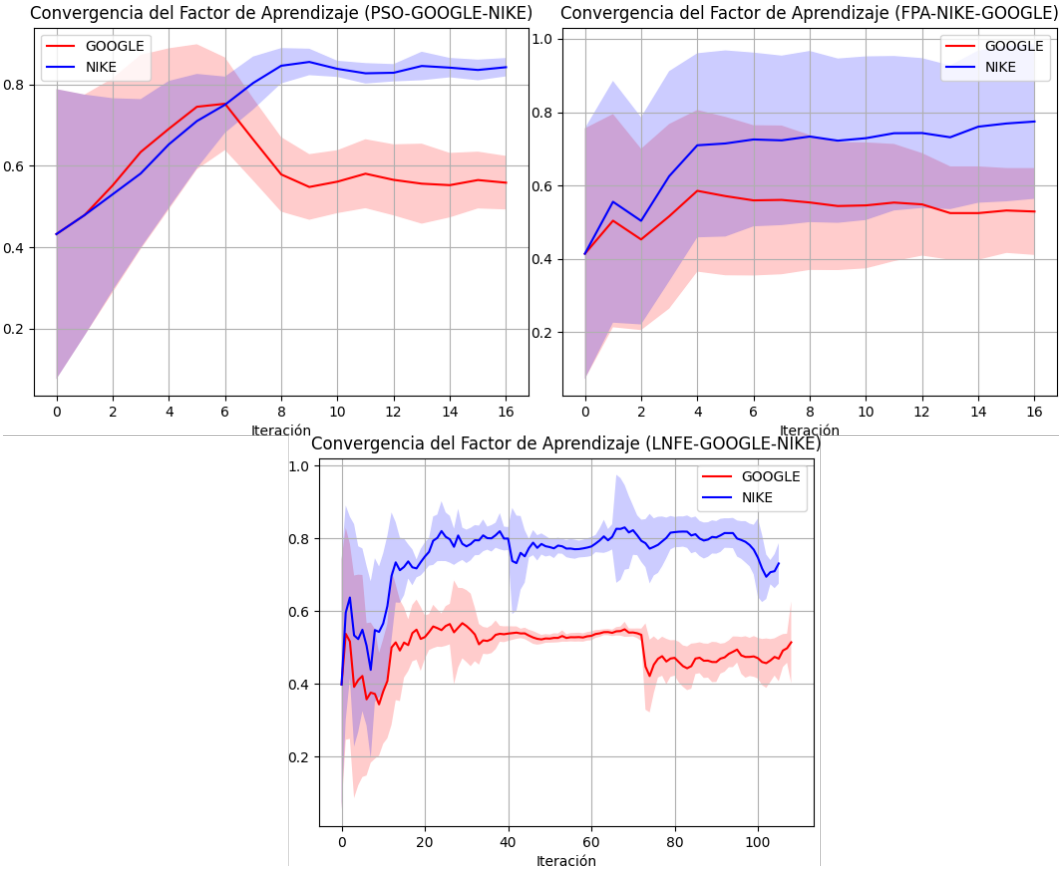


Figura 5.8: Convergencia de Factor de Aprendizaje. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia)

valor medio y el valor máximo del espacio de búsqueda. Nótese que al momento de asignar los valores del espacio de búsqueda no se utilizaron valores muy pequeños ni muy elevados para evitar que el entrenamiento sea lento o que el entrenamiento oscilara alrededor de un mínimo.

En el tamaño del lote, en el caso de las acciones de Google las soluciones obtenidas por las metaheurísticas fueron distintas, mientras que en caso de Nike fueron iguales. En el caso de FPA, hubo una divergencia de 18.1 % y 28.2 %, para el caso de las acciones de Google y para el caso de las acciones de Nike, respectivamente. En las gráficas de la Fig. 5.9, se aprecia que PSO inicia con mayor dispersión y conforme aumentan las iteraciones, la dispersión baja hasta converger en un valor, mientras que en FPA aumenta la dispersión en la última iteración. En el caso de LNFE, tiene un comportamiento errático porque la dispersión disminuye y aumenta en cada iteración; a pesar de eso, la dispersión después de

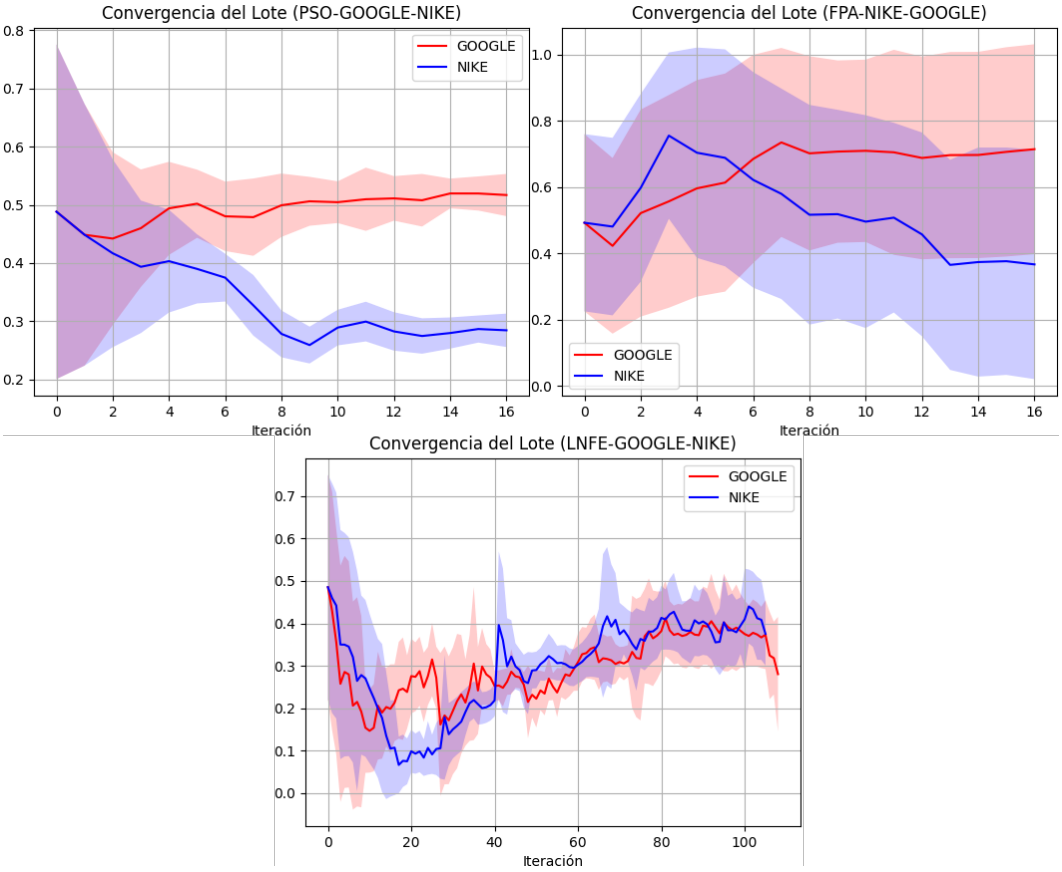


Figura 5.9: Convergencia de Tamaño del Lote. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia)

las primeras iteraciones es cercana a la dispersión de las últimas iteraciones de PSO y menor a la dispersión de FPA.

En la Fig. 5.10 se muestran las gráficas de convergencia del tamaño de la ventana. Estas gráficas muestran un comportamiento similar a las demás gráficas de convergencia: PSO con una convergencia suave, FPA mantiene su dispersión en cada iteración y LNFE converge en una menor cantidad de llamadas a función. En cada una de las soluciones el tamaño de la ventana es distinto.

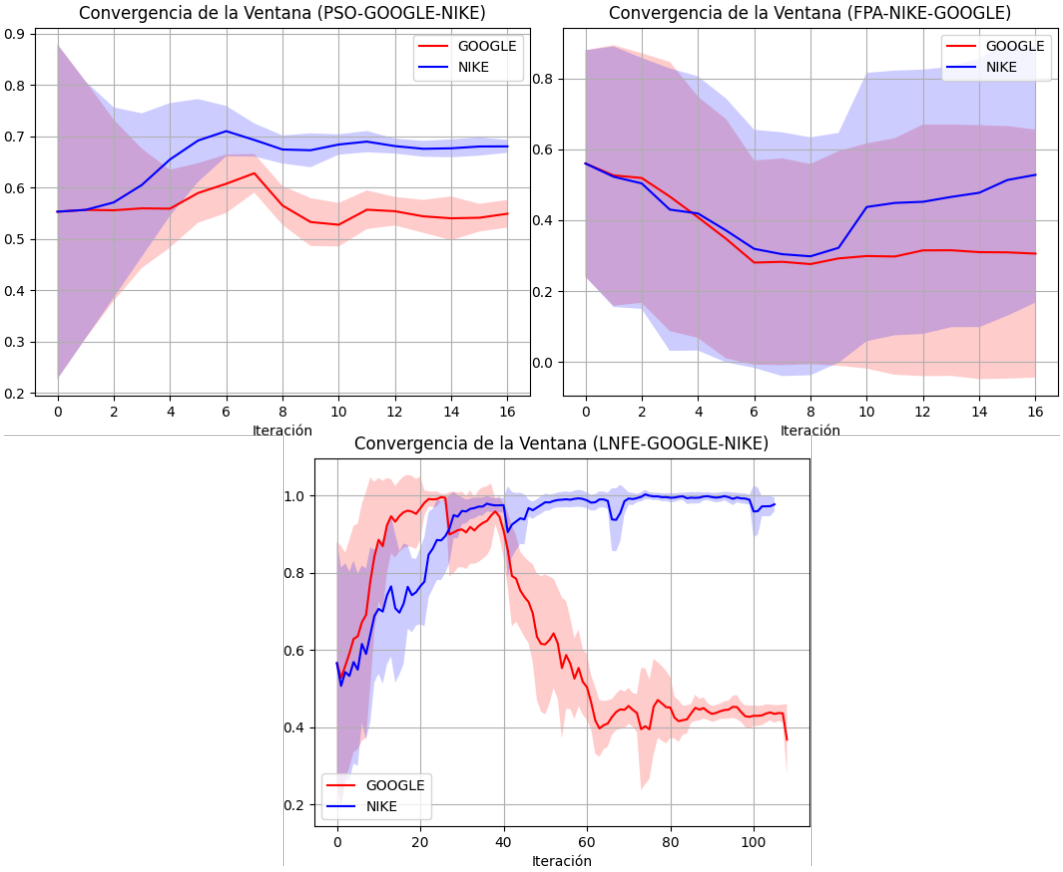


Figura 5.10: Convergencia de Tamaño de la Ventana. Las soluciones fueron escaladas entre $[0, 1]$. (Elaboración propia)

Conclusiones

La optimización con FPA provee de las mejores soluciones tanto para el caso de las acciones de Google como para las acciones de Nike, de igual manera FPA presenta los mejores rendimientos.

Las soluciones de LNFE, tanto para el caso de las acciones de Google como para las acciones de Nike, tienen el menor rendimiento, a pesar de no ser el menor valor *fitness* obtenido en la HPO, esto puede deberse a un sobre ajuste de la red LSTM, es necesario comprobarlo.

A pesar de que las soluciones encontradas por PSO, tanto para el caso de las acciones de Google como las acciones de Nike, tienen el menor valor *fitness* en la HPO, de entre las 3 metaheurísticas; su rendimiento en prueba fue superior a las soluciones encontradas por LNFE.

El análisis de las metaheúristcas nos muestra que no solo se necesita comparar la dispersión de la primera población contra la dispersión de la última población; sino que son las gráficas de convergencia las que dan mayor detalle del comportamiento de la optimización en cada iteración.

PSO muestra una búsqueda más equilibrada entre intensificación y exploración; porque en cada gráfica de convergencia el algoritmo inicia con una elevada dispersión, la cual baja lentamente comparada con LNFE, que la dispersión baja rápido en pocas iteraciones y FPA que mantiene su dispersión o no disminuye de manera significativa. LNFE, tiende a converger en una menor cantidad de iteraciones, con resultados satisfactorios. FPA tiene una convergencia baja, se nota en cada iteración de la optimización que mantiene una elevada dispersión, incluso en algunas variables diverge, mientras que en las variables que influyen significativamente en el rendimiento de la red LSTM, tiende a converger.

El tamaño del lote tiene valores distintos en el caso de las acciones de Google; porque es un valor que no afecta de manera significativa el rendimiento de la red y solo afecta en la velocidad de convergencia; pero esto no explica porque en el caso de las acciones de Nike todos los valores del tamaño del lote son iguales.

Los resultados del tamaño de la ventana dan indicios de que es un valor independiente de los demás hiperparámetros, como sucede con el tamaño del lote.

Las arquitecturas de las redes LSTM tienden a ser más simples que aquellas redes OSM, lo que indica que la sintonización de los hiperparámetros da rendimientos superiores a las redes OSM.

Finalmente, se reafirma que se puede realizar un pronóstico financiero de las acciones de Google y Nike con una red LSTM optimizando sus hiperparámetros con metaheurísticas (PSO, FPA y LNFE); y que los resultados superan a redes LSTM las cuales se han ajustado sus hiperparámetros sin metaheurísticas.

Capítulo 6

Discusiones

En el presente capítulo se presentan las discusiones de los resultados experimentales. Las discusiones se presentan en dos secciones, la primera llamada Discusiones de soluciones, en donde se discuten los rendimientos de las soluciones; la segunda nombrada Discusiones de convergencia, en la cual se discute el comportamiento de las metaheurísticas en el proceso de optimización de hiperparámetros.

6.1. Discusiones de soluciones

En esta sección se discuten las soluciones obtenidas; desde los hiperparámetros que se obtuvieron cada hiperparámetro como tamaño de la ventana, número de celdas LSTM, estados ocultos, entre otros, hasta el contraste entre las soluciones obtenidas.

Los resultados experimentales muestran que las arquitecturas de las soluciones obtenidas con metaheurísticas son más simples que la OSM; porque las 2 arquitecturas obtenidas por las metaheurísticas tienen menos parámetros de entrenamiento, una con $37,729$ parámetros y otra con $91,351$ parámetros, mientras que la arquitectura de OSM tiene $260,065$ parámetros, lo que equivale a ser 6.8 veces más profunda que la primera red mencionada y 2.8 veces más profunda que la segunda red mencionada.

Las soluciones de OSM no compiten con las soluciones de las metaheurísticas. Esto puede deberse a que los hiperparámetros obtenidos por las metaheurísticas son aproximaciones

óptimas.

Los algoritmos metaheurísticos se ajustaron para realizar aproximadamente 425 llamadas a función; por esta razón, los tiempos de ejecución de los experimentos para las acciones de Google fueron similares (62.33 minutos en promedio) y los tiempos de los experimentos para las acciones de Nike fueron similares (177.06 minutos en promedio).

La razón del porqué los tiempos de ejecución de los experimentos para el caso de las acciones de Google es menor al tiempo de ejecución de los experimentos de las acciones de Nike, se debe al tamaño de las bases de datos. La base de datos para las acciones de Google tiene $4,388$ datos y en el caso de las acciones de Nike son $10,376$ datos.

El hecho de que LNFE reduzca la población a 16 individuos después de la primera iteración, causa que el algoritmo tenga menor diversidad de soluciones. A pesar de eso, las soluciones fueron satisfactorias, compitiendo con PSO y FPA; esto indica que LNFE puede tener resultados satisfactorios con una población pequeña.

La razón de que las soluciones de LNFE y OSM realicen predicciones con un error que va aumentando en el periodo de 2018 a 2022 puede deberse a que la solución obtenida tenga un cierto grado de sobre ajuste, lo que impide que la red trabaje con nuevos datos de los precios de las acciones.

Las curvas de aprendizaje y las curvas de convergencia muestren que en el caso de las acciones de Nike se encuentren por debajo de las acciones de Google puede deberse a que la base de datos de las acciones de Google es menor a la base de datos de las acciones de Nike; porque al entrenar la red LSTM en el caso de las acciones de Nike, tiene más ejemplos de los que aprender. Empero, no se pueden dar explicaciones concluyentes al tener diferentes tamaños de ventana para cada configuración, porque el tamaño del conjunto de entrenamiento depende del tamaño de la ventana; entre más grande sea el tamaño de la ventana, menor será el conjunto de entrenamiento y entre más pequeño sea el tamaño de la ventana, el conjunto de entrenamiento será más grande.

Los valores del número de épocas es distinto en cada solución; esto puede deberse a que el número de épocas depende de los demás hiperparámetros; como número de celdas y

estados ocultos.

El factor de aprendizaje encontrado en todos casos de las acciones de Google fue igual; mientras que para Nike solo difirió en un caso; esta información aunada a lo visto en las gráficas de convergencia (Fig. 5.8), nos da información de que es un valor consistente en el entrenamiento de una red LSMT para realizar un pronóstico financiero; quizá esta consistencia pueda repetirse en otro tipo de problemas.

En el caso de las acciones de Googel el tamaño de lote fue distinto en cada solución. Esto se puede explicar porqué el tamaño del lote afecta en la velocidad de entrenamiento, no obstante, no explica por qué en el caso de las acciones de Nike el tamaño del lote fue igual.

La configuración de hiperparámetros que retornó FPA en el caso de las acciones de Google provee la predicción con el mejor rendimiento con el menor riesgo[1] en comparación con las demás soluciones (PSO-GOOGLE y LNFE-GOOGLE). En el caso de la configuración de PSO-GOOGLE, se tiene un pronóstico medio con mayor riesgo; y en el caso de LNFE-GOOGLE tiene el peor rendimiento con riesgo medio.

La configuración de hiperparámetros que provee PSO en el caso de las acciones de Nike brinda una predicción con el mejor rendimiento con un riesgo medio, en comparación con las demás soluciones (FPA-NIKE y LNFE-NIKE). Mientras que la solución de FPA-NIKE, tiene la predicción de rendimiento media con el menor riesgo; en la solución LNFE-NIKE obtuvo la peor predicción con mayor riesgo. Esta comparación se realizó sin considerar la solución OSM-NIKE.

Los rendimientos de los pronósticos de las soluciones encontradas por las metaheurísticas compiten entre sí y superan a las soluciones obtenidas sin metaheurísticas. Los rendimientos de OSM pueden deberse a su profundidad, porque las redes más profundas tienden a sobre ajustarse, impidiendo tener un adecuado rendimiento con datos nuevos; mientras que las redes menos profundas tienen menos probabilidad de sobre-ajustarse.

[1]En finanzas, cuando se dice que existe mayor riesgo, es porque hay mayor incertidumbre de lo que podrá pasar en un futuro. Ésta incertidumbre se puede medir por la dispersión. Por eso un modelo de pronósticos que tenga mayor dispersión, tiene mayor riesgo en sus predicciones.

A pesar de que las soluciones obtenidas por PSO, FPA y LNDE compiten entre sí, PSO y FPA dejan por detrás a LNFE, porque LNFE tiene menor rendimiento y el error de predicción de LNFE va aumentando conforme la frontera de predicción se acerca a 2022; este fenómeno que también se repite con las soluciones de OSM.

6.2. Discusiones de convergencia

En esta sección se discute el comportamiento que tuvieron las metaheurísticas a lo largo de la búsqueda de optimización de hiperparámetros.

LNFE muestra una menor convergencia que PSO en la última iteración, a pesar de que LNFE es un algoritmo que converge en una menor cantidad de iteraciones [77]. No obstante, LNFE tiene la característica de aumentar la exploración (dispersión) al reiniciar la población cuando no se mejora el valor *fitness* después de cierto número de iteraciones. Esto explica que LNFE tenga mayor dispersión que en la última iteración de PSO.

El número de celdas para todas las soluciones convergen en un mismo valor; esto indica la consistencia con la teoría que dice que el rendimiento depende en mayor medida de la arquitectura. Por lo que las metaheurísticas muestran un comportamiento con tendencia hacia una arquitectura de una celda con 150 estados ocultos, para generar un rendimiento con el menor error posible.

En el caso de la convergencia del número de épocas, cada valor de las soluciones es distinto, porque el número de épocas es consistente cuando los hiperparámetros están sintonizados y no cambian. Si se contrasta la convergencia para los casos de las acciones de Google que tienen soluciones distintas, el número de épocas difiere para cada valor. En el caso de las acciones de Nike, con PSO y FPA los demás hiperparámetros (número de celdas, estados ocultos y tamaño del lote) son más consistentes; dando como resultado el mismo número de épocas. En el caso de LNFE, cambia el factor de aprendizaje y tamaño de la ventana, por lo que cambia el número de épocas; esto también da indicios de que el tamaño de la ventana no tiene efecto significativo sobre el rendimiento del pronóstico de la red LSTM.

El número de estados ocultos fue consistente en la mayoría de las soluciones a excepción de la solución de PSO para las acciones de Google. A pesar de ello, esta información aunada con la del número de celdas indican que para obtener un pronóstico con rendimientos deseados no es necesario tener una red con varias celdas LSTM, sino que es necesario aumentar el número de estados ocultos para obtener un mejor rendimiento.

El hecho de que el tamaño del lote afecte en la velocidad de aprendizaje explica porqué en las gráficas de convergencia (Fig. 5.9) LNFE aumenta y disminuye la dispersión en cada iteración, también explica porqué FPA aumento su dispersión. Inclusive en el caso de las acciones de Google todos los valores del tamaño del lote fueron distintos. Sin embargo, no explica porqué en el caso de las acciones de Nike todos los valores del tamaño de lote son iguales; quizás la razón se encuentra en que los valores de los demás hiperparámetros son similares, incluso los valores del tamaño de la ventana (que a pesar de ser distintos son valores cercanos al límite superior del espacio de búsqueda).

Los valores del tamaño de la ventana son distintos en cada solución. En el proceso de optimización tienen un comportamiento similar al tamaño de lote, el cual es un hiperparámetro que solo afecta en la velocidad aprendizaje. Esta información indica que el tamaño de la ventana puede ser un valor que tenga poca influencia en el rendimiento de la red LSTM para que esta realice un pronósticos financieros. Si bien es claro que el tamaño de lote es independiente de otros hiperparámetros como los de la arquitectura, también comienza a aparecer información sobre la poca influencia del tamaño la ventana en el pronóstico financiero.

El análisis de la dispersión vs. iteración de cada metaheurística, nos da información del comportamiento exploratorio o intensivo de la metaheurística. En este trabajo se nota el equilibrio entre exploración e intensificación de PSO; en el caso de FPA se nota que el algoritmo es más exploratorio; mientras que LNFE es más intensivo.

El análisis de la dispersión vs. iteración no solo nos permite dar conclusiones de las metaheurísticas, sino también del comportamiento de las soluciones y da información de posibles razones del comportamiento de las soluciones.

Capítulo 7

Conclusiones y trabajos a futuro

La investigación realizada determina que es posible optimizar los hiperparámetros de una red LSTM, brindando a la red la posibilidad de realizar un pronóstico financiero. La optimización de los hiperparámetros se logró con tres metaheurísticas PSO, FPA y LNFE; las cuales son metaheurísticas basadas en poblaciones. Con la optimización de hiperparámetros la red LSTM logra pronosticar el precio de cierre del día siguiente de las acciones de Google y de las acciones de Nike.

Las metaheurísticas basadas en poblaciones utilizadas en la presente tesis no son los únicos algoritmos que han demostrado resultados satisfactorios en la optimización de hiperparámetros, p. ej. el GA es ampliamente utilizado en la optimización de hiperparámetros de distintos modelos de ML para realizar tanto pronósticos financieros como en otros campos como generación de datos, clasificación, entre otros. La diferencia que tiene GA con las metaheurísticas basadas en poblaciones, es que estas pueden utilizar representaciones de soluciones más sencillas, trabajando sobre el espacio de los números continuos y no sobre números binarios.

La representación que se utilizó en la presente tesis es más simple que las representaciones que utiliza en algoritmos genéticos, lo cual facilita la optimización de hiperparámetros, porque al mantener las metaheurísticas trabajando sobre el espacio de los continuos no requiere representaciones complejas.

Cuando los hiperparámetros están sintonizados con las metaheurísticas, las soluciones obtenidas no solo superan a una red LSTM optimizada sin metaheurísticas, sino que además las redes son menos profundas o más simples.

El número de épocas es una variable que solo afecta la velocidad de entrenamiento, por lo que su valor es distinto en cada una de las soluciones obtenidas por las metaheurísticas. Así se puede realizar una optimización de hiperparámetros fijando el número de épocas y luego, en caso de ser necesario, entrenar con más épocas la red neuronal obtenida por la metaheurística.

Los valores del tamaño de la ventana difieren en cada solución. Esto coincide con la literatura, pues en ella no existe consenso al designar un tamaño de ventana. Puede ser un hiperparámetro que no influya de manera significativa en un pronóstico de acciones financieras.

Al encontrar varias arquitecturas similares en soluciones las metaheurísticas son consistentes al buscar la arquitectura de una red LSTM. Esto refuerza la teoría que dice que la arquitectura de una red neuronal determina el rendimiento de la misma. Lo cual tiene sentido, porque la optimización busca minimizar la función de pérdida de una red LSTM y si no, la arquitectura no es la adecuada y el error de pérdida será elevado.

El algoritmo PSO, es un algoritmo de búsqueda equilibrada entre exploración e intensificación, porque la dispersión de sus soluciones va bajando de manera suave en cada iteración hasta converger. FPA es un algoritmo más exploratorio que intensivo, porque la dispersión de sus soluciones se mantiene en cada iteración y en algunos casos aumenta en lugar de disminuir. Mientras que LNFE, es un algoritmo con búsqueda más intensivo que exploratoria, porque la dispersión de sus soluciones es elevada en la primera iteración y disminuye hasta converger en pocas iteraciones.

La hipótesis de la presente tesis se confirma con las acciones de Google y Nike, no obstante en el estado del arte se han pronosticado otros activos financieros con redes LSTM, desde acciones financieras como: IBM, ABX, entre otras; hasta índices como: S&P 500, Dow Jones Industrial Average, NASDAQ, Shanghai Stock Exchange, entre otros. Por ello es po-

sible que el método de optimización de hiperparámetros para que una red LSTM realice pronósticos financieros, pueda generalizarse para diferentes activos financieros bajo la premisa que LSTM puede realizar pronósticos financieros para diferentes activos y gran parte activos financieros que cotizan en diferentes mercados financieros como S&P500. Considerando además que los pronósticos se han hecho a índices de bolsa, los cuales ponderan las acciones que cotizan en la bolsa lo que implica que de manera implica se consideran otras acciones financieras.

Las redes LSTM si bien son ampliamente utilizadas, en los trabajos más recientes hay una tendencia de hibridar redes las LSTM con otros algoritmos como métodos optimización de hiperparámetros; también se hibridan con otros modelos como ARIMA y CNN. La litera muestra que hibridar redes da resultados superiores a las redes no híbridas.

Otra tendencia es el de desarrollar sistemas completos de comercio automatizado; en este tipo de sistemas por lo general se usan redes LSTM para realizar el pronóstico y/o señales de inversión. Sin embargo, existen otro tipo de propuestas como las *Graph* CNNs, las cuales pueden analizar varios aspectos de los mercados financieros a la vez; en trabajos futuros podría contrastarse las redes LSTM con las *Graph* CNNs.

Analizar la convergencia comparando la dispersión de la población de la primera iteración contra la última iteración y mediante gráficas de convergencia, permite una mejor entendimiento del comportamiento de los algoritmos metaheurísticos; el análisis de convergencia ayuda a seleccionar un algoritmo para un determinado problema.

El EDA avaluado muestra mayor convergencia en una menor cantidad de iteraciones con resultados satisfactorios, en trabajos futuros se pueden probar otros algoritmos de la misma clase como CMA-ES [49].

La metodología propuesta para optimización de hiperparámetros de una red LSTM utilizando metaheurísticas da resultados satisfactorios; sin embargo es necesario probar la metodología con diferentes acciones, fijando el número de épocas, utilizando un conjunto de validación [1] y reducir el conjunto de entrenamiento. Utilizar un conjunto de validación

[1] En ML, se utiliza *rolling cross-validation* como método de validación en series de tiempo [24, 42].

daría resultados más robustos.

El estado del arte muestra no solo la aplicación de optimización de hiperparámetros, sino también la selección de características, así como el uso de diferentes tipos de entrada, desde la serie de tiempo, indicadores técnicos, indicadores macroeconómicos, hasta análisis de sentimientos; en trabajos futuros se puede llevar a cabo el proceso completo de AutoML [38] considerando el preprocesamiento de datos y selección de características.

En el estado del arte se hibridan distintos modelos de DL, en trabajos futuros se puede llevar a cabo la selección de distintos modelos de DL; esto podría utilizarse para ver los efectos que tienen distintas arquitecturas de DL mediante el análisis de la población y su valores *fitness*.

Es necesario realizar el proceso de optimización de hiperparámetros con más acciones para ver si se repite el patrón de simplicidad en la arquitectura y verificar si se repite o existe similitud en las arquitecturas de red LSTM. En tales casos se puede utilizar LNFE, el cual converge en una menor cantidad de iteraciones y llamadas a función. Realizar la optimización de hiperparámetros con más metaheurísticas permitirá dar más formalidad a la hipótesis.

Es necesario realizar la optimización de hiperparámetros con distintas representaciones de variables, p. ej. una representación binaria y compararlo con la representación utilizada en este trabajo. Porque realizar la comparación permitira decir de manera concisa que una representación es mejor que otra para el problema de optimización de hiperparámetros, en el caso particular de la presente tesis o de otro tipo de problemas.

Es necesario realizar una comparación de la correlación de las variables (hiperparámetros) contra el rendimiento de la red para encontrar cuáles son las variables que tienen mayor influencia en el rendimiento de la red. La información de la correlación puede dar información que oriente la optimización de hiperparámetros, p. ej. para fijar límites de búsqueda en trabajos futuros o modificar algún algoritmo añadiendo funciones heurísticas para sesgar la búsqueda a una región prometedora. Para ello se pueden utilizar las poblaciones que se obtienen en cada iteración de las metaheurísticas.

Es necesario repetir la experimentación aumentando los espacios de búsqueda para hiperparámetros como número de estados ocultos, épocas y tamaño de la ventana, y comparar los resultados con los de la presente tesis; porque algunos valores de las soluciones llegaron a los límites de su espacio de búsqueda.

Es necesario llevar a cabo un estudio comparativo entre predicción de valores y predicción de tendencias de las acciones financieras porque aún existe una controversia entre cual de las predicciones es mejor, si la de valores o la de tendencias.

En trabajos futuros se puede experimentar con una función *fitness* (función de pérdida) la cual considere los rendimientos de inversión, por ejemplo el índice Sharpe *Cross-Entropy* [87].

La literatura muestra otros algoritmos para la optimización de hiperparámetros como DRL [23] y DL, v.gr. HELP [75]. En trabajos futuros se pueden probar estos algoritmos y contrastarlos con PSO, FPA, LNFE.

También es necesario hacer una comparación entre GRU y LSTM, porque en la literatura muestra resultados satisfactorios y en casos supera a la red LSTM con menor costo computacional; sin embargo, en los últimos años se ha dejado de utilizar sin razón alguna, por eso es necesario realizar una comparación entre estas dos redes.

Finalmente, es necesario realizar la optimización de hiperparámetros utilizando el algoritmo SEED, porque este es un objetivo que quedó pendiente en la presente investigación.

Apéndice A

Artículo COMIA

En el presente apéndice se presenta el artículo publicado en el Congreso Mexicano de Inteligencia Artificial en su edición XIV el cual fue realizado como parte de la investigación de tesis.

Optimización de hiperparámetros de una red Long Short Term Memory para pronósticos financieros

Francisco J. Pedroza-Castro, Alfonso Rojas-Domínguez*, Martín Carpio, Manuel Ornelas-Rodríguez, and Héctor Puga

Tecnológico Nacional de México/Instituto Tecnológico de León, 37290, León, Guanajuato, México.
✉ alfonso.rojas@gmail.com

Resumen El proceso de optimización de hiperparámetros de una red neuronal artificial, usualmente se lleva cabo de manera manual usando Grid Search o Random Search. En este artículo, estudiamos una metodología para optimizar hiperparámetros de una red Long Short Term Memory (LSTM) usando dos metaheurísticas bio-inspiradas, Particle Swarm Optimization, y Flower Pollination Algorithm y una Estimation of Distribution Algorithm (EDA), Low Number of Function Evaluation, para que la red realice un pronóstico financiero del precio de cierre del siguiente día de las acciones de Google y Nike. Los resultados muestran que las redes LSTM obtenidas mediante el proceso de optimización de hiperparámetros son más simples, resultando en menor tiempo de entrenamiento, y mayor rendimiento, que las redes no optimizadas con metaheurísticas. Los errores de prueba obtenidos de las soluciones son de 10E-4 hasta 10E-6. Adicionalmente realizamos una comparación de las metaheurísticas, concluyendo que la EDA encuentra una solución esperada con menor cantidad de llamadas a función, lo que se traduce en menor costo computacional y tiempo de ejecución, con resultados satisfactorios.

Palabras clave: Long Short Term Memory, Estimation of Distribution Algorithm, Bio-inspired metaheuristics, Algorithm for a Low Number of Function Evaluations, Particle Swarm Optimization, Flower Pollination Algorithm, Pronóstico Financiero.

1. Introducción

Inversionistas, academia e investigadores, del campo de pronósticos financieros se han visto interesados por el campo de la inteligencia computacional, ya que las investigaciones muestran ventajas significativas en contraste con los métodos clásicos, como análisis técnico, análisis fundamental, y métodos estadísticos [12, 13, 21].

Las investigaciones en inteligencia computacional aplicadas a pronósticos financieros, muestran que el uso de redes neuronales híbridas con algoritmos evolutivos, obtienen un mayor rendimiento en comparación con las redes no-híbridas. Dichas hibridaciones van desde la optimización de parámetros hasta hiperparámetros [12, 13, 21]. Este último es un campo del Automated Machine Learning, conocido como Hyperparameter Optimization (HPO). Por lo que en este artículo nos referimos a la hibridación para ajuste de hiperparámetros como HPO [11].

De hecho, el proceso de ajuste de hiperparámetros de una Red Neuronal Artificial, comúnmente se lleva a cabo usando Random Search o Grid Search [15]. HPO busca realizar este proceso de manera automatizada utilizando, comúnmente, diferentes técnicas como Algoritmos Genéticos y Métodos Bayesianos [11, 15].

El presente artículo, muestra una metodología propia para llevar a cabo la optimización de hiperparámetros, probando 3 metaheurísticas: 2 algoritmos bio-inspirados, Particle Swarm Optimization y Flower Pollination Algorithm, y 1 algoritmo perteneciente a las Estimation of Distribution Algorithm, Estimation of Distribution Algorithm Low Number of Function Evaluation (EDALNFE, en este artículo nos referiremos a él como LNFE).

Estos algoritmos se compararon y adicionalmente se contrastaron con una red LSTM que fue ajustada de manera manual por [16]. Los resultados obtenidos muestran que la optimización de

* Investigador CONACYT.

hiperparámetros con metaheurísticas diseñan una red más simple con mejores rendimientos en comparación de la hechas manualmente.

El trabajo está organizado como sigue: en la Sección 2, presentamos diferentes trabajos que se han utilizado para realizar pronósticos financieros utilizando redes neuronales artificiales, investigaciones que optimizan hiperparámetros en pronósticos financieros y de otros campos. En la Sección 3, se describe la teoría de la red LSTM y las metaheurísticas. En la Sección 4, exponemos la metodología propuesta para realizar la optimización de hiperparámetros y comparación de metaheurísticas. En la sección 5 se muestran los resultados experimentales y discusiones, y finalmente en la Sección 6 las conclusiones de nuestro trabajo.

2. Trabajos relacionados

Los investigadores en pronósticos financieros han encontrado que el uso de inteligencia computacional genera resultados satisfactorios que superan a los métodos clásicos [4, 12–14, 16, 21]. Estos métodos van desde un Algoritmo Genético (AG) [4, 7], Lógica Difusa [4, 6], Sistemas Expertos [4], y Redes Neuronales Artificiales (RNAs) [18], entre otros. Las RNAs se han utilizado mayormente, especialmente la red LSTM, que de acuerdo a la literatura, son las que mayor rendimiento tienen [12–14, 19, 21, 24]

En la literatura para pronósticos financieros usando redes neuronales, del periodo de 2017 a 2019, cerca del 51 % de los artículos revisados utilizan algún tipo de Red Neuronal Recurrente (RNR), de ese porcentaje, el 61 % utilizan redes LSTM, la razón es la misma que se menciona en el párrafo anterior [21], además de ser relativamente fácil de aplicar.

Kumar *et. al* (2021) optimizan hiperparámetros de una red LSTM para pronósticos de tendencias de activos financieros, utiliza PSO y FPA. Su trabajo lo comparan con un modelo sin uso de metaheurísticas, los resultados muestran que utilizar metaheurísticas para la optimización, genera resultados superiores a los casos donde no se usa metaheurísticas [13].

En otro trabajo se realiza un sistema de comercio automatizado usando un AG y un RNA, obteniendo 72.5 % de precisión y 23.3 % de Rendimiento Anual Neto. En su trabajo encontraron que existen tres factores importantes que afectan el rendimiento del pronóstico: Pre-procesamiento de datos adecuado y presentación de los datos, estrategias óptimas de comercio, y la estructura del modelo a pronosticar [7].

En otros trabajos se realiza un sistema de comercio utilizando una red LSTM [22], su red la llevan a un sistema simulado considerando los costos de transacción, y diferentes versiones de la red LSTM, con lo cual llegan a tener rendimientos de hasta 228.94 % en el mejor de los casos.

Sezer *et. al* (2020), además, identifica que en diversos trabajos de pronósticos financieros, con redes neuronales artificiales, el ajuste de hiperparámetros es de vital importancia para un adecuado rendimiento de la red al realizar el pronóstico de activos financieros. Consideran que encontrar los mejores hiperparámetros de las redes neuronales es un problema significativo.

De hecho HPO optimizan los hiperparámetros usando Métodos Bayesianos (MB) como Sequential Model-Based Global Optimization (SMBGO), The Gaussian Process Approach (GPA), Tree-structured Parzen Estimator Approach (TSPEA) [2,3], entre otros algoritmos. Por ejemplo, se ha realizado optimización de hiperparámetros en campos diferentes a pronósticos financieros, donde se usa GA y MB [11, 15], con resultados satisfactorios. También se ha hecho uso de Aprendizaje por Refuerzo Profundo [5], pero en este caso los resultados superan a GA y MB, incluso, en trabajos relacionados, se ha hecho uso de una red LSTM para la optimización, teniendo resultados satisfactorios y en un tiempo reducido [13].

Otro trabajo de optimización de hiperparámetros de una red tipo LSTM fue el de [1], quien consideran que es una área de oportunidad para las metaheurísticas con el objeto de buscar los hiperparámetros de la red mencionada. Ellos proponen utilizar Grey Wolf Optimizer (GWO), con lo cual encuentran que una red LSTM de arquitectura simple, la cual puede tener el mismo rendimiento que una red más compleja. Atribuyéndolo a la adecuada combinación de hiperpárametros.

En este trabajo, siguiendo la línea marcada por [12], utilizaremos algoritmos evolutivos para la optimización de hiperparámetros, los cuales son dos bio-inspirados ya utilizados en la literatura, PSO, y FPA, los cuales han tenido resultados satisfactorios al optimizar los hiperparámetros de

una red LSTMA; adicionalmente utilizaremos una EDA, EDALNFE que de acuerdo a su diseño hace una menor cantidad de llamadas a función, lo cual puede ser una ventaja, ya que las otras metaheurísticas tiene que hacer llamadas a función a toda la población en cada iteración; en el presente caso, que se usa RNAs, la llamada a función implica entrenar la red en cada iteración, generando un alto costo computacional. El proceso de optimización se seguirá utilizando una metodología propuesta por nosotros. Al final contrastamos cada uno de los algoritmos con una red LSTM que se ajustó sin metaheurísticas por [16].

3. Teoría

3.1. Long Short Term Memory (LSTM)

La red LSTM es un tipo de RNR que puede trabajar con hasta 1000 retrasos en el tiempo, evitando los problemas del *exploding-gradient* y el *vanishing-gradient* [20]. La red trabaja simulando compuertas lógicas que permiten eliminar o agregar información a un *Cell State* (5) como se describe a continuación[1]: primero Forget gate (2) decide que información se olvida o se mantiene; después, la Input gate (1) y Candidate gate (4) deciden que información se agrega; finalmente se genera el Hidden state (6) usando Output gate (3) y la Cell state (5) [10]. Las ecuaciones para llevar dicho proceso se muestran a continuación. En la Figura 1 se muestra el funcionamiento de la red LSMT.

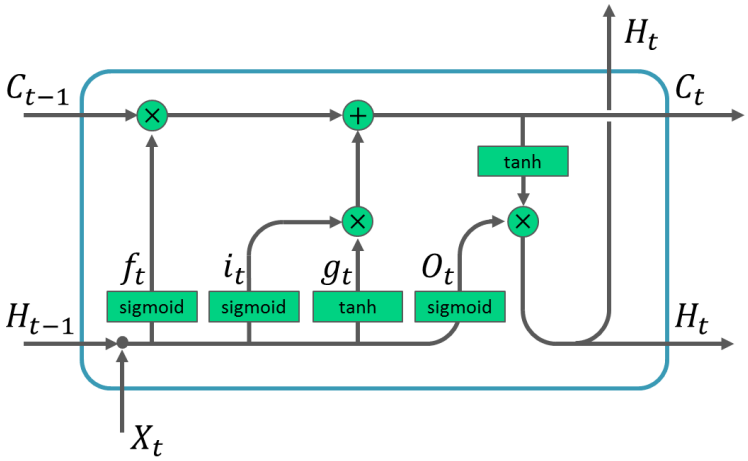


Figura 1. Estructura de una celda LSTM (Elaboración propia).

$$\textbf{Input gate:} \quad i_t = \text{sig}(x_t U_i + H_{t-1} W_i + b_i), \quad i_t \in \mathbb{R}^{d_H} \tag{1}$$

$$\textbf{Forget gate:} \quad f_t = \text{sig}(x_t U_f + H_{t-1} W_f + b_f), \quad f_t \in \mathbb{R}^{d_H} \tag{2}$$

$$\textbf{Output gate:} \quad O_t = \text{sig}(x_t U_o + H_{t-1} W_o + b_o), \quad O_t \in \mathbb{R}^{d_H} \tag{3}$$

$$\textbf{Candidate gate:} \quad g_t = \tanh(x_t U_g + H_{t-1} W_g + b_g), \quad g_t \in \mathbb{R}^{d_H} \tag{4}$$

$$\textbf{Cell state:} \quad C_t = f_t \otimes C_{t-1} + i_t \otimes g_t, \quad C_t \in \mathbb{R}^{d_H} \tag{5}$$

$$\textbf{Hidden state:} \quad H_t = O_t \otimes \tanh(C_t) \tag{6}$$

$$\textbf{State:} \quad s_t = R_{LSTM}(s_{t-1}, x_t) = [c_t; H_t], \quad s_t \in \mathbb{R}^{2d_H} \tag{7}$$

donde U y $W \in \mathbb{R}^{d_x \times d_H}$ son los pesos, $x_t \in \mathbb{R}$. Si se utiliza tamaños de lote d_b, entonces: $s_t \in \mathbb{R}^{2d_H \times d_b}$, C_t, H_t, i_t, f_t, O_t, $g_t \in \mathbb{R}^{d_H \times d_b}$, y $b \in \mathbb{R}^{d_b \times d_H}$.

[1] En este trabajo, $\otimes$ representa *element-wise* o el producto Hadamard, sig$(\cdot)$ es la función sigmoide y $\tanh(\cdot)$ función tangente hiperbólica.

3.2. Particle Swarm Optimization (PSO)

Esta metaheurística se inspira en el comportamiento social de insectos, animales y humanos. El algoritmo inicia con una población aleatoria, donde cada partícula de la población se considera una solución potencial, las cuales se van desplazando en el espacio de búsqueda con el objeto de mejorar la mejor solución encontrada $\mathbf{g}^*$ en la iteración previa, usando la información de los vecinos $\mathbf{g}^*$ y su mejor posición $\mathbf{p}_i$ (en la que alguna vez estuvo) [9]. Este concepto se presenta en la regla de actualización (8).

$$\mathbf{v}_i^{New} = \chi(w\mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{g}^* - \mathbf{x}_i)) \tag{8}$$

donde $\mathbf{v}_i$ es la velocidad de la partícula i, $\mathbf{x}_i$ es la posición de la partícula i, w es la inercia o peso de inercia, $\mathbf{g}^*$ es la mejor posición global, $\varphi_1 = R_1 c_1$, $\varphi_2 = R_2 c_2$, donde c_1 es el coeficiente cognitivo, c_2 el coeficiente social, $R_1, R_2 \sim \mathcal{U} \in [0, 1]$, y χ el coeficiente de constricción dado por la Ec. (9). La Ec. (10) es la regla actualización de la posición de la partícula. El pseudocódigo del PSO se muestra en Algoritmo 1.

$$\chi = \frac{2}{\left|2 - \varphi' - \sqrt{\varphi'^2 - 4\varphi'}\right|}, \quad \varphi' = c_1 + c_2, \varphi' > 4 \tag{9}$$

$$\mathbf{x}_i^{New} = \mathbf{x}_i + \mathbf{v}_i^{New} \tag{10}$$

Algoritmo 1 PSO, asumiendo minimización

Generación de población inicial de manera aleatoria $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$.
Encontrar la mejor solución $\mathbf{g}^*$ en X.
while *Criterio de paro* no se cumpla, **do**
    for partícula $\mathbf{x}_i$ **do**
        if $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ **then**
            $\mathbf{p}_i \leftarrow \mathbf{x}_i$; actualizar la mejor posición de la partícula i.
        end if
        Identificar la mejor posición de la iteración $\mathbf{g}^*$.
        Actualizar velocidad $\mathbf{v}_i$ vía (8).
        Actualizar posición $\mathbf{x}_i$ vía (10).
    end for
end while
return La mejor solución encontrada $\mathbf{g}^*$.

3.3. Flower Pollination Algorithm (FPA)

El FPA es un algoritmo inspirado en el proceso de polinización de las plantas bajo la óptica del proceso de optimización [25]. Tanto FPA como PSO son algoritmos bio-inspirados, los cuales son similares al seguir un proceso de generación de población inicial y mover las soluciones en el espacio de búsqueda, no obstante FPA añade ruido usando una distribución Lévy para explorar, mientras que PSO solo usa números aleatorios uniformes. El FPA usa dos conceptos de manera general, polinización global y polinización local, para realizar la búsqueda. Las Ecs. (11) a (13) representan dichas ideas. El pseudocódigo se muestra en el Algoritmo 2.

Algoritmo 2 FPA, asumiendo minimización

Crear población inicial de manera aleatoria $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$.
Encontrar la mejor solución $\mathbf{g}^*$.
Define el sego de polinizavión $\rho \in [0, 1]$.
while *Criterio de paro* no es encontrado, **do**
    for por cada flor/gameto $\mathbf{x}_i$ ($i = 1$ to n) **do**
        if $rand < \rho$ **then** ▷ Donde $rand \in [0, 1]$.
            Crea números aleatorios por cada variable de la solución usando el vuelo de Lévy.
            Crea solución temporal $\mathbf{x}'$ con la polinización global Ec. (11).
        else
            Crea un número aleatorio ϵ de manera uniforme ($\mathcal{U} \in [0, 1]$) por cada variable.
            Crea solución temporal $\mathbf{x}'$ con la polinización local Ec. (14).
        end if
        if $f(\mathbf{x}') < f(\mathbf{x}_i)$ **then**
            $\mathbf{x}_i \leftarrow \mathbf{x}'$
        end if
    end for
    Identifica la mejor solución $\mathbf{g}^*$ de la iteración.
end while
return La mejor solución encontrada $\mathbf{g}^*$.

La polinización global se define por (11), donde $\mathbf{x}_i^t$ es una flor/gameto, $\mathbf{g}^*$ la mejor solución (flor/gameto) global, γ es el factor de escalamiento del vuelo Lévy, y $L(\lambda)$ es un número aleatorio obtenido mediante la distribución del vuelo de Lévy de acuerdo a la Ec. (12).

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \gamma L(\lambda) \otimes (g^* - \mathbf{x}_i^t) \tag{11}$$

$$L(\lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \cdot \frac{1}{s^{1+\lambda}}, \quad s \gg s_0 > 0, \quad s = \frac{U}{|V|^{1/\lambda}} \tag{12}$$

donde $V \sim \mathcal{N}(0, 1)$ y $U \sim \mathcal{N}(0, \sigma_{fpa}^2)$, con desviación estándar σ_{fpa} obtenida con la Ec. (13).

$$\sigma_{fpa}^2 = \left[\frac{\Gamma(1+\lambda)}{\lambda \Gamma((1+\lambda)/2)} \cdot \frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}} \right]^{1/\lambda} \tag{13}$$

La polinización local se define por (14), donde $\mathbf{x}_k^t$ y $\mathbf{x}_j^t$ son dos flores/gametos tomadas de manera aleatoria de la población. Y ϵ es un número aleatorio en el intervalo $[0, 1]$.

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \epsilon \otimes (\mathbf{x}_j^t - \mathbf{x}_k^t) \tag{14}$$

3.4. Algorithm for a Low Number of Function Evaluations (EDALNFE)

LNFE[2] pertenece a la familia de Estimation of Distribution Algorithms (EDAs). La idea de una EDA es iniciar con una población la cual se usa para crear una media y varianza que es utilizada para generar una nueva muestra poblacional mediante una distribución de probabilidad previamente definida.

La idea central del LNFE es crear un vector direccional con la correlación de Pearson, para crear una nueva población. La ventaja de este algoritmo es que hace una menor cantidad de llamadas a función [17]. El pseudocódigo se muestra en el Algoritmo 3[3].

Este algoritmo inicia con una población, cada solución es evaluada y se le asigna una probabilidad de acuerdo a su fitness. La mejor solución y su fintess, X_{elite}, F_{elite}, se usa para generar una media y varianza (ver [23]). Si el fitness no mejora después de n iteraciones la población se reinicia

[2] En este articulo nos referimos a EDALNFE como LNFE.
[3] Este algoritmo trabaja en el espacio $[0, 1]$, lo cual separa al problema del algoritmo.

preservando la mejor solución. La nueva población se genera mediante dos distribuciones (ver línea 8 en Algoritmo 3), una distribución normal multivariada y una distribución sobre el vector dirigido a una dirección prometedora. En cada iteración se verifica que el vector proyección obtenido por la correlación de Pearson mejore al mejor fitness, de lo contrario se reinicia la población. Este proceso se repite hasta cumplir con el criterio de paro (para más detalles ver [17]).

Algoritmo 3 Algorithm for a Low Number of Function Evaluations (EDALNFE)

1: $N_p \leftarrow 8D + 2$; $C_{elite} \leftarrow 0; C_{resets} \leftarrow 0$
2: $X \leftarrow \text{initialize}(d, N_p)$ ▷ Inicia población de manera aleatoria X.
3: $F \leftarrow \text{evaluate}(X, f(\cdot))$ ▷ Donde $f(\cdot)$ es la función fitness.
4: $[ibest, ps] \leftarrow \text{selection}(F)$
5: $[X_{elite}, F_{elite}] \leftarrow [X_{ibest(1)}, F_{ibest(1)}]$
6: $f_{reset} \leftarrow F_{elite(1)}$ ▷ Guardar la mejor solución.
7: **while** *Criterio de paro* ($var_c > var_{min}$) no se cumple, **do**
8: $\left[\hat{X}, U\right] \leftarrow \text{generateNewIndividuals}(X, F, X_{elite}, ps, ibest, U \leftarrow rand(D))$
9: $\hat{F} \leftarrow \text{evaluate}(\hat{X}, f(\cdot))$ ▷ Evaluar la nueva población.
10: **if** $\left|F_{elite} - F_{ibest(1)}\right| > th_{elite}$ **then** ▷ Si no mejora
11: $C_{elite} \leftarrow 0$ ▷ Reiniciar población.
12: $[X_{elite}, F_{elite}] \leftarrow \left[X_{ibest(1)}, F_{ibest(1)}\right]$ ▷ Actualizar la mejor solución elite.
13: **else**
14: $C_{elite} \leftarrow C_{elite} + 1$ ▷ Incrementar el contador de elite.
15: **end if**
16: **if** $C_{elite} = MaxC_{elite}$ **then** ▷ Si la petición de la mejor solución se repite en un limite, reiniciar.
17: $C_{resets} \leftarrow C_{resets} + 1$ ▷ Incrementar el contador de reinicio.
18: $X_{elite} \leftarrow X_{ibest(1,\ldots,C_{resets})}$
19: $F_{elite} \leftarrow F_{ibest(1,\ldots,C_{resets})}$
20: $X \leftarrow \text{reinitialize}(D, Np - 1, C_{resets}, X_{elite})$ ▷ Reiniciar población.
21: $F \leftarrow \text{evaluate}(X, f(\cdot))$ ▷ Evaluar la nueva población.
22: $[X, F] \leftarrow [[X, X_{elite}], [F, F_{elite}]]$ ▷ Agregar la mejor solución elite a la nueva población.
23: $[ibest, F_{elite}] \leftarrow \text{selection}(F)$
24: $[X_{ibest}, ps] \leftarrow \left[X_{ibset(1)}, F_{ibest(1)}\right]$
25: $C_{elite} \leftarrow 0$
26: **if** $F_{elite(1)} = f_{reset}$ or $C_{resets} = MaxC_{resets}$ **then** ▷ Si no mejora, detener el algoritmo.
27: break
28: **else**
29: $f_{reset} \leftarrow F_{elite(1)}$ ▷ Si la solución mejora actualizarla.
30: **end if**
31: **end if**
32: $[X, F] \leftarrow [X(ibest), F(ibest)]$ ▷ Ordenar las soluciones de acuerdo a su valor fitness.
33: $[X, F] \leftarrow [X(1, \ldots, Np/2), F(1, \ldots, Np/2)]$ ▷ Mantener la mitad de la población con mejor fitness.
34: $[X, F] \leftarrow \left[\left[X, \hat{X}\right], \left[F, \hat{F}\right]\right]$ ▷ Añadir nuevas soluciones a la población.
35: **end while**
36: **return** La mejor solución encontrada, $[x_{elite}, f_{elite}]$

4. Metodología

Para realizar la optimización de hiperparámetros de la red LSTM y pueda así pronosticar los precios de cierre del siguiente día de las acciones de Google y Nike. Utilizamos 2 algoritmos bio-inspirados (PSO y FPA) y una EDA (LNFE). Para realizar el entrenamiento de las redes LSTM usamos el optimizador Adam y, como función de pérdida el error cuadrático medio. Seleccionamos

el 65 % de la serie de tiempo del precio de cierre para el conjunto de entrenamiento y 35 % para el conjunto de prueba. Los datos se obtuvieron de la web Yahoo Finance[4]. Los periodos seleccionados para las acciones de Google fuero de 24/08/2004 a 21/01/2022; para el caso de Nike el periodo fue de 02/12/1980 a 24/01/2022. Estas acciones fueron las mismas que usaron [16]; se replicó el trabajo de estos últimos para compararlo con las metaheurísticas. Es menester resaltar que [16] en su trabajo únicamente muestra 4 configuraciones de red LSTM, cambiando únicamente el número de épocas, sin mostrar otras configuraciones con demás hiperparámetros. Al modelo de [16] nos referiremos como Optimización sin Metaheurística (OSM).

En cada experimento se decidió normalizar los datos usando $min - max$ en el intervalo entre $[0, 1]$.

Los hiperparámetros que se optimizaron se muestran en la primera columna del Cuadro 2, y su símbolo en la segunda columna. Los hiperparámetros Cl_s, H_s, α, W_s, B_s, y E_s, fueron seleccionados como lo hace [13] (ver Cuadro 2). Los parámetros de control del PSO fueron seleccionados como sugiere [9] en el caso de FPA se seleccionaron como sugiere [25]. En ambos casos (PSO y FPA) el número de soluciones iniciales se seleccionan como sugiere [13]. En el caso de LNFE la población fue de 26, ya que el diseño del algoritmo sólo así lo permite. Estos parámetros son mostrados en el Cuadro 1.

Cuadro 1. Parámetros usados en metaheuristicas.

Parámetro	**Valor**
PSO	
Coeficiente cognitivo, c_1	2.05
Coeficiente social, c_2	2.05
Peso de inercia, w	0.8
Número de partículas (Soluciones)	25 (Usado en [12])
Número de iteraciones	16
FPA	
Sesgo de polinización, ρ	0.8
Factor de escalamiento de vuelo Lévy, γ	0.1
Número Flores/gametos (Soluciones)	25 (Usado en [12])
Número de iteraciones	16
LNFE	
Número de soluciones	26

Tanto PSO como FPA trabajan sobre el espacio de los números continuos. Por lo que decidimos mapear los valores al espacio de los discretos redondeando los números como lo sugiere [8] para solucionar problemas de variables mixtas. De tal modo las soluciones tienen valores de acuerdo al Cuadro 2.

El espacio de tamaño de lote se seleccionó como se usa típicamente en RNAs. Las épocas se seleccionaron de acuerdo a experimentos previos y el trabajo de [16]. El tamaño de la ventana se usa de acuerdo a las investigaciones realizadas por [21]. El número de estados ocultos se seleccionó para reducir el espacio de búsqueda, no obstante, se incluye la solución de [16]. El número de celdas LSTM se selecciona como sugiere [13] (ver Cuadro 2).

Para realizar una comparación justa de las metaheurísticas decidimos utilizar un mínimo de 400 llamadas a función, dicho número se fijó por experimentos previos. Para entrenar la red se consideró añadir condiciones de paro para disminuir el costo computacional, consistiendo detener el entrenamiento de no mejora la pérdida de la red después de 5 épocas. La metodología de optimización de hiperparámetros se puede ver en la Figura 2.

Por otro lado, también se decidió utilizar la misma semilla de generación de número aleatorios de las soluciones iniciales para comparar a los algoritmos bajo condiciones similares.

[4] https://finance.yahoo.com/

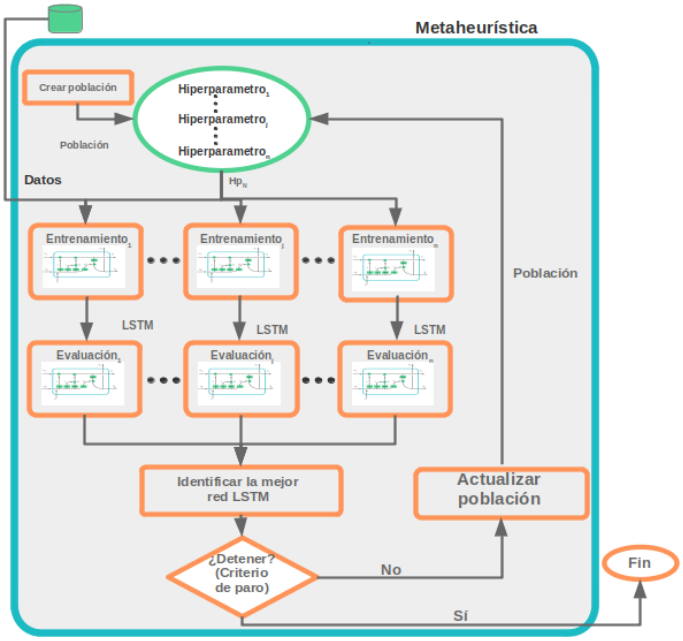


Figura 2. Metodología de optimización. Hp_N representa el N hiperparámetro. (Elaboración propia)

Una vez obtenidas las soluciones por cada metaheurística, se realizaron 31 experimentos de entrenamiento de cada solución para obtener evidencia estadística de cada solución.

Por último, para revisar el fitness de convergencia, realizamos gráficas de convergencia, así como comparar sus desviaciones estándar de la primera generación a la última iteración de cada metaheurística.

Cuadro 2. Hiperparámetros y espacio de búsqueda.

Hiperparámetro	**Símbolo**	**Espacio de búsqueda**	**Límite**
Celda LSTM	Cl_s	$\{ 1, 2, 3, 4, 5, 6, 7, 8\}$	$[1, 8]$
Estado oculto	H_s	$\{ 1, 20, 50, 80, 96, 110, 150\}$	$[1, 7]$
Tamaño de la ventana	W_s	$\{ 20, 40, 60, 80, 90, 100\}$	$[1, 6]$
Factor de aprendizaje	α	$\{ 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$	$[1, 5]$
Tamaño de lote	B_s	$\{ 8, 16, 32, 64\}$	$[1, 4]$
Número de épocas	E_s	$\{ 12, 25, 50, 100, 150\}$	$[1, 5]$

5. Resultados experimentales y discusiones

Los experimentos se realizaron en una computadora con procesador RYZEN 5 5600x, una tarjeta gráfica GPU-GTX-1050TI, y RAM de 16GB. Se utilizó el lenguaje Python, usando principalmente NumPy, matplotlib y Keras de TensorFlow. El tiempo de ejecución fue de entre 4 y 7 horas tanto para PSO, FPA y LNFE, para cada uno de los conjuntos de datos (Google y Nike).

Los hiperparámetros encontrados por cada una de las metaheuristicas se encuentra en el Cuadro 3.

Se analizó la convergencia de cada uno de los algoritmos. En la Figura 3 se muestran algunas gráficas de convergencia ya que de agregar todas las imágenes el número de páginas sería mayor a lo requerido. Adicionalmente en el Cuadro 4 se muestran las diferencias de las dispersión de la primera población y la última población de cada una de las variables, de cada algoritmos y de

cada acción [5]. En el Cuadro 4 se aprecia que en PSO la convergencia en cada variable disminuyó a excepción de FPA, que incluso en algunos casos aumenta la dispersión. En apariencia PSO muestra mejor convergencia que LNFE, no obstante como se aprecia en la Figura 4 y 3 en 400 llamadas a función PSO llegó a la solución mostrada en el Cuadro 3, no obstante LNFE, llegó a la solución en una menor cantidad de iteraciones y con menor cantidad de llamadas a función.

En la última iteración LNFE tiene mayor dispersión a comparación de PSO, empero el algoritmo esta diseñado para renovar su población si se repite el fitness después de n iteraciones, por lo que explicaría una mayor dispersión que PSO en su población final (ver Cuadro 4). Por otro lado, FPA no disminuyó en gran medida su dispersión en la población final, la cual se mantuvo en cada iteración como se muestra en la Figura 3 y el Cuadro 4, incluso en algunos casos aumentó su dispersión.

En el Cuadro 3 se muestran las soluciones encontradas, así como los resultados de 31 experimentos de entrenamiento de cada una de las redes. Como se aprecia en las soluciones encontradas por las metaheurísticas, la arquitectura de la red LSTM es similar en cada uno de los casos a excepción de LSTM-PSO-Nike[6]. Empero, cada solución es más simple que las redes LSTM-OSM-Google y LSTM-OSM-Nike, inclusive la media de la pérdida es menor en todos los casos. En el caso de LSTM-PSO-Google, se muestra que es el mejor fitness entre LSTM-FPA-Google, LSTM-LNFE-Google y LSTM-OSM-Google, no obstante la media de los 31 experimentos muestra que LSTM-LNFE-Google tiene mejor rendimiento, mientras que en el caso de LSTM-FPA-Nike y LSTM-LNFE-Nike muestra un mejor rendimiento que LSTM-PSO-Nike, con fitness similar como se aprecia en la Figura 6. Empero LNFE, encontró la solución con una menor cantidad de llamadas a función como se ven Figura 3.

Cuadro 3. LSTM optimizada para realizar el pronóstico de las acciones.

	Google				**Nike**			
Hiperparámetro	**PSO**	**FPA**	**LNFE**	**OSM**	**PSO**	**FPA**	**LNFE**	**OSM**
Cl_s	1	1	1	4	1	1	1	4
H_s	150	150	150	96	110	150	150	96
W_s	60	80	40	50	90	100	40	50
α	10^{-4}	10^{-3}	10^{-4}	10^{-2}	10^{-4}	10^{-4}	10^{-4}	10^{-2}
B_s	16	32	8	16	16	16	8	16
E_s	50	150	100	100	50	150	150	100
Datos de solución	**PSO**	**FPA**	**LNFE**	**OSM**	**PSO**	**FPA**	**LNFE**	**OSM**
Llamadas a función:	400	400	401	-	400	400	402	-
Fitness:	3.18E-5	3.41E-5	3.32E-5	-	5.25E-6	4.23E-6	1.24E-7	-
Media de pérdida:	3.53E-5	2.91E-5	2.91E-5	1.86E-3	5.35E-6	3.96E-6	3.97E-6	1.02E-3
Media de prueba:	2.60E-4	3.28E-4	1.49E-4	2.22E-3	1.78E-4	1.04E-4	1.08E-4	2.97E-3
Desviación estándar de pérdida:	2.10E-6	1.58E-6	1.04E-6	7.80E-5	1.93E-7	9.13E-8	1.51E-7	5.61E-5
Desviación estándar de prueba:	1.29E-4	1.88E-4	6.47E-5	2.01E-4	6.01E-5	4.05E-5	2.38E-5	5.48E-4

[5] Las soluciones fueron escaladas en el intervalo $[0, 1]$ para poder hacer una comparación relativa.

[6] Cuando nos referimos a LSTM-PSO-Nike significa, red LSTM optimizada con PSO para pronosticar las acciones de Nike. Del mismo modo para la combinación de las diferentes siglas.

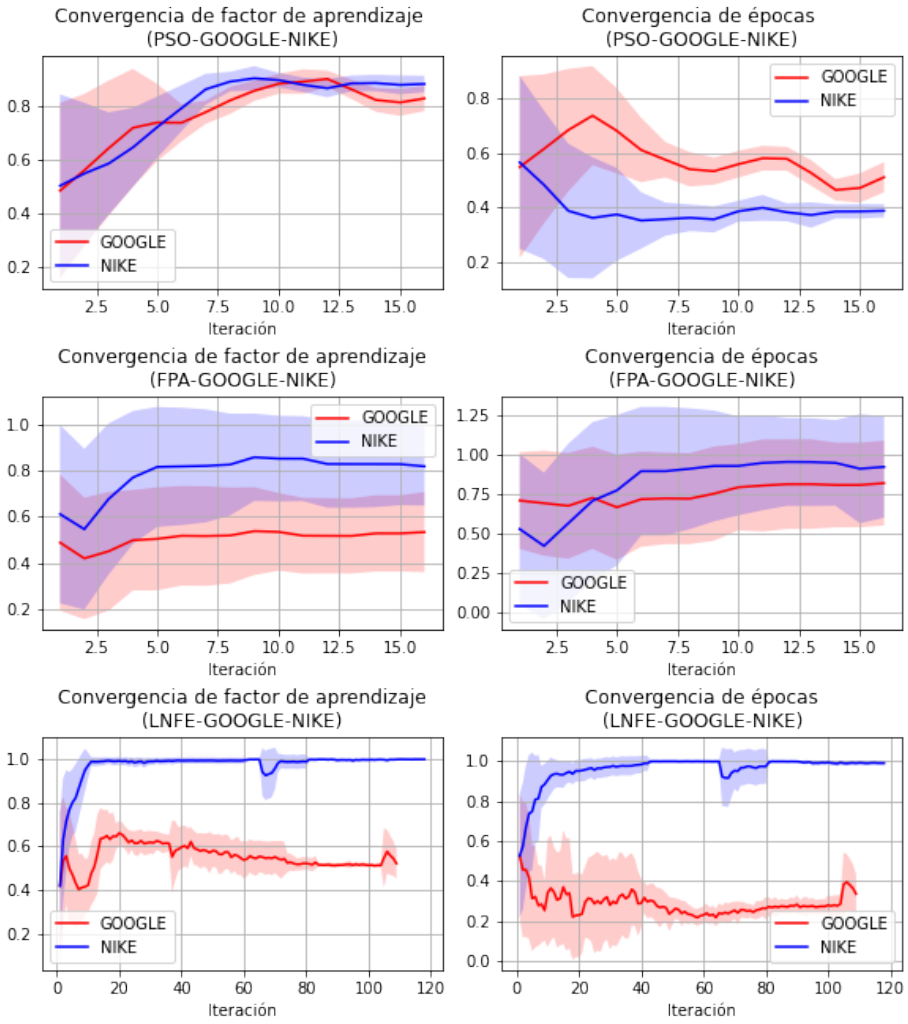


Figura 3. Convergencia del factor de aprendizaje y número de épocas. Las gráficas están escaladas en el intervalo $[0, 1]$.

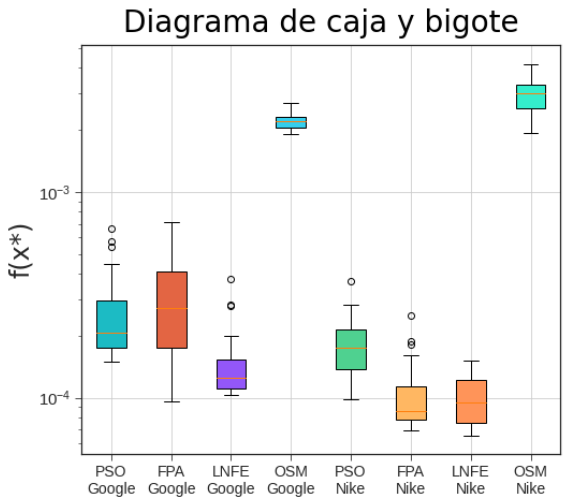


Figura 6. Diagrama de caja y bigote de prueba de 31 experimentos.

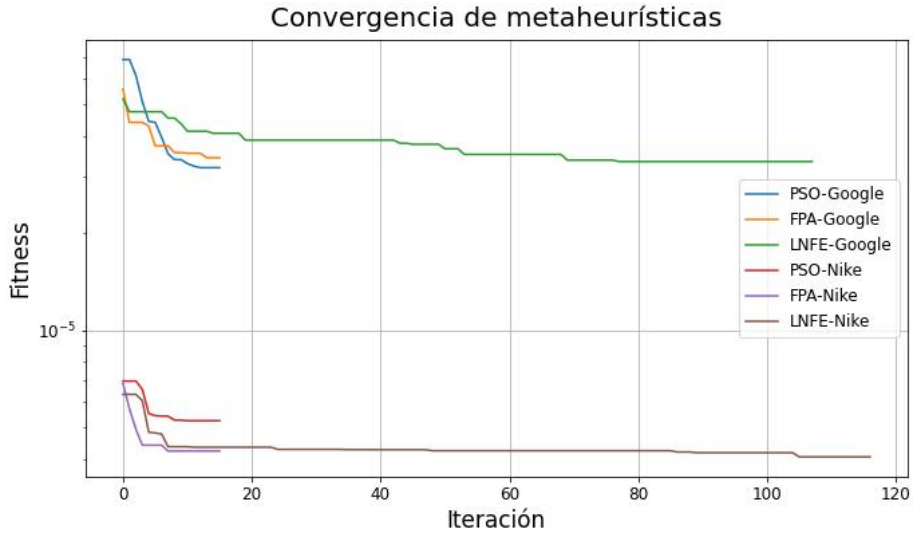


Figura 4. Convergencia (fitness vs. Iteración) de la optimización de hiperparámetros de la red LSTM.

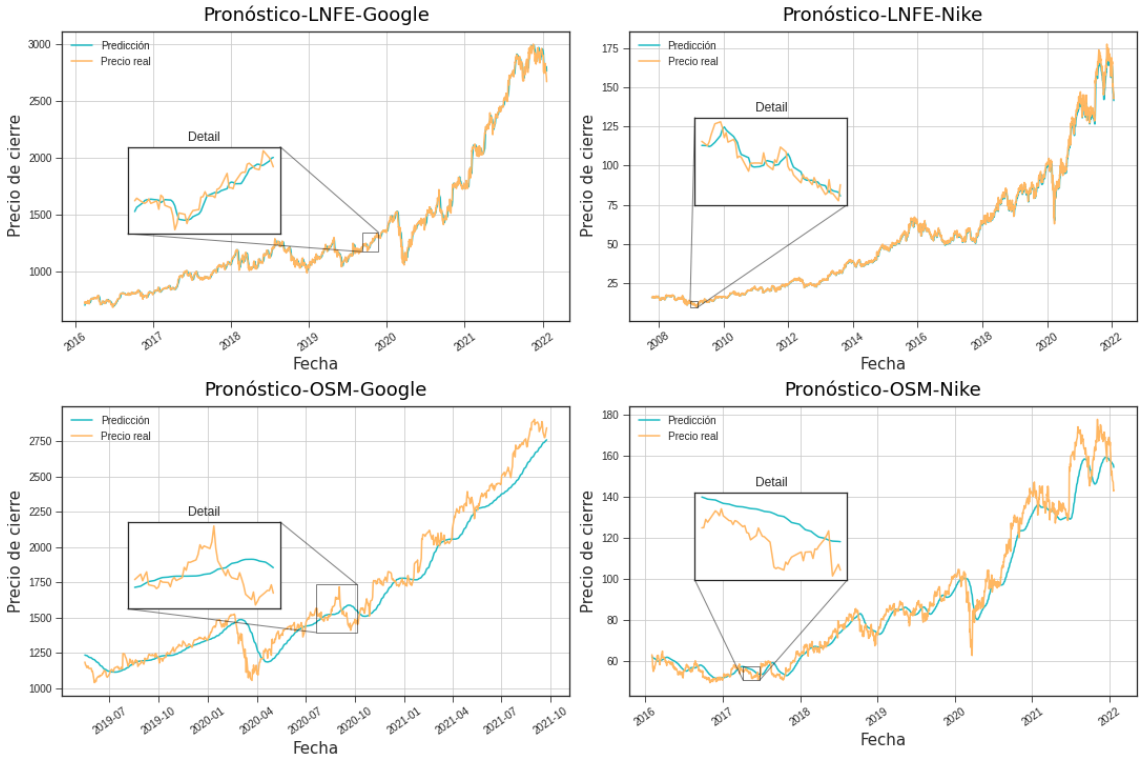


Figura 5. Pronóstico promedio de 31 experimentos de LNFE y OSM.

Cuadro 4. Diferencia de desviación estándar entre la primera y última generación de cada variable.

	PSO		FPA		LNFE	
Hiperparámetro	**Google**	**Nike**	**Google**	**Nike**	**Google**	**Nike**
Cl_s	0.23	0.26	0.19	0.13	0.26	0.23
H_s	0.27	0.25	0.32	0.34	0.29	0.28
W_s	0.29	0.26	0.10	0.09	0.25	0.23
α	0.31	0.28	0.12	0.20	0.27	0.33
B_s	0.23	0.20	−0.03	−0.03	0.17	0.18
E_s	0.29	0.27	0.04	0.12	0.21	0.30

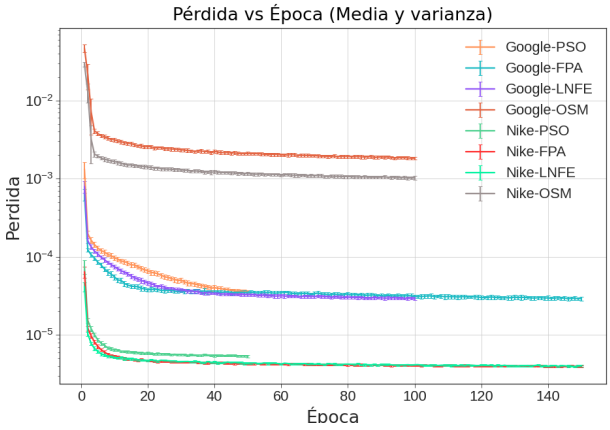


Figura 7. Media y varianza de 31 experimentos de entrenamiento de la red optimizada con PSO, FPA, LNFE y OSM.

Los resultados promedio de pronósticos de 31 experimentos de LNFE y OSM se muestra en la Figura 5. En dichas imágenes se aprecia la mejora sustancial de usar LNFE, lo cual sucede de igual manera con los demás pronósticos y se puede apreciar en el Cuadro 3. No se presentan los demás experimentos para evitar tener un número excesivo de imágenes. Dichos pronósticos se logran con una red LSTM más simple que las redes LSTM-OSM, similar a lo que refiere [1] quién asegura que encontrando las hiperparámetros adecuados se puede obtener la misma precisión a una red más compleja, e incluso se supera el rendimiento, como se aprecia en en este caso en el diagrama de caja y bigote de la Figura 6 y en la gráfica de entrenamiento de la Figura 7.

6. Conclusiones

Las topologías que retornaron cada una de las metaheurísticas fueron similares, a excepción de PSO-Nike, sin embargo esto da indicios de que se puede usar una misma arquitectura tanto para Nike como Google, y usar con menor cantidad de Cl_s, usando más estados ocultos, y diferentes configuraciones de hiperparámetros; de la misma manera en el caso de α que se repite en todos a excepción de FPA-Google. Mientras que LNFE retorna soluciones iguales tanto para Google como Nike, y convergencia claramente definida en una menor cantidad de llamadas a función logrando rendimientos similares a PSO-Google, lo cual sugiere continuar probando con más acciones y diferentes EDAs, por ejemplo, CMA-ES.

Por otro lado la convergencia de LNFE puede aumentar su dispersión en la población final, empero con soluciones consistentes, pues el mismo algoritmo guarda la mejor solución aunque se reinicie su población, por lo cual se puede correr el algoritmo con una menor cantidad de llamadas a función e iteraciones obteniendo resultados satisfactorios.

En referente a la metodología para optimizar la metaheurística, muestra resultados satisfactorios. Se sugiere probar la metodología fijando el número de E_s, y reduciendo el conjunto de

entrenamiento, siguiendo un proceso similar de optimización Random Search, pero automatizado; de tal modo que se puede reducir el tiempo de búsqueda, y al final el resultado solo entrenarlo con un mayor número de E_s y con un conjunto de entrenamiento mayor.

Es menester puntualizar que mantener a los algoritmos trabajando en el espacio de los continuos y luego mapear cada solución a números enteros da resultados satisfactorios.

Se sugiere experimentar con un número mayor de acciones, para saber si la arquitectura se repite, quizás en tales casos solo será necesario usar una LNFE, ya que implica menor costo computacional.

Por último, usar gráficas de convergencia permite un análisis más detallado del comportamiento de los algoritmos para saber que algoritmo usar para determinado caso, en el presente, pronósticos.

Agradecimientos: Este trabajo es parcialmente financiado por CONACYT de México a través: Beca de Posgrado No. 774627 (F. Pedroza) y Proyecto CÁTEDRAS-2598 (A. Rojas).

Referencias

1. Aufa, B.Z., Suyanto, S., Arifianto, A.: Hyperparameter setting of lstm-based language model using grey wolf optimizer. In: 2020 International Conference on Data Science and Its Applications (ICoDSA). pp. 1–5. IEEE (2020)
2. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. Advances in neural information processing systems **24** (2011)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research **13**(2) (2012)
4. Cavalcante, R.C., Brasileiro, R.C., Souza, V.L., Nobrega, J.P., Oliveira, A.L.: Computational intelligence and financial markets: A survey and future directions. Expert Systems with Applications **55**, 194–211 (2016)
5. Chen, S., Wu, J., Chen, X.: Deep reinforcement learning with model-based acceleration for hyperparameter optimization. In: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). pp. 170–177. IEEE (2019)
6. Chourmouziadis, K., Chatzoglou, P.D.: An intelligent short term stock trading fuzzy system for assisting investors in portfolio management. Expert Systems with Applications **43**, 298–311 (2016)
7. Evans, C., Pappas, K., Xhafa, F.: Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation. Mathematical and Computer Modelling **58**(5-6), 1249–1266 (2013)
8. Filomeno Coelho, R., Xiao, M., Guglielmetti, A., Herrera, M., Zhang, W.: Investigation of three genotypes for mixed variable evolutionary optimization. In: Advances in evolutionary and deterministic methods for design, optimization and control in engineering and sciences, pp. 309–319. Springer (2015)
9. Gendreau, M., Potvin, J.Y., et al.: Handbook of metaheuristics, thir edition, vol. 272. Springer (2019)
10. Goldberg, Y.: Neural network methods for natural language processing. Synthesis lectures on human language technologies **10**(1), 1–309 (2017)
11. Gorgolis, N., Hatzilygeroudis, I., Istenes, Z., Gyenne, L.G.: Hyperparameter optimization of lstm network models through genetic algorithm. In: 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA). pp. 1–4. IEEE (2019)
12. Kumar, G., Jain, S., Singh, U.P.: Stock market forecasting using computational intelligence: A survey. Archives of Computational Methods in Engineering **28**(3), 1069–1101 (2021)
13. Kumar, K., Haider, M., Uddin, T.: Enhanced prediction of intra-day stock market using metaheuristic optimization on RNN–LSTM network. New Generation Computing **39**(1), 231–272 (2021)
14. Li, A.W., Bastos, G.S.: Stock market forecasting using deep learning and technical analysis: a systematic review. IEEE access **8**, 185232–185242 (2020)
15. Li, W., Ng, W.W., Wang, T., Pelillo, M., Kwong, S.: Help: An lstm-based approach to hyperparameter exploration in neural network learning. Neurocomputing **442**, 161–172 (2021)
16. Moghar, A., Hamiche, M.: Stock market prediction using LSTM recurrent neural network. Procedia Computer Science **170**, 1168–1173 (2020)
17. Mora, K.M.F., Marín, J.A., Cerda, J., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., Olvera-López, J.A.: Pattern Recognition: 12th Mexican Conference, MCPR 2020, Morelia, Mexico, June 24–27, 2020, Proceedings, vol. 12088. Springer Nature (2020)
18. Ozbayoglu, A.M., Gudelek, M.U., Sezer, O.B.: Deep learning for financial applications: A survey. Applied Soft Computing **93**, 106384 (2020)

19. Qu, Y., Zhao, X.: Application of lstm neural network in forecasting foreign exchange price. In: Journal of Physics: Conference Series. vol. 1237, p. 042036. IOP Publishing (2019)
20. Schmidhuber, J., et al.: Long short-term memory. Neural Comput **9**(8), 1735–1780 (1997)
21. Sezer, O.B., Gudelek, M.U., Ozbayoglu, A.M.: Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied soft computing **90**, 106181 (2020)
22. Silva, T.R., Li, A.W., Pamplona, E.O.: Automated trading system for stock index using LSTM neural networks and risk management. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020)
23. Valdez-Peña, S.I., Hernández-Aguirre, A., Botello-Rionda, S.: Approximating the search distribution to the selection distribution in edas. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. pp. 461–468 (2009)
24. Wen, Y., Lin, P., Nie, X.: Research of stock price prediction based on pca-lstm model. In: IOP Conference Series: Materials Science and Engineering. vol. 790, p. 012109. IOP Publishing (2020)
25. Yang, X.S. (ed.): Nature-inspired computation and swarm intelligence: Algorithms, theory and applications. Academic Press (2020)

COMIA 2022

La Sociedad Mexicana de Inteligencia Artificial (SMIA) y la Escuela de Sistemas Biológicos e Innovación Tecnológica (ESBIT) de la Universidad Autónoma Benito Juárez de Oaxaca

OTORGAN ESTE CERTIFICADO A

Francisco Javier Pedroza Castro, Alfonso Rojas Domínguez, Juan Martin Carpio Valadez, Manuel Ornelas Rodriguez y Héctor J. Puga

por la presentación del artículo titulado

Hyperparameters optimization of Long-Short Term Memory networks for financial forecasting

en el XIV Congreso Mexicano de Inteligencia Artificial - COMIA 2022

Oaxaca, Oaxaca, México, del 25 al 27 de mayo de 2022

Dr. Ildar Batyrshin
Presidente SMIA

Dr. Hiram Ponce Espinosa
Presidente Comité de Programa

Dr. Gilberto Ochoa Ruíz
Presidente Comité de Programa

Dra. Iris I. Méndez Gurrola
Presidente Comité de Programa

Dra. Patricia Batres Mendoza
Comité Local COMIA

Apéndice B

Artículo ISCI

En el presente apéndice se presenta el artículo publicado en el International Seminar of Computational Intelligence ISCI 2022 el cual fue realizado como parte de la investigación de tesis.

Automated Machine Learning to improve stock-market forecasting using PSO and LSTM networks

Francisco J Pedroza-Castro, Alfonso Rojas-Domínguez [0000-0003-1818-4162]✉
Martín Carpio [0000-0002-1191-2676]

National Technological Institute of Mexico / Technological Institute of León,
37290, León, Guanajuato, México.
✉ alfonso.rojas@gmail.com

Abstract. The Automated Machine Learning (AutoML) process for feature selection, model creation and hyper-parameter optimization was performed. The AutoML process is applied to generate a machine learning model for the forecasting of Google's stock prices. For this reason the feature selection task is performed based on a set of 11 technical indicators, including the time series of the stock prices. Model selection is carried out between two models, a Long-Short Term Memory (LSTM) by itself or combined with Convolutional Neural Network (CNN). Hyper-parameter optimization is done on the learning rate, the windows size, the batch size, the epochs, the number of CNN filters and the CNN's kernel size. To perform the AutoML process, a bio-inspired population metaheuristic, Particle Swarm Optimization (PSO), is used. The experimental results show that the errors of the AutoML-generated solutions are in the order of 1E-2 to 9E-4 and that a CNN-LSTM network has better performance than an LSTM network by itself.

Keywords: Automated Machine Learning, Stock Forecasting, LSTM networks.

1 Introduction

Forecasting the time series of financial stocks is a challenge for investors, academics and researchers. Currently, these three have advanced the use of Computational Intelligence (CI) as a tool for financial forecasting from fuzzy logic [1][2], evolutionary algorithms (EA) [3], different Machine Learning (ML) algorithms such as Deep Learning (DL) [4], to hybridizing algorithms such as DL with EA [5]. In the present work, we hybridize a Long-Short Term Memory (LSTM) network with an EA, particularly the Particle Swarm Optimization (PSO) algorithm as the optimization engine to perform the Hyper-parameter Optimization (HPO) [6][7][8] and the feature selection tasks in a simplified version of Automated Machine Learning (AutoML).

Regarding the feature selection task, in the literature some authors report that the use of Convolutional Neural Networks (CNNs) to extract or generate new features from raw data provides better forecasting results [9], while other authors state that the use of Technical Indicators (TI) achieve better performance [10]. In the present work during the AutoML process the optimization metaheuristic selects whether to use

CNN or TI in a process directed toward improving the model performance in forecasting of Google's stocks.

In Section 2 we review works related to HPO and AutoML in financial forecasting. Section 3 describes the CNN, LSTM, and Particle Swarm Optimization. In Section 4 we describe our methodology to perform AutoML. The experimental results are reported in Section 5. Finally, our conclusions and directions for future work are offered in Section 6.

2 Related Work

According to our review of the literature, the most widely used DL models are the LSTM networks and their variations or hybridization with other techniques. A summary of the review is provided in Table 1. In less recent research the LSTM network is also the most commonly used [10].

Table 1. Models optimized as reported in the literature.

Author	Model	Hyper-parameters	Optimization Method
Chung & Shin [11]	LSTM	Number of cells LSTM (Cl_s), Hidden strates (H_s) and Window size (W_s)	Genetic Algorithm (GA)
Bhandari *et al.* [12]	LSTM	Cl_s, H_s, Epochs (E_s), Optimizer, learning rate (α), W_s	GA
Kumar *et al.* [13]	Bi-LSTM-ARIMA	H_s, α, E_s, dropout, E_s and feature selection	GA
Dang *et al.* [14]	LSTM-MEMD	H_s, W_s, α, Batch size (B_s) and E_s	Artificial Bee Colony (ABC)
Kumar *et al.* [7]	LSTM	Cl_s, H_s, W_s, E_s and feature selection	PSO and FPA
Dang *et al.* [15]	LightGBM-NSGA-II-SW	α , max leaf size, and min leaf size.	Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II)

The differences between the most recent research and previous works are: 1) That more recently the objective is not limited to perform forecasting only, but also to design automated trading systems. 2) That recent works do not use the LSTM networks by themselves, but hybridize these with other algorithms, for example to perform feature selection and hyper-parameter optimization (in an AutoML approach). 3) That not only is the LSTM network hybridized to perform some part of the AutoML process, but it is also combined with other models to generate forecasts or investment signals; for example, LSTM networks have been used with an ARIMA (AutoRegressive Integrated Moving Average) model to generate financial forecasts [13].

Regarding the Data Preprocessing and Feature Engineering task, there are different approaches to financial forecasting, using as input either technical indicators or the raw data with simple preprocessing, such as normalization. In those cases that use inputs without complex preprocessing, LSTM networks are used in conjunction with other types of networks, such as CNNs which generate the internal representations for

forecasting. In one work the reported performance was up to 6 orders of magnitude better than that of simple LSTM networks [16].

In another work an LSTM network with Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (LSTM-CEEMDAN) has been used for financial forecasting; this model was compared against a Support Vector Machine (SVM) and a simple LSTM, the results show that LSTM-CEEMDAN outperforms the SVM and the LSTM, with up to 3 and 10 orders of magnitude better respectively [17].

Chen *et al.* (2021) [18] reported that they used Graph CNN (GC-CNN) to analyze stock behavior and market information such as volatility and correlated stocks. The experimental results show that the model has an annual return of up to 33.42%, outperforming models such as CNN-LSTM that obtained an annual return of 6.65%. However, hyper-parameter and feature tuning is not considered, which would be like comparing models that are not in their best conditions.

The AutoML process to perform financial forecasting was used with population metaheuristic algorithms (p-metaheuristics) including GA, ABC, FPA, PSO and NSGA-II [19].

3 Theoretical background

The two machine learning models available to the AutoML process are briefly described in this section. The Optimization engine (PSO, a bioinspired metaheuristic) is also described.

3.1 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a type of multilayer network used to transform raw data (i.e. perform feature extraction) by means of convolution filters which constitute the weights or parameters of the network [20]. In ML parlance, the convolution operation is a simple linear combination of the input data where the coefficients of the combination are the weights of the network. Additionally, a CNN includes non-linear activation functions (typically the Rectified Linear Unit, ReLU), and pooling layers that enable dimensionality reduction. After the feature-extraction layers a classification or regression stage can be included, in the form of a Support Vector Machine or of traditional Fully-Connected layers. Fig. 1 illustrates the structure of a typical CNN for image classification.

The weights of a CNN, i.e. the convolution filters, are adjusted iteratively through application of gradient descent by means of the Stochastic Backpropagation (BP) algorithm, which computes the gradient of the network weights with respect to a loss function (a measure of the network's error) and performs the required adjustments to minimize the error of the network; this process is called *training* of the network.

To provide stability to the BP algorithm, the training instances (the data over which the network's error for a given iteration is computed) are provided to the network in small groups called mini-batches. The magnitude of the weight adjustments, which affects the training speed, is regulated by a hyper-parameter called *learning rate*.

As can be seen, the performance of a CNN depends on the training process and the architecture of the network. Thus, the hyper-parameters of a CNN are the number of training iterations, called *epochs*; the number of convolutional layers; the filters per layer; the learning rate; and the size of the mini-batches. In this work, all of these hyper-parameters are automatically adjusted by means of an optimization engine, which is described in Section 3.3.

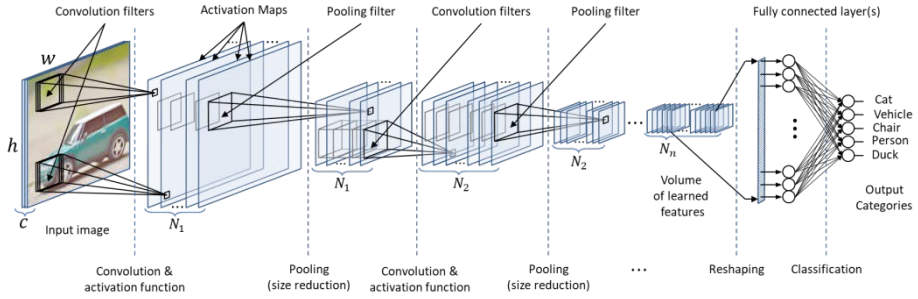


Figure 1: An schematic diagram of a typical CNN for classification

Long-Short Term Memory

The second type of ML model that can be selected is the Long-Short Term Memory (LSTM). LSTM networks are designed to process data sequences; it has been shown that an LSTM can process sequences of up to 1000 values at a time, while avoiding the exploding-gradient and the vanishing-gradient problems [21].

An LSTM network works by simulating logic gates that allow deleting or adding information to a Cell State (7). First a Forget gate (4) decides what information is forgotten or kept; then, the Input gate (3) and the Candidate gate (6) decide what information is added; finally the Hidden state (8) is generated using the Output gate (5) and the Cell state (7) [21]. The equations for this process are shown below. Fig. 2 illustrates the structure of an LSMT cell [21].

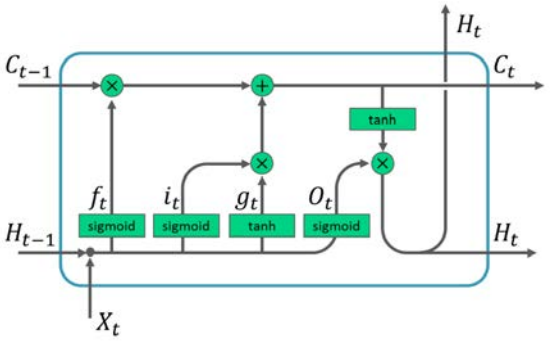


Figure 2: Schematic diagram of an LSTM Cell.

$$\textbf{Input gate: } i_t = sig(x_t U_i + H_{t-1} W_i + b_i),\ i_t \in \mathbb{R}^{d_H} \quad (1)$$

$$\textbf{Forget gate: } F_t = sig(x_t U_f + H_{t-1} W_f + b_f),\ f_t \in \mathbb{R}^{d_H} \quad (2)$$

$$\textbf{Output gate: } O_t = sig(x_t U_o + H_{t-1} W_o + b_o),\ O_t \in \mathbb{R}^{d_H} \quad (3)$$

$$\textbf{Candidate gate: } G_t = tanh(x_t U_g + H_{t-1} W_g + b_g),\ G_t \in \mathbb{R}^{d_H} \quad (4)$$

$$\textbf{Cell state: } C_t = f_t \otimes c - t - 1 + i_t \otimes G_t,\ C_t \in \mathbb{R}^{d_H} \quad (5)$$

$$\textbf{Hidden state: } H_t = O_t \otimes tanh(C_t) \quad (6)$$

$$\textbf{State: } s_t = R_{LSTM}(s_{t-1}, x_t) = [c_t; H_t],\ s_t \in \mathbb{R}^{2d_H} \quad (7)$$

where $U \in \mathbb{R}^{d_x \times d_H}$ and $W \in \mathbb{R}^{d_x \times d_H}$ are the network weights, $x_t \in \mathbb{R}$ is the input, $sig(\cdot)$is the sigmoid function, $tanh(\cdot)$ is the hyperbolic tangent function, and $\otimes$ represents the element-wise product [21].

An LSTM network can be formed by connecting C_t and H_t in a loop, back into the cell's inputs, making the LSTM a type of Recurrent Neural Network. This is equivalent to having several LSTM cells connected in sequence and forming a layer, as illustrated in Fig 3. An LSTM network may also include multiple of those layers.

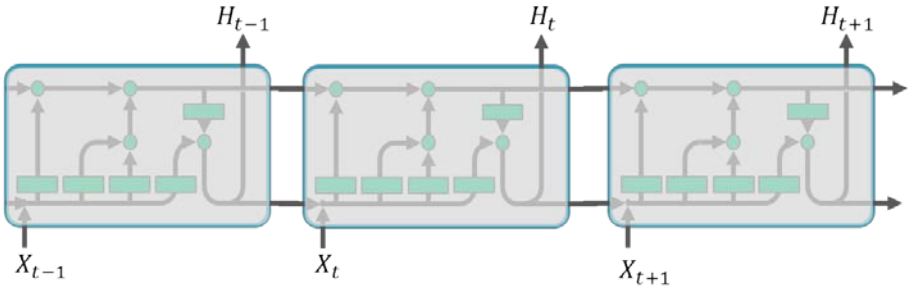


Figure 3: Schematic diagram of an LSTM layer with several cells.

3.2 Optimization Engine

As described in Section 1, AutoML is an automated optimization process through which several ML models are trained and evaluated on test data. This process is driven by an optimization engine, which is used to identify the model with the smallest error on the specific ML task of interest to the user.

There are many algorithms, generally known as metaheuristics that can be used as the optimization engine in AutoML, and these are organized in families according to their characteristics. There are evolutionary algorithms, such as Differential Evolution and Genetic Algorithms; population-based algorithms such as Particle Swarm Opti-

mization (PSO, also characterized as swarm intelligence); estimation of distribution algorithms such as the Univariate Marginal Distribution Algorithm; and many bio-inspired metaheuristics, such as the Bat Algorithm, Ant Colony Optimization, etc.

Due to its flexibility and effectiveness, in this work the PSO is employed as the optimization engine for AutoML. PSO is a metaheuristic inspired by the social behavior of insects, animals and humans. This algorithm starts with a random population of so called *particles*; each particle constitutes a potential solution to the optimization problem at hand. Then, while a termination criterion is not met, PSO will generate new candidate solutions by moving the existing particles in the search space, with the goal of finding better solutions than the currently known candidate solutions. The population of particles is thus iteratively updated until the termination criterion (e.g. a maximum number of iterations) is met and PSO returns the best solution that was found. The pseudocode of PSO is given in Algorithm 1 and the equations employed to update the particles are described below.

According to PSO, a particle $\vec{x}$ that moves with velocity $\vec{v}$ will be updated using the best position $\vec{p}$ that it has previously occupied and the best solution $\vec{p}_b$ in the current population. This idea is shown in the update velocity (8),

$$\vec{v}_i \leftarrow \gamma(w\vec{v}_i + \varphi_1(\vec{p}_i - \vec{x}_i) + \varphi_2(\vec{p}_b - \vec{x}_i)) \tag{8}$$

were $\vec{v}_l$ is the velocity of particle i, $\vec{x}_i$ is the particle i, w is the inertial weight, $\vec{p}_b$ is the best particle found so far, $\varphi_1 = R_1 c_1$, $\varphi_2 = R_2 c_2$, c_1 is the cognitive coefficient, c_2 is the social coefficient, R_1 and R_2 are uniform random values between 0 and 1, γ is the constriction coefficient given by (9), and the particle update equation is (10).

$$\gamma = \frac{2}{|2-\phi^2-4\phi|}, \qquad \phi = c_1 + c_2 > 4 \tag{9}$$

$$\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i \tag{10}$$

Algorithm 1 PSO, assuming minitization

1: Generate random population $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\}$.
2: Finding the best solution $\vec{p}_b$ in X.
3: **while** *Stop criterion* is not met, **do**
4: **for** particle $\vec{x}_i$ **do**
5: **if** $f(\vec{x}_i) < f(\vec{p}_i)$ **then**
6: $\vec{p}_i \leftarrow \vec{x}_i$; update the best position of particle i.
7: **end if**
8: Identify the best position of the iteration $\vec{p}_b$.
9: Update velocity $\vec{v}_i$ via (10).
10: Update position $\vec{x}_i$ via (12).
11: **end for**
12: **end while**
13: **return** The best solution found $\vec{p}_b$.

4 Methodology

This section describes the experiments performed to obtain the solutions via the AutoML process using PSO as the optimization engine. Provided that PSO, as any other metaheuristic is a semi-stochastic method which depends on initial solutions and other factors, the AutoML solutions found may vary between different executions of this process. Because of this, the AutoML process is executed 10 times, to generate 10 different "optimal" solutions. After the AutoML process, the 10 solutions returned are compared with each other by performing 33 trials, each consisting in re-training and evaluating their performance in forecasting of the stock prices. This multi-trial evaluation is performed to provide statistical support to our final conclusions.

To perform the AutoML process the PSO algorithm and the rest of the system was implemented using several Python libraries including NumPy, Pandas, Scikit-Learn and TensorFlow. The metaheuristic iteratively performs feature selection and design of the LSTM network with the objective of finding a configuration that minimizes the error of the network for the forecasting of the closing price of Google's stocks. The fitness function used is the Mean Squared Error (MSE). In each iteration, the metaheuristic creates a set of candidate solutions; each solution is a vector containing the hyper-parameters and features that will be given as inputs to the network. The ultimate objective is to iteratively improve the solution, so that the metaheuristic finally returns the network with the lowest possible MSE. This process is shown in Fig. 4.

In each fitness-function call, an LSTM network is designed, the features that will be the input to this network are selected, and the training of the LSTM network is performed. For the training, the closing price of Google's stock from 2004-11-08 to 2022-01-21 was used, with 64% of the data used as the training set; 21% of the data used for validation; and 15% employed for testing of the network. To reduce the execution time and avoid overfitting, early stopping was employed. The Adam optimizer was used to train the LSTM network.

In this work, the selection of characteristics is based on a set of technical indicators (TIs, shown in Table 2) which were selected according to [7],[21],[22] and [23]. The selection of hyper-parameters (Table 4) was carried out according to [7] and [8]. The parameters of the PSO algorithm (Table 3) where set according to [24] and [25].

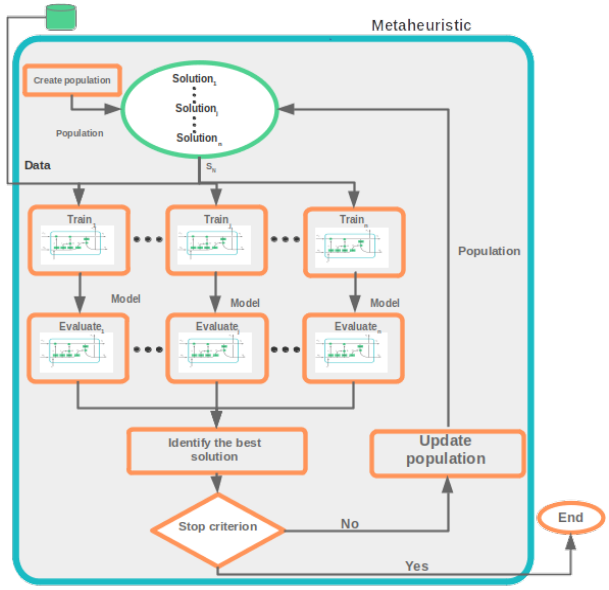


Figure 4: AutoML Methodology based on PSO

Table 2: Technical Indicators.

	Name	Symbol	Search Space
1	Price Rate Of Change	ROC	{0, 1}
2	Simple Moving Average	SMA	{0, 1}
3	Exponential Moving Average	EMA	{0, 1}
4	Moving Average Convergence Divergence	MACD	{0, 1}
5	Relative Strength Index	RSI	{0, 1}
6	Commodity Channel Index	CCI	{0, 1}
7	William R	WR	{0, 1}
8	Money Flow Index	MFR	{0, 1}
9	Force index	RFI	{0, 1}
10	Average True Range	TR	{0, 1}
11	Ease Of Movement Indicator	EMI	{0, 1}
12	Close	Close	{0, 1}

Table 3: Parameters of the PSO metaheuristic.

Parameter	Value
Constriction coefficient	$c_1 = c_2 = 2.05$
Inertial weight	0.8
Population size	86
Iterations	16

Table 4: Hyper-parameters and model.

	Hyper-parameters and model	Symbol	Search Space
1	CNN layer	CNN	{0, 1}
2	No. Filters CNN layer	FC_s	{1, 20, 50, 80, 96, 110, 150}
3	Kernel size	KC_s	{1, 20, 50, 80, 96}
4	LSTM cell	Cl_s	{1, 2, 3, 4, 5, 6, 7, 8}
5	Hidden state	H_s	{1, 20, 50, 80, 96, 110, 150}
6	Windows size	W_s	{20, 40, 60, 80, 90, 100}
7	Learning Rate	α	{1E-1, 1E-2, 1E-3, 1E-4, 1E-5}
8	Batch size	B_s	{8, 16, 32, 64}
9	Epochs	E_s	{12, 25, 100, 150}

5 Experimental Results

The execution time for each experiment was between 3 and 4 hours. Each solution achieves an MSE in the order of 1E-5. The MSE on test data are in the range of 1E-1 to 1E-5. The 10 AutoML solutions (S-1 to S-10) obtained are shown as the columns in Table 5. The last three rows in Table 5 report the statistics of the MSE obtained by each solution in the multi-trial evaluation. The whole set of results are presented as boxplots in Fig. 5. Additionally, the forecasting results of the top four solutions are presented in Fig. 6.

The results show that even though a network may be deeper, it does not necessarily achieve a better performance. This may be due to the fact that a deeper network can be more prone to overfitting. The best performing solutions have 110 and 150 hidden states, while the solutions with 96 hidden states have the lowest performance. However, the best solutions (S-2, S-3 and S-4) contain very different amount of parameters. The best solution (S-2) contains approximately 53k parameters, while the second and third best solutions (S-3 and S-4, respectively according to Fig. 3) contain about 94k parameters each (see Table 5).

In all but one solution (S-3), the learning rate was set to α=1E-3, indicating a consistency in the optimal value of this parameter for this problem. The batch size values of the solutions are either 16 or 32, thus giving indications that these two batch sizes promote the satisfactory performance of the model.

Only one solution (S-2) has a CNN layer, and this solution is the one with the best performance, as shown in Fig. 5. This is consistent with previous work that has found that the use of CNNs promotes better forecasting. However, we have no explanation as to why the AutoML process only generated one solution with CNN layers.

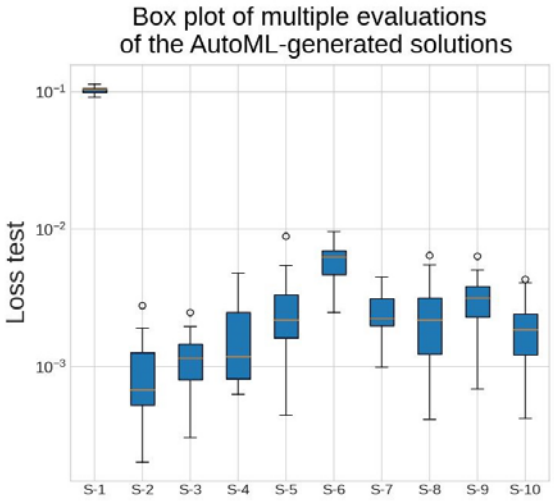


Figure 5: Loss test of 33 experiments of train solutions.

Table 5: The ten AutoML-generated Solutions and their performance metrics.

	Symbol	S-1	S-2	S-3	S-4	S-5	S-6	S-7	S-8	S-9	S-10
Features	Close	0	1	1	1	1	1	1	1	1	1
	ROC	0	1	1	1	0	1	0	0	0	1
	SMA	0	1	1	1	1	1	1	1	1	0
	EMA	1	1	0	1	0	0	0	0	0	1
	MACD	1	0	0	0	0	1	0	0	0	0
	RSI	0	0	0	0	0	1	0	0	0	1
	CCI	1	1	0	0	1	1	1	1	1	0
	WR	1	1	1	0	0	0	0	1	0	0
	MFR	1	0	0	1	1	1	1	1	1	1
	RFI	1	0	1	1	1	0	0	0	0	0
	TR	1	0	0	0	1	1	0	0	0	0
	EMI	1	1	0	1	0	0	0	0	0	0
Sum of Features		8	7	5	7	6	8	4	5	4	5
Hyper-parameters	CNN	0	1	0	0	0	0	0	0	0	0
	FC_s	0	110	0	0	0	0	0	0	0	0
	KC_s	0	2	0	0	0	0	0	0	0	0
	Cl_s	6	1	1	1	1	1	1	1	1	1
	H_s	96	110	150	150	150	96	150	110	110	150
	W_s	100	40	60	60	60	60	60	90	80	90
	α	1E-3	1E-3	1E-4	1E-3	1E-3	1E-3	1E-3	1E-3	1E-3	1E-3
	B_s	16	32	16	16	16	32	32	16	16	32
	E_s	100	50	100	25	25	100	100	50	100	50
Amount of Parameters		410,977	55,333	93,751	94,951	39,649	40,801	93,151	39,265	38,881	93,751
Performance	Fitness	3E-5	4E-5	5E-5	3E-5	5E-5	4E-5	4E-5	3E-5	5E-5	6E-5
	Mean	1E-2	9E-4	1E-3	1E-3	2E-3	5E-3	2E-3	2E-3	3E-3	2E-3
	Std. Dev.	5E-3	5E-4	5E-4	1E-3	1E-3	1E-3	9E-4	1E-3	1E-3	9E-4

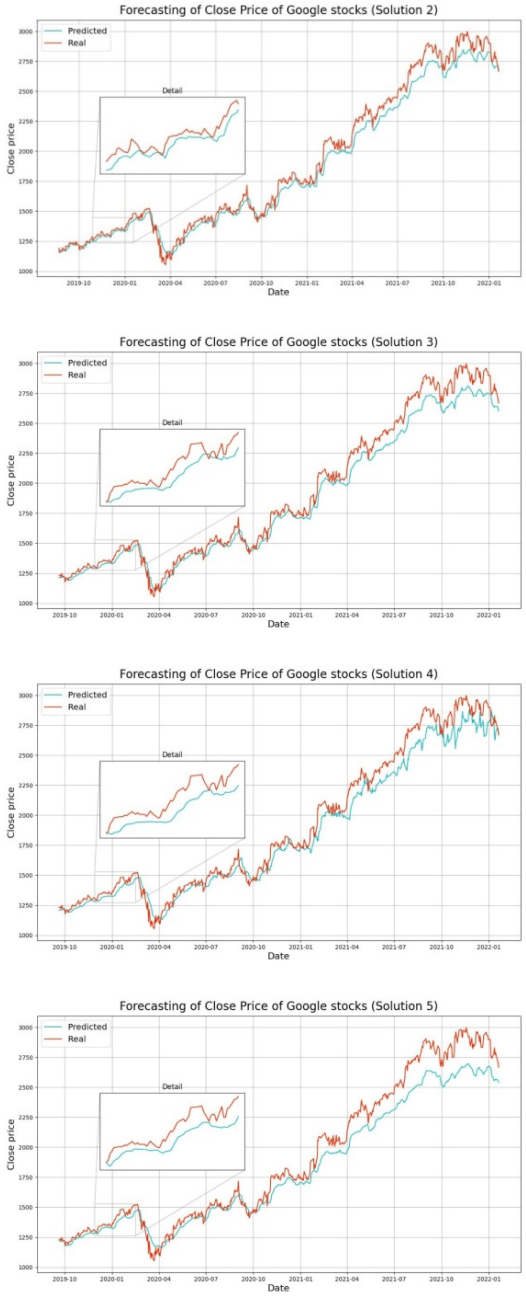


Figure 6: Stock forecasting of AutoML Solutions 2, 3, 4 and 5 (top to bottom, respectively).

Regarding the technical indicators, SMA is selected in solutions S-2 to S-9. SMA is a technical indicator for smoothing the time series, and its repeated selection in most solutions indicates that smoothing the time series promotes a better forecasting.

On the other hand, RSI is only found in two solutions (S-6 and S-10), indicating that knowing whether or not the financial stock is overbought has little influence on the performance of the model (in other words, the trend has no significant effect on the performance of the solution obtained by the metaheuristic).

Similarly, EMA also serves to determine the trend and is found in solutions S-1, S-2, S-4 and S-10. S-2, obtained the best performance and S-1 the worst performance. However, S-2 processes the features through a CNN layer to generate new features, and this may have a significant impact on the performance of the network well beyond the particular selection of features that constituted the input to the network.

6 Conclusions

PSO manages to carry out the AutoML process, although being a non-deterministic algorithm, the solutions it finds are not always the best. It was found that a CNN-LSTM network has better performance than an LSTM network with technical indicators, but this solution was found only in one out of the 10 experiments performed. This may be due to the fact that the solutions obtained are in the order of 1E-4 and that the weights of the networks are created randomly with a different seed each time, in conjunction with the possibility that the fitness function is not bijective.

In future work, we will seek to carry out the AutoML process with more control over the generation of random numbers to better understand this problem. At this point it is still unknown the actual reason for which the metaheuristic did not choose to produce more solutions with CNN-LSTM networks.

Solutions with deeper artificial neural networks do not always generate better results, to obtain deeper networks with better performance their hyper-parameters must be tuned optimally, but this is a challenge when the networks contain a large number of parameters, as is the case with deeper networks.

In future work, we will seek to analyze the influence of independent data sources on network performance, considering that a good amount of solutions obtained via the AutoML process may be sufficient to carry out this analysis, which is also an objective of the AutoML paradigm.

We also consider applying the AutoML process on different problems, such as the classification problem, as well as using different algorithms as the optimization engine to carry out the AutoML process.

Acknowledgments: This work was supported by the National Council of Science and Technology (CONACYT) of Mexico through Postgraduate Scholarship 774627 (F. Pedroza) and Research Grant CÁTEDRAS-2598 (A. Rojas).

References

1: R.C. Cavalcante, R.C. Brasileiro, V.L. Souza, J.P. Nobrega, A.L. Oliveira, Computational intelligence and financial markets: A survey and future directions. Expert Systems with Applications 55, 194–211, 2016.
2: K. Chourmouziadis, P.D. Chatzoglou, An intelligent short term stock trading fuzzy system for assisting investors in portfolio management. Expert Systems with Applications 43, 298–311, 2016.
3: C. Evans, K. Pappas, F. Xhafa, Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation. Mathematical and Computer Modelling 58(5-6), 1249–1266, 2013.
4: S. Chopra, D. Yadav, A. Chopra, Artificial neural networks based Indian stock market price prediction: before and after demonetization. J. Swarm Intel. Evol. Comput. 8(174), 2, 2019.
5: H. Chung, K. S. Shin, Genetic algorithm-optimized Long Short-Term Memory network for stock market prediction. Sustainability 10(10), 3765, 2018.
6: G. Kumar, S. Jain, U. Singh, Stock market forecasting using computational intelligence: A survey. Archives of Computational Methods in Engineering 28(3), 1069–1101, 2021.
7: K. Kumar, M. Haider, T. Uddin, Enhanced prediction of intra-day stock market using metaheuristic optimization on RNN–LSTM network. New Generation Computing 39(1), 231–272, 2021.
8: T. R. Silva, A. W. Li, E.O. Pamplona, Automated trading system for stock index using LSTM neural networks and risk management. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE, 2020.
9: A. Tsantekidis, N. Passalis , A. Tefas, J. Kanniainen, M. Gabbouj, A. Iosifidis, Using deep learning for price prediction by exploiting stationary limit order book features. Applied Soft Computing 93, 106401, 2020.
10: O.B. Sezer, M.U. Gudelek, A.M. Ozbayoglu, Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied soft computing 90, 106181, 2020.
11: H. Chung, K.s Shin, Genetic algorithm-optimized Long Short-Term Memory network for stock market prediction. Sustainability 10(10), 3765, 2018.
12: H.N. Bhandari, B. Rimal, N.R Pokhrel, R. Rimal, K.R. Dahal, R.K. Khatri, Predicting stock market index using LSTM. Machine Learning with Applications p. 100320, 2022.
13: R. Kumar, P. Kumar, Y. Kumar, Three stage fusion for effective time series forecasting using Bi-LSTM-ARIMA and improved DE-ABC algorithm. Neural Computing and Applications pp. 1–17, 2022.
14: C. Deng, Y. Huang, N. Hasan, Y. Bao, Multi-step-ahead stock price index forecasting using Long Short-Term Memory Model with Multivariate Empirical Mode Decomposition. Information Sciences, 2022.
15: S. Deng, C. Xiao, Y. Zhu, Y. Tian, Z. Liu, T. Yang, Dynamic forecasting of the Shanghai Stock Exchange index movement using multiple types of investor sentiment. Applied Soft Computing p. 109132, 2022.

16: A. Kanwal, M.F. Lau, S.P. Ng, K.Y Sim, S. Chandrasekaran, BiCuDNNLSTM1dCNN—A hybrid deep learning-based predictive model for stock price prediction. Expert Systems with Applications 202, 117123, 2022.
17: Y. Lin, Y. Yan, J. Xu, Y. Liao, F. Ma, Forecasting stock index price using the CEEMDAN-LSTM model. The North American Journal of Economics and Finance 57, 101421, 2021.
18: W. Chen, M. Jiang, W.G. Zhang, Z. Chen, A novel graph convolutional feature based convolutional neural network for stock trend prediction. Information Sciences 556, 67–94, 2021.
19: Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al., Recent advances in convolutional neural networks. Pattern recognition 77, 354–377, 2018.
20: J. Schmidhuber, et al., Long Short-Term Memory. Neural Comput 9(8), 1735–1780, 1997.
21: O. Bustos, A. Pomares-Quimbaya, Stock market movement forecast: A systematic review. Expert Systems with Applications 156, 113464 , 2020.
22: W. Chen, M. Jiang, M., W.G. Zhang, Z. Chen, A novel graph convolutional feature based convolutional neural network for stock trend prediction. Information Sciences 556, 67–94, 2021.
23: H. Chung, K.s, Shin, Genetic algorithm-optimized Long Short-Term Memory network for stock market prediction. Sustainability 10(10), 3765, 2018.
24: M. Gendreau, J.Y Potvin., et al., Handbook of metaheuristics, third edition, vol. 272, Springer, 2019.

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLOGICO NACIONAL DE MÉXICO

TIJUANA INSTITUTE OF TECHNOLOGY
IEEE - Computational Intelligence Society Mexico-Chapter
In cooperation with HAFSA Association

CERTIFICATE OF PRESENTATION

Tijuana, Mexico, August 22, 2022

This is to confirm that: **Francisco J. Pedroza-Castro, Alfonso Rojas- Domínguez, Martín Carpio, Manuel Ornelas-Rodríguez, Héctor Puga.**, have presented during the technical program of the International Seminar of Computational Intelligence ISCI 2022 held in Tijuana, Mexico (online) on 23-25 of August, 2022. With the Talk: **Automated Machine Learning to improve LSTM to forecasting stock markets using metaheuristics**

This participation has been very important for the academic success of the ISCI 2022 International seminar.

Prof. Oscar Castillo, D.Sc.
Program Chair of ISCI 2022
IEEE CIS Mexico Chapter Chair

Prof. Patricia Melin, D.Sc.
General Chair of ISCI 2022
IEEE CIS Mexico Chapter Vice Chair

IEEE - Computational Intelligence Society
Mexico-Chapter

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

IEEE CIS-MEXICO CHAPTER

MAESTRIA EN CIENCIAS DE LA COMPUTACIÓN

Apéndice C

Experimentos realizados con SEED

En este apéndice se muestran los experimentos realizados con el algoritmo SEED para optimizar los hiperparámetros de una red LSTM y de este modo la red LSTM pueda pronosticar el precio de cierre que tendrán al día siguiente las acciones de Google y las acciones de Nike.

Los parámetros para correr el algoritmo fueron los siguientes: 16 iteraciones (425 llamadas a función), con un criterio de paro de máximo de 10^{-7}.

El tiempo de ejecución para el caso de Google fue de 28.64 minutos; mientras que para el caso de Nike fue de 67.72 minutos. Las soluciones encontradas por SEED se muestran en la Tabla C.1. La evidencia estadística muestra un rendimiento satisfactorio con una red LSTM simple de profundidad para el caso de SEED-GOOGLE. Tanto para el caso de SEED-GOOGLE como SEED-NIKE el número de épocas fue menor en comparación de las otras metaheurísticas aplicadas en esta tesis, indicando que se puede tener rendimientos satisfactorios cuando los hiperaparámetros se encuentra sintonizados. En la Fig. C.1 se muestra el pronóstico realizado por las soluciones encontradas por SEED.

Si bien, la evidencia experimental-estadística muestra resultados satisfactorios, no sucede lo mismo con el comportamiento de convergencia (Fig. C.2); en la optimización se esperaría que la búsqueda se realizara de manera distinta y no con una media y dispersión similar en cada iteración.

A pesar de ello, las soluciones obtenidas por SEED muestran ser más simples y se

	Resultados SEED	
	Acciones de Google	**Acciones de Nike**
Hiperpárametros		
Cl_s	1	1
H_s	80	150
W_s	20	20
α	10^{-3}	10^{-4}
B_s	8	20
E_s	12	12
Información de solución		
No. de Iteraciones:	16	16
Llamadas a función:	425	425
Fitness:	5.97E-5	4.96E-6
Promedio de perdida:	6.27E-5	4.91E-6
Promedio de prueba:	7.05E-4	1.78E-4
Desviación estándar de perdida:	3.17E-6	1.39E-7
Desviación estándar de prueba:	6.86E-4	6.77E-5

Tabla C.1: Soluciones encontradas por SEED así como los resultados de su rendimiento.

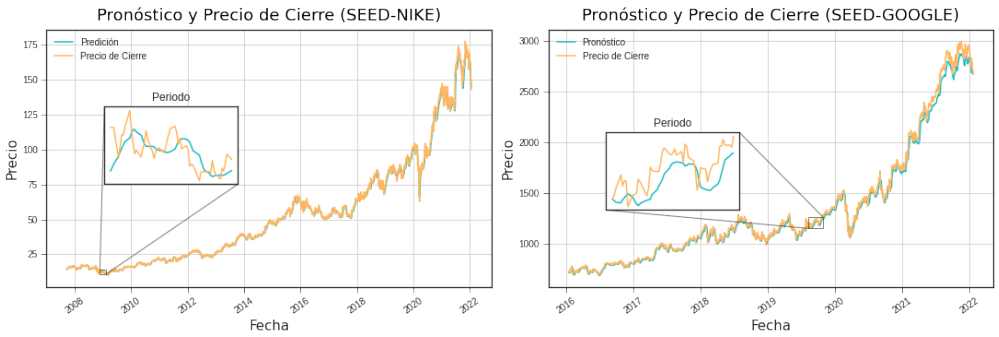


Figura C.1: Diagrama de caja y bigote de Prueba de pronóstico promedio de 33 experimentos. (Elaboración propia)

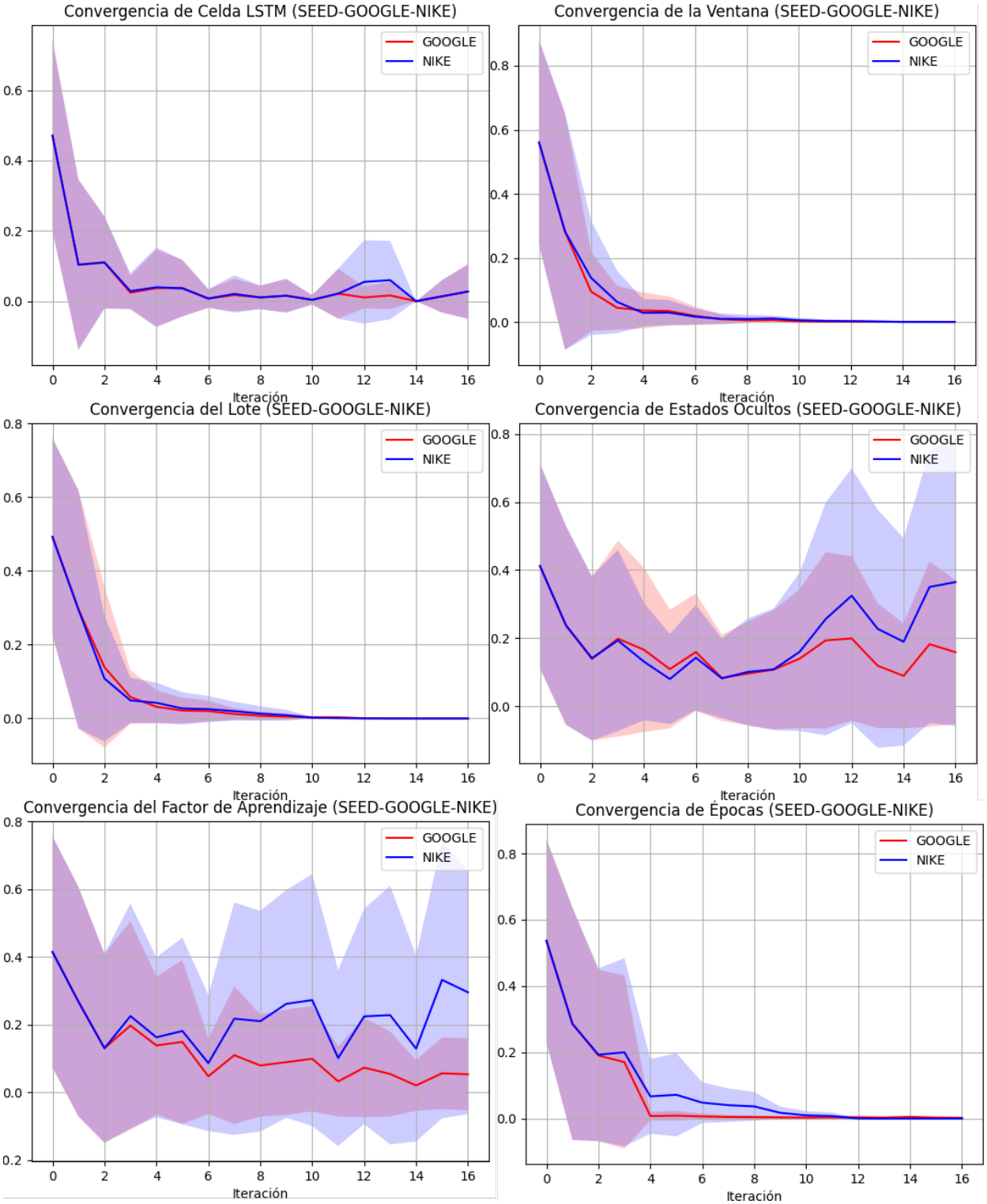


Figura C.2: Diagrama de caja y bigote de Prueba de pronóstico promedio de 33 experimentos. (Elaboración propia)

entrenan con 12 épocas, un número reducido en comparación de las soluciones obtenidas con PSO, FPA y LNFE. Incluso en el caso de pronosticar las acciones de Nike, LNFE supera a las soluciones de las demás metaheurísticas, aunque no pasa lo mismo en el caso de las acciones de Google; no obstante, se puede afirmar que las soluciones de SEED, compiten con las demás soluciones de las metaheurísticas y supera a las soluciones OSM.

Apéndice D

Experimentos con red LSTM para realizar pronósticos

En el presente anexo se muestran algunos de los experimentos realizados con la red LSTM para pronosticar las acciones de Apple. Estos experimentos se llevaron a cabo con el objeto de observar el efecto del número de estados ocultos en el rendimiento de la red neuronal.

Los experimentos se llevaron a cabo fijando los siguientes hiperparámetros: factor de aprendizaje de 10^{-1}, una celda LSTM, dos épocas, Error Cuadrático Medio tanto como función de pérdida como función de evaluación y se utilizó el Optimizador Adam.

Los resultados experimentales muestran que aumentar el número de estados ocultos mejora el rendimiento de la red LSTM; este comportamiento se aprecia en la gráfica de la Fig. D.1 de pérdida contra estados ocultos (Loss vs. hidden). La gráfica de la Fig. D.1 también nos dice que usando datos de entrenamiento existe una tendencia a la baja de la pérdida y que a pesar de que existen algunos aumentos de la pérdida en prueba, el valor sigue siendo bajo. Si realizamos un acercamiento a la imagen en los últimos valores (Fig. D.2) se puede notar que la pérdida sigue bajando y el cambio en prueba es mínimo; no obstante, esa diferencia solo se aprecia en las gráficas de predicción que se muestran en las Figs. D.3 a D.10.

En las gráficas de pronósticos, la curva azul muestra la serie de tiempo original, la línea

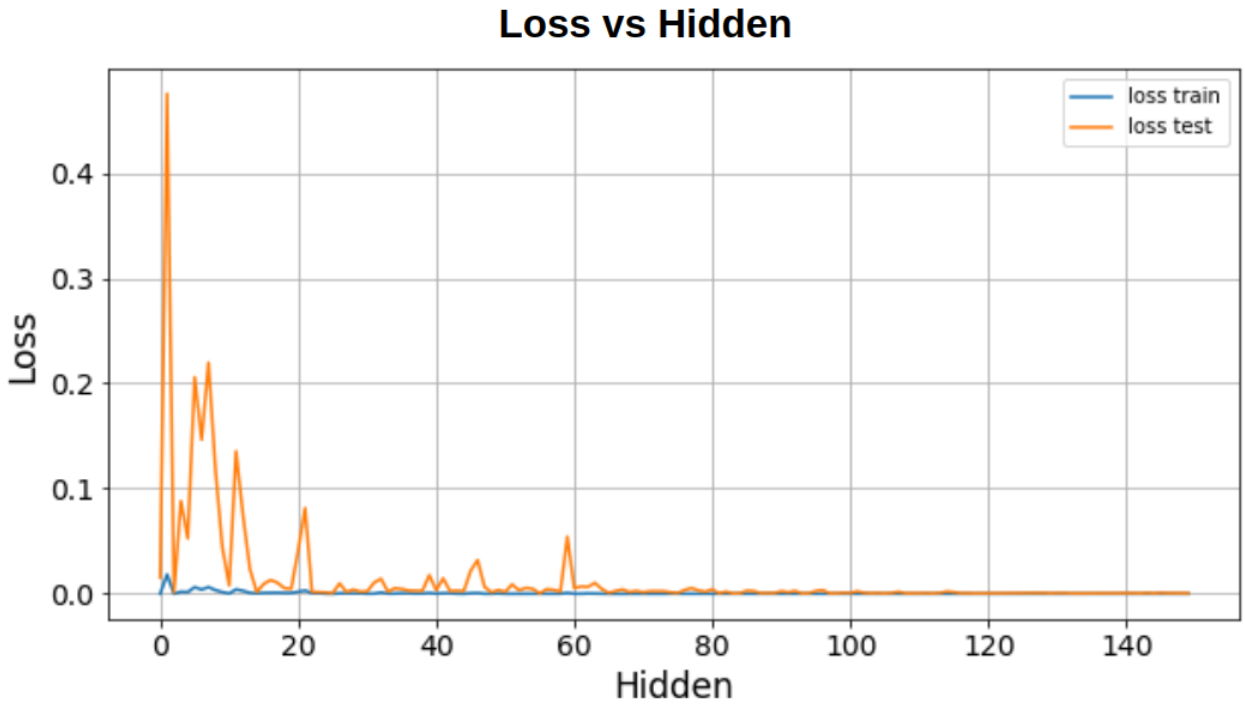


Figura D.1: Pérdida vs. número de estados ocultos. (Elaboración propia)

anaranjada es la predicción con datos de entrenamiento y la línea verde es la predicción con datos de prueba (evaluación).

En la gráfica de la Fig. D.9 se muestra el pronóstico de una red con 125 estados ocultos, la cual tiene el mejor rendimiento y cuando se aumenta el número de estados ocultos aumenta el error como se muestra en la Fig. D.10 con 150 estados ocultos, sin embargo, no se puede concluir que si se aumenta de 125 estados ocultos se comienza a aumentar el error, porque conforme aumenta el número de estados ocultos, el comportamiento del error oscila, empero tienen una tendencia a la baja como se muestra en la Fig. D.1.

El efecto oscilatorio que se produce al aumentar el número de estados ocultos se aprecia en las Figs. D.3 y D.4, porque un estado oculto tiene mejor rendimiento de predicción que 2 estados ocultos, no obstante, al aumentar el número de estados ocultos, mejora el rendimiento.

De manera general, se puede decir que aumentar el número de estados ocultos muestra un comportamiento oscilatorio del error, pero con tendencia a disminuir el error mejorando la predicción.

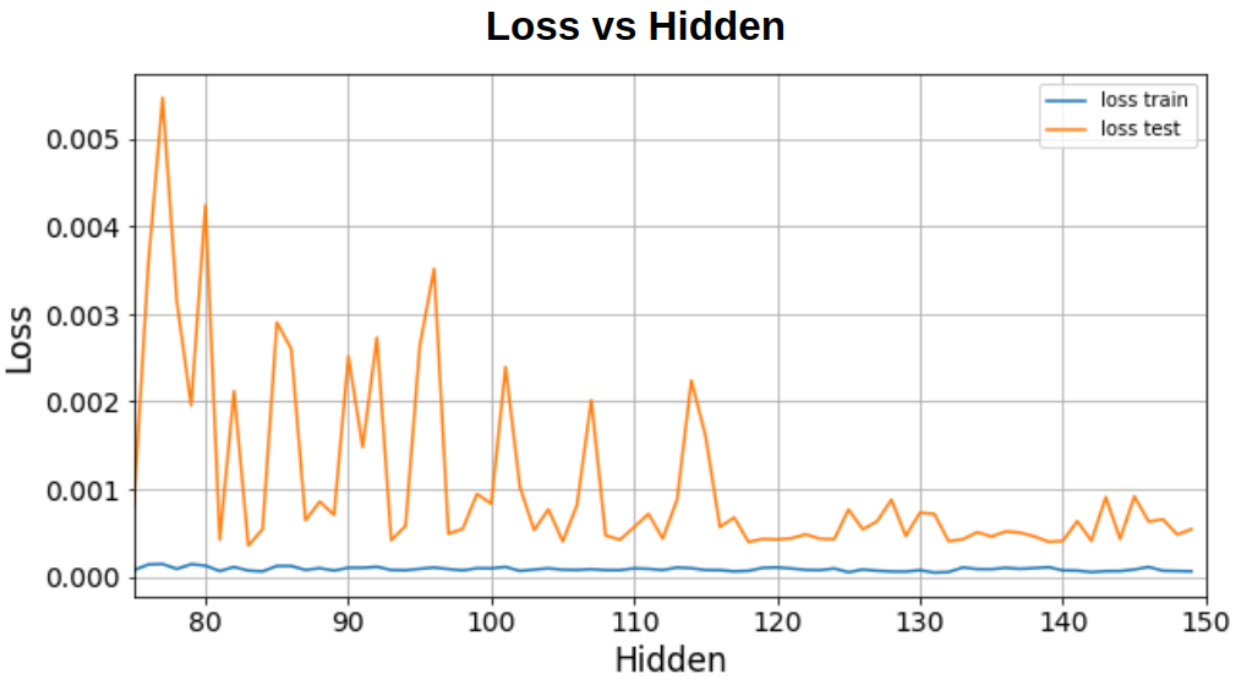


Figura D.2: Acercamiento de pérdida vs. número de estados ocultos. (Elaboración propia)

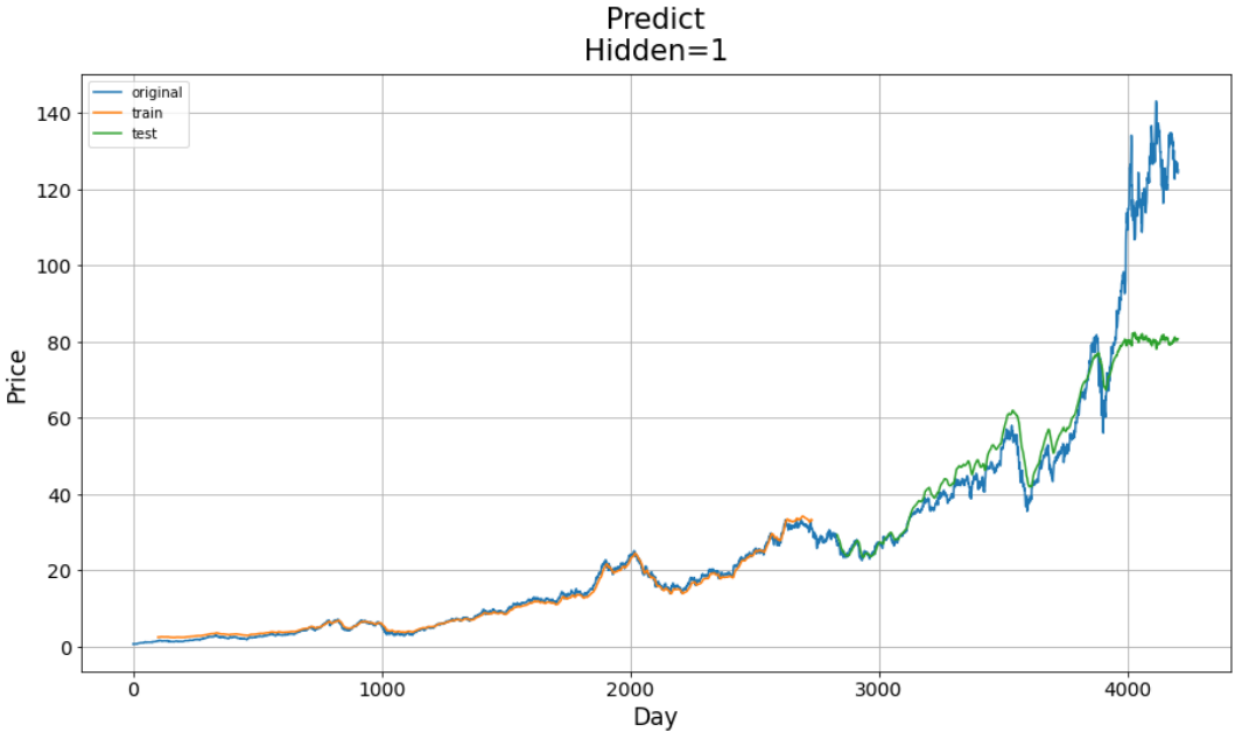


Figura D.3: Predicción de una red LSTM con un estado oculto. (Elaboración propia)

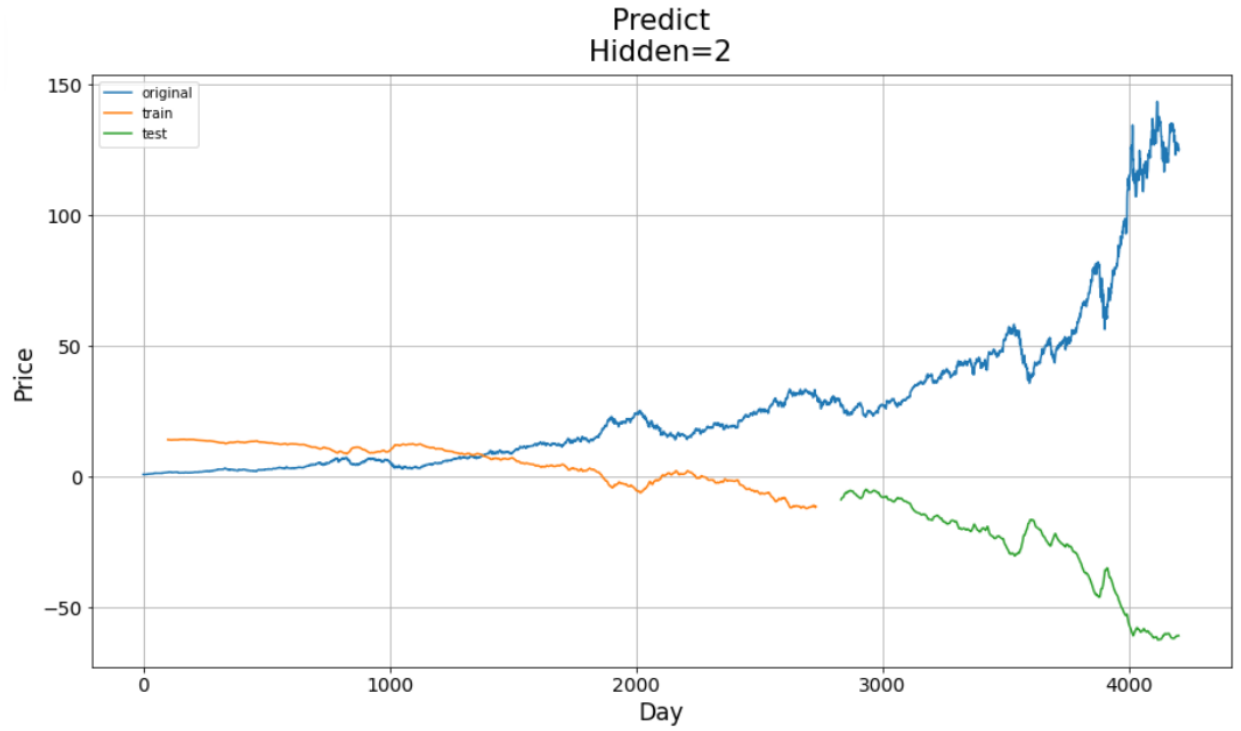


Figura D.4: Predicción de una red LSTM con dos estados ocultos. (Elaboración propia)

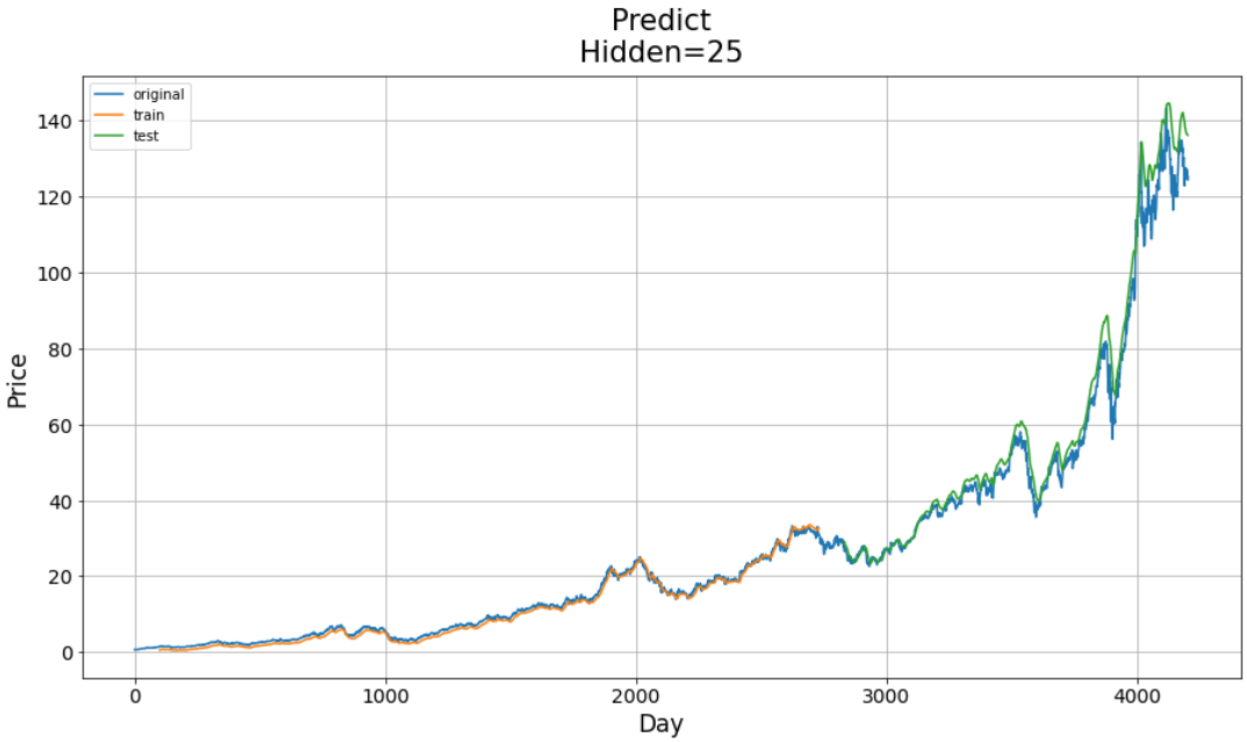


Figura D.5: Predicción de una red LSTM con 25 estados ocultos. (Elaboración propia)

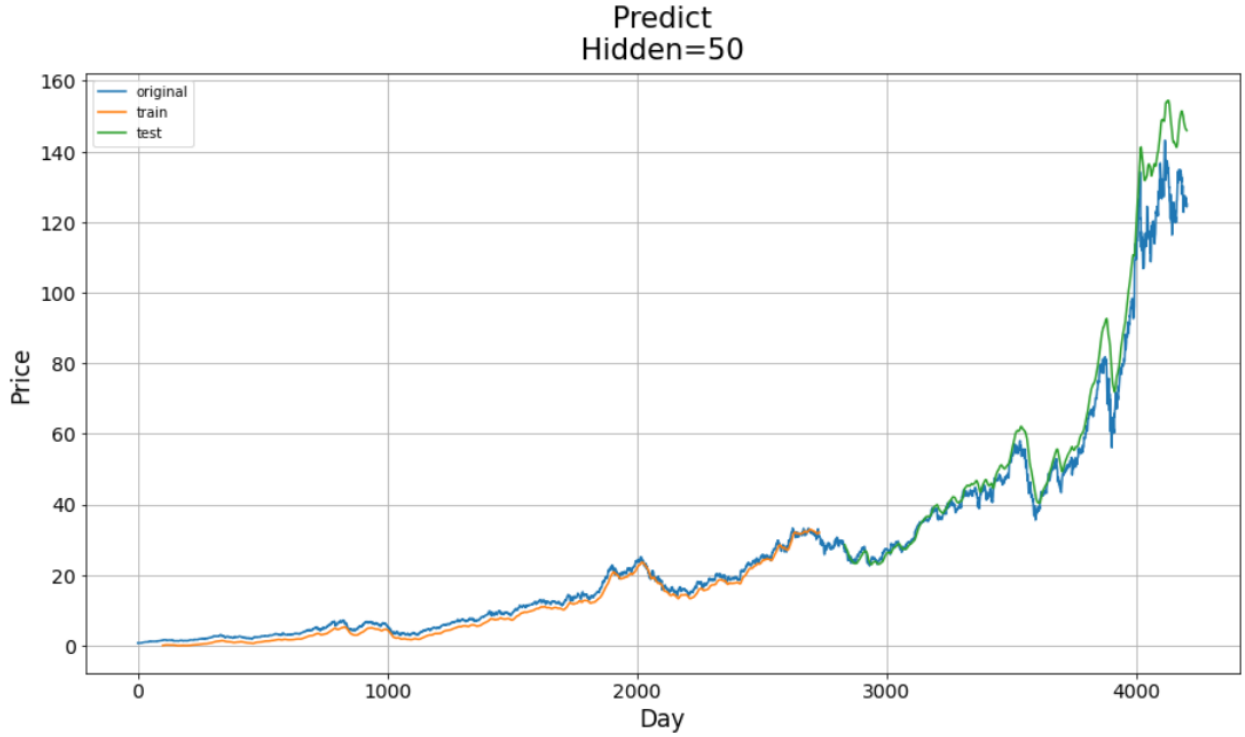


Figura D.6: Predicción de una red LSTM con 50 estados ocultos. (Elaboración propia)

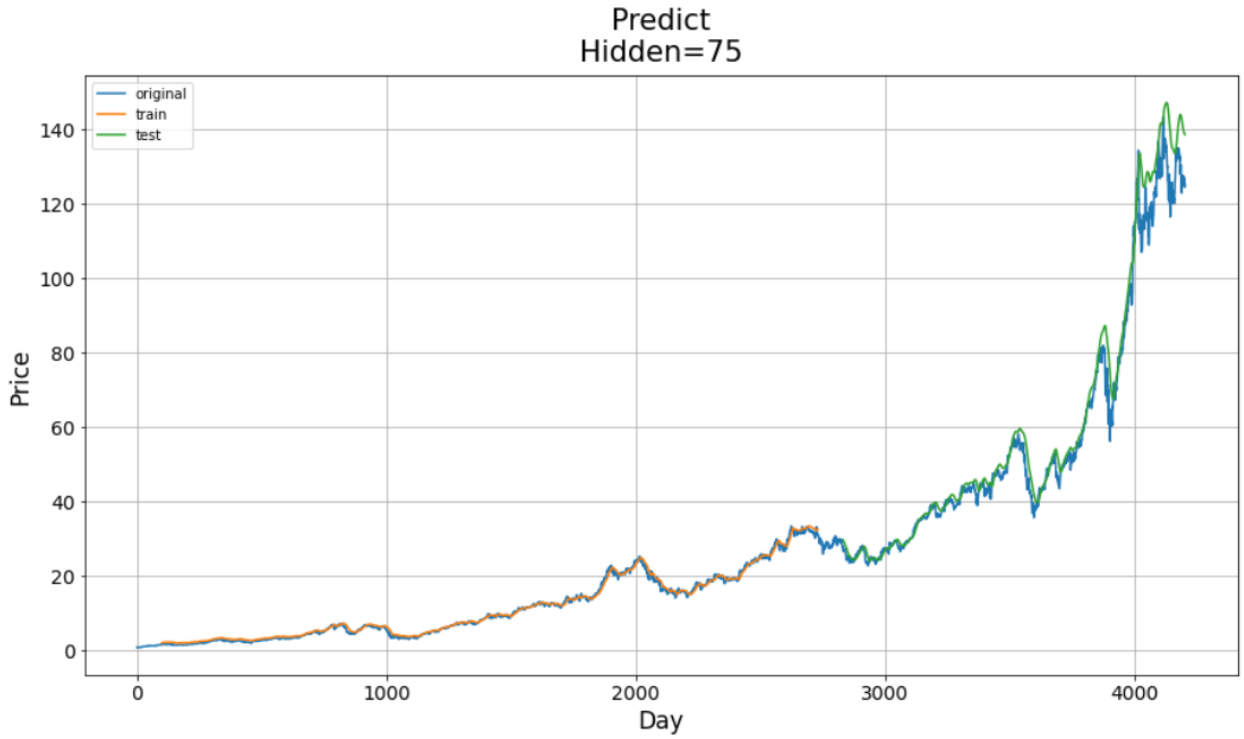


Figura D.7: Predicción de una red LSTM con 75 estados ocultos. (Elaboración propia)

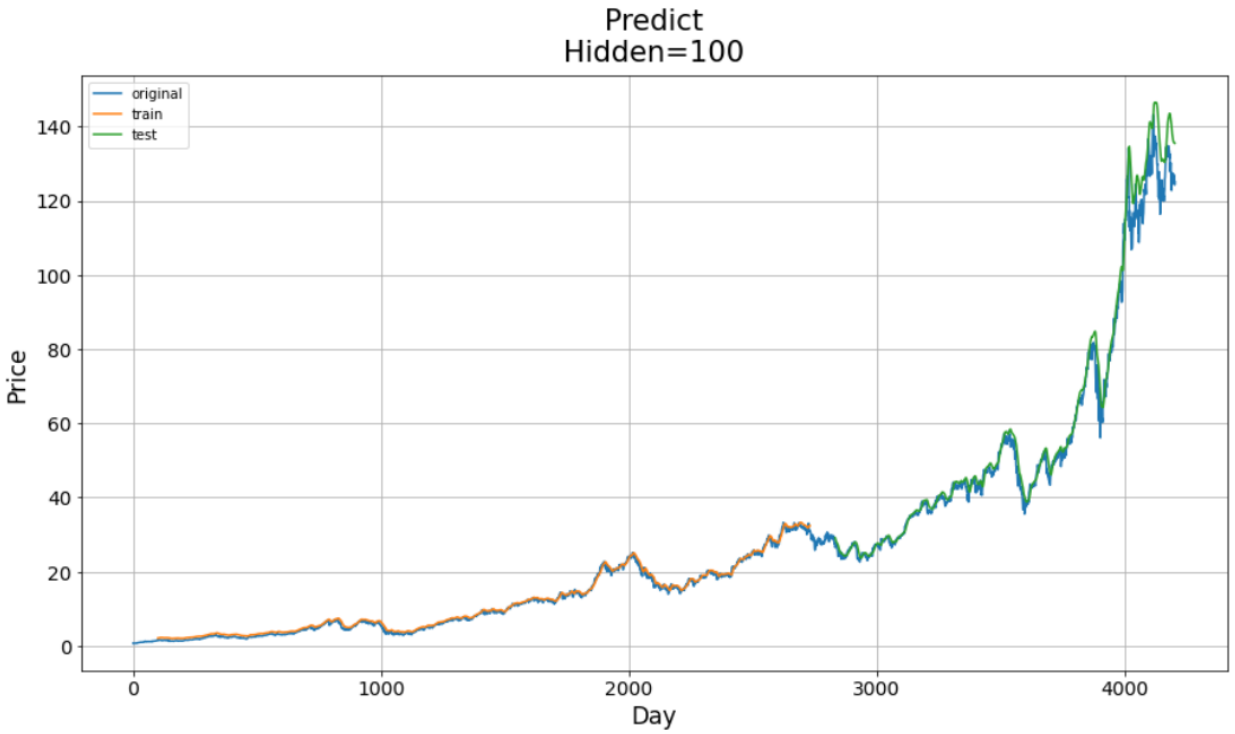


Figura D.8: Predicción de una red LSTM con 100 estados ocultos. (Elaboración propia)

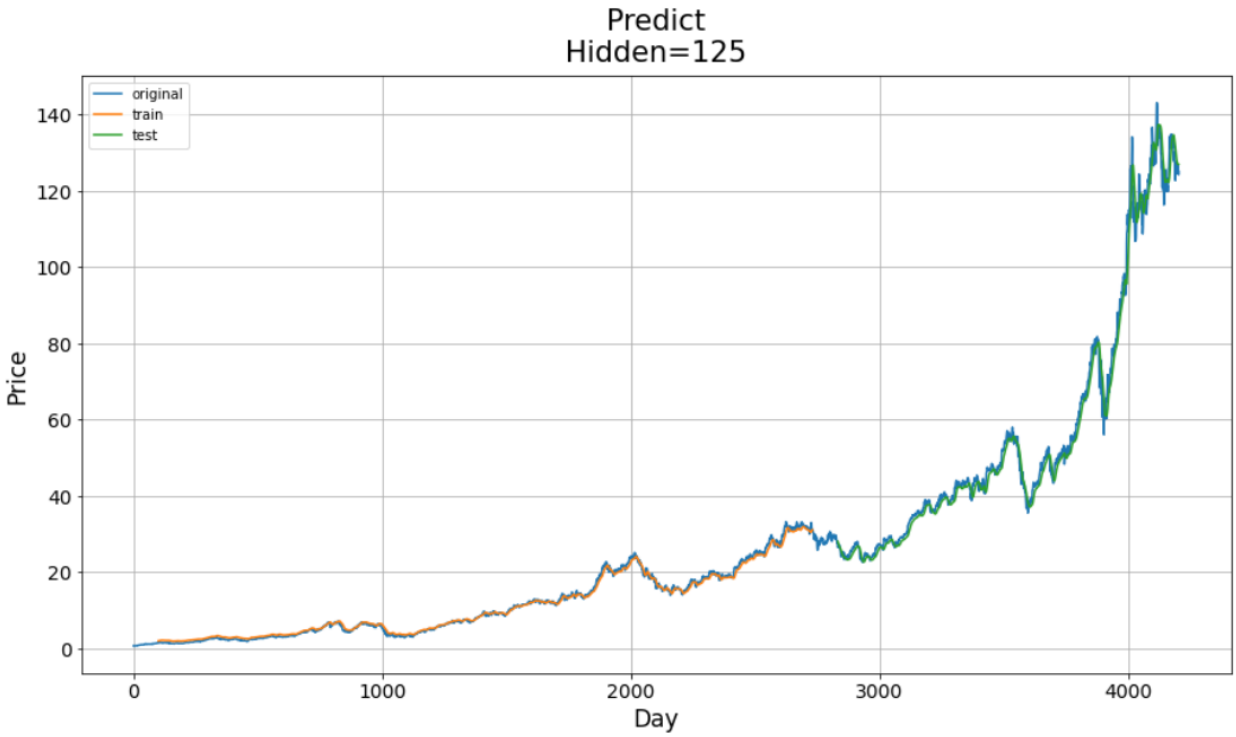


Figura D.9: Predicción de una red LSTM con 125 estados ocultos. (Elaboración propia)

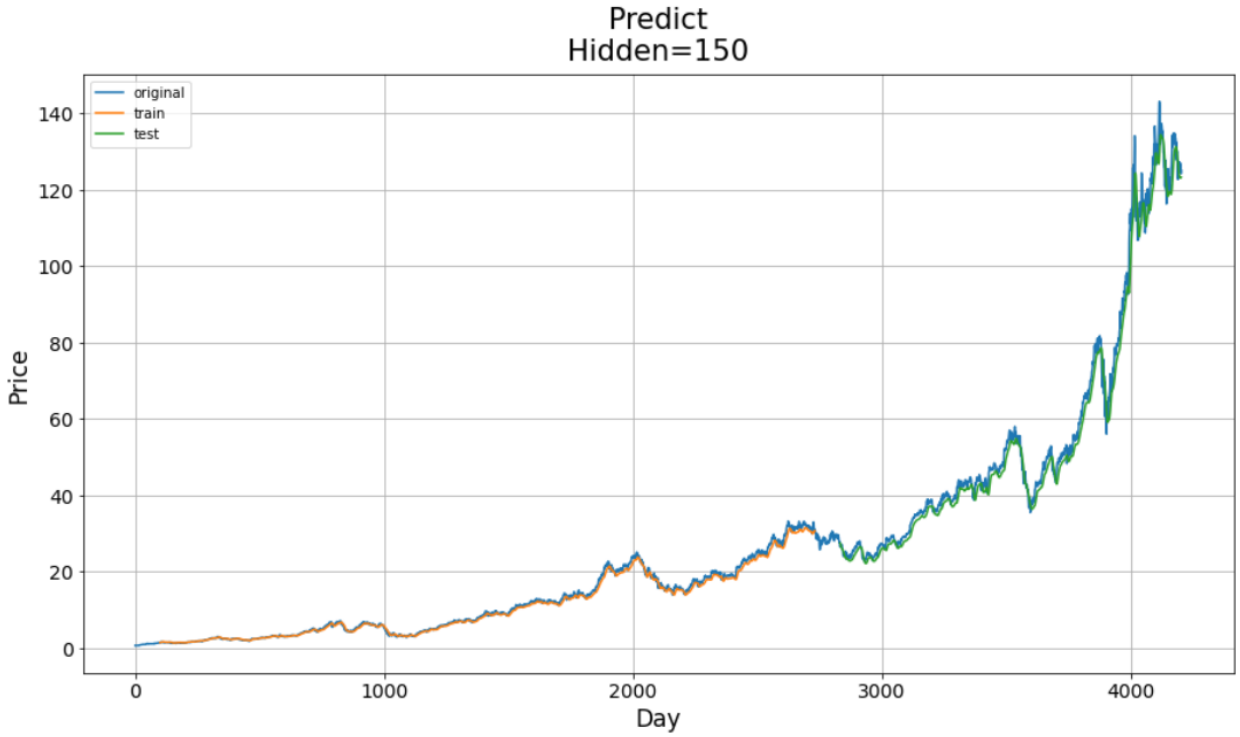


Figura D.10: Predicción de una red LSTM con 150 estados ocultos. (Elaboración propia)

Bibliografía

[1] Adebiyi, A.A., Adewumi, A.O., Ayo, C.K.: Comparison of ARIMA and artificial neural networks models for stock price prediction. Journal of Applied Mathematics **2014** (2014)

[2] Ahmadi, E., Jasemi, M., Monplaisir, L., Nabavi, M.A., Mahmoodi, A., Jam, P.A.: New efficient hybrid candlestick technical analysis model for stock market timing on the basis of the support vector machine and heuristic algorithms of imperialist competition and genetic. Expert Systems with Applications **94**, 21–31 (2018)

[3] Alpaydin, E.: Introduction to machine learning. MIT press (2020)

[4] de Anda-Suárez, J., Carpio-Valadez, J.M., Puga-Soberanes, H.J., Calzada-Ledesma, V., Rojas-Domínguez, A., Jeyakumar, S., Espinal, A.: Symmetric-approximation energy-based estimation of distribution (SEED): a continuous optimization algorithm. IEEE Access **7**, 154859–154871 (2019)

[5] Ansarullah, S.I., Mohsin Saif, S., Abdul Basit Andrabi, S., Kumhar, S.H., Kirmani, M.M., Kumar, D., et al.: An Intelligent and Reliable Hyperparameter Optimization Machine Learning Model for Early Heart Disease Assessment Using Imperative Risk Attributes. Journal of healthcare engineering **2022** (2022)

[6] Asadi, S., Hadavandi, E., Mehmanpazir, F., Nakhostin, M.M.: Hybridization of evolutionary Levenberg-Marquardt neural networks and data pre-processing for stock market prediction. Knowledge-Based Systems **35**, 245–258 (2012)

[7] Atsalakis, G.S., Valavanis, K.P.: Surveying stock market forecasting techniques–part ii: Soft computing methods. Expert systems with applications **36**(3), 5932–5941 (2009)

[8] Bakhoda, A., Yuan, G.L., Fung, W.W., Wong, H., Aamodt, T.M.: Analyzing CUDA workloads using a detailed GPU simulator. In: 2009 IEEE international symposium on performance analysis of systems and software. pp. 163–174. IEEE (2009)

[9] Banik, S., Sharma, N., Mangla, M., Mohanty, S.N., Shitharth, S.: LSTM based decision support system for swing trading in stock market. Knowledge-Based Systems **239**, 107994 (2022)

[10] Bas, E., Egrioglu, E., Kolemen, E.: Training simple recurrent deep artificial neural network for forecasting using particle swarm optimization. Granular Computing **7**(2), 411–420 (2022)

[11] Bazrkar, M.J., Hosseini, S.: Predict Stock Prices Using Supervised Learning Algorithms and Particle Swarm Optimization algorithm. Computational Economics pp. 1–22 (2022)

[12] Berzal, F.: Redes neuronales & deep learning: Volumen I & II. Independently published (2018)

[13] Bhandari, H.N., Rimal, B., Pokhrel, N.R., Rimal, R., Dahal, K.R., Khatri, R.K.: Predicting stock market index using LSTM. Machine Learning with Applications p. 100320 (2022)

[14] Bisoi, R., Dash, P.K.: A hybrid evolutionary dynamic neural network for stock market trend analysis and prediction using unscented kalman filter. Applied Soft Computing **19**, 41–56 (2014)

[15] Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time series analysis: forecasting and control. John Wiley & Sons (2015)

[16] Brasileiro, R.C., Souza, V.L., Fernandes, B.J., Oliveira, A.L.: Automatic method for stock trading combining technical analysis and the artificial bee colony algorithm. In: 2013 IEEE Congress on Evolutionary Computation. pp. 1810–1817. IEEE (2013)

[17] Bustos, O., Pomares-Quimbaya, A.: Stock market movement forecast: A Systematic review. Expert Systems with Applications **156**, 113464 (2020)

[18] Cavalcante, R.C., Brasileiro, R.C., Souza, V.L., Nobrega, J.P., Oliveira, A.L.: Computational intelligence and financial markets: A survey and future directions. Expert Systems with Applications **55**, 194–211 (2016)

[19] Chakraborty, P., Pria, U.S., Rony, M.R.A.H., Majumdar, M.A.: Predicting stock movement using sentiment analysis of Twitter feed. In: 2017 6th International Conference on Informatics, Electronics and Vision & 2017 7th International Symposium in Computational Medical and Health Technology (ICIEV-ISCMHT). pp. 1–6. IEEE (2017)

[20] Chande, T.S.: Beyond technical analysis: How to develop and implement a winning trading system, vol. 2. John Wiley & Sons (1999)

[21] Chang, P.C., et al.: A novel model by evolving partially connected neural network for stock price trend forecasting. Expert Systems with Applications **39**(1), 611–620 (2012)

[22] Chen, J.: SVM application of financial time series forecasting using empirical technical indicators. In: 2010 International Conference on Information, Networking and Automation (ICINA). vol. 1, pp. V1–77. IEEE (2010)

[23] Chen, S., Wu, J., Chen, X.: Deep Reinforcement Learning with Model-Based Acceleration for Hyperparameter Optimization. In: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). pp. 170–177 (2019). https://doi.org/10.1109/ICTAI.2019.00032

[24] Chen, W., Jiang, M., Zhang, W.G., Chen, Z.: A novel graph convolutional feature based convolutional neural network for stock trend prediction. Information Sciences **556**, 67–94 (2021)

[25] Chen, Y., Wu, J., Zhang, Y., Fu, L., Luo, Y., Liu, Y., Li, L.: Research on Hyperparameter Optimization of Concrete Slump Prediction Model Based on Response Surface Method. Materials **15**(13), 4721 (2022)

[26] Chen, Y., Wu, J., Wu, Z.: China's commercial bank stock price prediction using a novel k-means-LSTM hybrid approach. Expert Systems with Applications **202**, 117370 (2022)

[27] Chiang, W.C., Enke, D., Wu, T., Wang, R.: An adaptive stock index trading decision support system. Expert Systems with Applications **59**, 195–207 (2016)

[28] Chopra, S., Yadav, D., Chopra, A.: Artificial neural networks based Indian stock market price prediction: before and after demonetization. J Swarm Intel Evol Comput **8**(174), 2 (2019)

[29] Chung, H., Shin, K.s.: Genetic algorithm-optimized Long Short-Term Memory network for stock market prediction. Sustainability **10**(10), 3765 (2018)

[30] Colasanto, F., Grilli, L., Santoro, D., Villani, G.: BERT's sentiment score for portfolio optimization: a fine-tuned view in Black and Litterman model. Neural Computing and Applications pp. 1–15 (2022)

[31] Dai, W., Wu, J.Y., Lu, C.J.: Combining nonlinear independent component analysis and neural network for the prediction of asian stock market indexes. Expert systems with applications **39**(4), 4444–4452 (2012)

[32] Demir, S., Mincev, K., Kok, K., Paterakis, N.G.: Introducing technical indicators to electricity price forecasting: A feature engineering study for linear, ensemble, and deep machine learning models. Applied Sciences **10**(1), 255 (2019)

[33] Deng, C., Huang, Y., Hasan, N., Bao, Y.: Multi-step-ahead stock price index forecasting using Long Short-Term Memory Model with Multivariate Empirical Mode Decomposition. Information Sciences (2022)

[34] Deng, S., Xiao, C., Zhu, Y., Tian, Y., Liu, Z., Yang, T.: Dynamic forecasting of the Shanghai Stock Exchange index movement using multiple types of investor sentiment. Applied Soft Computing p. 109132 (2022)

[35] Evans, C., Pappas, K., Xhafa, F.: Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation. Mathematical and Computer Modelling **58**(5-6), 1249–1266 (2013)

[36] Fama, E.F.: Random walks in stock market prices. Financial analysts journal **51**(1), 75–80 (1995)

[37] Ferreira, T.A., Vasconcelos, G.C., Adeodato, P.J.: A new intelligent system methodology for time series forecasting with artificial neural networks. Neural Processing Letters **28**(2), 113–129 (2008)

[38] Feurer, M., Hutter, F.: Hyperparameter optimization. In: Automated machine learning, pp. 3–33. Springer, Cham (2019)

[39] Filomeno Coelho, R., Xiao, M., Guglielmetti, A., Herrera, M., Zhang, W.: Investigation of three genotypes for mixed variable evolutionary optimization. In: Advances in evolutionary and deterministic methods for design, optimization and control in engineering and sciences, pp. 309–319. Springer (2015)

[40] Gendreau, M., Potvin, J.Y., et al.: Handbook of metaheuristics, thir edition, vol. 272. Springer (2019)

[41] Ghasemiyeh, R., Moghdani, R., Sana, S.S.: A hybrid artificial neural network with metaheuristic algorithms for predicting stock price. Cybernetics and Systems **48**(4), 365–392 (2017)

[42] Ghosh, P., Neufeld, A., Sahoo, J.K.: Forecasting directional movements of stock prices for intraday trading using LSTM and random forests. Finance Research Letters **46**, 102280 (2022)

[43] Göçken, M., Özçalıcı, M., Boru, A., Dosdoğru, A.T.: Stock price prediction using hybrid soft computing models incorporating parameter tuning and input variable selection. Neural Computing and Applications **31**(2), 577–592 (2019)

[44] Goldberg, Y.: Neural network methods for natural language processing. Synthesis lectures on human language technologies **10**(1), 1–309 (2017)

[45] Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)

[46] Gujarati, D.N.: Essentials of econometrics. SAGE Publications (2021)

[47] Hadavandi, E., Ghanbari, A., Abbasian-Naghneh, S.: Developing an evolutionary neural network model for stock index forecasting. In: International Conference on Intelligent Computing. pp. 407–415. Springer (2010)

[48] Hafezi, R., Shahrabi, J., Hadavandi, E.: A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price. Applied Soft Computing **29**, 196–210 (2015)

[49] Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation **9**(2), 159–195 (2001)

[50] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[51] Huang, C.F.: A hybrid stock selection model using genetic algorithms and support vector regression. Applied Soft Computing **12**(2), 807–818 (2012)

[52] Huang, Q., Kong, Z., Li, Y., Yang, J., Li, X.: Discovery of trading points based on bayesian modeling of trading rules. World Wide Web **21**(6), 1473–1490 (2018)

[53] Hutter, F., Kotthoff, L., Vanschoren, J.: Automated machine learning: methods, systems, challenges. Springer Nature (2019)

[54] Illa, P.K., Parvathala, B., Sharma, A.K.: Stock price prediction methodology using random forest algorithm and support vector machine. Materials Today: Proceedings **56**, 1776–1782 (2022)

[55] Inthachot, M., Boonjing, V., Intakosum, S.: Artificial neural network and genetic algorithm hybrid intelligence for predicting thai stock price index trend. Computational intelligence and neuroscience **2016** (2016)

[56] Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W.M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al.: Population based training of neural networks. arXiv preprint arXiv:1711.09846 (2017)

[57] Jasemi, M., Kimiagari, A.M., Memariani, A.: A modern neural network model to do stock market timing on the basis of the ancient investment technique of japanese candlestick. Expert Systems with Applications **38**(4), 3884–3890 (2011)

[58] Johanson, M.B., Hughes, E., Timbers, F., Leibo, J.Z.: Emergent Bartering Behaviour in Multi-Agent Reinforcement Learning. arXiv preprint arXiv:2205.06760 (2022)

[59] Kanwal, A., Lau, M.F., Ng, S.P., Sim, K.Y., Chandrasekaran, S.: BiCuDNNLSTM-1dCNN—A hybrid deep learning-based predictive model for stock price prediction. Expert Systems with Applications **202**, 117123 (2022)

[60] Kao, L.J., Chiu, C.C., Lu, C.J., Yang, J.L.: Integration of nonlinear independent component analysis and support vector regression for stock price forecasting. Neurocomputing **99**, 534–542 (2013)

[61] Kilinc, H.C., Haznedar, B.: A Hybrid Model for Streamflow Forecasting in the Basin of Euphrates. Water **14**(1), 80 (2022)

[62] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[63] Koster, R., Balaguer, J., Tacchetti, A., Weinstein, A., Zhu, T., Hauser, O., Williams, D., Campbell-Gillingham, L., Thacker, P., Botvinick, M., et al.: Human-centered mechanism design with Democratic AI. arXiv preprint arXiv:2201.11441 (2022)

[64] Kourentzes, N., Barrow, D.K., Crone, S.F.: Neural network ensemble operators for time series forecasting. Expert Systems with Applications **41**(9), 4235–4244 (2014)

[65] Kristjanpoller, W., Minutolo, M.C.: A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. Expert Systems with Applications **109**, 1–11 (2018)

[66] Kumar, G., Jain, S., Singh, U.P.: Stock market forecasting using computational intelligence: A survey. Archives of Computational Methods in Engineering **28**(3), 1069–1101 (2021)

[67] Kumar, K., Haider, M., Uddin, T.: Enhanced prediction of Intra-day Stock Market Using metaheuristic Optimization on RNN–LSTM Network. New Generation Computing **39**(1), 231–272 (2021)

[68] Kumar, R., Kumar, P., Kumar, Y.: Three stage fusion for effective time series forecasting using Bi-LSTM-ARIMA and improved DE-ABC algorithm. Neural Computing and Applications pp. 1–17 (2022)

[69] Kumar Chandar, S.: Fusion model of wavelet transform and adaptive neuro fuzzy inference system for stock market prediction. Journal of Ambient Intelligence and Humanized Computing pp. 1–9 (2019)

[70] Larrañaga, P., Lozano, J.A.: Estimation of distribution algorithms: A new tool for evolutionary computation, vol. 2. Springer Science & Business Media (2001)

[71] Lasfer, A., El-Baz, H., Zualkernan, I.: Neural network design parameters for forecasting financial time series. In: 2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO). pp. 1–4. IEEE (2013)

[72] Lee, M.C.: Using support vector machine with a hybrid feature selection method to the stock trend prediction. Expert Systems with Applications **36**(8), 10896–10904 (2009)

[73] Li, A.W., Bastos, G.S.: Stock market forecasting using deep learning and technical analysis: A systematic review. IEEE Access **8**, 185232–185242 (2020)

[74] Li, R., Han, T., Song, X.: Stock price index forecasting using a multiscale modelling strategy based on frequency components analysis and intelligent optimization. Applied Soft Computing p. 109089 (2022)

[75] Li, W., Ng, W.W., Wang, T., Pelillo, M., Kwong, S.: HELP: An LSTM-based approach to hyperparameter exploration in neural network learning. Neurocomputing **442**, 161–172 (2021)

[76] Lin, Y., Yan, Y., Xu, J., Liao, Y., Ma, F.: Forecasting stock index price using the CEEMDAN-LSTM model. The North American Journal of Economics and Finance **57**, 101421 (2021)

[77] Lopez-Farias, R., Valdez, S.I., Garcia-Robledo, A.: Parameter Calibration of the Patch Growing Algorithm for Urban Land Change Simulations. In: 2021 Mexican International Conference on Computer Science (ENC). pp. 1–8. IEEE (2021)

[78] Lu, C.J.: Integrating independent component analysis-based denoising scheme with neural network for stock price prediction. Expert Systems with Applications **37**(10), 7056–7064 (2010)

[79] Majhi, B., Anish, C.: Multiobjective optimization based adaptive models with fuzzy decision making for stock market forecasting. Neurocomputing **167**, 502–511 (2015)

[80] Malagrino, L.S., Roman, N.T., Monteiro, A.M.: Forecasting stock market index daily direction: A Bayesian Network approach. Expert Systems with Applications **105**, 11–22 (2018)

[81] Moghar, A., Hamiche, M.: Stock market prediction using LSTM recurrent neural network. Procedia Computer Science **170**, 1168–1173 (2020)

[82] Murphy, J.J.: Study Guide to Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications. Penguin (1999)

[83] Nasukawa, T., Yi, J.: Sentiment analysis: Capturing favorability using natural language processing. In: Proceedings of the 2nd international conference on Knowledge capture. pp. 70–77 (2003)

[84] Nayak, S., Misra, B., Behera, H.: Artificial chemical reaction optimization of neural networks for efficient prediction of stock market indices. Ain Shams Engineering Journal **8**(3), 371–390 (2017)

[85] Neto, M.C.A., Tavares, G., Alves, V.M., Cavalcanti, G.D., Ren, T.I.: Improving financial time series prediction using exogenous series and neural networks committees. In: The 2010 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2010)

[86] Nti, I.K., Adekoya, A.F., Weyori, B.A.: A systematic review of fundamental and technical analysis of stock market predictions. Artificial Intelligence Review **53**(4), 3007–3057 (2020)

[87] Oncharoen, P., Vateekul, P.: Deep learning using risk-reward function for stock market prediction. In: Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence. pp. 556–561 (2018)

[88] Pagolu, V.S., Reddy, K.N., Panda, G., Majhi, B.: Sentiment analysis of twitter data for predicting stock market movements. In: 2016 international conference on signal

processing, communication, power and embedded system (SCOPES). pp. 1345–1350. IEEE (2016)

[89] Pang, X., Zhou, Y., Wang, P., Lin, W., Chang, V.: An innovative neural network approach for stock market prediction. The Journal of Supercomputing **76**(3), 2098–2118 (2020)

[90] Pinto, J.M., Neves, R.F., Horta, N.: Boosting trading strategies performance using VIX indicator together with a dual-objective evolutionary computation optimizer. Expert Systems with Applications **42**(19), 6699–6716 (2015)

[91] Pulido, M., Melin, P., Castillo, O.: Particle swarm optimization of ensemble neural networks with fuzzy aggregation for time series prediction of the Mexican stock exchange. Information Sciences **280**, 188–204 (2014)

[92] Rajab, S., Sharma, V.: An interpretable neuro-fuzzy approach to stock price forecasting. Soft Computing **23**(3), 921–936 (2019)

[93] Ren, R., Wu, D.D., Liu, T.: Forecasting stock market movement direction using sentiment analysis and support vector machine. IEEE Systems Journal **13**(1), 760–770 (2018)

[94] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. nature **323**(6088), 533–536 (1986)

[95] Russell, S.J., Norvig, P.: Inteligencia Artificial: un enfoque moderno. No. 04; Q335, R8y 2004. (2004)

[96] Samuelson, P.A., Basic, U., et al.: Fundamentos del análisis económico (1971)

[97] Senapati, M.R., Das, S., Mishra, S.: A novel model for stock price prediction using hybrid neural network. Journal of the Institution of Engineers (india): Series B **99**(6), 555–563 (2018)

[98] Sezer, O.B., Gudelek, M.U., Ozbayoglu, A.M.: Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied Soft Computing **90**, 106181 (2020)

[99] Shen, W., Guo, X., Wu, C., Wu, D.: Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. Knowledge-Based Systems **24**(3), 378–385 (2011)

[100] Silva, T.R., Li, A.W., Pamplona, E.O.: Automated trading system for stock index using LSTM neural networks and risk management. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020)

[101] Souissi, B., Ghorbel, A.: Upper confidence bound integrated genetic algorithm-optimized Long Short-Term Memory network for click-through rate prediction. Applied Stochastic Models in Business and Industry (2022)

[102] Talbi, E.G.: Metaheuristics: from design to implementation. John Wiley & Sons (2009)

[103] Thakur, N., Karmakar, S., Soni, S.: Time series forecasting for uni-variant data using hybrid GA-OLSTM model and performance evaluations. International Journal of Information Technology pp. 1–6 (2022)

[104] Valdez-Peña, S.I., Hernández-Aguirre, A., Botello-Rionda, S.: Approximating the search distribution to the selection distribution in EDAs. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. pp. 461–468 (2009)

[105] Victoria, A.H., Maragatham, G.: Automatic tuning of hyperparameters using bayesian optimization. Evolving Systems **12**(1), 217–223 (2021)

[106] Viñuela, P.I., León, I.M.G.: Redes de neuronas artificiales: un enfoque práctico. Prentice Hall (2004)

[107] Wang, J., Sun, T., Liu, B., Cao, Y., Wang, D.: Financial markets prediction with deep learning. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 97–104. IEEE (2018)

[108] Wang, L., Wang, F., He, D., Li, K., Liu, Y.: Soft sensing of feed pulp concentration in Thickener Based on Cross Validation LSTM Hyper-parameters Optimization. In: 2021 33rd Chinese Control and Decision Conference (CCDC). pp. 5011–5016. IEEE (2021)

[109] Wen, Y., Lin, P., Nie, X.: Research of stock price prediction based on PCA-LSTM model. In: IOP Conference Series: Materials Science and Engineering. vol. 790, p. 012109. IOP Publishing (2020)

[110] Xiong, T., Bao, Y., Hu, Z., Chiong, R.: Forecasting interval time series using a fully complex-valued RBF neural network with DPSO and PSO algorithms. Information Sciences **305**, 77–92 (2015)

[111] Yan, Z., Zhou, K., Zhu, X., Chen, H.: Application of MEA-LSTM Neural Network in Stock Balance Prediction. In: The International Symposium on Computer Science, Digital Economy and Intelligent Systems. pp. 60–71. Springer (2022)

[112] Yang, X.S.: Nature-inspired computation and swarm intelligence: Algorithms, theory and applications (2020)

[113] Yunpeng, C., Xiaomin, S., Peifa, J.: Probabilistic modeling for continuous EDA with Boltzmann selection and Kullback-Leibeler divergence. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. pp. 389–396 (2006)

[114] Zeybek, S., Pham, D.T., Koç, E., Seçer, A.: An Improved Bees Algorithm for Training Deep Recurrent Networks for Sentiment Classification. Symmetry **13**(8), 1347 (2021)

[115] Zhang, X., Chen, X., Yao, L., Ge, C., Dong, M.: Deep neural network hyperparameter optimization with orthogonal array tuning. In: International conference on neural information processing. pp. 287–295. Springer (2019)

[116] Zheng, S., Trott, A., Srinivasa, S., Parkes, D.C., Socher, R.: The AI economist: Optimal Economic Policy Design via Two-level Deep Reinforcement Learning. arXiv preprint arXiv:2108.02755 (2021)

[117] Zhong, X., Enke, D.: Forecasting daily stock market return using dimensionality reduction. Expert Systems with Applications **67**, 126–139 (2017)

[118] Zhu, T., Li, Y., Li, Z., Guo, Y., Ni, C.: Inter-Hour Forecast of Solar Radiation Based on Long Short-Term Memory with Attention Mechanism and Genetic Algorithm. Energies **15**(3), 1062 (2022)