Facultad de Administración y Dirección de Empresas

Trabajo de Fin de Grado

Aplicación de técnicas de Machine Learning para el análisis y predicción de acciones: The Coca-Cola Company

Alicia Pastor Fernández

2024/2025

Trabajo de Fin de Grado presentado en la Facultad de Administración y Dirección de Empresas de la Universidad de Santiago de Compostela para la obtención del Grado en Empresa y Tecnología

Agradecimientos

Me gustaría mostrar mi más sentido agradecimiento:

A mi madre y a mi hermana,

Por enseñarme el amor más puro y sincero. Por confiar en mis decisiones, apoyarme sin condiciones y ofrecerme siempre vuestro consejo cuando más lo necesitaba.

A mis amigos,

Por enseñarme a sentirme en casa incluso lejos de ella. Por acogerme como una gallega más y hacer que Lugo también forme parte de mí.

A mi mejor amiga,

Por escucharme y entenderme, incluso cuando ni yo misma sabía explicarme. Por seguir a mi lado a pesar de la distancia. Por estar presente en los momentos más difíciles y seguir siendo, tras tantos años, mi mayor apoyo.

A mi pareja,

Por enseñarme lo que es el amor en su forma más sincera. Por estar a mi lado incluso cuando todo parecía cuesta arriba. Por ser refugio en medio del caos y alegría en los días grises.

A mi perro,

Por ser mi compañero más leal. Por acompañarme en cada aventura y por hacer especiales nuestros paseos. Por recordarme que la felicidad también está en las cosas pequeñas.

Y, por último, a mi padre,

Por ser mi ejemplo de esfuerzo, amor y perseverancia. Por confiar en mí durante estos cuatro años, desde aquel primer día camino a Lugo, hasta el último. Por mantener la fe en mí incluso cuando yo misma la perdía. Por apoyarme incondicionalmente en cada decisión y ofrecerme siempre tus consejos y tu tiempo, sin pedir nunca nada a cambio. Por estar orgulloso de mí como hija, y por hacerme sentir profundamente afortunada de tenerte como padre.

Resumen

Este Trabajo de Fin de Grado tiene como objetivo elaborar un modelo de predicción bursátil fundamentado en métodos de ML, que resulte exacto, comprensible y sencillo de entender. Para ello, se realizó un análisis de varios algoritmos supervisados utilizados en el sector financiero, resaltando su habilidad para identificar patrones complejos en datos de tiempo real, algo que sobrepasa las restricciones de los modelos estadísticos convencionales.

A continuación, se pusieron en práctica y contrastaron varios modelos en Python, tales como regresión lineal, árboles de decisión, random forest y redes neuronales, empleando información histórica auténtica del mercado bursátil. Cada modelo fue analizado a través de métricas de evaluación, lo que facilitó la identificación del más eficiente.

El trabajo se acompaña con representaciones gráficas y cuadros comparativos que simplifican la interpretación de los resultados, guiando el proyecto no solo hacia una perspectiva técnica, sino también pragmática. Se intentó desarrollar un instrumento entendible para usuarios no especialistas, que puedan emplearlo como soporte para calcular el precio futuro de las acciones.

Finalmente, se sugieren estrategias futuras de optimización, tales como la utilización de datos externos, la incorporación de modelos mixtos y la construcción de una interfaz más intuitiva. En resumen, este proyecto evidencia la capacidad del aprendizaje automático en el estudio financiero y su uso efectivo en la toma de decisiones en el mercado bursátil.

El número de palabras del presente trabajo es de 10.908.

Índice

Índice de abreviaturas

AI.	Artificial Intelligence
ANN.	Artificial Neural Network
DNN.	Deep Neural Network
DP.	Deep Learning
KNN.	K-Nearest Neighbors
LSTM.	Long Short-Term Memory
MAE.	Mean Absolute Error
MAPE.	Mean Absolute Percentage Error
ML.	Machine Learning
MSE.	Mean Squared Error
RF.	Random Forest
RMSE.	Root Mean Squared Error
SMAPE.	Symmetric Mean Absolute Percentage Error

Índice de tablas

Índice de figuras

1 INTRODUCCIÓN

El aprendizaje automático (Machine Learning, ML) ha demostrado ser una herramienta eficaz en el ámbito financiero, especialmente en la predicción de precios bursátiles, al superar las limitaciones de los modelos estadísticos tradicionales frente a datos no lineales y ruidosos (Nabipour et al., 2020; Rundo et al., 2019).

Este proyecto propone aplicar y comparar distintos algoritmos de ML con el fin de desarrollar un modelo predictivo de precios de acciones que sea preciso, accesible para usuarios no expertos y acompañado de visualizaciones claras que faciliten su interpretación (Sarker, 2021).

1.1. Contexto y motivación del proyecto

El mercado bursátil es un ambiente complejo y cambiante, en el que anticipar el comportamiento futuro de las acciones supone un desafío significativo. Históricamente, este tipo de análisis se ha tratado a través de modelos estadísticos, los cuales, a pesar de ser útiles, suelen ser insuficientes para identificar relaciones no lineales o fluctuaciones bruscas del mercado. En este escenario, los modelos de aprendizaje automático han evidenciado una superior habilidad para ajustarse a datos complejos y aumentar la exactitud de las proyecciones (Nabipour et al., 2020; Sarker, 2021).

Por lo tanto, el objetivo principal de este estudio es utilizar estas técnicas para elaborar un modelo predictivo que no solo sea exacto, sino que también sea accesible y sencillo de utilizar para cualquier individuo interesado en calcular el precio futuro de las acciones de una empresa multinacional como The Coca-Cola Company. La motivación no solo consiste en contrastar modelos desde una perspectiva técnica, sino en desarrollar una herramienta útil que pueda ser empleada por inversores individuales o cualquier usuario sin habilidades avanzadas en programación.

Este método tiene como objetivo democratizar el acceso a la predicción bursátil a través de una solución simple, práctica y fundamentada en datos reales, que fusiona la potencia del aprendizaje automático con una implementación nítida en Python.

1.2. Objetivos del TFG

El propósito principal de este trabajo es desarrollar, aplicar y perfeccionar un modelo de predicción bursátil fundamentado en métodos de aprendizaje automático.

Basándonos en este objetivo general, se proponen los siguientes objetivos particulares:

- Examinar los modelos de aprendizaje automático más empleados en labores de proyección financiera, resaltando sus particularidades, beneficios y restricciones.
- Implementar y entrenar varios de estos modelos en un ambiente real utilizando el lenguaje Python.
- Analizar el desempeño de cada modelo utilizando indicadores objetivos como el RMSE, MAE, basándose en el tipo de salida esperada
- Elegir el modelo que proporcione los resultados más destacados en términos de exactitud y estabilidad.
- Optimizar el modelo seleccionado a través de la modificación de hiperparámetros para mejorar su desempeño.
- Acompañar la evolución del modelo con ilustraciones gráficas y cuadros comparativos que demuestren de manera clara los resultados alcanzados, posibilitando una interpretación visual e intuitiva del desempeño de cada algoritmo.

En resumen, el proyecto tiene como objetivo implementar el aprendizaje automático en el análisis bursátil a través de un modelo predictivo exacto, accesible y de fácil interpretación, complementado con resultados visuales que simplifiquen su entendimiento.

Este trabajo se organiza de manera gradual para simplificar su entendimiento.

- La sección 2 (Marco Teórico) trata los principios básicos del mercado bursátil, el aprendizaje automático y los métodos de pronóstico y valoración de modelos.
- El diseño del sistema de predicción, la recolección de datos y las herramientas utilizadas se detallan en la sección 3 (Metodología).
- La sección 4 (Desarrollo del Modelo) detalla la exploración de datos, puesta en marcha, capacitación, verificación y optimización.
- La sección 5 (Resultados) expone las proyecciones y el estudio de exactitud.
- La sección 6 (Aplicación Práctica) detalla el uso del modelo para realizar decisiones de inversión y sus restricciones.

- Finalmente, la sección 7 (Conclusiones) compila los objetivos logrados, contribuciones realizadas y futuras estrategias de investigación.

2 MARCO TEÓRICO

Primeramente, se debe definir la base teórica para poder comprender la metodología aplicada en este trabajo. Se van a abordar los conceptos básicos del análisis de carteras de inversión, junto al concepto de ML, sus técnicas de predicción, y, por último, lo métodos de evaluación existentes.

2.1 Fundamentos del mercado de valores

El mercado de valores tiene un papel fundamental sobre el desarrollo económico, ya que, proporciona un sistema de asignación de recursos financieros (Atje & Jovanovic, 1993).

Una definición conceptual del mercado de valores es la siguiente: institución central que, a través de la organización corporativa, permite reunir los pequeños ahorros de miles de individuos para transformarlos en grandes sumas de capital. Este capital, una vez acumulado, es dirigido por los grandes actores financieros, lo que convierte al mercado bursátil en el barómetro que refleja la prosperidad de la economía de una nación (Huebner, 1910).

Alrededor del concepto del mercado de valores, existen otras ideas también relevantes que evalúan el desarrollo de los mercados financieros. Primeramente, la liquidez, indica la facilidad con la que se pueden comprar y vender valores en el mercado sin generar un impacto significativo en los precios; se destaca que mercados de mayor tamaño tienden a ser más líquidos (Demirgiic-Kunt & Levine, 1996). Por otro lado, está la volatilidad, entendida como la variación de los precios de los valores a lo largo del tiempo sirviendo de medida para la estabilidad o inestabilidad del mercado (Demirgiic-Kunt & Levine, 1996). Por último, la integración internacional, referida al grado de conexión del mercado de valores con los mercados globales, lo que implica un acceso ampliado a capitales internacionales y una mayor interacción en el ámbito financiero mundial (Demirgiic-Kunt & Levine, 1996).

Después de analizar conceptos relacionados con el mercado financiero, se puede entender la diferencia que existe entre dicho mercado y otros mercados pertenecientes a otros sectores. En el mercado financiero existe un alto grado de volatilidad y riesgo, ya que, el precio de las acciones, bonos, etc. fluctúan dependiendo del comportamiento de las empresas, de los mercados financieros o de otros factores internacionales. Esto puede conllevar a altos

rendimientos sujetos a elevados riesgos. El riesgo en el mercado de valores es no predecible y está sujeto a especulaciones futuras (Demirgiic-Kunt & Levine, 1996).

2.2 Introducción al aprendizaje automático

Existen muchas definiciones sobre aprendizaje automáticos, ya que, es una tecnología novedosa y que aún hoy en día se siguen descubriendo aspectos de ella. Una definición sobre este concepto sería el siguiente: el ML, se define como la capacidad de los sistemas computacionales para aprender a partir de datos, mejorando su desempeño en tareas específicas sin necesidad de ser reprogramados explícitamente (Janiesch et al., 2021).

Su funcionamiento es distinto al de otras maquinas, los algoritmos de ML extraen patrones, regularidades y relaciones complejas directamente de los datos de entrenamiento, facilitando así el trabajo con conjuntos de datos de gran volumen y dimensionalidad (Janiesch et al., 2021).

Profundizando más sobre el concepto, es necesario conocer la jerarquía entre las disciplinas existentes en el campo de la AI (Ver Figura 1).

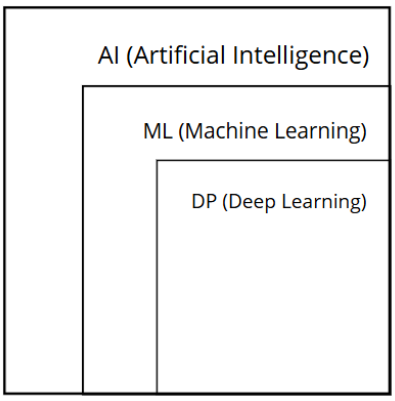

Figura 1. Jerarquía dentro de la AI. Elaboración propia

En los primeros desarrollos dentro del campo de la AI se centraban en desarrollar una tecnología que dotara a los ordenadores de capacidad para imitar el comportamiento humano, permitiendo así replicar u optimizar la toma de decisiones basados en humanos. Esto permitía que las maquinas tomaran decisiones independientes o con poca influencia humana. Su funcionamiento se basaba en la generación de enunciados codificados en lenguajes formales, dónde un ordenador puede razonar basándose en unas reglas de inferencia lógica. No obstante, este arquetipo se enfrentaba a múltiples limitaciones, entre ellos, la dificultad de los humanos

para explicar el conocimiento implícito para la realización correcta de tareas complejas (Janiesch et al., 2021).

El ML supera estas limitaciones. En líneas generales, el ML puede optimizar el rendimiento de cualquier programa informático mediante el desarrollo de experiencia basado en el procesamiento previo de múltiples tareas. El funcionamiento de este modelo se basa en tratar de aplicar determinados algoritmos que aprenden de manera autónoma a partir de unos datos de entrenamiento específicos, permitiendo al ordenador encontrar información o patrones ocultos por debajo de los datos, sin la necesidad de una previa programación explicita (Janiesch et al., 2021).

Los casos en los que este modelo aumenta la optimización con relación al resto, es en aquellos dónde los datos tienen elevada dimensión cómo la clasificación, regresión y agrupación. En función del problema y de los datos disponibles, se distinguen cuatro tipos de ML (Sarker, 2021):

- Aprendizaje supervisado. El modelo trabaja con datos que vienen etiquetados. Es utilizado para tareas de clasificación y regresión, dónde el modelo predice una etiqueta de salida para cada dato de entrada.
- Aprendizaje no supervisado. Este enfoque se trabaja con datos sin etiquetas, por lo tanto, solo pueden describir estructuras o patrones subyacentes en el conjunto de datos. En este caso, las técnicas utilizadas son el clustering o la reducción de dimensionalidad.
- Aprendizaje semi-supervisado. Este método combina elementos del aprendizaje supervisado y no supervisado. Se utiliza cuando solo una parte de los datos cuenta con etiquetas, aprovecha la información de ambos grupos de datos para mejorar la precisión del modelo. Este método es el menos utilizado, ya que, la mayoría de los conjuntos de datos suelen pertenecer a una misma categoría.
- Aprendizaje por refuerzo. En este último caso, un agente externo interactúa con su entorno y aprende a tomar decisiones basadas en la retroalimentación que recibe en forma de recompensas o penalizaciones, permitiendo así aprender de experiencias pasadas para actuar en el futuro.

Una vez habiendo analizado el concepto en profundidad de ML, debe analizarse un submodelo llamado DP.

Primeramente, se deben tener en cuenta dos conceptos que diferencian ambos modelos: Artificial Neural Network (ANN) y Deep Neural Network (DNN) (Ver Figura 2).

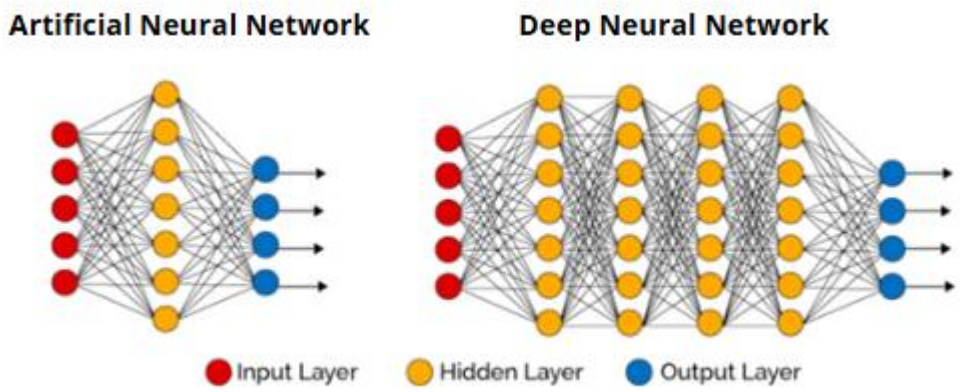

Figura 2. ANN y DNN. Adaptado de (Yang et al., 2023)

La red neuronal artificial, ANN, está inspirada en el principio de procesamiento de información en sistemas biológicos, las ANN representan las unidades de procesamiento presentes en el cerebro humano, llamadas neuronas artificiales. Al igual que las sinapsis en un cerebro, cada conexión entre neuronas transmite señales. Dichas señales solo son procesadas por neuronas posteriores si se supera un determinado umbral según una función de activación. Por lo general, las neuronas se organizan en redes de diferentes capas. La capa de entrada recibe datos del exterior y la capa de salida produce el resultado final. En el medio, puede haber cero o más capas ocultas que son responsables de procesar la información (Janiesch et al., 2021).

La red neuronal profunda, DNN, en cambio, suelen constar de más de una capa oculta organizada. Suelen contener neuronas avanzadas a diferencia de las ANN simple. Si en el anterior tipo se habla de una función de activación simple, en DNN se utilizan operaciones avanzadas o múltiples activaciones en una neurona. Estas características permiten que las DNN se alimenten con datos de entrada sin procesar y descubran una representación necesaria para la tarea de aprendizaje correspondiente (Janiesch et al., 2021).

La diferencia entre ML y DP radica en la utilización de los tipos de redes neuronales. Por un lado, el ML emplea ANN para solucionar tareas como la clasificación y la regresión. No obstante, cuando el objetivo es manejar datos más complejos, se emplea el DNN.

El ML comprende una variedad de técnicas, incluyendo el DP. Estos modelos tienen la capacidad de realizar tareas de gran complejidad y sobrepasan a los algoritmos de ML en las aplicaciones que manejan datos de texto, imagen, video o audio (Janiesch et al., 2021).

2.3 Técnicas de predicción en series temporales

En este apartado se desarrollan diversas técnicas de predicción de series temporales, abarcando procedimientos tradicionales y otros más innovadores integrados en múltiples aplicaciones informáticas.

2.3.1 Métodos estadísticos tradicionales

Estos métodos han sido la base de análisis de series temporales en finanzas, es importante tenerlos en cuenta por su elevada importancia e influencia sobre modelos actuales (Rundo et al., 2019):

- ARIMA (AutoRegressive Integrated Moving Average). Este modelo es una generalización del modelo ARMA, su principal ventaja es la capacidad de transformar una serie no estacionaria en estacionaria mediante diferencias finitas de los puntos de datos.
- GRACH (Generalized Autoregressive Conditional Heteroskedasticity). Esta modelo amplia ARIMA, ya que, permite capturar la heterocedasticidad condicional en los datos, es decir, permite modelar y predecir la volatilidad cambiante en el tiempo.

2.3.2 Métodos de Aprendizaje Supervisado

Estos procedimientos tratan la predicción de tendencias desde un enfoque de clasificación o regresión, mediante indicadores técnicos derivados de información histórica. Entre ellos se incluyen (Nabipour et al., 2020):

- Árbol de decisión (Decision Tree). Modelo basado en la partición del espacio de características mediante umbrales, generado un árbol en el que cada hoja representa una decisión o predicción. La facilidad de interpretación del modelo es una ventaja de este; por el contrario, la construcción de árboles demasiado complejos provoca un sobreajuste.

- Random Forest (RF). Conjunto de árboles de decisión con aleatoriedad en tres niveles: muestra de datos, selección de variables por nodo y subconjunto de variables en cada división. Esto mejora la diversidad y robustez del modelo.
- Support Vector Classifier (SVC). Basado en encontrar el hiperplano optimo que separa las clases en el espacio de características, maximizando el margen entre ellas.
- Naïve Bayes. Es un tipo de clasificador probabilístico basado en el teorema de Bayes, el cual asume fuerte independencia entre las características a causa del valor de la variable de clase.
- K-Nearest Neighbors (KNN). Es un algoritmo de clasificación que atribuye una clase a una nueva muestra basándose en la mayoría de sus K vecinos más próximos, establecidos por una métrica de distancia concreta (Prasath et al., 2017).
- Regresión Logística (Logistic Regression). Es un clasificador que asigna observaciones a distintas clases. Utiliza un tipo de función característico el cual permite transformar su salida en un valor de probabilidad (entre 0 y 1).
- Red Neuronal Artificial (ANN). Es un clasificador caracterizado por estar compuesto de múltiples capas conectadas entre sí. A medida que aumenta el número de capas ocultas, la red se convierte más profunda y es capaz de aprender representaciones más complejas.

2.3.3 Métodos de Deep Learning

Estos métodos permiten capturar dependencias temporales y relaciones complejas en series secuenciales (Nabipour et al., 2020).

- Recurrent Neural Networks (RNN). Diseñadas para mejorar datos en secuencia, las RNN tienen conexiones repetitivas que permiten que los datos anteriores afecten la salida actual. Son apropiadas para modelas series temporales, aunque pueden tener problemas de desvanecimiento de gradiente.
- Long Short-Term Memory (LSTM). Corresponde a una versión de las RNN que incluye mecanismos de puertas para mantener la información en

secuencias largas, superando ciertas limitaciones de las RNN convencionales.

2.3.4 Técnicas de Ensamblado

Estos métodos combinan múltiples modelos de aprendizaje para mejorar la precisión y robustez de las predicciones en comparación a los modelos individuales. Algunos de estos métodos son (Lin et al., 2012):

- Bagging. En este método, se crean diversos subconjuntos de datos a partir del conjunto de entrenamiento a través de muestreo con reemplazo. Basándose en estos subconjuntos, se capacitan diferentes clasificadores autónomos y finalmente, se consigue la predicción final mediante la medición de sus resultados o mediante el uso de votación mayoritaria.
- Boosting. A diferencia del Bagging, en el boosting los clasificadores no se capacitan de manera independiente, sino de manera secuencial. Cada clasificador se capacita basándose en los fallos cometidos por el anterior, otorgándole mayor relevancia a las instancias que no fueron correctamente clasificadas. Al finalizar el proceso, los modelos se fusionan evaluando su exactitud para generar una predicción final sólida.

2.4 Métodos de evaluación de modelos

La evaluación de modelos es fundamental para medir la precisión y robustez de las predicciones. En este contexto, se utilizan métricas que comparan el valor de los resultados predichos y los observados. En esta sección van a analizarse el conjunto de métricas más utilizadas, permitiendo generar una visión conjunto de los métodos existentes (Botchkarev, 2019; Chai & Draxler, 2014).

- Mean Absolute Erros (MAE). Se define como la media aritmética de las diferencias absolutas entre los valores predichos y los observados. Es decir, mide el error promedio sin distinguir la dirección del error. Esta métrica es intuitiva y fácil de interpretar.
- Mean Squared Erros (MSE). Se define como la media de los errores al cuadrado, que es adquirida al incrementar al cuadrado la diferencia entre los valores estimados y los reales y posteriormente,

promediándolas. Esta métrica penaliza con dureza los errores significativos, ya que, estos se elevan al cuadrado.

- Root Mean Squared Error (RMSE). Se obtiene aplicando la raíz cuadrada al MSE. Refleja la dispersión de los errores, siendo sensible a errores atípicos.
- Coeficiente de Determinación (R^2). Se define como el porcentaje de la varianza total de la variable dependiente que el modelo explica. Esta medida ofrece una evaluación global de la calidad del ajuste.
- Mean Absolute Percentage Error (MAPE). Representa el promedio de los errores absolutos expresados en términos porcentuales, lo que facilita la valoración de la magnitud del error con relación a los valores reales. No obstante, su aplicación pierde efectividad cuando los valores son próximos a cero, dado que pueden surgir porcentajes extremadamente elevados o inestables.
- Symmetric Mean Absolute Percentage Error (SMAPE). SMAPE se plantea como una variante de MAPE que busca corregir ciertos sesgos, mediante una formulación que simetriza la diferencia entre los valores observados y predichos. Esto posibilita conseguir una evaluación más balanceada de los errores porcentuales.

3 METODOLOGÍA

El presente apartado detalla la metodología empleada para la creación del sistema de predicción bursátil. Se tratan todas las fases que conforman el proceso: desde la organización global del sistema, a través de la recolección y tratamiento de los datos, hasta las herramientas y tecnologías empleados en la implementación.

3.1 Diseño del sistema de predicción

El propósito del sistema de predicción que se pretende desarrollar para este proyecto es proyectar el comportamiento futuro del precio de acciones bursátiles, en concreto, predecir el valor de cierre de los días próximos en función de sus datos históricos. Para ello, se han probado diversas técnicas de ML y análisis de series temporales, evaluando su eficacia en términos de precisión predictiva.

La estructura del sistema se organiza en distintas fases, las cuales se representan en la Figura 3:

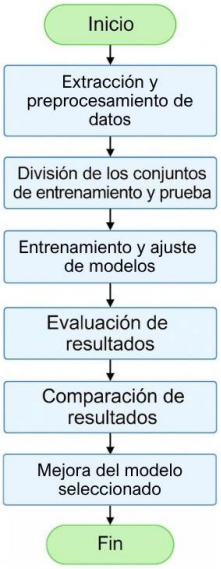

Figura 3. Diagrama de flujo sobre fases de proyecto. Elaboración propia.

Se han considerado y probado distintos algoritmos, todos ellos explicados previamente en el Marco Teórico de este proyecto:

- Modelos estadísticos como ARIMA
- Métodos de Aprendizaje Supervisado como Regresión Lineal y Logística, Árbol de Decisión, RF y ANN
- Métodos de DP como LSTM
- Diversas técnicas de ensamblado como Bagging

Por último, los Métodos de evaluación de modelos utilizados fueron:

- Mean Absolute Erros (MAE)
- Mean Squared Erros (MSE)
- Coeficiente de Determinación (R^2)
- Root Mean Squared Error (RMSE)

El diseño del sistema se ha basado en un enfoque iterativo y comparativo que ha permitido evaluar distintos modelos de predicción y seleccionar el más adecuada en función de su rendimiento.

3.2 Selección y obtención de los datos

Para el desarrollo del sistema de predicción se utilizaron datos históricos del recio de acciones obtenidos mediante la librería *Yahooquery,* una API que permite extraer información financiera actualizada y fiable directamente de Yahoo Finance. Se optó por utilizar esta herramienta como fuente de datos tras revisar el estudio Nabipour et al., (2020), donde se empleaba esta misma fuente para la predicción bursátil.

Para la creación del sistema de predicción de este proyecto se seleccionó la empresa The Coca-Cola Company como activo de referencia, por su solidez y a la estabilidad que ha mostrado históricamente en su comportamiento. Según el análisis actualizado de Macroaxis, (2025), Coca-Cola presenta una volatilidad histórica baja, con un coeficiente de skewness de 0.34 y de kurtosis de 1.75, valores que reflejan una distribución simétrica y con colas poco extremas. Adicionalmente, el interés por Coca-Cola como activo para estudios predictivos está ampliamente respaldado en la literatura académica, uno de estos es el estudio de Dong, (2024) Coca-Cola es identificada como una de las compañías más apropiadas para la aplicación de modelos de ML, debido a la alta calidad y profundidad de sus datos históricos, su inclusión en el índice S&P 500 y su perfil como blue-chip consolidada. Los autores subrayan que este tipo de activos es especialmente adecuado para validar estrategias predictivas por su comportamiento relativamente predecible frente a compañías de alta volatilidad o de reciente creación. Otro aspecto relevante es el carácter multinacional de Coca-Cola, con operaciones en más de 200 países (The Coca Cola Company, 2025). Esto permite enriquecer los modelos predictivos incorporando variables macroeconómicas internacionales (como el índice VIX, los tipos de cambio USD/EUR o la evolución de otros índices bursátiles como el IBEX35), dotando al sistema de predicción de una capacidad de generalización superior y de mayor aplicabilidad en escenarios globales.

Los datos abarcan el periodo comprendido entre enero del 2015 y enero del 2025 y su evolución se observa en la Figura 4:

Figura 4. Evolución precios 2015- 2025 KO. Fuente: (Yahoo! Finance, 2025)

3.3 Herramientas y tecnologías utilizadas

Para el desarrollo del sistema de predicción se empleó el lenguaje de programación Python 3.11.7, por su versatilidad y amplio ecosistema de librerías especializadas en análisis de datos, ML y representación de datos. La implementación se realizó íntegramente en notebook de Jupyter, lo que permitió combinar líneas de código, resultados, facilitando la ejecución individual de los diferentes modelos.

Entre las librerías utilizadas se encuentran:

- Yahoquery: utilizado para la descarga automática de datos financieros desde Yahoo Finance.
- Pandas y Numpy: útiles para el procesamiento, limpieza y transformación de datos.
- Matplotlib y seaborn: utilizadas para la visualización de resultados y la generación de gráficos comparativos entre valores reales y predichos.
- Scikit-learn: utilizada en la implementación de modelos tradicionales de ML, como regresión lineal, logística y RF, además de funciones de particionado de datos y cálculo de métricas de evaluación.
- Statsmodels: útil para la creación y ajuste de modelos ARIMA.
- Tensorflow y Keras: útil para la creación de modelos de DP, como el modelo LSTM.

El entorno de trabajo fue ejecutado de forma local, aunque las pruebas fueron diseñadas para ser fácilmente reproducibles en entornos en la nube como Google Colab para facilitar su uso y compresión.

Gracias a esta combinación de herramientas, fue posible implementar un flujo completo de trabajo: desde la adquisición de datos, pasando por el procesamiento de estos, por la creación de los modelos hasta la evaluación y comparación de los distintos modelos generados, garantizando su desarrollo modular y fácilmente comprensible para su posterior uso.

4 DESARROLLO DEL MODELO

En este capítulo se describe de manera detallada la metodología empelada para el desarrollo del sistema de predicción del precio de las acciones. El objetivo ha sido diseñar un pipeline completo, desde la obtención de los datos históricos hasta la generación de predicciones y su evaluación cuantitativa.

4.1 Exploración y análisis de los datos

Antes de comenzar a entrenar los diversos modelos predictivos, se llevó a cabo una exploración detallada de los datos históricos obtenidos de la acción de CocaCola, su abreviatura reconocible por la API de yahooquery es KO.

El objetivo principal de esta fase fue tratar de comprender la estructura del conjunto de datos, identificar patrones y detectar posibles inconsistencias o valores atípicos que pudieran afectar al rendimiento de los modelos.

Antes de abordar la fase del modelado predictivo, re realizó un análisis exploratorio para identificar qué variables del entorno financiero tienen un impacto más significativo sobre la acción de Coca-Cola (KO).

Por ello, se aplicó un modelo de regresión lineal (OLS), utilizando como variable dependiente el rendimiento de KO (Rend_KO) y como variables explicativas: Rend_IBEX, Rend_S&P500, Rend_VIX, Rend_EUR/USD y CPI_YoY. Tres de estas variables pertenecen a mercados de valores, otra, un tipo de cambio de moneda y el último, el Índice de Precios al Consumo (CPI) medido en variación interanual (YoY), es un indicador de inflación.

```
Linear regression (OLS)
Data     : Base_de_datos_def_TFG_Alicia_Pastor
Response variable    : Rend_KO
Explanatory variables: Rend_IBEX, Rend_S_P500, Rend_VIX, Rend_EUR_USD, CPI_YoY
Null hyp.: the effect of x on Rend_KO is zero
Alt. hyp.: the effect of x on Rend_KO is not zero

             coefficient std.error t.value p.value
 (Intercept)       0.000     0.000   0.162   0.871
 Rend_IBEX        -0.028     0.017  -1.635   0.102
 Rend_S_P500      -0.096     0.025  -3.775  < .001 ***
 Rend_VIX         -0.010     0.003  -2.930   0.003 **
 Rend_EUR_USD      0.082     0.037   2.195   0.028 *
 CPI_YoY           0.000     0.000   0.597   0.551

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-squared: 0.008,  Adjusted R-squared: 0.006
F-statistic: 5.644 df(5,3639), p.value < .001
Nr obs: 3,645
```

Figura 5. Análisis de variables exploratorias. Elaboración propia

El resumen de la regresión obtenida se puede observar en la Figura 4, y los resultados obtenidos son los siguientes:

- Rend_S&P500 muestra un impacto estadísticamente significativo sobre el rendimiento de KO ($p < 0.001$), con un coeficiente negativo (-0.096). Esto indica que cuando el S&P500 baja, el rendimiento de Coca- Cola tiende a bajar ligeramente (Gareth James, 2021).
- Rend_VIX también resulta significativo ($p = 0.003$), con un efecto negativo. Esto significa que, un aumento de la volatilidad del mercado (VIX) suele perjudicar a KO, aunque sea una empresa estable (Gareth James, 2021).
- Rend_EUR/USD presenta un efecto positivo y significativo ($p = 0.028$). Cuando el euro se aprecia frente al dólar, el rendimiento de KO mejora, probablemente porque una parte de sus ingresos proviene del extranjero (Gareth James, 2021).
- Las demás variables (Rend_IBEX y CPI_YoY) no muestran significación estadística en este modelo, lo que indica que su influencia es limitada o nula a nivel diario.

Este análisis preliminar sugiere que existen factores del entorno macroeconómico y de los mercados internacionales que influyen en la evolución de la acción de Coca-Cola, destacando en particular la relación con el S&P 500 y el tipo de cambio EUR/USD.

El dataset original cuenta con 2516 filas y siete columnas, representadas en la Tabla 1:

VARIABLES	SIGNIFICADO	TIPO
Symbol	Símbolo de acción	float64
Open	Precio de apertura	float64
High	Precio máximo	float64
Low	Precio mínimo	float64
Close	Precio de cierre	float64
Adjclose	Precio ajustado	float64
Volume	Volumen de operaciones	int64
Dividends	Dividendos	float64

Tabla 1. Variables de la data set original. Elaboración propia.

La mayoría de las variables son de tipo decimal, esta representación tiene sentido, ya que, seis variables son continuas y representan precios en el mercado bursátil que no son enteros. Por otro lado, la variable volumen, la cual, representa la cantidad de acciones que se han negociado en un día, debe ser número enteros, ya que, no se pueden comprar 0,5 acciones.

Durante el análisis preliminar, se observó que las variables presentaban un comportamiento coherente con la evolución normal de un activo financiero líquido, con variaciones propias del mercado, pero sin la presencia de valores extremos o vacíos, lo cual facilito su tratamiento.

Las primeras diez filas del dataset original se muestran en la Tabla 2:

date	open	high	low	close	volume	adjclose	dividends
2015-01-02 00:00:00	42.2600	42.4000	41.8000	42.1400	9921100.0000	30.4375	0.0000
2015-01-05 00:00:00	42.6900	42.9700	42.0800	42.1400	26292600.0000	30.4375	0.0000
2015-01-06 00:00:00	42.4100	42.9400	42.2400	42.4600	16897500.0000	30.6686	0.0000
2015-01-07 00:00:00	42.8000	43.1100	42.5800	42.9900	13412300.0000	31.0515	0.0000
2015-01-08 00:00:00	43.1800	43.5700	43.1000	43.5100	21743600.0000	31.4270	0.0000
2015-01-09 00:00:00	43.4700	43.5600	42.9500	43.0300	12733500.0000	31.0803	0.0000
2015-01-12 00:00:00	43.0700	43.2000	42.4600	42.6400	11415800.0000	30.7987	0.0000
2015-01-13 00:00:00	42.8300	43.2400	42.4500	42.6300	12529500.0000	30.7914	0.0000
2015-01-14 00:00:00	42.0800	42.6000	42.0700	42.5600	13447600.0000	30.7409	0.0000
2015-01-15 00:00:00	42.8300	42.8600	42.1700	42.3800	11193100.0000	30.6109	0.0000

Tabla 2. Base de datos preprocesada. Elaboración propia

Tras realizar dicho análisis, se llevaron a cabo las siguientes tareas de análisis y limpieza:

- Conversión de la columna de fechas al formato *datatime* y establecimiento como índice temporal.
- Creación de la variable objetivo (*close_tomorrow*), que representa el precio de cierre del día siguiente. Esta transformación permitió convertir el problema en una tarea supervisada de predicción temporal hacia adelante.
- Eliminación de la última fila del dataset al no disponer de valor objetivo para el día siguiente.
- Eliminación de la columna *symbol* por ser redundante.

Otro procesamiento que se llevó a cabo antes de proceder al entrenamiento de los modelos es la normalización de los datos de las variables tipo *float64*. Se aplicó un escalado con el fin de que todas las variables numéricas formaran parte de un rango comparable y evitar que variables con mayores magnitudes dominen en el proceso de aprendizaje, dando más importancia a las variables más grandes, aunque no sean relevantes. Específicamente, la variable *volumen* tiene valores en el rango de miles, mientras que, el resto de las variables tiene el rango en unidades.

Las primeras diez filas del dataset tras las tareas de limpieza se muestran en la Tabla 3:

date	open	high	low	close	volume	adjclose	dividends
2015-01-02 00:00:00	0.1209	0.1191	0.1537	0.1292	0.1068	0.0584	0.0000
2015-01-05 00:00:00	0.1331	0.1353	0.1615	0.1292	0.3592	0.0584	0.0000
2015-01-06 00:00:00	0.1252	0.1344	0.1659	0.1382	0.2144	0.0637	0.0000
2015-01-07 00:00:00	0.1362	0.1392	0.1754	0.1532	0.1606	0.0725	0.0000
2015-01-08 00:00:00	0.1470	0.1522	0.1898	0.1678	0.2891	0.0811	0.0000
2015-01-09 00:00:00	0.1552	0.1520	0.1857	0.1543	0.1502	0.0731	0.0000
2015-01-12 00:00:00	0.1439	0.1418	0.1720	0.1433	0.1298	0.0667	0.0000
2015-01-13 00:00:00	0.1371	0.1429	0.1718	0.1430	0.1470	0.0665	0.0000
2015-01-14 00:00:00	0.1158	0.1248	0.1612	0.1410	0.1612	0.0653	0.0000
2015-01-15 00:00:00	0.1371	0.1321	0.1640	0.1360	0.1264	0.0623	0.0000

Tabla 3. Base de datos procesada. Elaboración propia

Como último paso antes del entrenamiento de los modelos, el conjunto de datos fue dividido en dos subconjuntos: uno para el entrenamiento del modelo (*x_train*), que representa el 80% de los datos (2012 datos) y otro para su validación (*x_test*), que representa el 20% restante de los datos (503 datos). Esta división 80/20 permite entrenar el modelo con una cantidad suficiente de información sin comprometer la evaluación de su rendimiento (Nabipour et al., 2020).

Esta preparación de datos ha permitido crear una base sólida para el entramiento de los modelos, permitiendo así trabajar con datos limpios, coherentes y escalados, que aseguran la capacidad de generalización futura del modelo.

4.2 Selección del modelo de Machine Learning

La selección de los modelos de ML aplicados en este trabajo se basó en la necesidad de abordar la predicción de precios bursátiles como un problema de serie temporal, donde los datos presentan dependencia en el tiempo y un comportamiento no lineal. Por ello, se optó por probar una combinación de modelos estadísticos, de Aprendizaje Supervisado y de DP.

Regresión lineal y logística. Estos modelos se utilizaron como punto de partida por su sencillez y por aplicación en tareas predictivas. Aunque no están diseñados específicamente

para series temporales, permiten establecer relacionales básicas entre las variables independientes y la variable objetivo (Nabipour et al., 2020).

ARIMA. Fue escogida como representación de los modelos estadísticos, dado su uso extendido en análisis financieros. Este modelo es capaz de comprender tendencias y componentes estacionales (Rundo et al., 2019).

Árbol de decisión y RF. Fueron elegidas por su robustez frente a datos ruidosos y por su bajo riesgo de sobreajuste gracias a la incorporación de múltiples árboles. Aunque no considera directamente la dimensión temporal (Nabipour et al., 2020).

ANN. Se decidió probar una red neuronal tradicional como un paso intermedio entre los modelos de Aprendizaje Supervisado y las arquitecturas de DP más avanzadas. Este modelo permite modelar relaciones no lineales. Sin embargo, no considera directamente la dimensión temporal (Nabipour et al., 2020).

LSTM. Se decidió aplicar este modelo, ya que, los anteriores no capturaban bien las secuencias temporales, factor muy relevante para este trabajo y que este modelo si adaptaba. Este modelo permite retener información a largo plazo y mejorar la dependencia entre eventos (Rundo et al., 2019).

La elección de estos modelos no fue arbitraria, sino que surgieron en la búsqueda de la solución a problemas de modelos anteriores.

4.3 Implementación de los modelos en Python

Una vez realizadas las tareas de preprocesamiento de datos se pueden iniciar las diferentes pruebas de ejecución de los diferentes modelos. En este apartado se detallarán cinco modelos, aunque solo se examinarán las líneas de código de tres de ellos.

Inicialmente, se implementó el modelo de Regresión Lineal empleando la librería stickt-learn. Este modelo busca establecer una conexión entre las variables independientes (X_train) y la variable objetivo (y_train). Posteriormente, se llevaron a cabo predicciones y evaluaciones sobre el conjunto de prueba (X_test).

El primer modelo ARIMA que se desarrolló se muestra en la Figura 6:

```
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Crear y entrenar el modelo ARIMA
arima_model = ARIMA(y_train, order=(100,1,1))
arima_fit = arima_model.fit()

# Hacer predicciones
y_pred_arima = arima_fit.forecast(steps=len(y_test))

# Evaluar el modelo
mae_arima = mean_absolute_error(y_test, y_pred_arima)
mse_arima = mean_squared_error(y_test, y_pred_arima)
r2_arima = r2_score(y_test, y_pred_arima)
rmse_arima = np.sqrt(mean_squared_error(y_test, y_pred_arima))

print(f"ARIMA")
print(f"MAE  : {mae_arima:.4f}")
print(f"MSE  : {mse_arima:.4f}")
print(f"R2   : {r2_arima:.4f}")
print(f"RMSE : {rmse_arima:.4f}")
```

Figura 6. Modelo ARIMA. Elaboración propia

En el caso del presente proyecto, el modelo ARIMA fue aplicado a través de la biblioteca Statsmodels y el módulo tsa.arima.model. Se importó únicamente la clase ARIMA. Posteriormente, se crea un modelo ARIMA con parámetros d, p, q ajustables. Primeramente, se determinan de la siguiente manera: (5, 0, 0), que significan; p= 5, que se utilizan los últimos 5 valores de la serie correspondiente a la parte autorregresiva del modelo (AR), d= 0, que no se utiliza diferenciación y q=0 que no se utiliza la parte de media móvil (MA). Luego, se entrena el modelo con los datos de entrenamiento y_train. Una vez entrenado, el modelo predice la misma cantidad de valores que hay en el conjunto de prueba y_test usando el método. forecast().

El tercer modelo implementado fue el RF, que constaba de 100 árboles de decisión para predecir valores continuos. Cada árbol se entrena con diversos subconjuntos aleatorios de datos y la predicción final es el promedio de las predicciones individuales de cada uno de los árboles. La ejecución se realiza de la misma manera que en los demás modelos, importándose de la librería scikit-learn.

El cuarto modelo fue la ANN y se muestra en la Figura 7:

```
import tensorflow as tf
from tensorflow import keras

# Definir la estructura de la red neuronal
model_nn = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),  # Capa oculta con 64 neuronas
    keras.layers.Dense(32, activation='relu'),  # Otra capa oculta con 32 neuronas
    keras.layers.Dense(1)  # Capa de salida para predecir el precio de cierre
])

# Compilar el modelo
model_nn.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Entrenar la red neuronal
history = model_nn.fit(X_train, y_train, epochs=50, batch_size=16, validation_data=(X_test, y_test), verbose=1)

# Hacer predicciones con el modelo de red neuronal
y_pred_nn = model_nn.predict(X_test).flatten()

# Evaluar el modelo
mae_nn = mean_absolute_error(y_test, y_pred_nn)
mse_nn = mean_squared_error(y_test, y_pred_nn)
r2_nn = r2_score(y_test, y_pred_nn)
rmse_nn = np.sqrt(mean_squared_error(y_test, y_pred_nn))

print(f"Red Neuronal")
print(f"MAE  : {mae_nn:.4f}")
print(f"MASE  : {mse_nn:.4f}")
print(f"R2   : {r2_nn:.4f}")
print(f"RMSE : {rmse_nn:.4f}")
```

Figura 7. Modelo ANN. Elaboración propia

Primeramente, se importa la biblioteca TensorFlow y se le asigna el alias tf, lo que facilita el uso de sus funcionalidades bajo ese nombre abreviado. Además, se importa el módulo Keras, que es una API que facilita la creación de redes neuronales de manera sencilla. Esta herramienta se una extensamente en estudios recientes de ML relacionados con el sector financiero. Munkhdalai et al., (2019) detalla la implementación de una red neuronal a través del uso de keras. Posteriormente, se define la red neuronal secuencial, que consta de dos capas ocultas de 64 y 32 neuronas con activación ReLU (activation= ”relu”) y una última capa (salida) con una neurona para predecir el precio de cierre. ReLU es una función de activación utilizada en las neuronas de una red con el objetivo de introducir no linealidad, frecuentemente utilizada en modelos de DP por su simplicidad computacional y eficacia para prevenir problemas como el desvanecimiento de gradiente (Cuomo et al., 2022).

Siguiendo los desarrollos de Cuomo et al., (2022), ReLU se destaca como una de las funciones más empleadas en combinación con sigmoid, tanh y leaky ReLU y su selección influye de manera considerable en la exactitud y estabilidad del modelo.

Después se elabora el modelo, especificando qué función debe reducirse y que métricas emplear para su evaluación. Por otro lado, el optimizador Adam (optimizar=” Adam”) se emplea para ajustar los pesos de la red durante el entrenamiento. El optimizador Adam se utilizó para

compilar la red, el cual fusiona la eficacia de técnicas como RMSProp y Momemtum, ajustando de manera dinámica la tasa de aprendizaje durante el entrenamiento. Su efectividad ha sido ampliamente validada en el entrenamiento de DP, tal y como se señala en (Cuomo et al., 2022), donde se destaca el uso extendido por si estabilidad y rapidez de convergencia.

Además, la función de pérdida, en este caso MSE (loss= “mse”), se establece cómo una métrica estándar en problemas de regresión continua, particularmente beneficiosa debido a su sensibilidad a errores grandes y su sencillez para optimizar a través de un gradiente descendiente (Cuomo et al., 2022).

Por último, se establece la métrica adicional que se mostrará durante el entrenamiento y evaluación del modelo, en este caso, se emplea el MAE (metrics= ”mae”). Luego, se entrena la red durante 50 épocas (epochs= 50), en cada una el modelo se procesa en lotes de 16 ejemplos simultáneamente (batch_size= 16). En el estudio de Munkhdalai et al., (2019) se detallan parámetros clave como el tamaño del lote (batch size= 32), el número de épocas (epochs= 1000) y la inclusión de la función de pérdida. Este método se empleó como punto de referencia en este proyecto. Finalmente, se realizan proyecciones del modelo y se evalúa de la misma manera que el resto de los modelos.

El último modelo ejecutado en este proyecto es LSTM y se muestra en la Figura 8:

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Crear modelo LSTM
model_lstm = Sequential([
    LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)),
    LSTM(50, return_sequences=False),
    Dense(25),
    Dense(1)
])

# Compilar el modelo
model_lstm.compile(optimizer='adam', loss='mse')

# Entrenar la LSTM
model_lstm.fit(X_train, y_train, epochs=50, batch_size=16, validation_data=(X_test, y_test), verbose=1)

# Hacer predicciones
y_pred_lstm = model_lstm.predict(X_test)

# Evaluar el modelo
mae_lstm = mean_absolute_error(y_test, y_pred_lstm)
mse_lstm = mean_squared_error(y_test, y_pred_lstm)
r2_lstm = r2_score(y_test, y_pred_lstm)
rmse_lstm = np.sqrt(mean_squared_error(y_test, y_pred_lstm))

print(f"LSTM")
print(f"MAE  : {mae_lstm:.4f}")
print(f"MSE : {mse_lstm:.4f}")
print(f"R2  : {r2_lstm:.4f}")
print(f"RMSE : {rmse_lstm:.4f}")
```

Figura 8. Primer modelo LSTM. Elaboración propia

Primero, se importa la biblioteca TensorFlow y se le otorga el alias tf, lo que facilita el uso de sus funcionalidades bajo ese nombre abreviado. Además, se importa el módulo Keras, que es una API que facilita la creación de redes neuronales de forma sencilla. Primero, se establece la estructura del modelo empleando una arquitectura secuencial con dos capas LSTM, cada una con 50 unidades y compuesta por dos capas Dense. La primera capa LSTM cumple dos funciones particulares con distintas funciones:

- Señala que la entrada es una secuencia de longitud X_train.shape[1], con una característica por paso temporal, en este caso, el precio de cierre.
- Remite toda la secuencia de salida, no solo el último valor (return_sequences=True).

La segunda capa LSTM proporciona únicamente el último estado de la secuencia, que luego se conecta a dos capas densas (Dense), la última de las cuales cuenta con una única neurona de salida. La estructura del modelo incluye capas densas (Dense layers o fully connected layers) que facilitan una transformación no lineal del vector de características producido por las capas LSTM, ajustando de manera flexible la salida hacia el valor objetivo (Cuomo et al., 2022).

Luego, se ejecutó el mismo proceso que con el modelo de Red Neuronal, se preparó el modelo, indicándole que función debe minimizar y que métricas utilizar para evaluarse. Se utiliza el optimizador Adam (optimizar= “Adam”), para ajustar los pesos de la red durante el entrenamiento. Además, se establece la función de pérdida, en este caso MSE (loss= “mse”). Posteriormente, el modelo se entrena durante 50 épocas (epochs= 50), en cada una de las cuales el modelo se procesa en lotes de 16 ejemplos a la vez (batch_size= 16). Finalmente, se predice y se evalúa el modelo.

4.4 Entrenamiento y validación

Una vez definida la arquitectura de los modelos, se puede proceder a la fase de entrenamiento y validación, cuyo objetivo es ajustar los parámetros para minimizar el error de la predicción del precio de las acciones.

Primeramente, se aplicó el modelo de Regresión Lineal, se exploraron distintas combinaciones de variables predictoras, incluyendo precios de cierre de días anteriores. También se intentó aplicar Bagging, con el objetivo de reducir la varianza del modelo y mejorar su capacidad de generalización.

El segundo modelo que se implantó fue ARIMA, al inicio no ofrecía buenos resultados, pero tras modificar determinados parámetros p, d, q ajustables, se determinó que la configuración (100, 1, 1) ofrecía el mejor rendimiento predictivo sobre los datos de precios históricos. Esta elección no fue arbitraria, sino que se basó en la observación de las propiedades de la serie temporal y en la comparación empírica de los errores obtenidos con distintos valores de (p, d, q).

El valor p= 100 indica que le modelo utiliza los últimos 100 valores pasados para predecir el siguiente. En series temporales financieras, donde pueden existir patrones a medio y largo plazo un número elevado de retardos permite capturar mejor estas dinámicas. Se seleccionó una diferenciación de orden d= 1, lo cual es consistente con la práctica habitual en el tratamiento de series financieras. Tal como se expone en el estudio de Rundo et al., (2019), el modelo ARIMA es particularmente adecuado para transformar series no estacionarias en series más estables mediante la aplicación del differencing. Por otro lado, la elección de q= 1 permite al modelo corregir errores recientes de predicción y mejora el ajuste final.

El tercer modelo fue RF, se llevaron a cabo diversas pruebas con el objetivo de optimizar su rendimiento. En primer lugar, se experimentó con el número de árboles (50, 100, 200, 300). Se observó que una profundidad excesiva favorecía el sobreajuste, mientras que una profundidad muy baja limitaba la capacidad predictiva. Finalmente, se optó por 100 árboles, ya que ofrecía un buen equilibrio entre precisión y tiempo de entrenamiento.

El cuarto modelo fue ANN. En primer lugar, se experimentó con la estructura de la red, probando distintas combinaciones de capas ocultas y número de neuronas por capa. Se testearon arquitecturas con una sola capa, con dos capas y con tres. Del mismo modo, se ajustó el número de neuronas en cada capa, probando valores como 32, 64 y 128. Finalmente, se observó que la arquitectura con dos capas ocultas de 64 y 32 neuronas respectivamente ofrecía el mejor equilibrio entre estabilidad y tiempo de ejecución intermedio. Además, se probó con la función de activación en las capas ocultas ReLU. Por otro lado, se llevaron pruebas variando parámetros de entrenamiento como el batch size (probando valores de 16, 32 y 64) y número de épocas (entre 50 y 200). Finalmente, se mantuvo los valores de 50 épocas con batch size de 16 permitiendo así obtener un buen rendimiento sin tener sobreajustes.

El último modelo fue LSTM. En primer lugar, se experimentó con la estructura de la red, probando distintas combinaciones de capas LSTM y Dense y el número de neuronas por capa. Se testearon arquitecturas con una sola capa, con dos capas, con tres capas y con cuatro,

alternando entre capas LSTM y Dense. Del mismo modo, se ajustó el número de neuronas en cada capa, probando valores como 32, 64 y 128. Finalmente, se observó que la arquitectura con dos capas LSTM de 50 unidades y dos capas Dense de 25 y 1 respectivamente ofrecía el mejor equilibrio entre estabilidad y tiempo de ejecución intermedio. Por otro lado, se llevaron pruebas variando parámetros de entrenamiento como el batch size (probando valores de 16, 32 y 64) y número de épocas (entre 50 y 200). Finalmente, se mantuvo los valores de 50 épocas con batch size de 16 permitiendo así obtener un buen rendimiento sin tener sobreajustes. Por último, se testeó la aplicación de una función de optimización Adam, por inspiración de varios estudios científicos.

4.5 Comparativa de resultados entre modelos

Tras haber implementado y evaluado los diferentes modelos, se realizó una comparativa global de resultados utilizando las métricas MAE, RMSE, R^2 y RMSE. Los resultados obtenidos se resumen en una Tabla 4, perteneciente al código y replicada aquí (los modelos se encuentran ordenados según el valor más optimo del MAE):

MODELO	MAE	MSE	R2	RMSE
LSTM	0.013202751	0.00028438	0.977393274	0.016863562
ANN	0.047422362	0.002575777	0.795238998	0.050752109
ARIMA	0.085234402	0.012624611	- 0.003591737	0.112359296
Regresión Lineal	0.392531818	0.272902476	0.98233447	0.522400685
Random Forest	1.19572334	4.582185441	0.70338585	2.14060399

Tabla 4. Resultados predictivos de los modelos. Elaboración propia

Los modelos que ofrecieron los resultados menos óptimos fueron los de Regresión Lineal y Random Forest.

En primer lugar, se aplicó el modelo de Regresión Lineal, pero tras ejecutarlo repetidas veces e intentar implementar diversas mejoras, no se pudo obtener resultados óptimos de este modelo, ya que, en línea sobre lo que se había comentado previamente, este modelo es adecuado para predecir valores numéricos continuos. Sin embargo, parte de la hipótesis de que existe una relación lineal entre las variables de entrada y las variables objetivo (Sarker, 2021).

En el contexto de predicción bursátil, esta suposición resulta excesivamente simplista, ya que los mercados financieros presentan dinámicas altamente no lineales y comportamientos temporales complejos.

Por otro lado, el modelo de RF no logró detectar tendencias dentro de los precios bursátiles. El motivo fundamental radica en la propia naturaleza del algoritmo, RF construye un conjunto de árboles de decisión, cada uno entrenado sobre subconjuntos aleatorios de datos. Cada árbol toma decisiones basadas en reglas estáticas sobre los valores de las variables de entrada, sin tener en cuenta el orden temporal de los datos, cómo ya se comentó anteriormente en este proyecto (Nabipour et al., 2020).

En el caso de series financieras, donde los valores futuros dependen fuertemente de la evolución pasada, esta independencia supuso una limitación clave. Aunque se intentó compensar esta carencia introduciendo variables de precios de días anteriores como predictores, resultó insuficiente. Esta conclusión encaja con estudios recientes, en los cuales señalas que cuando RF se aplica directamente sobre datos continuos en el dominio financiero, tiende a ofrecer resultados pobres en la predicción de tendencias. En Rundo et al., (2019) indica explícitamente que RF no es capaz de obtener información sobre la tendencia subyacente de los datos cuando se utilizan datos continuos.

El resto de las modelos (ARIMA, ANN y LSTM) fueron los que mostraron un rendimiento más competitivo en la predicción del precio de las acciones.

El modelo, a pesar de ser una técnica clásica, obtuvo resultados óptimos. Este comportamiento se debe a que ARIMA es capaz de modelar componentes autorregresivos y de media móvil. Sin embargo, como se destaca en Rundo et al., (2019), *"el modelo ARIMA tiene limitaciones definidas cuando se trabaja con datos de series de tiempo financieras que presentan asimetrías, intervalos de tiempo irregulares y alta/baja volatilidad"*. Esto implica que ARIMA no es capaz de capturar las relaciones lineales ni las dependencias a largo plazo.

Además, en este trabajo se observó que, aunque optimizando los parámetros (p, d, q) el modelo ARIMA alcanzaba resultados aceptables, seguía mostrando un MAE y un RMSE significativamente más altos que el modelo LSTM y un R^2 inferior, lo que confirma su dudosa capacidad de mejora y optimización frente a otros modelos.

El siguiente modelo con mejores resultados, es el modelo de ANN que, gracias a su capacidad para modelar relaciones no lineales entre las variables de entrada y las variables objetivo, permitió capturar ciertos patrones complejos. Sin embargo, las ANN tradicionales no están diseñadas para trabajar con datos secuenciales. Como se explica en Cuomo et al., (2022), las redes densas no incorporan estructuras para retener memoria temporal ni para modelar dependencias entre observaciones en el tiempo.

Como resultado, aunque las ANN ofreció un rendimiento aceptable, si se hubiera escogido cómo modelo final, el resultado habría sido un sistema con menor capacidad para anticipar cambios de tendencia y con peor capacidad de generalización frente a datos nuevos, ya que su estructura no aprovecha de forma efectiva la información temporal (Cuomo et al., 2022).

Finalmente, el modelo LSTM fue seleccionado como el modelo final para la predicción del precio de las acciones en este proyecto. A diferencia de los modelos probados, la arquitectura LSTM está específicamente diseñada para procesar series temporales y aprender dependencias a corto y largo plazo gracias a su estructura interna de celdas de memoria y puertas de control (Nabipour et al., 2020). Esta capacidad es fundamental en el contexto de los mercados financieros. Cómo señala el documento Rundo et al., (2019), las arquitecturas LSTM han demostrado superar consistentemente a modelos como ARIMA y ANN en tareas de predicción financiera, debido precisamente a su capacidad para modelar dinámicas temporales complejas. Además, en el trabajo Nabipour et al., (2020), se destaca que LSTM muestra una clara superioridad en predicción de movimientos de acciones, especialmente para datos continuos cuando el rendimiento de los modelos de ML es mucho menor.

Los resultados obtenidos en este trabajo son coherentes con estas evidencias previas: el modelo LSTM fue el único capaz de alcanzar un MAE y un RMSE muy reducidos, junto con un R^2 cercano a 1, demostrando una excelente capacidad de generalización.

4.6 Mejora del modelo final

Tras el análisis y la comparativa de los diferentes modelos, finalmente, se determinó que la arquitectura LSTM es la que ofrece el rendimiento más prometedor para realizar este trabajo. En este apartado se aborda el proceso de optimización del modelo, con el objetivo de perfeccionarlo. Por ello, se ha diseñado y un flujo de trabajo completo que integra de manera

coherente todas las fases del proceso, desde preprocesamiento de datos hasta visualización gráfica e impresión de resultados.

Primeramente, se importan las librerías, en estes caso: pandas, matplotlib, sklearn y tensolflow.keras (Ver Figura 9).

```
import datetime
import pandas as pd
import matplotlib.pyplot as plt
import sys

from yahooquery import Ticker

from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam
```

Figura 9. Bibliotecas del modelo LSTM. Elaboración propia

Previo a la función principal se ejecutan dos funciones auxiliares:

- Get_next_business_day (): esta función calcula el siguiente día hábil (ignorando sábados y domingos). Es útil ya que el mercado bursátil solo opera en días laborales y eliminado esos días se eliminan errores por valores vacíos.
- Create_sequences (): esta función convierte el DataFrame en secuencias que pueda usar el modelo LSTM. Es importante, ya que, este modelo no puede aprender directamente del DataFrame, necesita frecuencias para predecir el futuro.

Estas funciones se pueden observar en la Figura 10:

```
# FUNCIONES AUXILIARES

# La función devuelve el siguiente día hábil (ignorando sábados y domingos)
def get_next_business_day(date):
    next_day = date + datetime.timedelta(days=1)
    while next_day.weekday() >= 5:
        next_day += datetime.timedelta(days=1)
    return next_day

# La función crea secuencias a partir de un DataFrame con 1 columna "KO_close" escalada.
def create_sequences(df_in, window):
    X_list, y_list = [], []
    for i in range(len(df_in) - window):
        X_list.append(df_in.iloc[i:i+window].values)
        y_list.append(df_in.iloc[i+window]["KO_Close"])
    return np.array(X_list), np.array(y_list)
```

Figura 10. Funciones auxiliares del modelo LSTM. Elaboración propia

4.6.1 Configuración

En este apartado se definen la fecha de inicio de predicción, cuántos días a futuro se van a predecir y el número de días pasados que la red usará para aprender.

En este caso, la fecha de inicio de predicciones 2024-11-29 (una fecha aleatoria dentro del rango de datos que maneja el modelo) y predice los 5 días hábiles siguientes (comprendiendo así una semana laboral completa).

Por otro lado, se configura la estructura del modelo LSTM; window_size= 90, epochs= 50 y batch_size= 12. Estos datos se seleccionaron tras varios procesos de prueba/error, hasta que se encontró la combinación que retribuía mejores resultados. En general, con valores bajos en los distintos parámetros, descendía considerablemente la efectividad. Por otro lado, atribuir valores muy altos aumentaba el tiempo de procesamiento sin retribuir mejoras considerables. Por lo tanto, se aseguró encontrar aquellos valores que mantuvieran una relación óptima entre resultados y tiempos de procesamiento.

Posteriormente, se ejecuta una función que, al descargar los datos, establece automáticamente como fecha de inicio un rango de 10 años previos y como fecha de finalización, la fecha introducida por el usuario.

Por último, se ejecuta otra función que recoge unos días extras al número de días que se quiere predecir, para asegurarse que los días hábiles que se pretenden predecir estén contenidos dentro del rango de descargas y no haya datos incompletos por días festivos o no laborales.

El codigo explicado en los anteriores párrafos se observa en la Figura 11:

```
# 1) CONFIGURACIÓN
prediction_start_str = "2024-11-29"  # Fecha a partir de la cual predecimos
days_to_forecast = 5                 # Cantidad de días hábiles a predecir

    # LSTM
window_size = 90
epochs = 50
batch_size = 12

    # Convertir a datetime
prediction_start = datetime.datetime.strptime(prediction_start_str, "%Y-%m-%d")

    # Descargamos datos desde 10 años antes
ten_years_before = prediction_start.replace(year=prediction_start.year - 10)
start_date_str = ten_years_before.strftime("%Y-%m-%d")

    # Fecha buffer final
end_buffer_date = (prediction_start + datetime.timedelta(days=days_to_forecast+10)).strftime("%Y-%m-%d")
```

Figura 11. Configuración del modelo LSTM. Elaboración propia

4.6.2 Descargar datos KO

En este apartado, se descargan los datos de Coca Cola entre el periodo de tiempo establecido en el apartado anterior.

Posteriormente, se crean funciones de alerta por error, se reinicia el índice para dejar solo los datos de las columnas “date” y “close” y son renombradas. También, se ejecutan varias funciones para configurar la columna de fecha y se incluyen unas líneas de código que extraen datos hasta el día anterior de la fecha seleccionada.

Luego, se introduce otra función de identificación de errores y, por último, se genera la lista de los próximos 5 días hábiles a partir de la fecha de predicción. Luego se construye un diccionario que almacena el precio real de cierre de la acción para esas fechas.

El codigo explicado en los anteriores párrafos se observa en la Figura 12:

```
# 2) DESCARGAR DATOS KO
print(f"[INFO] Descargando datos KO desde {start_date_str} hasta {end_buffer_date}")
ko = Ticker("KO")
data = ko.history(start=start_date_str, end=end_buffer_date)

    # Filtrar multi-símbolo
data = data.loc["KO"]

if data.empty:
    print("[ERROR] No se encontraron datos. Revisar fechas o ticker.")
    sys.exit(0)

data = data.reset_index()
    # Dejar solo los valores de "date" y "close"
data = data[["date","close"]].copy()
data.rename(columns={"close":"KO_Close"}, inplace=True)

data["date"] = pd.to_datetime(data["date"])
data.sort_values("date", inplace=True)
data.set_index("date", inplace=True)

    # Cortar datos hasta el día anterior
day_before = prediction_start - datetime.timedelta(days=1)
train_df = data.loc[data.index <= day_before].copy()
```

```
if train_df.empty:
    print("No hay datos para entrenar. Revisa las fechas.")
    sys.exit(0)

    # Recolectar valores reales para los próximos 5 días hábiles
future_business_days = []
current_day = prediction_start
for _ in range(days_to_forecast):
    future_business_days.append(current_day)
    current_day = get_next_business_day(current_day)

real_dict = {}
for bd in future_business_days:
    if bd in data.index:
        real_dict[bd] = data.loc[bd,"KO_Close"]
    else:
        real_dict[bd] = None
```

Figura 12. Importación del modelo LSTM. Elaboración propia

4.6.3 Escalado y secuencias

En este apartado, se realizó un escalado de los datos mediante la técnica MinMaxScaler, que transforma los precios históricos al rango [0,1], esto permite reducir disparidades en la escala de los datos, acelerando la convergencia y mejorando la precisión del modelo (Zhang, 2023).

También, se generaron secuencias temporales de entrada para el modelo, utilizando ventanas de N días consecutivos, conde cada secuencia (x) permite predecir el precio del día siguiente (y).

El codigo explicado en los anteriores párrafos se observa en la Figura 13:

```
# 3) ESCALADO
    # 3.1) Instanciar el scaler
scaler = MinMaxScaler()

    # 3.2) Ajustar y transformar
df_scaled = pd.DataFrame(
    scaler.fit_transform(train_df),
    columns=train_df.columns,
    index=train_df.index

    # 3.3) Definición y entrenamiento del modelo LSTM

scaler = MinMaxScaler()
train_scaled = pd.DataFrame(scaler.fit_transform(train_df),
                            columns=train_df.columns,
                            index=train_df.index)

    # 3.4) Crear secuencias
X, y = create_sequences(train_scaled, window_size)
if X.shape[0] < 2:
    print("[ERROR] Muy pocos datos tras la creación de secuencias.")
    sys.exit(0)

X_train, y_train = X, y
```

Figura 13. Escalado y secuencias del modelo LSTM. Elaboración propia

4.6.4 Modelo LSTM

En este apartado se detalla la estructura del modelo:

- Primera capa LSTM (64 unidades): procesa las secuencias de entrada y devuelve la secuencia completa para la siguiente capa.
- Dropout (20%): se introduce tras cada capa LSTM para reducir el riesgo de sobreajuste.
- Segunda capa LSTM (64 unidades): genera un vector resumen de la secuencia procesada.
- Otra capa Dropout (20%).

- Capa de salida Dense (1 unidad): produce la predicción final, es decir, el precio de cierre previsto.

El modelo se compiló con el optimizador Adam y con una tasa de aprendizaje del 0,001 y con la función de pérdida MSE.

Para generar este modelo se introdujeron capas Dropout con una tasa del 20% como técnica de regularización, con el objetivo de reducir el riesgo de overfitting (Cuomo et al., 2022). Durante el entrenamiento, estas capas apagan aleatoriamente un 20% de las neuronas de la capa anterior, lo que fomenta la generalización de representaciones más robustas y mejora la capacidad de generalización del modelo.

El código que ejecuta el modelo LSTM descrito en los anteriores párrafos se muestra en la Figura 14:

```
# 4) MODELO LSTM
model = Sequential()
model.add(LSTM(64, return_sequences=True, input_shape=(window_size,1)))
model.add(Dropout(0.2))
model.add(LSTM(64))
model.add(Dropout(0.2))
model.add(Dense(1))

model.compile(optimizer=Adam(learning_rate=0.001), loss="mean_squared_error")
model.summary()
```

Figura 14. Modelo LSTM final. Elaboración propia

4.6.5 Entrenamiento y predicción

Primeramente, el modelo se entrena sobre las secuencias generadas anteriormente. El entrenamiento se realizó durante el número de épocas y con el batch size definidos en la configuración del experimento, reservando un 10% de los datos para validación interna. Esta fase permite que el modelo ajuste sus pesos para minimizar el error de predicción sobre los datos históricos.

Tras el entrenamiento, se procedió a realizar una predicción iterativa de los 5 días hábiles futuros. En cada iteración:

- Se toma la última ventana disponible (last_window_data) como entrada a la red.
- El modelo genera la predicción para el día siguiente.
- La predicción se desescala utilizando los parámetros del escalado MinMaxScaler.

- La predicción se guarda junto a la fecha correspondiente.
- La ventana se actualiza incorporando la nueva predicción y se repite el proceso para el siguiente día hábil.

El código explicado en los anteriores párrafos se observa en la Figura 15:

```
# 5) ENTRENAMIENTO Y PREDICCIÓN ITERATIVA DE 5 DÍAS LABORALES
history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,
                    validation_split=0.1, verbose=1)

last_window_data = train_scaled.iloc[-window_size:].copy()
predictions = []
current_pred_date = prediction_start

for i in range(days_to_forecast):
    X_pred = np.array([last_window_data.values])
    pred_scaled = model.predict(X_pred)[0,0]
    # Desescalado
    data_min = scaler.data_min_[0]
    data_max = scaler.data_max_[0]
    data_range = data_max - data_min
    pred_value = pred_scaled * data_range + data_min

    predictions.append((current_pred_date, pred_value))

    # Actualiza la ventana
    new_row = pd.DataFrame([[pred_scaled]], columns=["KO_Close"])
    last_window_data = pd.concat([last_window_data.iloc[1:], new_row], ignore_index=True)

    current_pred_date = get_next_business_day(current_pred_date)
```

Figura 15. Entrenamiento y predicción del modelo LSTM. Elaboración propia

4.6.6 Evaluación

Una vez obtenidas las predicciones para los 5 días hábiles futuros, se compararon las predicciones del modelo (y_pred) con los valores reales (y_true), previamente recogidos en la estructura (real_dict). También se incluyó un sistema que emite un aviso en caso de no disponer de valores reales para las fechas previstas.

El código explicado en los anteriores párrafos se observa en la Figura 16:

```
# 6) CALCULAR MÉTRICAS (MAE, MSE, RMSE, MAPE, etc.)
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Creamos las listas
y_true = []
y_pred = []

for (fd, val_pred) in predictions:
    val_real = real_dict.get(fd, None)
    if val_real is not None:
        y_true.append(val_real)
        y_pred.append(val_pred)

if len(y_true) > 0:
    y_true_arr = np.array(y_true)
    y_pred_arr = np.array(y_pred)

    mae = mean_absolute_error(y_true_arr, y_pred_arr)
    mse = mean_squared_error(y_true_arr, y_pred_arr)
    r2 = r2_score(y_true_arr, y_pred_arr)
    rmse = np.sqrt(mse)

    print("MÉTRICAS DE ERRORES")
    print(f"MAE  : {mae:.3f}")
    print(f"MSE  : {mse:.3f}")
    print(f"R2   : {r2:.4f}")
    print(f"RMSE : {rmse:.3f}")
else:
    print("No se han encontrado valores reales en esas fechas futuras, no se calculan métricas.")
```

Figura 16. Evaluación del modelo LSTM. Elaboración propia

4.6.7 Representación e impresión

Por último, para facilitar la interpretación de los resultados, se generó un gráfico que muestra tano el comportamiento histórico del recio de la acción como las predicciones del modelo LSTM para los 5 días hábiles futuros.

Además del gráfico, se implementó una función para mostrar en consola las predicciones del modelo, junto con los valores reales disponibles, en un formato claro y legible.

El código explicado en los anteriores párrafos se observa en las Figuras 17 y 18:

```
# 7) CREAMOS LA GRAFICA
plt.figure(figsize=(10,5))
plt.plot(train_df.index, train_df["KO_Close"], label="Histórico KO_Close", color="blue")
plt.axvline(prediction_start, color="green", linestyle="--", label="Inicio Predicción")

# Mostrar las predicciones
pred_x = [p[0] for p in predictions]
pred_y = [p[1] for p in predictions]
plt.scatter(pred_x, pred_y, color="red", marker="x", s=100, label="Predicciones")
plt.plot(pred_x, pred_y, color="red", linestyle="--")

# Mostrar los datos reales
real_x = []
real_y = []
for bd, realv in real_dict.items():
    if realv is not None:
        real_x.append(bd)
        real_y.append(realv)
if real_x:
    plt.scatter(real_x, real_y, color="orange", marker="o", s=100, label="Reales")
    plt.plot(real_x, real_y, color="orange", linestyle="--")

plt.title(f"Predicción LSTM de {days_to_forecast} días hábiles a partir de {prediction_start_str}")
plt.xlabel("Fecha")
plt.ylabel("Precio KO_Close")
plt.legend()
plt.tight_layout()
plt.show()
```

Figura 17. Representación gráfica del modelo LSTM. Elaboración propia

```
# Imprimir en consola
print("\n========== PREDICCIONES 5 DIAS LABORALES ==========")
for (fd, val) in predictions:
    r = real_dict[fd]
    if r is not None:
        print(f"{fd.strftime('%Y-%m-%d')} => Pred: {val:.3f}, Real: {r:.3f}")
    else:
        print(f"{fd.strftime('%Y-%m-%d')} => Pred: {val:.3f}, Real: None")
print("==================================================\n")
```

Figura 18. Impresión del modelo LSTM. Elaboración propia

5 RESULTADOS

En este apartado se presentan y analizan los resultados obtenidos tras la aplicación del modelo LSTM desarrollado para la predicción del precio de las acciones de Coca-Cola (KO).

Se han seleccionado varias fechas de referencia, incluyendo periodos de comportamiento estable y otros afectados por eventos extraordinarios, como crisis sanitarias, con el fin de contrastar la capacidad del modelo para adaptarse a diferentes situaciones.

5.1 Predicciones y análisis de resultados

Tras el entrenamiento y validación del modelo LSTM, se procedió a realizar predicciones para los 5 días hábiles posteriores a la fecha seleccionada (29 de noviembre de 2023).

A continuación, en la Tabla 5, se muestra la comparación entre los valores predichos por el modelo y los valores reales observados en el mercado:

FECHA	PREDICCIÓN	REAL
2023-11-29	58.553	58.230
2023-12-02	58.553	58.440
2023-12-03	58.558	58.640
2023-12-04	58.569	58.570
2024-12-05	58.580	58.660

Tabla 5. Predicción modelo LSTM 1. Elaboración propia

Las predicciones muestran un comportamiento notablemente cercano a los valores reales, con pequeñas diferencias en el rango de 0,1 puntos.

Además, la evolución temporal de la serie y las predicciones puede observarse en la Figura 19:

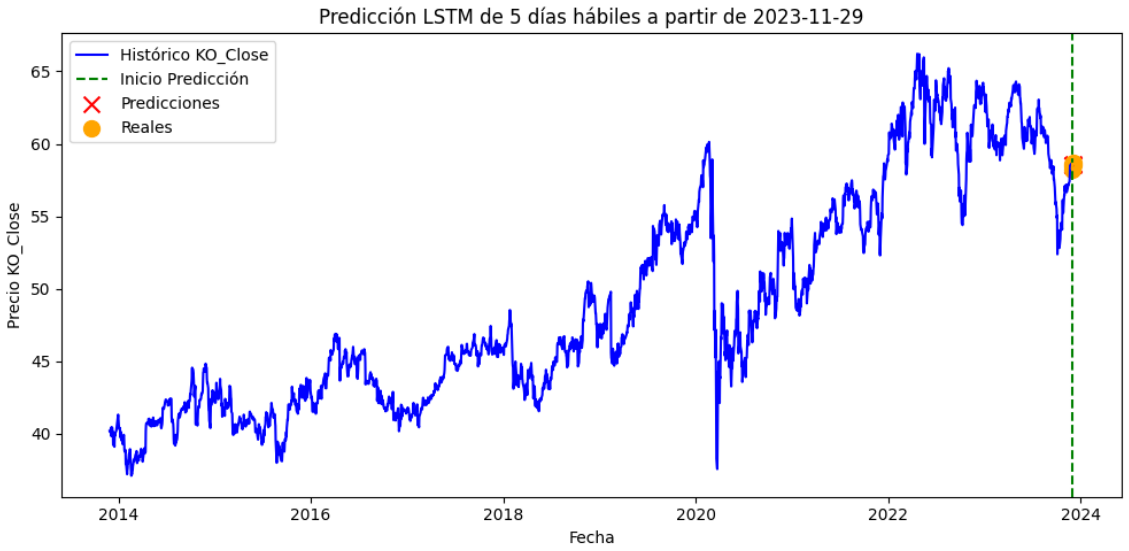

Figura 19. Representación predicción modelo LSTM 1. Elaboración propia

El gráfico permite comparar visualmente la precisión de las predicciones del modelo frente a la evolución real de los precios, facilitando la detección de aciertos y errores y observar la evolución del valor de los datos en comparación con años anteriores.

Adicionalmente y a modo de comprobación, se ejecutó el modelo varias veces con fechas distintas. A continuación, en la Tabla 6, se muestran los resultados de dos de esas ejecuciones, junto con la Figura 20, que representa su evolución de manera visual:

FECHA	PREDICCIÓN	REAL
2014-03-19	38.444	38.140
2014-03-20	38.493	38.450
2014-03-21	38.524	38.440
2014-03-24	38.548	38.400
2014-03-25	38.571	38.620

Tabla 6. Predicción modelo LSTM 2. Elaboración propia

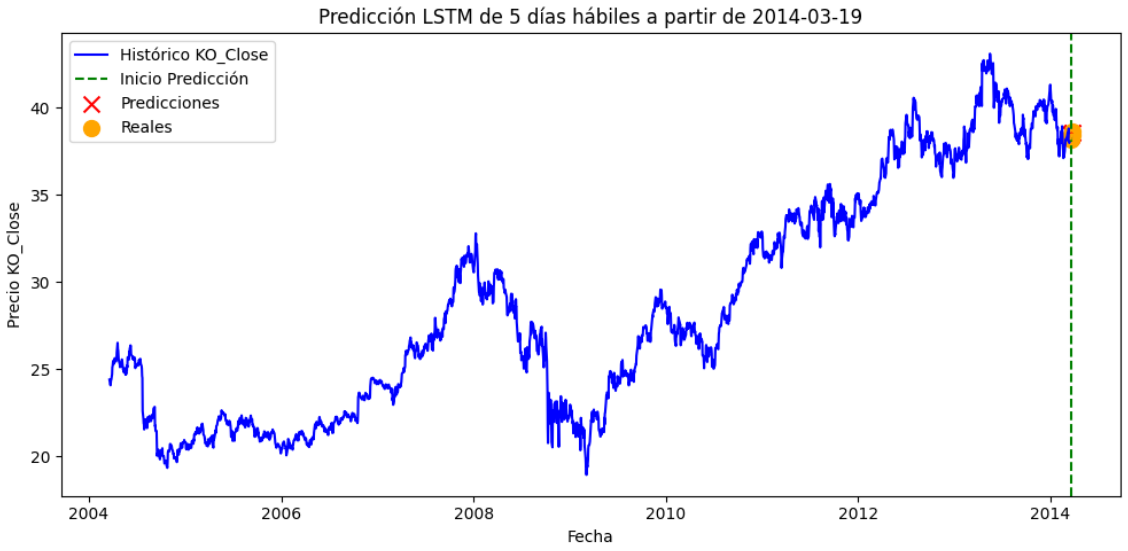

Figura 20. Representación predicción modelo LSTM 2. Elaboración propia

Adicionalmente, se llevó a cabo la ejecución del modelo en una fecha concreta, el inicio de la pandemia COVID en Estados Occidentales, para comprobar la efectividad del modelo en casos excepcionales.

Los resultados y su representación se observan en la Tabla 7 y la Figura 21:

FECHA	PREDICCIÓN	REAL
2020-03-16	48.371	45.260
2020-03-17	48.534	47.180
2020-03-18	48.710	44.850
2020-03-19	48.880	41.830
2020-03-20	49.031	38.300

Tabla 7. Predicción modelo LSTM 3. Elaboración propia

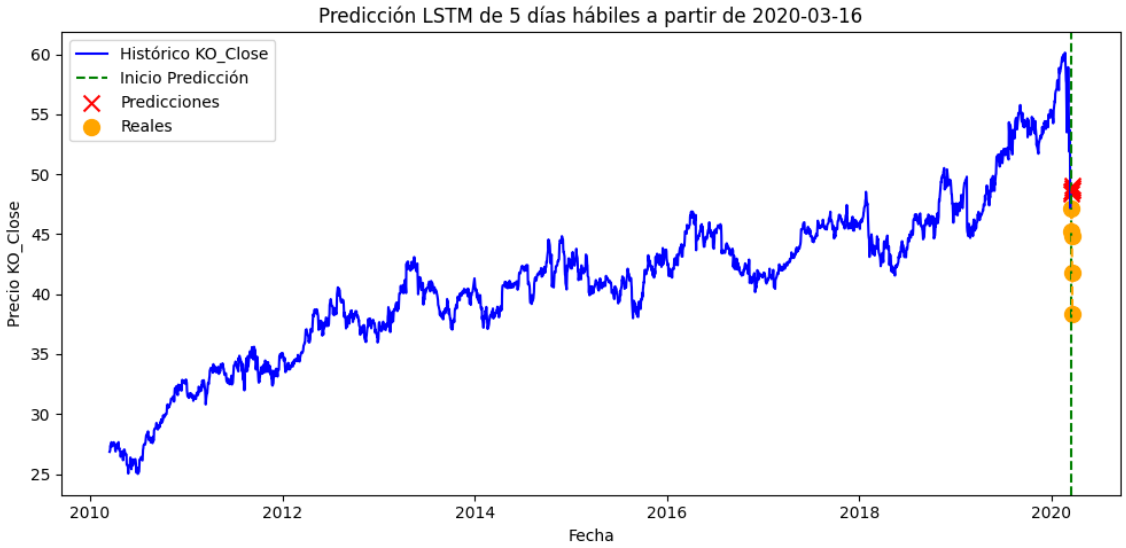

Figura 21. Representación predicción modelo LSTM 3. Elaboración propia

Los resultados muestran que el modelo ha sido capaz de realizar predicciones bastante precisas en periodos de relativa estabilidad. Diversos estudios recientes señalan que, en tareas de predicción bursátil a corto plazo, los modelos basados en LSTM suelen alcanzar valores de MAE entre 0.1 y 0.5 y de RMSE entre 0.15 y 0.8 en condiciones normales de mercado (Nabipour et al., 2020; Rundo et al., 2019). En el presente trabajo, los valores obtenidos en periodos estables se encuentran en dichos rangos, lo cual refuerza la validez del modelo desarrollado.

Por el contrario, en contextos de alta volatilidad y situaciones excepcionales, como la crisis provocada por la pandemia en marzo de 2020, la capacidad del modelo se ve notablemente reducida, reflejando que el modelo no logra capturar correctamente los cambios abruptos en los precios de la acción. Este tipo de situaciones son foco de atención en muchos estudios, en los cuales indican que, en contextos de alta volatilidad, como la pandemia de COVID-19, es común que el desempeño del modelo se vea afectado, presentando mayores errores y valores negativos en R^2 (Gerlein et al., 2016).

5.2 Discusión sobre la precisión del modelo

El modelo LSTM desarrollado en este trabajo ha mostrado una capacidad notable para predecir con precisión el comportamiento de las acciones de Coca-Cola (KO) en escenarios de mercado estables y previsibles. Tal y como se ha evidenciado en el apartado anterior, en periodos recientes como el de noviembre de 2023, el modelo ha logrado un muy bajo error MAE y RMSE, lo cual es consistente con la literatura que destaca la eficacia de las redes LSTM para la predicción de series temporales financieras (Nabipour et al., 2020; Rundo et al., 2019).

Sin embargo, el rendimiento del modelo disminuye significativamente en contextos caracterizados por alta incertidumbre o eventos extraordinarios, como se observa en el ejemplo correspondiente a marzo de 2020 (crisis sanitaria por COVID-19). Este comportamiento era previsible, ya que el modelo se basa únicamente en el histórico de precios de la acción, sin incorporar variables exógenas ni indicadores que reflejen el entorno macroeconómico o el sentimiento de los inversores (Rundo et al., 2019). De hecho, en el análisis de variables macroeconómicas realizado, la variable CPI_YoY fue descartada debido a su escasa relevancia estadística en la predicción del precio de KO a corto plazo.

Este resultado confirma una de las limitaciones conocidas de los modelos basados exclusivamente en aprendizaje de series temporales: su dificultad para anticipar cambios de régimen, rupturas estructurales (Nabipour et al., 2020). En otras palabras, aunque el LSTM es muy competente para extrapolar tendencias aprendidas en el pasado, carece de mecanismos para detectar o adaptarse a eventos imprevistos como crisis económicas, guerras, pandemias o colapsos sectoriales (Nabipour et al., 2020).

En resumen, el modelo actual es una herramienta eficaz para la predicción rutinaria en mercados eficientes, pero no debe considerarse un sistema infalible en situaciones de crisis o de fuerte disrupción, limitación que es ampliamente reconocida en la literatura especializada (Nabipour et al., 2020; Rundo et al., 2019).

6 APLICACIÓN PRÁCTICA

Una vez entrenado y validado los resultados del modelo predictivo, resulta fundamental analizar su utilidad más allá del entrono experimental. Este apartado desarrolla las aplicaciones prácticas del modelo, realizando comprobaciones sobre la versión definitiva del proyecto. Asimismo, se representan las limitaciones inherentes al uso del modelo en escenarios reales. Todo ello, permite valorar el grado de aplicabilidad del sistema.

6.1 Uso del modelo para tomar decisiones de inversión

El modelo ya generado tiene un apartado en concreto (apartado 5), el cual, está configurado para comparar las predicciones generadas con el modelo con los valores reales generados. Pero, este modelo solo es válido para aquellas fechas en las que exista ya unos valores reales con los que comparar.

Por lo tanto, como el modelo que buscamos está creado con el objetivo de predecir precios de cierre a futuro, se debe eliminar el apartado ya mencionado, porque ya se comprobó la fiabilidad del modelo.

Se optó por establecer como fecha de inicio el primer día hábil de la semana inmediatamente posterior al momento de redacción del presente trabajo (2025-06-22). Esto permite que el modelo comience a operar en un entorno "desconocido" de forma prospectiva, reflejando un uso real. Las predicciones se ven reflejadas en la Tabla 9, junto a los valores reales que obtuvieron las acciones, introducidas posteriormente a través de (Yahoo! Finance, 2025), el error de predicción del modelo se encuentra alrededor del 3%:

FECHA	PREDICCIÓN	REAL
2025-06-22	71.906	69.74
2025-06-23	71.808	69.77
2025-06-24	71.765	70.21
2025-06-25	71.726	69.63
2025-06-26	71.683	69.47

Tabla 8. Predicción modelo LSTM aplicable. Elaboración propia.

La representación de la predicción del modelo se observa en la Figura 22:

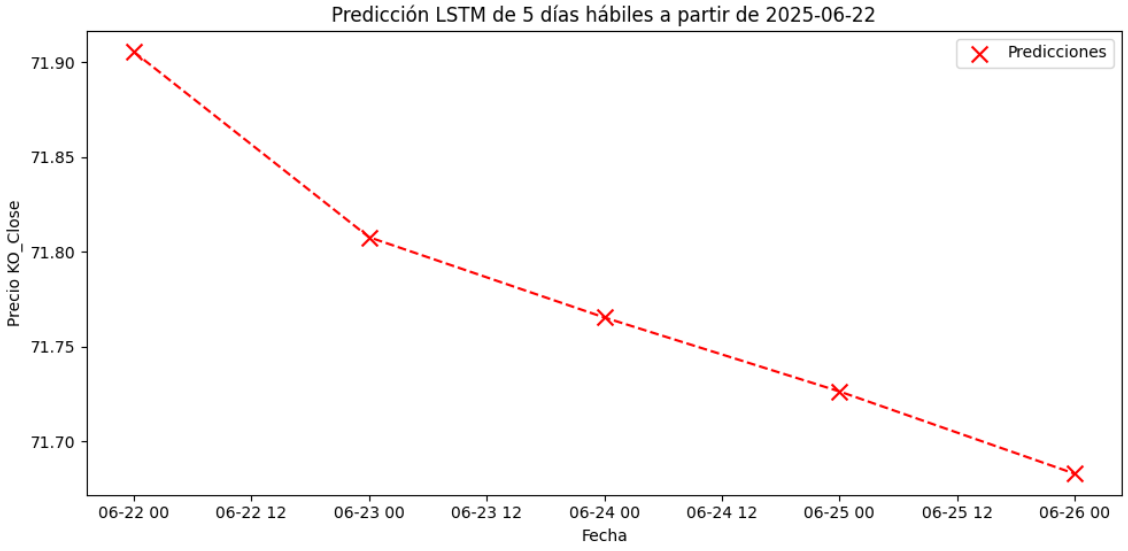

Figura 22. Representación predicción modelo aplicable. Elaboración propia

Los resultados predichos muestran una ligera tendencia descendente, sin oscilaciones abruptas, lo cual coincide con el comportamiento histórico de activos estables como Coca-Cola.

Aunque no se dispone todavía de los valores reales para validar esta predicción, el marco metodológico sostiene la credibilidad de que el modelo genera un rendimiento robusto.

Por otro lado, se decidió implementar la versión final del modelo (aquella descrita en este apartado) en la plataforma Google Colab.

El notebook puede abrirse directamente desde el siguiente enlace:

TFG_AliciaPastor – Google Colab

Esta herramienta, permite ejecutar código Python directamente desde el navegador, sin necesidad de instalar bibliotecas ni configurar entornos locales complejos (Google Developers, 2025). Gracias a este software cualquier usuario de la red podrá ejecutar el modelo y visualizar las predicciones generadas.

6.2 Limitaciones del modelo en entornos reales

Aunque los resultados obtenidos han sido satisfactorios en términos de precisión, es necesario subrayas una serie de limitaciones relevantes que deben tenerse en cuenta antes de considerar su uso real en mercados financieros:

- Incapacidad de anticipar eventos inesperados. Situaciones exógenas como crisis económicas, conflictos geopolíticos o crisis sanitarias pueden provocar movimientos abruptos en el precio nunca vistos y, por lo tanto, fuera de consideración por el modelo.
- Perdida de relevancia de impacto de variables externas. Al utilizar únicamente precios históricos, el modelo no incorpora factores externos como indicadores macroeconómicos o sentimientos en la población, por tanto, está perdiendo información sobre tendencias (Goodell et al., 2021).
- Cambios estructurales del mercado (concept drift). Los modelos pueden perder efectividad si se enfrentan a nuevos regímenes de mercado no representados en los datos históricos para el entrenamiento (Lin et al., 2012).
- Coste computacional en modelos avanzados. A mayor complejidad del modelo, se incrementan los recursos computacionales necesarios para

ejecutar el modelo. Esto puede causar dificultad en su implementación en dispositivos con recursos limitados (Janiesch et al., 2021).

- Sensibilidad al sobreajuste (overfitting). Los modelos de ML, especialmente aquellos con arquitecturas complejas, tienden a ajustarse demasiado a los datos de entrenamiento, provocando un deterioro en su capacidad de generalizar en entornos reales, donde las condiciones cambian constantemente (Janiesch et al., 2021).

En resumen, el modelo desarrollado presenta un rendimiento sólido en entornos controlados, pero su aplicación en escenarios reales requiere precaución, ya que, las predicciones pueden verse afectadas por múltiples factores adversos.

7 CONCLUSIONES

Luego del desarrollo integral del proyecto, en esta sección se muestran las conclusiones más relevantes obtenidas. Se examina el nivel de cumplimiento de los objetivos iniciales, las contribuciones efectuadas en el campo de la predicción bursátil mediante el ML, y se sugieren caminos futuros para seguir y perfeccionar el trabajo. Estas conclusiones posibilitan evaluar el verdadero alcance del modelo creado y su utilidad práctica.

7.1 Resumen de los objetivos alcanzados

Los objetivos establecidos al comienzo del trabajo se han alcanzado de manera satisfactoria. Se pusieron en práctica y contrastaron varios algoritmos de ML empleados en la predicción bursátil, empleando Python y un conjunto de datos históricos auténticos. Mediante la utilización de métricas se valoró el desempeño de cada modelo.

Luego del análisis comparativo, se optó por la red neuronal LSTM como el modelo con los mejores resultados, gracias a su habilidad para identificar patrones temporales y proporcionar predicciones más estables y exactas que los demás algoritmos evaluados (Nabipour et al., 2020). El modelo se mejoró a través de la modificación de hiperparámetros, lo que incrementó aún más su desempeño.

Además, se complementó el desarrollo con representaciones gráficas y tablas que simplificaron la comparación y entendimiento de los resultados, logrando de esta manera el propósito de desarrollar una herramienta práctica para aquellos interesados en anticipar el cambio en el precio de las acciones sin requerir habilidades técnicas sofisticadas.

7.2 Contribuciones del trabajo

El presente trabajo ha logrado proporcionar una visión amplia de los distintos logaritmos de Machine Learning más utilizados en la predicción de tendencia del mercado bursátil, calificándolas y describiéndolas según el enfoque predictivo de este proyecto.

Por otro lado, se han comprobado múltiples logaritmos que teóricamente encajaban con los requerimientos del modelo, pero se comprobó prácticamente si también lo hacían. Con el modelo más optimo, se procedió a mejorar el resultado y generar el código que permitiera predecir precios de acciones de Coca Cola en el futuro. Además, se ha subrayado a importancia del desarrollo de modelos interpretables y robustos que no solo busquen maximizar métricas de rendimiento.

Finalmente, se ha enfatizado el valor de las métricas de evaluación adecuadas y del entendimiento profundo del comportamiento de los modelos bajo diferentes condiciones de mercado.

7.3 Líneas futuras de investigación

El análisis realizado permite vislumbrar múltiples líneas de investigación futuras que podrían enriquecer aún más el campo de estudio en la intersección entre finanzas y ML:

- En primer lugar, integrar variables adicionales que proporcionen contexto económico o informacional, como indicadores de volatilidad (por ejemplo, VIX), tipos de interés, datos macroeconómicos relevantes, o incluso métricas de sentimiento extraídas de noticias o redes sociales (Rundo et al., 2019).
- En segundo lugar, considerar la inclusión de métodos de aprendizaje incremental, capaces de adaptarse dinámicamente a nuevos datos sin necesidad de reentrenamiento completo (Sarker, 2021).
- En tercer lugar, explorar modelos híbridos que combinen algoritmos de aprendizaje automático tradicionales con modelos de aprendizaje profundo (Nabipour et al., 2020).
- Por último, desarrollar una interfaz gráfica más intuitiva y accesible para el usuario final. Pese a que el modelo predictivo ha sido correctamente implementado en Python, su manejo aún demanda habilidades técnicas elementales.

En conclusión, estas líneas generan nuevas posibilidades para la creación de modelos más exactos, versátiles y fiables, que puedan incorporarse eficazmente en estrategias reales de inversión, administración de riesgos y automatización financiera.

8 BIBLIOGRAFÍA

Atje, R., & Jovanovic, B. (1993). Stock markets and development. *European Economic Review*, *37 (2-3)*, 632–640. https://doi.org/10.1016/0014-2921(93)90053-D

Botchkarev, A. (2019). A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, *14*, 45–79. https://doi.org/10.28945/4184

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? -Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, *7*, 1247–1250. https://doi.org/10.5194/gmd-7-1247-2014

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, *92*. https://doi.org/10.1007/s10915-022-01939-z

Demirgiic-Kunt, A., & Levine, R. (1996). Stock Market Development and Financial Intermediaries: Stylized Facts. *The World Bank Economic Review*, *10 (2)*, 291–321. https://doi.org/10.1093/wber/10.2.291

Dong, W. (2024). Analysis of Financial Performances for Coca-Cola: Comparison with PepsiCo, Keurig Dr Pepper and Monster Beverage. *Advances in Economics, Management and Political Sciences*, *107*(1), 87–93. https://doi.org/10.54254/2754-1169/107/2024GA0108

Gareth James, D. W. T. H. R. T. (2021). *An Introduction to Statistical Learning with Applications in R* (Second Edition). https://doi.org/https://doi.org/10.1007/978-1-0716-1418-1

Gerlein, E. A., McGinnity, M., Belatreche, A., & Coleman, S. (2016). Evaluating machine learning classification for financial trading: An empirical approach. *Expert Systems with Applications*, *54*, 193–207. https://doi.org/10.1016/j.eswa.2016.01.018

Goodell, J. W., Kumar, S., Lim, W. M., & Pattnaik, D. (2021). Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis. In *Journal of Behavioral and Experimental Finance* (Vol. 32). Elsevier B.V. https://doi.org/10.1016/j.jbef.2021.100577

Google Developers. (2025, February 18). *Instalación de Python. Notebook de Colab*. https://developers.google.com/earth-engine/guides/python_install-colab?hl=es-419

Huebner, S. S. (1910). Scope and functions of the stock market. *The ANNALS of the American Academy of Political and Social Science, 35(3)*, 1–23. https://doi.org/10.1177/000271621003500301

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*, 685–695. https://doi.org/10.1007/s12525-021-00475-2/Published

Lin, W. Y., Hu, Y. H., & Tsai, C. F. (2012). Machine learning in financial crisis prediction: A survey. In *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews: Vol. 42 (4)* (pp. 421–436). https://doi.org/10.1109/TSMCC.2011.2170420

Macroaxis. (2025). *The Coca Cola Stock Volatility*. https://www.macroaxis.com/volatility/KO/The-Coca-Cola

Munkhdalai, L., Munkhdalai, T., Namsrai, O. E., Lee, J. Y., & Ryu, K. H. (2019). An empirical comparison of machine-learning methods on bank client credit assessments. *Sustainability (Switzerland)*, *11*(3). https://doi.org/10.3390/su11030699

Nabipour, M., Nayyeri, P., Jabani, H., Shahab, S., & Mosavi, A. (2020). Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data; A Comparative Analysis. *IEEE Access*, *8*, 150199–150212. https://doi.org/10.1109/ACCESS.2020.3015966

Prasath, V. B. S., Alfeilat, H. A. A., Hassanat, A. B. A., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., & Salman, H. S. E. (2017). Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier -- A Review. *National Institutes of Health*, *7(4)*. https://doi.org/10.1089/big.2018.0175

Rundo, F., Trenta, F., di Stallo, A. L., & Battiato, S. (2019). Machine learning for quantitative finance applications: A survey. *Applied Sciences (Switzerland)*, *9, 5574*(24). https://doi.org/10.3390/app9245574

Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. In *SN Computer Science* (Vol. 2, Issue 3). Springer. https://doi.org/10.1007/s42979-021-00592-x

The Coca Cola Company. (2025). *Country selector*. https://www.coca-cola.com/country-selector

Yahoo! Finance. (2025). *The Coca-Cola Company (KO)*. https://es.finance.yahoo.com/quote/KO/

Yang, J., Lin, N., Zhang, K., Jia, L., Zhang, D., Li, G., & Zhang, J. (2023). A Parametric Study of MPSO-ANN Techniques in Gas-Bearing Distribution Prediction Using Multicomponent Seismic Data. *Remote Sensing*, *15, 3987*(16). https://doi.org/10.3390/rs15163987

Zhang, K. (2023). *LSTM Neural Network in Stock Price Prediction*. 848–856. https://doi.org/10.2991/978-94-6463-198-2_87