

Predicción de series temporales aplicada a activos financieros



Universitat
Oberta
de Catalunya

Antonio de los Mozos Alonso

MU Ingeniería Computacional y
Matemática
Área de Inteligencia Artificial

Nombre Tutor/a de TF

Carlos Gaitán Poyatos

**Profesor/a responsable de la
asignatura**

Carles Ventura Royo

Fecha Entrega

07/06/2023

**Firma del director autorizando la
entrega final del TFM:**



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	Predicción de series temporales aplicada a activos financieros
Nombre del autor:	Antonio de los Mozos Alonso
Nombre del consultor/a:	Carlos Gaitán Poyatos
Nombre del PRA:	Nombre y dos apellidos
Fecha de entrega (mm/aaaa):	06/2023
Titulación o programa:	Máster en Ingeniería Computacional y Matemática
Área del Trabajo Final:	Inteligencia Artificial
Idioma del trabajo:	Castellano
Palabras clave	Series temporales, Redes neuronales, Activos financieros.

Resumen del Trabajo

Este trabajo se centra en la predicción de series temporales, concretamente el precio de activos financieros cotizados en bolsa, mediante redes neuronales LSTM (Long Short-Term Memory).

Las series temporales son conjuntos de datos secuenciales que se recopilan en intervalos de tiempo regulares. En el ámbito financiero, la predicción precisa del precio de los activos es de suma importancia para los participantes del mercado.

Las redes neuronales LSTM son un tipo de red neuronal recurrente que puede modelar secuencias largas y capturar dependencias a largo plazo en los datos.

El trabajo se estructura en varias etapas. En primer lugar, se busca una fuente de datos fiable y estable que permita obtener los datos de cualquier activo financiero de forma estructurada. Luego, se realiza un preprocesamiento de los datos para asegurarse de que sean adecuados para el entrenamiento de la red neuronal LSTM añadiendo indicadores como medias móviles o rendimiento diario. A continuación, se construye y entrena el modelo de red neuronal LSTM utilizando los datos preprocesados. Durante el entrenamiento, el modelo aprende a capturar patrones y relaciones en las secuencias temporales, ajustando los pesos de las conexiones neuronales para minimizar el error de predicción. Después de entrenar el modelo, se realiza una evaluación de su rendimiento. Se utilizan métricas adecuadas, como el error cuadrático medio (MSE), para comparar las predicciones del modelo con los valores reales de los precios. Finalmente se genera una solución software que permite realizar todos estos procedimientos de forma sencilla.

Abstract

This paper focuses on the prediction of time series, specifically the price of listed financial assets, using LSTM (Long Short-Term Memory) neural networks.

Time series are sequential data sets that are collected at regular time intervals. In finance, accurate prediction of asset prices is key to market participants.

LSTM neural networks are a type of recurrent neural network that can model long sequences and capture long-term dependencies in the data.

The work is structured in several stages. First, a reliable and stable data source is identified to obtain the data for any financial asset in a structured way. Then, a preprocessing of the data is performed to ensure that it is suitable for training the LSTM neural network by adding indicators such as moving averages or daily returns. Next, the LSTM neural network model is built and trained using the preprocessed data. During training, the model learns to capture patterns and relationships in the temporal sequences, adjusting the weights of the neural connections to minimize prediction error. After training the model, an evaluation of its performance is performed. Appropriate metrics, such as mean square error (MSE), are used to compare model predictions with actual price values. Finally, a software solution is generated that allows to perform all these procedures in a simple way.

Índice

1.	Introducción.....	1
1.1.	Contexto y justificación del Trabajo.....	1
1.2.	Objetivos del Trabajo	1
1.3.	Impacto en sostenibilidad, ético-social y de diversidad	2
1.3.1.	Dimensión sostenibilidad.	2
1.3.2.	Dimensión comportamiento ético y de responsabilidad social (RS)...	2
1.3.3.	Dimensión diversidad, género y derechos humanos.	2
1.4.	Enfoque y método seguido.....	3
1.5.	Planificación del Trabajo.	3
1.5.1.	Planificación.....	3
1.5.2.	Herramientas.	5
1.6.	Breve resumen de productos obtenidos	7
1.7.	Breve descripción de los otros capítulos de la memoria	7
2.	Estado del arte	9
3.	Materiales y métodos	11
3.1.	Aspectos relevantes del diseño y desarrollo.	11
3.1.1.	Librería YFinance.....	11
3.1.2.	Librería Apache Spark.	11
3.1.3.	Redes neuronales LSTM.	13
3.1.4.	Simulación Montecarlo.....	15
3.2.	Metodología de desarrollo.....	15
3.3.	Productos obtenidos.....	16
3.3.1.	Módulo de gestión de datos.	17
3.3.2.	Módulo de cálculos financieros.	18
3.3.3.	Módulo de visualización.	18
3.3.4.	Módulo de inteligencia artificial.	21
3.3.4.1.	Predicción.....	21
3.3.4.2.	Simulación	23
3.3.5.	Programa principal.....	24
3.3.6.	Experimentos.....	24
3.4.	Valoración económica.....	25
4.	Resultados	26
4.1.	Funcionalidades del software desarrollado.	26
4.1.1.	Descarga de datos.....	26
4.1.2.	Actualizar todos los activos.....	26
4.1.3.	Ver gráfico histórico de precios de un activo.....	26
4.1.4.	Ver gráfico de rendimiento diario de un activo.....	27
4.1.5.	Crear portfolio.....	27
4.1.6.	Calcular matriz de correlación de portfolio.....	27
4.1.7.	Calcular matriz de covarianza de portfolio.....	28
4.1.8.	Entrenar modelo con portfolio.....	28
4.1.9.	Obtener predicción con portfolio.....	29
4.1.10.	Entrenar modelo con portfolio para simulación.....	30
4.1.11.	Simulación Montecarlo de portfolio.....	30
4.1.12.	Ver tabla de activos.....	32

4.1.13.	Ver tabla de portfolios.....	32
4.1.14.	Ver tabla de activo.....	32
4.2.	Comparativa Redes Neuronales.....	33
4.3.	Estudio de optimizadores y funciones de perdida.....	34
4.4.	Viabilidad del software como herramienta de predicción.....	39
4.	Conclusiones y trabajos futuros	40
4.1.	Conclusiones.....	40
4.2.	Consecución de los objetivos.....	40
4.3.	Planificación y metodología.....	40
4.4.	Impactos previstos.....	40
4.5.	Líneas de trabajo futuras.....	41
5.	Glosario.....	42
6.	Bibliografía	44

Lista de figuras

Figura 1: Planificación (Diagrama de Gantt)	3
Figura 2: Detalle de los componentes del producto final.	7
Figura 3: Ejemplo de estructura de un RDD de Spark	12
Figura 4: Ejemplo de DAG de Spark	13
Figura 5: Comparación entre una Red Neuronal y una Red Neuronal Recurrente	13
Figura 6: Estructura de una red neuronal LSTM	14
Figura 7: Resultado de la simulación Montecarlo con el software diseñado en este proyecto. Simulación de 400 días con 100 trazas.	15
Figura 8: Pasos de la metodología Agile	16
Figura 9: Menú del software desarrollado	16
Figura 10: Log de activos financieros descargados	17
Figura 11: Log de portfolios creados (1)	18
Figura 12: Log de portfolios creados (2)	18
Figura 13: Detalle del gráfico de velas para el activo Google en los últimos 3 años.	19
Figura 14: Detalle del gráfico de rendimiento para el activo Google en la crisis de 2008.	20
Figura 15: Detalle del gráfico de predicción para el activo JP Morgan.	20
Figura 16: Detalle del gráfico de simulación para el activo BBVA para 50 días con 20 trazas.	21
Figura 17: Arquitectura de la red neuronal LSTM	22
Figura 18: Proceso para generar trazas aleatorias	23
Figura 19: Detalle de los experimentos realizados.	25
Figura 20: Descarga del activo GOOGL (Google).	26
Figura 21: Actualizar todos los activos en disco.	26
Figura 22: Creación de un portfolio con acciones de DELL (DELL) y GameStop (GME)	27
Figura 23: Matriz de correlación del portfolio DELL - GME	28
Figura 24: Matriz de covarianza del portfolio DELL - GME.	28
Figura 25: Detalle de entrenamiento de portfolio con los activos DELL y GME	29
Figura 26: Detalle de predicciones para DELL y GME	30
Figura 27: Simulación para el activo DELL para 50 días con 20 trazas.	31
Figura 28: Simulación para BBVA de 500 días con 100 trazas.	31
Figura 29: Simulación para BBVA de 500 días con 100 trazas. Foco solo en la media.	32
Figura 30: Resultado Adadelta + Mean Absolute Error	34
Figura 31: Resultado Adadelta + Mean Squared Error	35
Figura 32: Resultado Adadelta + Mean Squared Logarithmic Error	35
Figura 33: Resultado Adam + Mean Absolute Error	36
Figura 34: Resultado Adam + Mean Squared Error	36
Figura 35: Resultado Adam + Mean Squared Logarithmic Error	37
Figura 36: Resultado RMSprop + Mean Absolute Error	37
Figura 37: Resultado RMSprop + Mean Squared Error	38
Figura 38: Resultado RMSprop + Mean Squared Logarithmic Error	38

Lista de tablas

Tabla 1: Fecha de entrega de las PECs	5
Tabla 2: Comparación de modelos.	33
Tabla 3: Rendimiento de optimizadores y funciones de pérdida.	34

1. Introducció

1.1. Contexto y justificación del Trabajo

La predicción del comportamiento de activos financieros cotizados en los mercados de valores es algo que se lleva intentando desde su origen. Tanto desde el punto de vista económico como desde el punto de vista matemático, es algo realmente atractivo para los interesados por este mundo. Aplicar nuevas tecnologías como las redes neuronales puede aproximarnos cada vez más a tener “algo” capaz de predecir estos mercados, al menos en el corto y medio plazo.

Conseguir este objetivo no es una tarea sencilla, ya que el precio de un activo financiero depende de infinitas variables externas, por lo que buscar una predicción a través de su evolución histórica de precios es una tarea realmente difícil. Actualmente se usan regresiones y tendencias, tanto del activo, como del propio mercado y otros activos del mismo sector, para hacer una estimación de cuál será el precio en un futuro. Recientemente también se han empezado a aplicar tecnologías como redes neuronales.

El resultado de este trabajo se espera que sea un software con las funcionalidades necesarias como para hacer predicciones sobre el valor a futuro de activos financieros cotizados en bolsa.

1.2. Objetivos del Trabajo

En vista de la magnitud del problema a resolver, es complicado que este proyecto de una solución final a este. Lo que busca de este proyecto es utilizar tecnología como las redes neuronales para resolver el problema de predicción de series temporales. Por ello, los objetivos que se persiguen son estos:

- Obtener datos de acciones en mercados cotizados, procesarlos, y almacenarlos de forma estructurada. Este objetivo busca obtener una fuente confiable de datos, generar nuevos datos y estadísticas sobre los datos originales, y almacenarlos de tal forma que puedan ser usados posteriormente.
- Desarrollar una un software basado en una red neuronal capaz de predecir el precio de un activo cotizado a futuro. Se buscará la tecnología que permita hacer predicciones a futuro del precio de un activo cotizado en bolsa en función de su histórico. La idea no es llegar a dar un precio exacto, pero si ser capaces de ver tendencias y valores más y menos probables.
- Se compararán diferentes arquitecturas y modelos.

- La visualización de los resultados se hará mediante gráficas interactivas.

1.3. Impacto en sostenibilidad, ético-social y de diversidad.

1.3.1. Dimensión sostenibilidad.

Este proyecto tiene impacto a nivel sostenible, ya que el procesamiento de los datos, y el entrenamiento de la red neuronal, hasta llegar a obtener información de valor, es muy pesado computacionalmente, por lo que necesita de cantidades significativas de energía eléctrica.

Este impacto dependerá del sistema en el que se ejecute el programa, de donde obtenga la energía este, lo eficiente que sea, y otros muchos parámetros. En cualquier caso, siempre podremos medir este impacto registrando el consumo eléctrico del sistema, y comparándolo con los porcentajes de producción de energía eléctrica de la zona, para saber cuánta energía viene de fuentes sostenibles y no sostenibles.

Buscando la optimización de los cálculos y del rendimiento del software, podemos ayudar a mejorar en esta dimensión.

1.3.2. Dimensión comportamiento ético y de responsabilidad social (RS).

Este proyecto permite generar predicciones de activos financieros a futuro, lo que podría llegar a usarse para obtener dinero en bolsa.

A partir de la crisis del 2008 se generaron diferentes regulaciones a nivel mundial de los mercados de capitales, las dos principales son: Basilea III y MIFID II, en ellas se endurecen las medidas necesarias para poder vender productos financieros a clientes o dar asesoramiento financiero, entre otras muchas medidas.

Cabe destacar que en ningún caso deberían usarse los resultados generados con este programa como asesoramiento financiero. Invertir siempre lleva un riesgo que puede hacer que pierdas todo tu dinero.

1.3.3. Dimensión diversidad, género y derechos humanos.

El software ha sido diseñado para que pueda usarse por cualquier persona, independientemente de su género, su religión, su raza, o su ideología. En esta versión está limitado a una interfaz de consola de comandos, por lo que personas no familiarizadas con este tipo de interfaces, o personas con alguna discapacidad, pueden encontrar difícil su uso.

Los datos que usa el programa como input son de libre acceso, y no reflejan nada más que el precio de activos financieros en bolsa.

Los menús del programa, así como el software, están escritos en inglés para facilitar que personas de todo el mundo puedan usarlo independientemente de su idioma.

1.4. Enfoque y método seguido

La estrategia elegida está basada en el desarrollo de software bajo metodologías ágiles. Se planifican franjas temporales en las que se va iterando para acabar generando un entregable con valor. Dado que este proyecto tiene fases de investigación y pruebas, estas fases se incluirán también dentro del marco agile.

Esta forma de trabajar es la más adecuada ya que permite ir entregando valor poco a poco y ver como el trabajo va evolucionando; además, al ser incremental, permite detectar fallos y/o mejoras en el proyecto con más antelación que, por ejemplo, un desarrollo en cascada.

Las principales fases en las que se irá iterando para generar valor se definen a continuación. Al finalizar cada fase se deberá haber desarrollado un software que se integrará en el producto final.

1.5. Planificación del Trabajo.

En este apartado veremos la planificación y herramientas usadas para el desarrollo del trabajo.

1.5.1. Planificación.

La planificación inicial se muestra a continuación en un diagrama de Gantt:

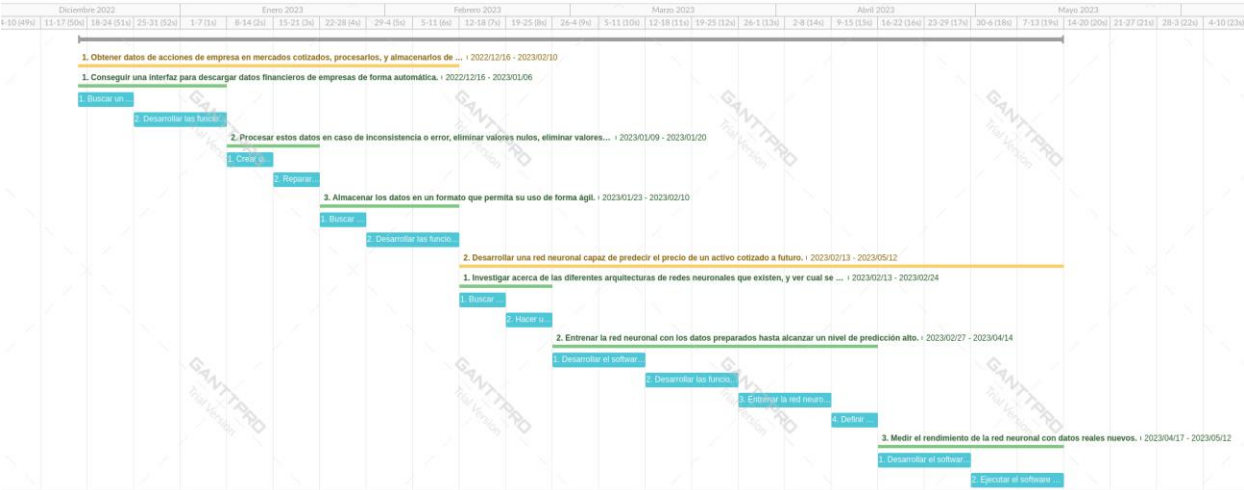


Figura 1: Planificación (Diagrama de Gantt)

Se desglosan todas las tareas a continuación:

1. Obtener datos de acciones de empresa en mercados cotizados, procesarlos, y almacenarlos de forma estructurada.
 1. Conseguir una interfaz para descargar datos financieros de empresas de forma automática.
 1. Buscar un API/librería que incorporar en el software que permita acceder a los datos financieros necesarios.
 2. Desarrollar las funciones necesarias para poder descargar la información de forma sencilla.
 2. Procesar estos datos en caso de inconsistencia o error, eliminar valores nulos, eliminar valores incompletos o erróneos.
 1. Crear una función de detección de datos erróneos.
 2. Reparar los datos detectados como erróneos.
 3. Almacenar los datos en un formato que permita su uso de forma ágil.
 1. Buscar el formato óptimo de almacenamiento.
 2. Desarrollar las funciones necesarias para poder almacenar la información en un formato que permita su uso posterior.
2. Desarrollar una red neuronal capaz de predecir el precio de un activo cotizado a futuro.
 1. Investigar acerca de las diferentes arquitecturas de redes neuronales que existen, y ver cual se puede adecuar más al objetivo que se persigue.
 1. Buscar papers, documentos, ensayos... sobre otros trabajos similares y ver cuales dan los mejores resultados.
 2. Hacer una tabla de pros y contras para decidir cuál es el más conveniente para este trabajo.
 2. Entrenar la red neuronal con los datos preparados hasta alcanzar un nivel de predicción alto.
 1. Desarrollar el software necesario para dividir el conjunto de datos.
 2. Desarrollar las funciones de entrenamiento y validación.
 3. Entrenar la red neuronal, registrar las estadísticas.
 4. Definir que entrenamiento es el que mejor resultado da.
 3. Medir el rendimiento de la red neuronal con datos reales nuevos.

1. Desarrollar el software necesario para poder usar la red neuronal con datos en tiempo real.
2. Ejecutar el software y medir el rendimiento del mismo.

Se cumplieron todas las fechas y además se incorporaron algunas funcionalidades extra al software.

Dadas las fechas de entrega de las PEC, se agruparon en la PEC2 y PEC3 el grueso del desarrollo. En la PEC2 se entregó hasta el punto 2.2.1 Desarrollar el software necesario para dividir el conjunto de datos. En la PEC3 se entregó desde el punto 2.2.2 Desarrollar las funciones de entrenamiento y validación, hasta el punto 2.3.2 Ejecutar el software y medir el rendimiento sobre datos actuales.

PEC	Fecha	Entregable
PEC 1	16/12/2022	Documento inicial
PEC 2	10/03/2023	Desarrollo fase 1 [1.1.1 - 2.2.1]
PEC 3	05/05/2023	Desarrollo fase 2 [2.2.2 – 2.3.2]
PEC 4	07/06/23	Memoria

Tabla 1: Fecha de entrega de las PECs

1.5.2. Herramientas.

Las herramientas utilizadas para llevar a cabo este proyecto se listan a continuación:

- Python 3.8 como lenguaje de programación, por su versatilidad y compatibilidad con las librerías de procesamiento de datos y machine learning.
- PyCharm como entorno de desarrollo por su compatibilidad con Python.
- Git y Github como repositorio para almacenar el código generado.

Principales librerías utilizadas:

- PySpark [\[3.3.0\]](#) para manejo de datos y almacenamiento en formato parquet.
- Pandas [\[1.4.3\]](#) para manejo de datos.
- YFinance [\[0.2.3\]](#) y Requests [\[2.28.1\]](#) para poder conectar con la fuente de datos.
- Keras [\[2.9.0\]](#) y Tensorflow [\[2.9.1\]](#) para trabajar con redes neuronales.
- Numpy [\[1.21.5\]](#) para realizar cálculos matemáticos.

1.6. Breve resumen de productos obtenidos

El producto obtenido es un software ejecutable con todas las funcionalidades descritas. El código está almacenado en el siguiente repositorio de Github: https://github.com/antonioldm96/TFM_AI_Trading

Dentro de este repositorio, en la carpeta “project” está el producto ejecutable, el archivo principal donde está en main es “app.py”; en la carpeta “lab” hay ficheros, tanto Python como Notebooks, que se han usado como experimentos para llegar al producto final; en la carpeta “doc” se encuentra la documentación del proyecto; en la carpeta “data” están los datos de los activos financieros; y en la carpeta “model_weights” están los pesos de los modelos entrenados.

Dentro de “project” están las distintas funcionalidades desarrolladas, en el paquete “io_stockdata” están las funcionalidades para obtener los datos, procesarlos, y almacenarlos; en el paquete ia_predict están las funcionalidades para usar los diferentes modelos desarrollados; en el paquete “finance” están las funcionalidades que permiten realizar cálculos financieros; y en el paquete “display” están las funcionalidades de visualización.

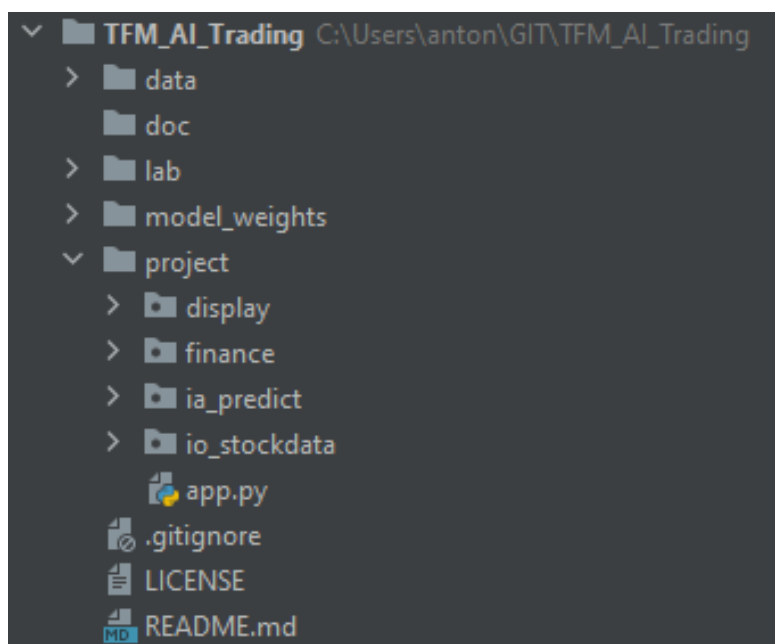


Figura 2: Detalle de los componentes del producto final.

1.7. Breve descripción de los otros capítulos de la memoria

A continuación, se hará una breve descripción del resto de capítulos:

- Capítulo 2 - Estado del arte: Análisis del estado actual de este problema, revisión de documentos y metodologías seguidas para resolverlo.
- Capítulo 3 - Materiales y métodos: Descripción y análisis en detalle de todos los desarrollos y experimentos realizados hasta llegar a la solución final del proyecto.

- Capítulo 4 - Resultados: Revisión de la solución final, de los resultados obtenidos, y demostración de esta.
- Capítulo 5 – Conclusiones: Detalle de las conclusiones obtenidas de este proyecto.
- Capítulo 6 – Glosario: Definición de términos y acrónimos utilizados a lo largo del documento.
- Capítulo 7- Bibliografía: Documentación usada en el desarrollo del proyecto.
- Capítulo 8 – Anexos: Otro tipo de materiales como manual

2. Estado del arte

La literatura sobre la predicción de activos financieros es bastante extensa, ya que es algo que se ha intentado desde la creación de estos. Aplicar nuevas tecnologías a este campo es algo más reciente, pero igualmente, es algo muy perseguido, ya que puede reportar beneficios económicos directos. La mayoría de los investigadores llegan a la misma conclusión, es muy difícil predecir activos financieros, sobre todo a largo plazo, ya que la mayoría de las veces son situaciones externas y arbitrarias lo que hacen variar su valor (guerras, pandemias). Si bien es cierto, bajo un entorno económico estable, sí que se pueden hacer predicciones aproximadas sobre el comportamiento de un activo, simplemente atendiendo a los factores y resultados económicos de la empresa en cuestión, y de las empresas o sectores correlacionados con esta. Algo muy usado recientemente, y muy efectivo, es el uso de series temporales, y redes neuronales capaces de aprender de estas.

Estos son algunos de los documentos más representativos sobre el estado del arte actual:

An innovative neural network approach for stock market prediction: En este paper los autores proponen usar redes LSTM para predecir la tendencia de alguno de los índices de referencia, para ello elaboran un dataset con datos del índice, lo dividen, y lo usan para entrenar una red LSTM que aprende las características particulares de la serie temporal.

Stock Market Prediction Using LSTM Recurrent Neural Network: En este paper, de forma similar al anterior, utilizan los datos de algunos activos financieros para entrenar una red LSTM obteniendo una red capaz de predecir el precio de activos a futuro.

The application research of neural network and BP algorithm in stock price pattern classification and prediction: Este paper utiliza también una red neuronal LSTM, pero los autores en este caso han buscado entrenar la red con datos más frecuentes, con diferencias de minutos, con el objetivo de detectar patrones en el día a día de los mercados. Hay que destacar que utilizan un algoritmo de Backpropagation para calibrar el modelo.

Predicting the Direction of Stock Market Index Movement Using an Optimized Artificial Neural Network Model: Este paper muestra cómo aplicar algoritmos genéticos para obtener una red neuronal capaz de predecir el precio de un activo financiero. No parece demostrar demasiada eficacia, puesto que la red neuronal no es recurrente, por lo que simplemente acaban prediciendo tendencias como si de una regresión se tratase.

Como vemos, las aproximaciones a este problema suelen ser similares, usando algún tipo de red neuronal para predecir a futuro el precio de un activo. En el caso de las redes LSTM se enfrentan a problemas como conjuntos de datos con mucha volatilidad, lo que complica el aprendizaje de la red, o las

limitaciones de las redes LSTM, que son capaces de predecir un número de outputs limitado para un conjunto de inputs, por lo que usarlas para predecir a futuro es algo complejo.

En este proyecto se usa una red LSTM de forma similar a los anteriores, y se toman dos aproximaciones: una donde se entrena a la red neuronal con ventanas de 100 días, y donde para cada día tenemos 9 inputs, y la red produce el siguiente día como output; esta aproximación es similar a alguna de las propuestas en papers anteriores. La segunda aproximación es usar algo similar a una simulación Montecarlo con la red LSTM para introducir valores semi aleatorios a la red de tal forma que esta pueda generar una traza a largo plazo, por ejemplo, a un año vista. Esta segunda aproximación es algo más novedosa.

3. Materiales y métodos

En este capítulo se mostrarán los materiales y métodos utilizados para el desarrollo del proyecto.

3.1. Aspectos relevantes del diseño y desarrollo.

Desde el comienzo de este proyecto se busca que el resultado sea un producto software con una interfaz que permita diferentes acciones. Para llegar a ello, en cada fase se han hecho pequeñas iteraciones de análisis, experimentación y desarrollo, integrando todo el software en el producto final.

A continuación, se describen las librerías más relevantes utilizadas en el proyecto.

3.1.1. Librería YFinance

Esta librería permite obtener datos financieros de forma estructurada. Su fuente de datos es la página Yahoo Finance, y según la documentación de la librería, obtienen los datos haciendo Web Scraping. Se probó la fiabilidad y rendimiento de la librería, y los resultados fueron satisfactorios, los datos que devuelve siempre han sido correctos, y la disponibilidad ha sido del 100%.

Permite diferentes filtros para obtener la información requerida. Para este proyecto en concreto la función más usada ha sido “download” filtrando por nombre del activo (Ticker), periodo que indica desde cuando hasta cuando queremos los datos, un día, una semana, un mes, un año, o el máximo posible (Period), e intervalo que pueden ser minutos, horas, días, entre otros (Interval).

3.1.2. Librería Apache Spark.

La librería Apache Spark es usada para gestión y procesamiento de grandes volúmenes de datos. Su principal ventaja frente a bases de datos convencionales es que es capaz de cargar el conjunto de datos en memoria mediante RDD, y se aprovecha del paralelismo de los procesadores para realizar operaciones simultáneamente. Además, registra el histórico de operaciones en el DAG.

El concepto de RDD hace referencia a Resilient Distributed Datasets, es importante destacarlo ya que estas tres siglas son las que identifican una tecnología como Spark:

- Resilient: en español Resiliente, indica que en caso de que un RDD sea destruido o corrompido, Spark es capaz de regenerarlo a través de un histórico.
- Distributed: en español Distribuido, indica que los RDD se distribuyen en particiones en memoria.
- Dataset: es un conjunto de datos.

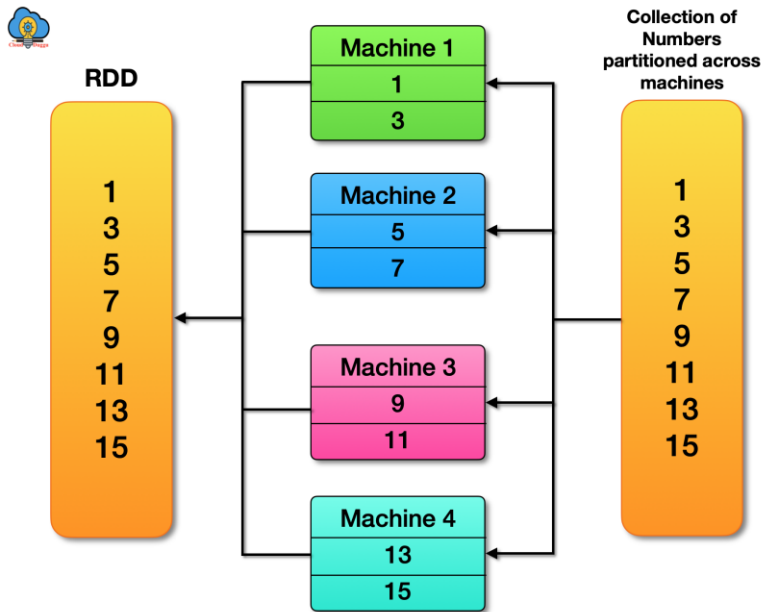


Figura 3: Ejemplo de estructura de un RDD de Spark

El concepto de DAG hace referencia a Directed Acyclic Graph, en este caso, este concepto explica como Spark construye en su motor interno las consultas, haciendo que cada operación sea un paso en el grafo, siendo capaz de optimizarlas, y ejecutarlas de forma paralela, lo que permite que los RDD anteriormente mencionados puedan regenerarse. Cabe destacar que Spark ejecuta las consultas de forma perezosa, esto quiere decir, que hasta que no es necesario, por ejemplo, mostrar el resultado de una consulta, Spark no la ejecuta, simplemente va construyendo y optimizando la consulta en forma de DAG.

- Directed: es un grafo dirigido, esto quiere decir que las relaciones van en un sentido.
- Acyclic: es un grafo que no puede tener ciclos, es decir, todas las líneas de relaciones tienen que llegar a un fin.
- Graph: como su propio nombre indica, es un grafo.

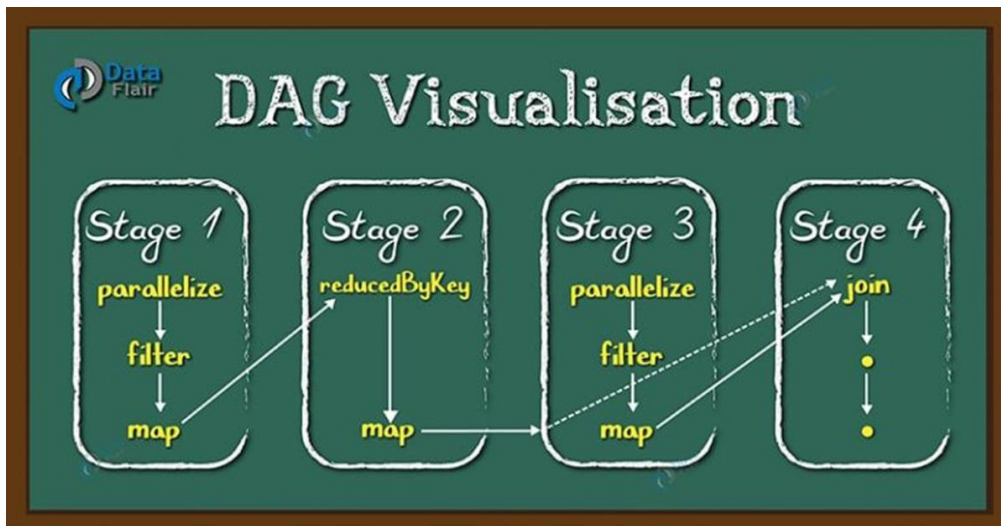


Figura 4: Ejemplo de DAG de Spark

3.1.3. Redes neuronales LSTM.

Las redes neuronales LSTM (Long short-term memory) son un tipo de Red Neuronal Recurrente con unas características muy concretas, a continuación, se detalla su estructura y funcionamiento:

En primer lugar, es necesario explicar que es una Red Neuronal Recurrente (RNN), este tipo de redes, a diferencia de las Redes Neuronales normales, tienen una capa de "células" que son capaces de capturar el output generado y mezclarlo con el nuevo input, de esta forma la red es capaz de aprender de inputs recurrentes, ya que siempre va a "recordar" el output generado para el anterior input usándolo como nuevo input.

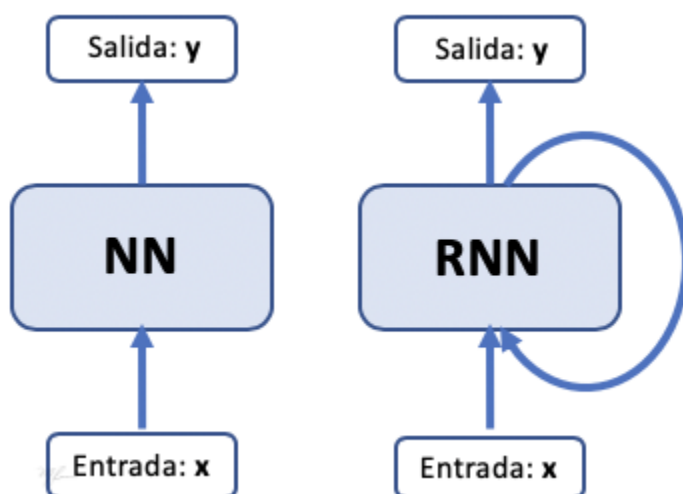


Figura 5: Comparación entre una Red Neuronal y una Red Neuronal Recurrente

A continuación, podemos explicar el funcionamiento de una red LSTM, que parte del concepto de RNN.

LONG SHORT-TERM MEMORY NEURAL NETWORKS

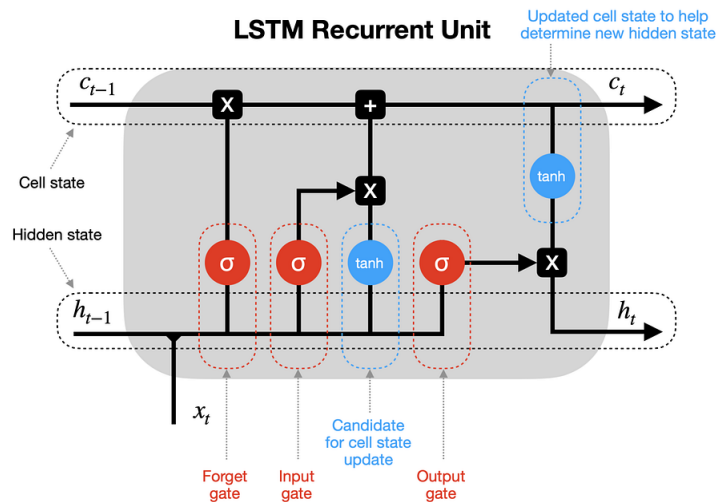


Figura 6: Estructura de una red neuronal LSTM

Las redes LSTM, como su propio nombre indica son redes capaces no solo de recordar a corto plazo, también a largo plazo, por este motivo han sido seleccionadas para este proyecto.

Las redes LSTM basan su funcionamiento en unas “células de memoria” (cell state y hidden state) y puertas de control (gates) que regulan el flujo de información dentro de la red. Cada célula de memoria almacena información pasada y es capaz de añadir o borrar información a lo largo del tiempo.

Ahora vamos a ver el funcionamiento de cada una de las partes:

- Puerta de olvido (Forget gate): Esta puerta permite a la red decidir que información debe ser olvidada de la celda de memoria. La función de activación sigmoide determina que datos han de ser olvidados.
- Puerta de entrada (Input gate): Esta puerta permite introducir nueva información en la celda de memoria. La función de activación sigmoide determina que datos de la célula de memoria se actualizarán.
- Celda de memoria (Cell State): Tenemos dos celdas de memoria, la actual, y la candidata. Utilizan la función de tangente hiperbólica (\tanh) para procesar la información. En cada ciclo, la candidata se usa para calcular la información a partir de los inputs y el estado actual de la

celda, y una vez hecho el cálculo, esta pasa a ser la celda de memoria actual.

- Puerta de salida (Output gate): Esta puerta permite a la red propagar información hacia nuevas células de memoria. La función de activación sigmoide determina que datos de la célula de memoria se propagarán.

3.1.4. Simulación Montecarlo.

La Simulación Montecarlo es una técnica que permite estimar resultados probables en problemas complejos o inciertos mediante la generación de múltiples muestras aleatorias. El concepto es sencillo, si tenemos el problema modelado mediante algún tipo de modelo, y a este modelo le damos cientos o miles de muestras aleatorias, acabaremos teniendo un conjunto de resultados donde podremos observar valores más y menos probables.

En concreto, en este proyecto, nuestro modelo será la red neuronal LSTM, que habrá sido entrenada con el histórico del activo financiero que se requiera. La generación de muestras aleatorias se explicará en los siguientes apartados.

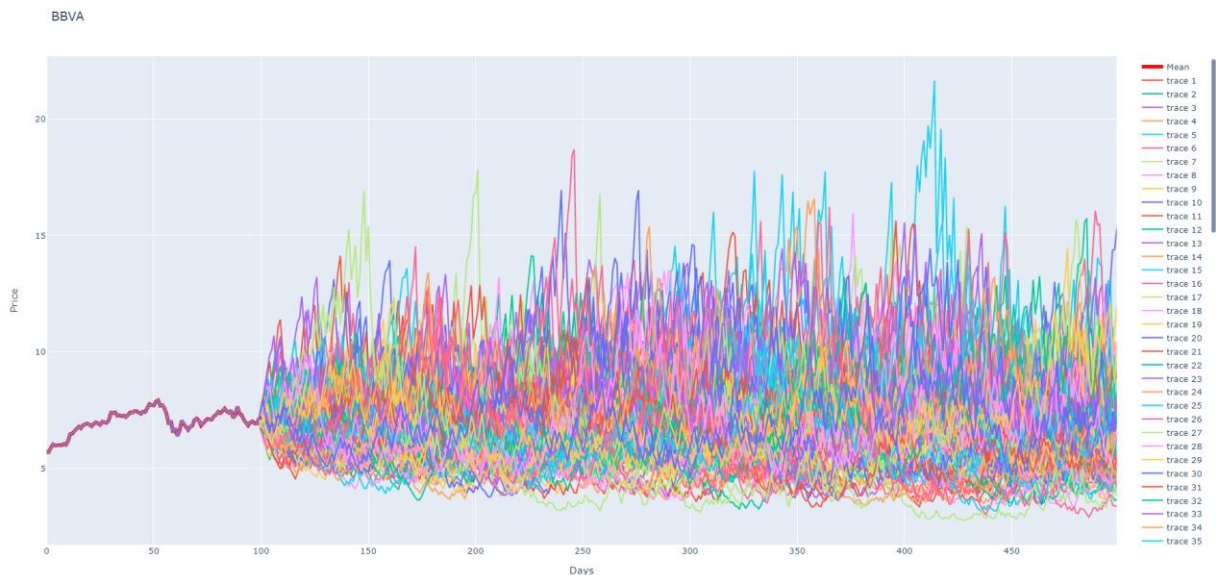


Figura 7: Resultado de la simulación Montecarlo con el software diseñado en este proyecto. Simulación de 400 días con 100 trazas.

3.2. Metodología de desarrollo.

La metodología de desarrollo que se ha seguido en este proyecto es Agile, el trabajo se planificó en pequeños pasos y por cada iteración se generaba un entregable que aportaba algún valor o funcionalidad al producto. Esta forma está triunfando ampliamente en el mundo del desarrollo del software porque permite ver la evolución del producto poco a poco, lo que hace que se puedan introducir cambios o ajustes de forma sencilla sin afectar a la planificación. En el apartado de planificación de esta memoria se puede ver en detalle todas las fases que se desarrollaron, así como las fechas de entrega.



Figura 8: Pasos de la metodología Agile

Si revisamos el menú el software podemos ver reflejadas estas pequeñas iteraciones, primero con funcionalidades de descarga de datos, luego funcionalidades de visualización, a continuación, funcionalidades de gestión de los datos, y, por último, funcionalidades de aprendizaje y simulación.

```
Select a valid option:
0. Exit
1. Download stock data.
2. Update ALL Stocks.
3. Show Stock graph.
4. Show Stock Performance graph.
5. Create Portfolio.
6. Calculate portfolio correlation matrix.
7. Calculate portfolio covariance matrix.
8. Train model with portfolio.
9. Get portfolio predictions.
10. Train Model Portfolio MonteCarlo Simulation.
11. Portfolio MonteCarlo Simulation.
12. Show Stocks Table.
13. Show Portfolio Table.
14. Show Stock Table.
```

Figura 9: Menú del software desarrollado

3.3. Productos obtenidos.

A continuación, se detallan los productos obtenidos. El software se encuentra en el repositorio git: https://github.com/antoniodlm96/TFM_AI_Trading

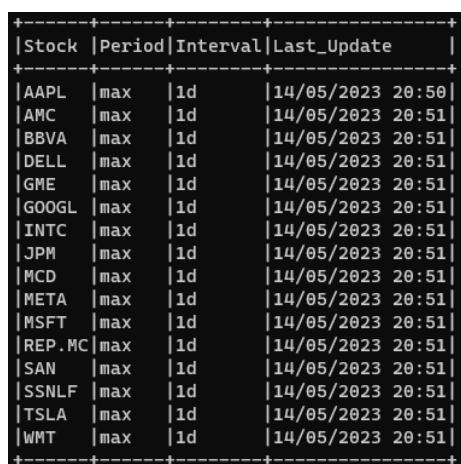
Primero veremos los diferentes módulos del software, y después el programa principal y los experimentos.

3.3.1. Módulo de gestión de datos.

La función principal de este módulo es descargar los datos de los activos financieros, procesarlos, añadir información útil a estos, guardarlos en disco, y recuperarlos cuando sea necesario. Para ello, se apoya en las librerías de YFinance y PySpark.

Las funciones de este módulo son:

- `calculate_performance`: Permite calcular el rendimiento del activo financiero entre los diferentes registros (habitualmente diarios) que existan en el dataframe. Genera una nueva columna en el dataframe con el rendimiento del registro actual respecto al anterior.
- `download_stock_data`: Permite descargar los datos de cualquier activo cotizado en bolsa. Añade a los datos de origen las medias móviles del precio de cierre del activo, de 30, 60 y 90 registros de tamaño. Si trabajamos con días, 30, 60 y 90 días.
- `write_stock_data`: Permite guardar en disco los datos del activo financiero.
- `read_stock_data`: Permite leer de disco los datos del activo financiero.
- `write_stock_log`: Cada vez que descargamos información del activo se genera un log, este log contiene el nombre del activo, le asigna un identificador, y muestra la última fecha de actualización. Esta función permite guardar este log.
- `read_stock_log`: Permite leer de disco el log de activos financieros.
- `write_portfolio_list`: Cada vez que se crea un nuevo portfolio se genera un registro en un log con los datos de este. Esta función permite guardar este log en disco.
- `read_portfolio_list`: Permite leer de disco el log de portfolios.



Stock	Period	Interval	Last_Update
AAPL	max	1d	14/05/2023 20:50
AMC	max	1d	14/05/2023 20:51
BBVA	max	1d	14/05/2023 20:51
DELL	max	1d	14/05/2023 20:51
GME	max	1d	14/05/2023 20:51
GOOGL	max	1d	14/05/2023 20:51
INTC	max	1d	14/05/2023 20:51
JPM	max	1d	14/05/2023 20:51
MCD	max	1d	14/05/2023 20:51
META	max	1d	14/05/2023 20:51
MSFT	max	1d	14/05/2023 20:51
REP.MC	max	1d	14/05/2023 20:51
SAN	max	1d	14/05/2023 20:51
SSNLF	max	1d	14/05/2023 20:51
TSLA	max	1d	14/05/2023 20:51
WMT	max	1d	14/05/2023 20:51

Figura 10: Log de activos financieros descargados

ID	Stock_List	Number_Shares_List	Buy_Dates_List
1	[BBVA, GOOGL, JPM, SAN]	[100, 100, 100, 100]	[2010-01-15 00:00:00, 2010-01-15 00:00:00, 2010-01-15 00:00:00, 2010-01-15 00:00:00]
2	[BBVA, JPM]	[100, 100]	[2015-01-01 00:00:00, 2015-01-01 00:00:00]
3	[DELL, BBVA, GOOGL, MSFT, MCD, JPM, SAN]	[100, 100, 100, 100, 100, 100, 100]	[2015-01-01 00:00:00, 2015-01-01 00:00:00, 2015-01-01 00:00:00, 2015-01-01 00:00:00, 2015-01-01 00:00:00, 2015-01-01 00:00:00, 2015-01-01 00:00:00]
4	[AAPL, TSLA]	[100, 100]	[2013-01-01 00:00:00, 2013-01-01 00:00:00]
5	[BBVA]	[100]	[2013-01-15 00:00:00]

Figura 11: Log de portfolios creados (1)

Performance_List	Min_Date	Max_Date
[[-0.6051649347397448, 7.096033712380348, 2.0700550633381223, -0.7958937063136813], [-0.2507869814746007, 1.1459433909684893], [2.709082000572947, 0.20202016796388048, 1.9179807405671472, 4.367790025176603, 1.5289497737601077, 1.0352103103006438, -0.20094560302596576], [7.800904191462520, 70.25849409312141], [-0.31147541511273086]]	[2010-01-15 00:00:00]	[2023-05-12 00:00:00]
	[2015-01-01 00:00:00]	[2023-05-12 00:00:00]
	[2016-08-17 00:00:00]	[2023-05-12 00:00:00]
	[2013-01-01 00:00:00]	[2023-05-12 00:00:00]
	[2013-01-15 00:00:00]	[2023-05-12 00:00:00]

Figura 12: Log de portfolios creados (2)

3.3.2. Módulo de cálculos financieros.

Este módulo permite realizar cálculos financieros sobre los datos, así como crear portfolios con diferentes activos. Las funciones que tiene son:

- `avg_daily_stock_return`: Permite calcular el rendimiento diario de un activo.
- `avg_stock_return`: Permite calcular el rendimiento medio de un activo a lo largo del tiempo.
- `covariance_matrix_portfolio`: Permite calcular la matriz de covarianza de los activos de un portfolio.
- `correlation_matrix_portfolio`: Permite calcular la matriz de correlación de los activos de un portfolio.
- `create_portfolio`: Permite crear un portfolio, cuando lo crea, calcula las fechas comunes de los activos del portfolio, los rendimientos medios, y permite registrar el número de acciones y fecha de compra de estas. Las fechas comunes se usarán más tarde para filtrar los datos, de tal forma que el programa trabaje con las fechas adecuadas, y un número de datos similares. En las figuras 11 y 12 se puede ver el detalle de datos calculados.

3.3.3. Módulo de visualización.

Este módulo permite visualizar las distintas gráficas que se generan en la ejecución del programa. Inicialmente se usó la librería `matplotlib`, pero finalmente se descartó por no ser interactiva. La librería usada finalmente es `plotly`, que genera las gráficas en un navegador web, y permite interacción con

estas, hacer zoom sobre una parte del gráfico, seleccionar una traza concreta entre otras funcionalidades.

Las funciones desarrolladas en este módulo son:

- `display_line_graph`: En desuso, se usó en las primeras versiones del software, permite generar gráficos de línea.
- `display_bar_graph`: En desuso, se usó en las primeras versiones del software, permite generar gráficos de barra.
- `interactive_candlestick_graph`: Permite generar un gráfico interactivo de “velas japonesas” del precio de apertura, máximo, mínimo y cierre en cada intervalo, de un activo financiero. Además, se muestran las medias móviles de 30, 60 y 90 días.
- `interactive_performance_graph`: Permite generar un gráfico donde se muestra la variación diaria de rendimiento a lo largo del tiempo de un activo.
- `interactive_performance_prediction`: Permite generar un gráfico donde se muestra el resultado del entrenamiento y predicción de un activo financiero.
- `interactive_simulation`: Permite generar un gráfico de la simulación Montecarlo aplicada a un activo financiero.



Figura 13: Detalle del gráfico de velas para el activo Google en los últimos 3 años.

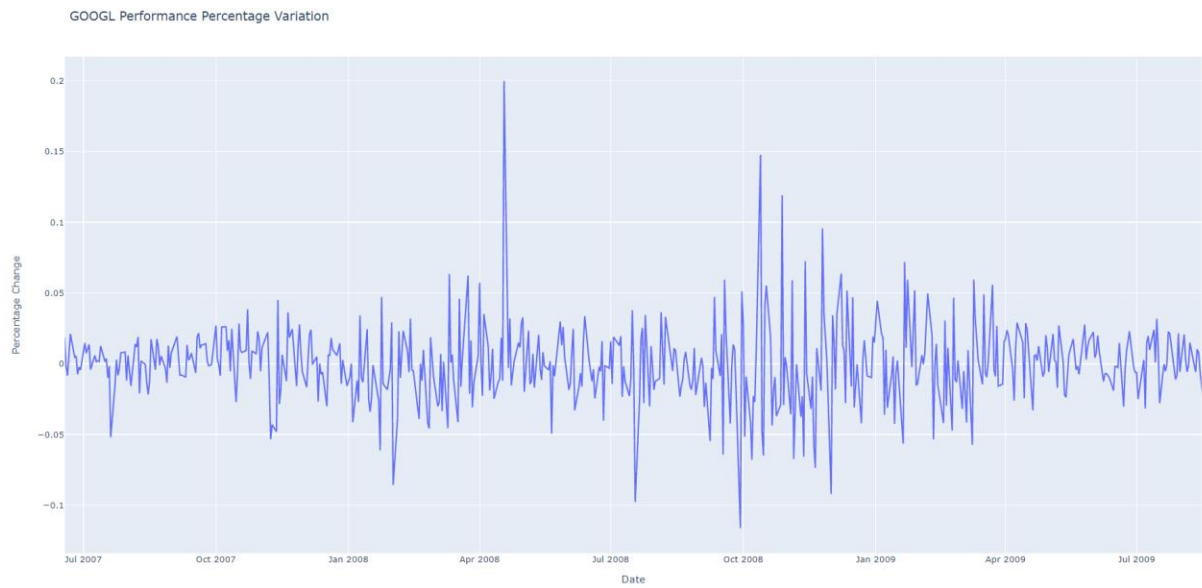


Figura 14: Detalle del gráfico de rendimiento para el activo Google en la crisis de 2008.



Figura 15: Detalle del gráfico de predicción para el activo JP Morgan.



Figura 16: Detalle del gráfico de simulación para el activo BBVA para 50 días con 20 trazas.

3.3.4. Módulo de inteligencia artificial.

Este módulo recoge las funcionalidades relacionadas con Inteligencia Artificial y simulación. Permite crear una red neuronal LSTM, entrenarla, usarla para hacer predicciones, y usarla para hacer Simulación Montecarlo. Tiene un parámetro global GLOBAL_DAYS_WINDOW que especifica el tamaño de las series temporales que reciben los modelos LSTM. Está configurado para que sean de 100 días.

3.3.4.1. Predicción.

Este modulo tiene las funciones que permiten crear una red LSTM que recibe como input una matriz de 100 registros y 9 columnas: 'Open', 'High', 'Low', 'Close', 'Volume', 'ma30', 'ma60', 'ma90', 'Performance'; como output genera el siguiente valor del precio de cierre. En este modulo se buscaba entrenar la red no solo con la serie temporal de precios de cierre, también con otros parámetros, para que la red pueda aprender patrones entre estos. La predicción que hace es correcta, pero no es realmente eficiente para hacer simulación de valores futuros.

Las funciones de este módulo son:

- **build_model_LSTM:** Permite construir el modelo de Red Neuronal LSTM. La configuración final de la red es esta:

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(1, 100, 200)	161600
lstm_3 (LSTM)	(1, 100)	120400
dense_2 (Dense)	(1, 50)	5050
dense_3 (Dense)	(1, 1)	51

```

Total params: 287,101
Trainable params: 287,101
Non-trainable params: 0

```

Figura 17: Arquitectura de la red neuronal LSTM

- `build_model_GRU`: Deprecado, en primeras versiones del proyecto se probó a usar redes GRU, pero las redes LSTM acabaron siendo superiores.
- `reshape_predictions`: Usado para poder usar el mismo normalizador (MinMaxScaler) con los datos de entrada y los datos de salida.
- `train_model`: Permite entrenar la red neuronal LSTM con un conjunto de datos que se divide en un 80% para entrenamiento y un 20% para validación. El entrenamiento se hace en 5 etapas.
- `get_prediction`: Permite obtener la predicción para el día siguiente a partir de una serie de días. El tamaño de la serie viene definido por el parámetro global comentado anteriormente.
- `save_model_weights`: Permite guardar en disco los pesos de la red neuronal asociada a un activo.
- `load_model_weights`: Permite cargar los pesos de la red neuronal almacenados en disco identificándolos por activo.
- `save_model_weights_porfolio`: Permite guardar en disco los pesos de la red neuronal asociada a un activo en un porfolio.
- `load_model_weights_porfolio`: Permite cargar los pesos de la red neuronal almacenados en disco identificándolos por activo y porfolio.
- `train_porfolio`: Permite realizar el entrenamiento de redes neuronales para cada activo dentro de un porfolio. Internamente llama a la función por cada activo de un porfolio.
- `predict_porfolio`: Permite predecir el siguiente valor de los activos de un porfolio. Internamente llama a la función `get_prediction` por cada activo de un porfolio.

3.3.4.2. Simulación

Este módulo al igual que el anterior permite crear una red LSTM, pero esta solo recibe la serie temporal de precios de cierre. La red es más versátil por solo recibir una serie temporal como input, y nos permite hacer Simulación Montecarlo que se explicará a continuación.

Las funciones de este módulo son:

- **build_model_LSTM:** Permite construir el modelo de Red Neuronal LSTM. La configuración final de la red es similar a la mostrada en la Figura 17.
- **build_model_GRU:** Deprecado, en primeras versiones del proyecto se probó a usar redes GRU, pero las redes LSTM acabaron siendo superiores.
- **train_model_sim:** Permite entrenar la red neuronal LSTM con un conjunto de datos que se divide en un 80% para entrenamiento y un 20% para validación. El entrenamiento se hace en 5 etapas.
- **get_simulation:** Esta función contiene el algoritmo de simulación. El algoritmo consta de varias fases: en una primera fase se introduce la serie temporal más actualizada posible, los últimos 100 días de histórico; a continuación, la red hace una predicción sobre estos 100 días, esta predicción se modifica ligeramente en función de la variabilidad que haya habido en los últimos 100 días, se calcula el margen mínimo y máximo de variabilidad, y dentro de este margen se calcula un valor aleatorio por el que se multiplica la predicción hecha por la red. Esta predicción aleatorizada se introduce como el siguiente día en la serie temporal, y esta se vuelve a introducir como input de la red neuronal.

Este proceso realizado de forma iterativa generará M trazas de N número de días. Estos dos parámetros se reciben como input desde la interfaz del programa.

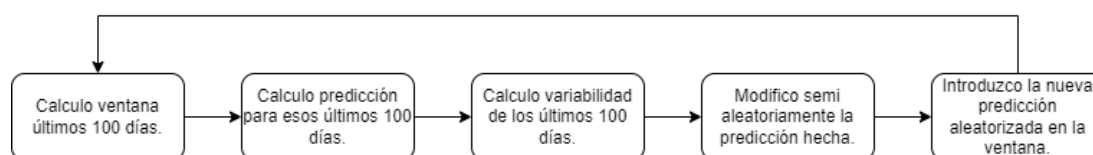


Figura 18: Proceso para generar trazas aleatorias

El proceso es computacionalmente pesado, podría optimizarse en un cluster para ejecutar procesos paralelos con varias tarjetas gráficas.

El resultado puede verse en figuras anteriores como la Figura 16 o la Figura 7.

- `save_model_weights_porfolio`: Permite guardar en disco los pesos de la red neuronal asociada a un activo en un porfolio.
- `load_model_weights_porfolio`: Permite cargar los pesos de la red neuronal almacenados en disco identificándolos por activo y porfolio.
- `train_porfolio_sim`: Permite realizar el entrenamiento de redes neuronales para cada activo dentro de un porfolio. Internamente llama a la función `train_model_sim` por cada activo de un porfolio.
- `simulate_porfolio`: Permite realizar la simulación para los activos de un porfolio. Internamente llama a la función `get_simulation` por cada activo de un porfolio.

3.3.5. Programa principal.

Este modulo contiene el proceso principal que ejecuta el menú y permite al usuario navegar entre las diferentes funcionalidades del programa.

El menú lo podemos ver en la Figura 9.

Las funciones de este módulo sirven como control de flujo del menú y para comprobar los inputs del usuario:

- `get_valid_period`: Permite comprobar que el periodo introducido por el usuario se encuentra entre los periodos válidos del API YFinance.
- `get_valid_interval`: Permite comprobar que el intervalo introducido por el usuario se encuentra entre los intervalos válidos del API YFinance.
- `main`: Contiene el flujo del programa principal. Inicialmente se realiza la configuración de las diferentes librerías que se usan en el programa. Se estructura en un loop infinito del que solo se puede salir con la opción "0". Las opciones 1 a 13 permiten seleccionar las opciones del menú.

3.3.6. Experimentos.

Este paquete contiene pequeños experimentos que se han realizado durante el alcance del proyecto para probar diferentes librerías y tecnologías, con el propósito de encontrar las más eficaces para abordar cada problema. Inicialmente se probaron las librerías `keras`, `tensorflow` y `yfinance` en los respectivos archivos. Una vez establecidas las bases del proyecto, y el conjunto de datos con el que trabajar, se desarrollaron los notebooks relacionados con la IA, tanto para predicción como para simulación.

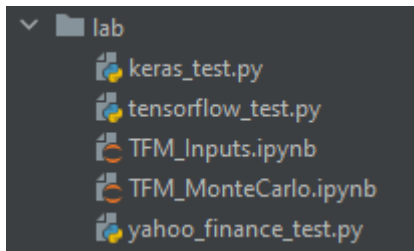


Figura 19: Detalle de los experimentos realizados.

3.4. Valoración económica.

La idea de este apartado es dar una valoración económica del producto. En este caso es algo complejo, porque por la propia naturaleza del producto desarrollado se podrían generar ganancias bastante relevantes. En cualquier caso, primero vamos a hacer un análisis de los gastos que ha supuesto este desarrollo:

- Las librerías usadas no requieren de ningún tipo de pago por su uso, todas se engloban dentro del paraguas de “Licencia Apache”.
- El IDE de desarrollo ha sido PyCharm 2022.2 (Community Edition), el cual tampoco requiere de pago por su uso.
- El desarrollo ha llevado unas 250 horas, las tarifas habituales en España por un desarrollador con conocimientos en Big Data y Machine Learning es de unos 50€ la hora, por lo que el coste total del desarrollo lo podemos estimar en 12500€.

Los ingresos de este proyecto, como hemos adelantado previamente, son difíciles de estimar, todo depende de la confianza en las estimaciones y simulaciones que genera el programa, de la cantidad de dinero a invertir, y de la aversión al riesgo del inversor.

4. Resultados

4.1. Funcionalidades del software desarrollado.

Es hora de analizar los resultados del software desarrollado. Tras todas las iteraciones hechas, llegamos a un producto donde las funcionalidades que se perseguían han sido desarrolladas.

Vamos a hacer un repaso sobre ellas:

4.1.1. Descarga de datos.

```
Please, input a valid stock name: GOOGL
Please, input a valid period:
Please, input a valid interval:
Downloading GOOGL stock data. Period: max Interval: 1d
[*****100%*****] 1 of 1 completed
Log Updated GOOGL_period=max_interval=1d
```

Figura 20: Descarga del activo GOOGL (Google).

El software nos solicita el nombre del activo que queremos descargar, y el periodo e intervalo que queramos, por defecto, el periodo es el máximo posible, y el intervalo es diario.

4.1.2. Actualizar todos los activos.

Permite actualizar todos los activos que tenemos almacenados en disco.

```
Downloading AAPL stock data. Period: max Interval: 1d
[*****100%*****] 1 of 1 completed
Log Updated AAPL_period=max_interval=1d

Downloading AMC stock data. Period: max Interval: 1d
[*****100%*****] 1 of 1 completed
Log Updated AMC_period=max_interval=1d

Downloading BBVA stock data. Period: max Interval: 1d
[*****100%*****] 1 of 1 completed
Log Updated BBVA_period=max_interval=1d

Downloading DELL stock data. Period: max Interval: 1d
[*****100%*****] 1 of 1 completed
Log Updated DELL_period=max_interval=1d

Downloading GME stock data. Period: max Interval: 1d
[*****100%*****] 1 of 1 completed
Log Updated GME_period=max_interval=1d
```

Figura 21: Actualizar todos los activos en disco.

4.1.3. Ver gráfico histórico de precios de un activo.

Permite ver el gráfico del histórico de precios de un activo, en formato de “velas japonesas” usando el precio de apertura, máximo, mínimo y cierre en cada intervalo. Además, se muestran las medias móviles de 30, 60 y 90 días.

La interfaz permite movernos a través del gráfico, seleccionar zonas y trazas concretas y hacer zoom. La Figura 13 es un ejemplo de este tipo de gráfico.

4.1.4. Ver gráfico de rendimiento diario de un activo.

Permite ver el gráfico del histórico de rendimiento diario. Con este gráfico podemos observar periodos de mayor o menor actividad en el activo, por ejemplo, fechas como la crisis del 2008 o la pandemia del Covid, se ven claramente con mayor actividad. La Figura 14 es un ejemplo de este tipo de gráfico.

4.1.5. Crear portafolio.

Permite crear un portafolio, para ello, el programa nos preguntará cuál de los activos descargados queremos usar. Tras hacer esta elección, nos pedirá el número de activos que tenemos, y la fecha de compra de estos. El proceso por debajo realizará los cálculos de fechas y rendimientos oportunos para dejar registrado el portafolio en la tabla de portafolios.

```
+-----+-----+-----+-----+
|Stock |Period|Interval|Last_Update |
+-----+-----+-----+-----+
|AAPL  |max   |1d      |20/05/2023 19:36|
|AMC   |max   |1d      |20/05/2023 19:36|
|BBVA  |max   |1d      |20/05/2023 19:36|
|DELL  |max   |1d      |20/05/2023 19:36|
|GME   |max   |1d      |20/05/2023 19:36|
|GOOGL |max   |1d      |20/05/2023 19:36|
|INTC  |max   |1d      |20/05/2023 19:36|
|JPM   |max   |1d      |20/05/2023 19:36|
|MCD   |max   |1d      |20/05/2023 19:36|
|META  |max   |1d      |20/05/2023 19:36|
|MSFT  |max   |1d      |20/05/2023 19:36|
|REP.MC|max   |1d      |20/05/2023 19:36|
|SAN   |max   |1d      |20/05/2023 19:36|
|SSNLF |max   |1d      |20/05/2023 19:36|
|TSLA  |max   |1d      |20/05/2023 19:36|
|WMT   |max   |1d      |20/05/2023 19:36|
+-----+-----+-----+-----+

Please, input a list separated by comma of stocks: DELL, GME
Invalid stock names are removed. List of stocks of the portfolio:
['DELL', 'GME']
Please, input the number of shares of DELL: 100
Please, input the date (format YYYY-MM-DD) when you bought DELL shares: 2010-01-01
Please, input the number of shares of GME: 100
Please, input the date (format YYYY-MM-DD) when you bought GME shares: 2010-01-01
```

Figura 22: Creación de un portafolio con acciones de DELL (DELL) y GameStop (GME)

4.1.6. Calcular matriz de correlación de portafolio.

Esta función calcula la matriz de correlación de los activos de un portfolio, para ello, nos mostrará una lista de los portfolios que tenemos creados, nos pedirá el ID del portfolio que queremos seleccionar, y calculará la matriz de correlación de los activos de este portfolio.

Esta matriz nos indica como de correlacionados están los activos financieros, 1 totalmente correlacionados, 0 no correlacionados, -1 totalmente negativamente correlacionados. Los valores intermedios entre estos umbrales indicarán en que medida están correlacionados. Por ejemplo, en la figura 23

```
Please, input a portfolio ID: 6
```

Correlations	DELL	GME
DELL	1.0	0.82471
GME	0.82471	1.0

Figura 23: Matriz de correlación del portfolio DELL - GME

4.1.7. Calcular matriz de covarianza de portfolio.

Al igual que la opción anterior, tendremos que seleccionar un ID de portfolio, y calculará la matriz de covarianza. Esta matriz es menos interpretable que la anterior, pero es útil a la hora de realizar ciertos cálculos financieros con los activos del portfolio.

```
Please, input a portfolio ID: 6
```

Covariance	DELL	GME
DELL	5.0E-4	1.7E-4
GME	1.7E-4	0.00643

Figura 24: Matriz de covarianza del portfolio DELL - GME.

4.1.8. Entrenar modelo con portfolio.

Esta funcionalidad permite entrenar un modelo con los activos de un portfolio. Nos pedirá el ID del portfolio, y, a continuación, realizará el entrenamiento del modelo. Los parámetros son, ventanas de 100 días para predecir el día siguiente, y 5 épocas de entrenamiento. El resultado del entrenamiento se puede ver en la estadística RMSE, y en el gráfico que nos devuelve el programa, similar a la Figura 15. Los pesos del modelo se guardan en disco para usarlos para hacer predicciones.

```

Please, input a portfolio ID: 6

*** Training model for stock: DELL
Epoch 1/5
1180/1180 [=====] - 16s 11ms/step - loss: 0.0044
Epoch 2/5
1180/1180 [=====] - 13s 11ms/step - loss: 0.0011
Epoch 3/5
1180/1180 [=====] - 13s 11ms/step - loss: 9.0651e-04
Epoch 4/5
1180/1180 [=====] - 13s 11ms/step - loss: 0.0013
Epoch 5/5
1180/1180 [=====] - 13s 11ms/step - loss: 7.4947e-04
10/10 [=====] - 1s 6ms/step

*** RSME:0.031435381677116375

*** Saving weights of model for stock: DELL

*** Training model for stock: GME
Epoch 1/5
1180/1180 [=====] - 14s 11ms/step - loss: 0.0030
Epoch 2/5
1180/1180 [=====] - 13s 11ms/step - loss: 0.0016
Epoch 3/5
1180/1180 [=====] - 13s 11ms/step - loss: 0.0022
Epoch 4/5
1180/1180 [=====] - 13s 11ms/step - loss: 0.0015
Epoch 5/5
1180/1180 [=====] - 13s 11ms/step - loss: 0.0014
10/10 [=====] - 0s 6ms/step

*** RSME:0.013091558554777815

*** Saving weights of model for stock: GME

```

Figura 25: Detalle de entrenamiento de porfolio con los activos DELL y GME

4.1.9. Obtener predicción con porfolio.

Relacionada con la funcionalidad anterior, carga los pesos del modelo guardados en disco, obtiene los últimos 100 días del histórico del activo financiero, y genera el siguiente día.

```
Please, input a portfolio ID: 6

*** Loading weights for stock: DELL Portfolio: 6
*** Getting prediction for DELL Portfolio: 6
1/1 [=====] - 1s 1s/step

*** Loading weights for stock: GME Portfolio: 6
*** Getting prediction for GME Portfolio: 6
1/1 [=====] - 0s 31ms/step

*** Predicted values for the stocks of your portfolio:
DELL: 46.46605682373047
GME: 24.335172653198242
```

Figura 26: Detalle de predicciones para DELL y GME

4.1.10. Entrenar modelo con portfolio para simulación.

Similar a la función de entrenamiento para predicción, elegimos el ID del portfolio, y se realizará el entrenamiento del modelo para cada activo financiero, al acabar, los pesos del modelo entrenado se guardarán para su uso posterior.

4.1.11. Simulación Montecarlo de portfolio.

Usando el modelo entrenado con la funcionalidad anterior, y el algoritmo descrito en el apartado [3.3.4.2](#) de esta memoria, podemos realizar una simulación Montecarlo que nos permitirá saber el aprendizaje real que ha hecho el modelo LSTM del histórico de precios del activo. En este caso el modelo también nos pedirá el ID del portfolio con el que queremos hacer la simulación, pero además nos pedirá cuantos días en el tiempo queremos hacer la simulación, y con cuantas trazas. Cuantos más días nos alejemos en el tiempo, más probabilidad de desviarnos hay, cuantas más trazas usemos, más información generaremos, por lo que menos desviación tendremos, aunque mayor coste computacional tendrá.

El resultado de la simulación será un gráfico donde la media de todas las trazas se verá en una línea más gruesa que las demás, de color rojo.



Figura 27: Simulación para el activo DELL para 50 días con 20 trazas.

En el caso de que tengamos muchas trazas, podemos hacer doble clic sobre la línea de la media en el menú derecho para ver solo esta.

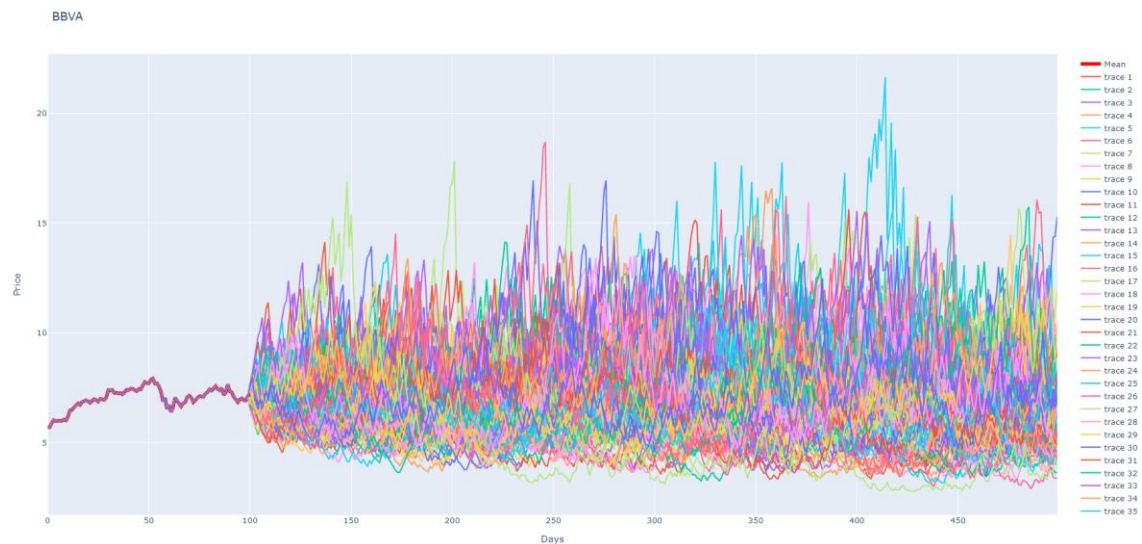


Figura 28: Simulación para BBVA de 500 días con 100 trazas.

En este caso, no podemos ver la media, pero si hacemos doble clic sobre ella en el menú derecho, veremos algo así:

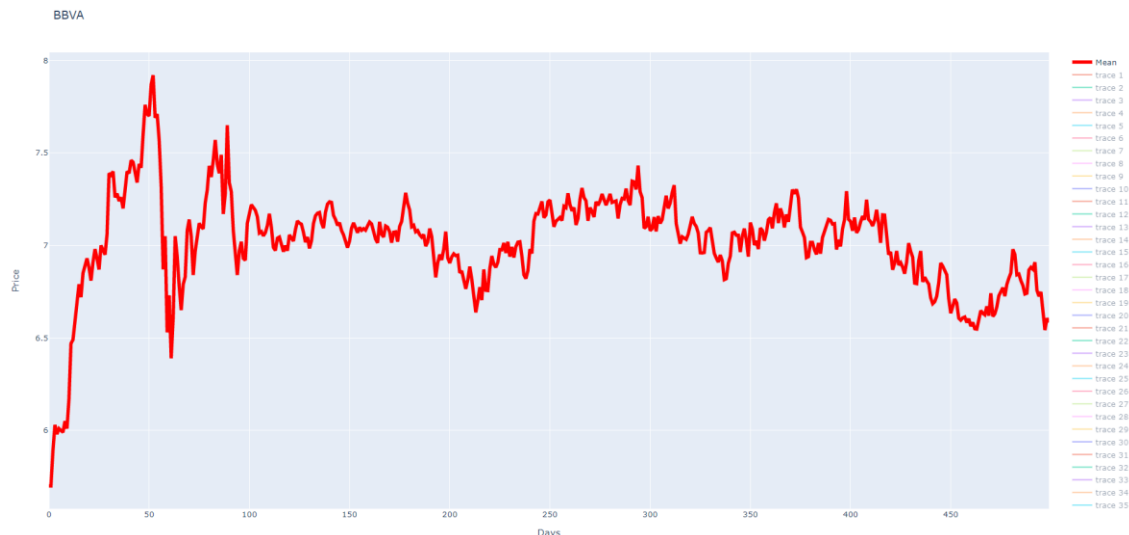


Figura 29: Simulación para BBVA de 500 días con 100 trazas. Foco solo en la media.

Ahora podemos ver perfectamente la media. Al hacer esto, las escalas del gráfico cambian, pero si analizamos un poco la tendencia de la media, cuadra con el precio del activo, su valor actual ronda los 6€, y recientemente ha tenido etapas de 7€, vemos como el modelo genera valores entorno a esos precios, y además viendo la tendencia negativa de los últimos años, replica que el valor poco a poco bajará de precio.

Esta es una de las variables que podemos obtener con esta simulación, pero también podemos obtener máximos, mínimos, valores más probables de la mitad por encima de la media, o valores más probables de la mitad por debajo de la media. El hecho de tener una simulación con decenas o centenas de trazas permite obtener la información que ha aprendido el modelo LSTM sobre el activo. Esta es la funcionalidad más potente de este proyecto.

4.1.12. Ver tabla de activos.

Esta funcionalidad permite ver la tabla de activos descargados.

4.1.13. Ver tabla de portfolios.

Esta funcionalidad permite ver la tabla de portfolios.

4.1.14. Ver tabla de activo.

Esta funcionalidad permite ver la tabla de un activo financiero. Realmente no es una función de utilidad para un usuario del programa, pero si es útil para desarrolladores, para que puedan ver cómo están estructurados los datos de un activo.

4.2. Comparativa Redes Neuronales.

En etapas tempranas del proyecto, se realizó una comparativa de las redes neuronales que posiblemente se podían usar para predecir el siguiente valor de una serie temporal. Se probaron las redes GRU, pero las redes LSTM ofrecieron un mejor rendimiento. El resto de las alternativas se descartaron por la dificultad de su uso para resolver este problema, y la baja efectividad de estas.

Tecnología	Pros	Contras
LSTM	Puede manejar series temporales	Aprende tendencias a un medio – corto plazo.
GRU	Puede manejar series temporales	Aprende tendencias a corto plazo.
Deep Feed Forward	Aprendizaje rápido	No es una red recurrente, no tiene capacidad de memorizar eventos en una serie temporal.
Red neuronal convolucional	Aprenden sobre el gráfico de la serie temporal que queremos predecir.	Sólo introducen una capa más de complejidad al problema, aprenden de un gráfico que por detrás tiene unos datos. ¿No es mejor usar los datos directamente? Es difícil que generen datos precisos.
Random Forest	Mejor explicabilidad que el resto de las alternativas.	Hay que generar múltiples datos para entrenarla, medias móviles, tendencias... Solo son capaces de clasificar el activo, por ejemplo, si tiene tendencia positiva o negativa.

Tabla 2: Comparación de modelos.

4.3. Estudio de optimizadores y funciones de perdida.

Una vez decidido que el mejor modelo para llevar a cabo el proyecto es LSTM, se ha hecho un estudio con algunos optimizadores y funciones de perdida distintos, para ver cual daba mejores resultados. Se han registrado los valores de la función de perdida, y el valor RMSE comparando los datos generados con los datos reales. A continuación, se muestra una tabla con la información obtenida.

	mean_squared_error	mean_absolute_error	mean_squared_logarithmic_error
adam	RSME:0.0187 loss: 6.3300e-04	RSME:0.0909 loss: 0.0191	RSME:0.0701 loss: 2.5188e-04
RMSprop	RSME:0.0485 loss: 6.9442e-04	RSME:0.0761 loss: 0.0214	RSME:0.0975 loss: 2.6640e-04
adadelata	RSME:0.1241 loss: 0.0026	RSME:0.1003 loss: 0.0284	RSME:0.1465 loss: 0.0011

Tabla 3: Rendimiento de optimizadores y funciones de pérdida.

También podemos ver a simple vista como aproxima la red neuronal en función del optimizador y función de perdida:



Figura 30: Resultado Adadelata + Mean Absolute Error



Figura 31: Resultado Adadelta + Mean Squared Error



Figura 32: Resultado Adadelta + Mean Squared Logarithmic Error



Figura 33: Resultado Adam + Mean Absolute Error

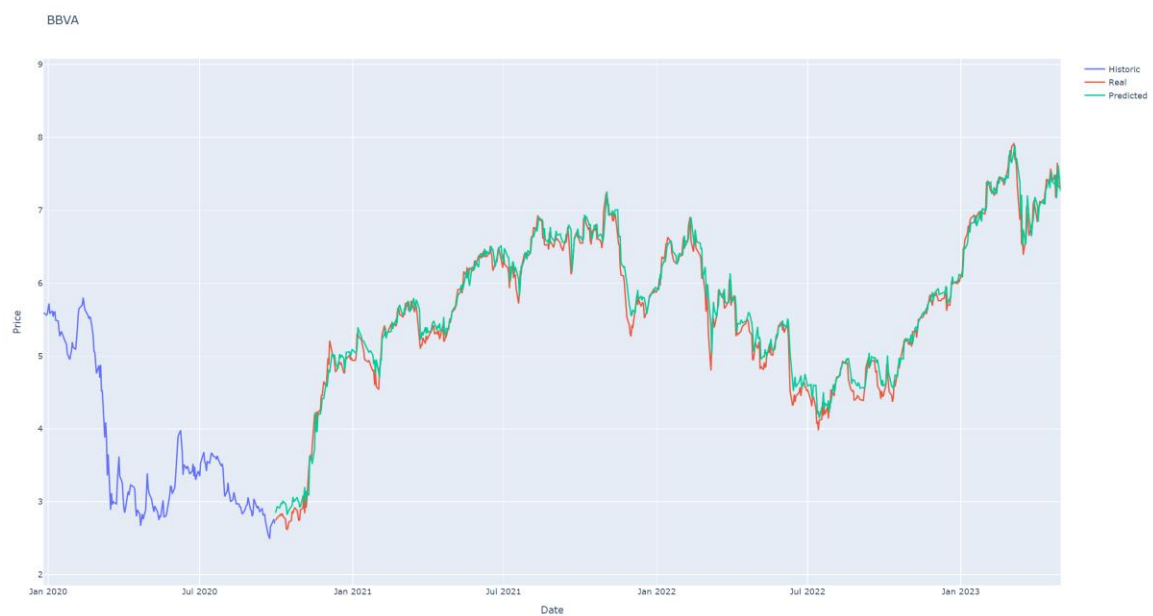


Figura 34: Resultado Adam + Mean Squared Error



Figura 35: Resultado Adam + Mean Squared Logarithmic Error



Figura 36: Resultado RMSprop + Mean Absolute Error



Figura 37: Resultado RMSprop + Mean Squared Error



Figura 38: Resultado RMSprop + Mean Squared Logarithmic Error

En vista de los resultados de la tabla y de los gráficos, podemos ver que el optimizador **adam**, con la función de perdida **mean_squared_error**, da los mejores resultados, **RMSprop** también da buenos resultados. Las funciones de perdida no afectan demasiado al resultado, en cambio, el optimizador usado sí que afecta, vemos como los modelos entrenados con el optimizador **adadelata** no son capaces de seguir la línea con precisión, y parece que están haciendo un cálculo de la media, en lugar de realmente prediciendo el siguiente valor.

4.4. Viabilidad del software como herramienta de predicción.

En este apartado vamos a analizar desde un punto de vista real, si este software puede ser usado realmente como una herramienta de predicción del precio de activos.

La idea de predecir algo es obtener sus últimos valores, y a partir de ahí estimar cual será su valor pasado X tiempo. El primer desarrollo que se hizo en este proyecto solo era capaz de generar el día siguiente a partir de los últimos 100 días, se intentó alargar el algoritmo para que predijera los siguientes 5 o 10 días, pero el resultado no fue el esperado, ya que lo único que hizo fue seguir la tendencia más reciente.

El segundo desarrollo, más robusto, permitió incorporar la simulación Montecarlo al modelo de red LSTM, siendo capaz de “extraer” la información que el modelo aprendió sobre el histórico de precios del activo. Este método si permite obtener un valor aproximado a futuro, aunque, una vez más, lo que estamos haciendo es estirar las tendencias que el modelo ha aprendido sobre el histórico. Pero ¿Realmente se puede obtener más información que esta a partir de la serie histórica? Parece que no, el precio del activo financiero esta completamente influenciado por infinitas variables externas, y poco o nada influenciado por su histórico, quizá solo por sus últimos valores.

Es por esto, que considero que lo máximo a lo que se puede llegar teniendo como conjunto de datos la serie histórica de precios, o, incluso, el volumen transaccionado, es a generar una serie de líneas de tendencias que van a replicar tendencias históricas, y nos pueden decir con más o menos exactitud el precio del activo en el futuro, si no sucede algún acontecimiento relevante a nivel macroeconómico.

Por tanto, ¿Se puede usar este software como herramienta de predicción? Sí, pero sin esperar una gran precisión de ella. Si no existen eventos de relevancia que influyan en el activo financiero, y este depende de si mismo, la tendencia de este seguirá su curso, y, por tanto, la tendencia que ha aprendido el modelo entrenado con el histórico, lo hará de similar manera.

4. Conclusiones y trabajos futuros

4.1. Conclusiones.

4.2. Consecución de los objetivos.

El objetivo principal del proyecto era generar un software capaz de predecir el precio de un activo financiero a futuro, por lo que este objetivo se ha cumplido. Además, todos los pequeños objetivos que se han ido definiendo también se han cumplido, generando un software funcional, con un menú interactivo, y que permite gestionar tanto los datos como los modelos entrenados. Podemos decir que los objetivos se han cumplido de forma satisfactoria.

4.3. Planificación y metodología.

La planificación de este proyecto se estableció en las etapas iniciales del mismo, se definieron objetivos generales, y objetivos específicos, se planificaron estos en un diagrama temporal buscando que se cumplieran las dependencias entre tareas, por ejemplo, no puedes desarrollar el modelo de IA si previamente no tienes un conjunto de datos de entrenamiento.

Una vez definida esta planificación y estos objetivos, el proyecto se desarrolló mediante metodologías Agile. La idea es ir haciendo pequeñas evoluciones cada pocas semanas que aporten algo de valor al proyecto, de tal forma que desde un inicio se pueda ver como poco a poco el software va tomando forma, y permitiendo corregir o modificar cualquier parte del software con el mínimo impacto en la planificación y tiempos de entrega.

Esto ha ayudado a que el proyecto se realice en tiempo y forma, siendo capaz de completar todos los objetivos, e incluso añadiendo alguna funcionalidad extra. Por tanto, podemos decir que la planificación fue adecuada, y que la metodología seguida ha ayudado a que el proyecto se complete en tiempo y forma, dando lugar al producto deseado.

4.4. Impactos previstos.

Los dos posibles impactos que hemos visto en la sección 1.3 son impacto en sostenibilidad por el coste computacional de ejecutar la simulación Montecarlo, y el impacto social/económico, ya que este software se puede entender como asesoramiento financiero, haciendo que alguna persona pueda poner en riesgo su dinero.

El impacto en sostenibilidad se ha tratado de mitigar haciendo el software lo más eficiente posible. Las librerías usadas se pueden ejecutar en modo CPU, lo cual es menos eficiente, pero en mi caso he configurado el programa y el entorno de ejecución para que se ejecuten con la GPU, lo cual no solo hace que se ejecute más rápido, también que sea más eficiente. Esto se podría mejorar aún más si el programa se decide ejecutar en un sistema de altas prestaciones con varias CPUs y varias GPUs, no solo mejoraría mucho en rendimiento, también al aprovechar el paralelismo y la eficiencia de estos sistemas, mejoraría la eficiencia de ejecución. Probablemente habría que hacer

alguna modificación en el software que haga que las distintas ejecuciones de la simulación se ejecuten en hilos o procesos paralelos.

El impacto social/económico es difícil de controlar, ya que este software está disponible de forma gratuita. Se pueden poner ciertas advertencias en el programa, pero quien lo quiera usar, lo hará igual. En un futuro, si la administración pública desarrolla un API que se pueda integrar con cualquier software y que, mediante el DNI electrónico, valide si eres mayor de edad, se podría hacer una verificación en este sentido, o si tienes estudios o ciertas certificaciones; pero de momento, esto no es posible.

La única medida real que se puede hacer en este sentido, y que se escapa del alcance del proyecto, es educar a la sociedad en economía y finanzas desde jóvenes, para que entiendan los riesgos y recompensas que el mundo de las inversiones puede tener.

4.5. Líneas de trabajo futuras.

Se han dejado varias líneas abiertas en el software que se pueden trabajar a partir de los resultados de la simulación. En primer lugar, combinando los resultados de la simulación con las funcionalidades de cálculo de matrices de correlación y covarianza, y cálculo del rendimiento, se puede obtener el riesgo y rendimiento a futuro del portfolio creado. Tenemos todos los datos, fechas, número de activos en posesión, rendimiento a futuro, volatilidad, y correlación de los activos.

Todas las funcionalidades que se desarrollaron como extras en el proyecto, tienen como objetivo en una línea de trabajo futura, acabar integradas dentro de las predicciones y simulaciones para que este software acabe siendo un producto “redondo” que permita gestionar portfolios de activos, calcular su valor presente, y su valor futuro gracias a las simulaciones hechas.

5. Glosario

- Series temporales: conjunto secuencial de valores ordenados en el tiempo.
- Redes neuronales: modelo matemático que simula el funcionamiento del cerebro para resolver problemas mediante el procesamiento de datos.
- Activos financieros: instrumentos negociables en mercados que representan un derecho de propiedad o de crédito en una entidad.
- Portafolio: colección de activos financieros que una persona o una entidad posee como su cartera de inversión.
- Mifid II: Markets in Financial Instruments Directive II es una regulación de la Unión Europea que busca fortalecer la protección al inversor y mejorar la transparencia en los mercados financieros.
- Basilea III: conjunto de normas internacionales para la regulación y supervisión bancaria, enfocado en fortalecer la solvencia y estabilidad financiera.
- Yahoo Finance: plataforma en línea que proporciona información financiera, noticias, y cotizaciones de acciones, y herramientas de análisis para inversores.
- Apache Spark: herramienta software de computación distribuida de código abierto que permite procesar grandes volúmenes de datos de manera rápida y eficiente.
- Notebook: Aplicación web que permite crear documentos interactivos que contienen código, texto explicativo, y visualizaciones.
- Agile: Metodología de gestión de proyectos que prioriza la flexibilidad, colaboración y adaptabilidad para entregar valor de manera incremental.
- RNN: Red Neuronal Recurrente, es un tipo de red neuronal que procesa secuencias de datos, capturando dependencias temporales para tareas como la predicción de series temporales.
- Backpropagation: es un algoritmo utilizado para entrenar redes neuronales, propagando el error desde la capa de salida hacia las capas ocultas para ajustar los pesos.
- LSTM: Long Short-Term Memory es un tipo de Red Neuronal Recurrente especializada en modelar secuencias largas capturando dependencias a largo plazo.
- GRU: Gated Recurrent Unit, es un tipo de red neuronal recurrente que simplifica la arquitectura de una LSTM para el procesamiento de secuencias en aprendizaje automático.
- Deep Feed Forward: también conocida como Red Neuronal Multicapa, es una arquitectura de redes neuronales en la que la información fluye en una dirección sin retroalimentación.
- CNN: Convolutional Neural Network, Red Neuronal Convolutiva, es un tipo de red neuronal especializada en el procesamiento y análisis de datos con estructura espacial, como imágenes.
- Random Forest: algoritmo de aprendizaje automático que combina múltiples árboles de decisión para realizar predicciones más precisas y robustas.

- RMSE: Root Mean Square Error es una métrica utilizada para evaluar el error promedio entre valores predichos y observados en modelos de regresión.
- MSE: Mean Square Error es una métrica utilizada para evaluar el error promedio entre valores predichos y observados en modelos de regresión.
- MSLE: Mean Square Logarithmic Error es una métrica utilizada para evaluar el error promedio entre valores predichos y observados en modelos de regresión.
- Función de pérdida: medida que cuantifica la diferencia entre los valores predichos y los valores reales en un modelo de aprendizaje automático.
- Optimizador: algoritmo que ajusta los parámetros de un modelo de aprendizaje automático para minimizar la función de pérdida.
- Adadelta: optimizador adaptativo utilizado en el entrenamiento de redes neuronales que ajusta la tasa de aprendizaje de manera adaptativa basada en la magnitud de los gradientes anteriores.
- Adam: Adaptive Moment Estimation es un optimizador popular utilizado en el entrenamiento de redes neuronales. Combina las ventajas de los algoritmos RMSprop y momentum.
- RMSprop: Root Mean Square Propagation es un algoritmo de optimización utilizado en el entrenamiento de redes neuronales, que ajusta la tasa de aprendizaje adaptativamente teniendo en cuenta la magnitud de los gradientes anteriores.
- Simulación Montecarlo: técnica numérica que utiliza números aleatorios para modelar y estimar resultados en problemas complejos y estocásticos.
- CPU: Unidad de Procesamiento Central, es el componente principal de un ordenador que realiza las operaciones y ejecuta los programas.
- GPU: Unidad de Procesamiento Gráfico, es un procesador especializado que acelera el procesamiento y renderización de gráficos y cálculos paralelos.
- Covid: es una enfermedad infecciosa causada por el virus SARS-CoV-2 que afecta principalmente el sistema respiratorio humano. Generó una pandemia global entre los años 2020 y 2021.

6. Bibliografía

An innovative neural network approach for stock market prediction [online] [marzo de 2020] [consulta: 4 de diciembre de 2022] Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, Victor Chang. Disponible en: https://www.researchgate.net/publication/322444003_An_innovative_neural_network_approach_for_stock_market_prediction

Predicting the Direction of Stock Market Index Movement Using an Optimized Artificial Neural Network Model [online] Zhen Wang, Kyushu University, JAPAN [19 de mayo de 2016] [consulta: 4 de diciembre de 2022] Mingyue Qiu, Yu Song. Disponible en: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0155133>

Stock Market Prediction Using LSTM Recurrent Neural Network [online] [abril de 2020] [consulta: 4 de diciembre de 2022] Adil Moghara, Mhamed Hamicheb. Disponible en: <https://www.sciencedirect.com/science/article/pii/S1877050920304865>

The application research of neural network and BP algorithm in stock price pattern classification and prediction [online] [febrero 2021] [consulta: 4 de diciembre de 2022] Dehua Zhang, Sha Lou. Disponible en: <https://www.sciencedirect.com/science/article/abs/pii/S0167739X20329861>

Apache Spark RDD Introduction [online] [consulta: 15 de mayo de 2023] <https://www.cloudduggu.com/spark/rdd-introduction/>

Directed Acyclic Graph DAG in Apache Spark [online] [consulta: 15 de mayo de 2023] <https://data-flair.training/blogs/dag-in-apache-spark/>

Introducción a Redes Neuronales Recurrentes (RNN) [online] [consulta: 15 de mayo de 2023] [http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/RNN LTS M/introduccion_rnn.html](http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/RNN_LTS_M/introduccion_rnn.html)

LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past [online] [consulta: 15 de mayo de 2023] <https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e>

¿Qué es la metodología Agile y por qué está de moda? [online] [consulta: 15 de mayo de 2023] <https://www.progressalean.com/metodologia-agile/>