Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos (ETSISI)

Trabajo de Fin de Grado
Grado en Ingeniería del Software

Predicción sobre mercado de valores con Redes Neuronales LSTM

Autor: Víctor A. Alcaraz López
Director: Dr. Fernando Ortega Requena

Madrid, Mayo 2022

Este Trabajo Fin de Grado se ha depositado en la ETSI Sistemas Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería del Software
Título: Predicción sobre mercado de valores con Redes Neuronales LSTM
Mayo 2022

Autor: Víctor A. Alcaraz López
Tutor:
Dr. Fernando Ortega Requena
Departamento de Sistemas Informáticos
ETSI Sistemas Informáticos
Universidad Politécnica de Madrid

A mi familia y a todas las personas que me han apoyado en estos años de estudio.

Resumen

La predicción del precio de las acciones supone un desafío en el área de Inteligencia Artificial. Es por ello que el propósito de este trabajo es mostrar si los actuales algoritmos de Deep Learning permiten realizar predicciones sobre series temporales con un error relativamente bajo.

La solución propuesta en este trabajo se basa en un análisis técnico. La metodología empleada consiste en la presentación de dos modelos basados en Redes Neuronales Recurrentes (de tipo LSTM), ambos modelos realizan una predicción a un día del precio de cierre de la acción de las empresas cotizadas en el índice NASDAQ-100. Se explica cómo recopilar, procesar los datos, entrenar y realizar predicciones con ambos modelos. Después se evalúa el error de ambos modelos con el indicador RMSE.

La principal conclusión del estudio es que el modelo que entrena con datos de todas las empresas en una única red neuronal (modelo 2) ofrece mejores resultados y su tiempo de entrenamiento es menor, teniendo un error medio (RMSE) en test de 56.78.

No obstante, se asume que el modelo cuenta con limitaciones dado que en un entorno tan complejo como la inversión bursátil se debe realizar también un análisis fundamental de las empresas. Además, se deben tener en cuenta aspectos macroeconómicos que influyen en los mercados financieros. Sin tener en cuenta estos aspectos resulta difícil realizar predicciones con un error lo suficientemente bajo como para llevar a cabo una inversión en el sector.

Abstract

Stock prediction is a challenge in the Artificial Intelligence area. That is why the purpose of this work is to show if the current Deep Learning algorithms can make predictions on time series with a relatively low error.

The solution proposed in this work is based on a technical analysis. The methodology used consists on the presentation of two models based on Recurrent Neural Networks (LSTM), both models make a one-day prediction of the stock closing price for the companies listed on the NASDAQ-100 index. This document explains how to collect data, process data, train the models, and make predictions with both models. Additionally, the error of both models is evaluated with the RMSE indicator.

The main conclusion of the study is that the model which trains with data from all the companies in a single neural network (model 2) offers better results and his training time is shorter, having a mean error (RMSE) in the test dataset of 56.78.

However, it is assumed that the model has limitations due to an environment as complex as stock market investment, a fundamental analysis of companies must also be carried out. In addition, macroeconomic aspects that influence financial markets must be taken into account. Without considering these aspects, it is difficult to make predictions with a low enough error to carry out an investment in the sector.

Palabras clave:

Inteligencia Artificial, Machine Learning, Deep Learning, Series temporales, LSTM, mercados financieros.

Tabla de contenidos

Acrónimos y definiciones

- EMH: Efficient Market Hypothesis
- AMH: Adaptive Market Hypothesis
- IA: Inteligencia artificial
- ML: Machine Learning
- DL: Deep Learning
- ANN: Artificial Neural Network
- FNN: Feed-Forward Neural Network
- RNN: Recurrent Neural Network
- API: Application Programming Interface
- LSTM: Long short-term memory
- MAE: Mean Absolute Error
- RMSE: Root Mean Square Error
- Outliers: valores atípicos de una muestra.
- EDA: Exploratory Data Analysis
- Epoch: ciclo completo que hace el modelo a través del conjunto de datos de entrenamiento
- Learning rate: factor de aprendizaje
- CNMV: Comisión Nacional del Mercado de Valores
- EAFI: Empresa de Asesoramiento Financiero)

1. Introducción

1.1 Motivación

En el ámbito de la inversión podemos encontrar muchos mercados como la bolsa, materias primas, forex o mercados de criptodivisas. Dentro de la inversión bursátil hay distintos índices que se corresponden con las plazas más importantes, algunos de los más relevantes son el Ibex 35 (España), Eurostoxx 50 (50 empresas más importantes de Europa), Dow Jones (EE.UU.), Nasdaq 100 (EE.UU.) o el S&P 500 (EE.UU.) [1].

Este trabajo supone para el autor una oportunidad de profundizar en el campo de la inteligencia artificial y concretamente en el tratamiento de series temporales. Además de aprender sobre algunos aspectos económicos imprescindibles para poder abordar el problema que se propone.

Las principales motivaciones de este proyecto son poner a prueba la capacidad de predicción en un entorno real de los algoritmos de inteligencia artificial disponibles actualmente así como aprender y adquirir destreza en el manejo de las principales librerías de tratamiento de datos para Python y Deep Learning (como Numpy, Pandas, Keras, etc).

1.2 Propuesta

Este trabajo propone la creación y optimización de dos modelos de predicción bursátil sobre el índice de referencia NASDAQ-100.

Se utiliza una arquitectura basada en Redes Neuronales Recurrentes (RNN), en concreto, una arquitectura de capas Long Short Term Memory apiladas (Stacked LSTM).

Se plantean dos modelos con una aproximación distinta al problema de predicción, en ambos casos se optimiza el modelo para tratar de obtener el mejor resultado con los datos disponibles y, por último, se evalúan los resultados obtenidos.

Ambos modelos presentados se pueden consultar en el siguiente enlace:
[victoralcarazlopez/StockPrediction (github.com)]

1.3 Objetivos

- Desarrollo de un modelo de predicción que permita tomar decisiones basadas en análisis técnico sobre el precio de una acción.
- Profundizar en el entendimiento de las tecnologías de Inteligencia Artificial y sus derivadas (Machine Learning y Deep Learning).
- Entendimiento del funcionamiento y dinámica del mercado de valores.
- Familiarizarse con el uso de librerías de tratamiento de datos e Inteligencia artificial como Numpy, Pandas, TensorFlow, Keras, etc.
- Entendimiento y puesta en marcha de una arquitectura de una red neuronal con capas apiladas.
- Optimización del tamaño de una red neuronal basado en el número y tipos de capas, así como, del tamaño de cada capa medido en número de neuronas.

- Aprender en un entorno real cómo llevar a cabo la recolección y gestión de datos para poder llevar a cabo una predicción.
- Entender cómo debe realizarse el pre-procesamiento de datos en Python (estandarización y normalización)
- Optimización de un modelo de predicción con redes neuronales en base a sus principales factores de ajuste como son learning-rate, número de Epochs, batch_size, etc.
- Realizar una evaluación de modelos de regresión dentro del ámbito del Deep Learning en base a su error..

2. Marco teórico

2.1 Mercado de valores

2.1.1 Introducción

De acuerdo con la Hipótesis del Mercado Eficiente (EMH), los precios de las acciones reflejan toda la información del mercado y no es posible vencer al mercado en general seleccionando acciones infravaloradas o sobrevaloradas [2]. Esta hipótesis sugiere que la única forma en que los inversores pueden vencer al mercado es asumiendo un mayor riesgo.

En el lado opuesto de la EMH, hay otra hipótesis llamada Hipótesis de Mercado Adaptativo (AMH). AMH es una teoría alternativa y sugiere que los inversores no siempre son racionales y las acciones no siempre cotizan a su valor objetivo.

Vemos, por tanto, que hay dos teorías contrapuestas, la primera dice que no es posible vencer al mercado y la otra dice que a veces los inversores pueden aprovechar las debilidades del mercado.

2.1.2 Análisis técnico vs análisis fundamental

En el ámbito de la inversión financiera encontramos dos tipos de análisis. El análisis fundamental estudia el valor de un activo en relación con su modelo de negocio, balances y registros financieros. Por otro lado, el análisis técnico estudia el comportamiento del precio de un activo durante un tiempo determinado basándose generalmente en precios, tendencias, análisis de gráficas, etc. [3]

Ambos tipos de análisis son interdependientes, si bien el análisis fundamental estudia la viabilidad de un negocio en un lapso de tiempo mayor (10 años o incluso más), el análisis técnico centra su atención en un periodo más corto (días en la mayoría de los casos).

El presente trabajo pretende centrarse en el análisis técnico, dado que este tipo de análisis apunta a un lapso de tiempo mucho menor y permite determinar si es un buen momento para entrar o salir en un activo.

2.1.3 NASDAQ-100

El NASDAQ-100 o NDX100 es un índice bursátil de Estados Unidos creado en 1985 que recoge los valores de las 100 compañías más importantes del sector de la industria de la tecnología, telecomunicaciones, salud, multimedia y otros servicios.

Figura 2.1. Evolución del índice NASDAQ-100 desde su creación (Fuente: [4])

La figura 2.1 muestra la evolución del índice desde 1985 hasta 2019. En esta figura se puede apreciar la burbuja ocurrida entre el año 1999 y 2001. También es destacable el crecimiento sostenido desde 2010 hasta la actualidad, habiendo multiplicado su valor por 5 en los últimos 10 años.

Sus futuros se negocian en el mercado de valores de Chicago. Este índice recoge algunas de las empresas del sector tecnológico más importantes como pueden ser Apple, Google/Alphabet, Amazon, Microsoft, Facebook, Tesla, etc.

2.2 Estado del arte

Hoy en día, es común que las empresas comerciales y los inversores apoyen sus decisiones en análisis técnicos y fundamentales. Pero esto no siempre fue así, el *forecasting,* consiste en llevar a cabo predicciones basadas en datos pasados y presentes.

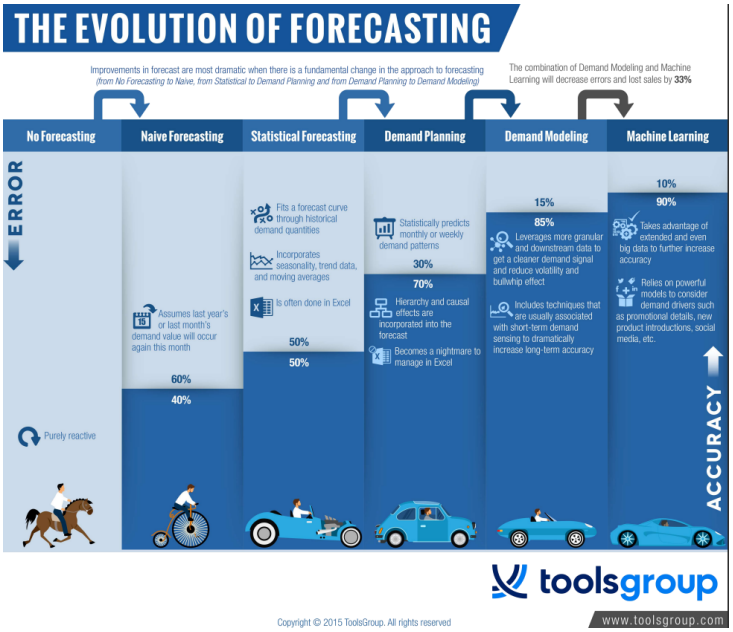


Figura 2.2: Evolución del forecasting (Fuente: [5])

La figura 2.2 muestra como con el tiempo, los seres humanos se han ido apoyando en distintas tecnologías para hacer *forecasting* (pronósticos).En un primer momento las predicciones eran 'ingenuas', esto es, se llevaban a cabo suponiendo que lo ocurrido hasta ahora volvería a ser similar en el futuro. En la actualidad existen modelos basados en Big Data, que tienen en cuenta redes sociales, usan Inteligencia Artificial, etc. Estos modelos han conseguido reducir el error de manera drástica a la hora de realizar predicciones.

En el análisis técnico, los algoritmos de aprendizaje automático y deep learning son realmente útiles para predecir los precios de las acciones en función de datos históricos [6]. Pero no solo la IA proporciona información a los inversores en análisis técnico. En el análisis fundamental, los algoritmos de ML de predicción de acciones evalúan a las empresas en función de distintos modelos, siendo uno de los más importantes el conocido como F-Score [6].

2.3 Inteligencia Artificial

En el mundo de la Inteligencia Artificial podemos abarcar principalmente cuatro tipos de problemas [7]:

- **Clasificación**: el resultado a predecir pertenece a un número limitado de clases.
- **Regresión**: el objetivo a predecir es un valor de entre infinitos valores posibles.
- **Clustering**: el objetivo es agrupar conjuntos de valores en base a características comunes.
- **Reducción de dimensionalidad**: el objetivo es mapear el conjunto de datos a un espacio menor que el original, generalmente reducir el número de variables en una colección de datos.

Gracias a la Inteligencia Artificial y algunas de sus ramas como son el Machine Learning o el Deep Learning se pueden crear modelos capaces de llevar a cabo funciones de clasificación y regresión.

Hablamos de clasificación cuando el resultado a predecir pertenece a un número limitado de clases, podrían ser 2 clases si hablamos de predecir si una determinada empresa va subir de precio o no. Por otro lado, los algoritmos de Inteligencia Artificial también permiten resolver problemas de regresión en los que el objetivo es predecir un valor numérico, en este caso, podría ser predecir el valor de cierre de una acción al día siguiente.

Existen distintas aproximaciones en el campo de la IA para un problema de regresión. Entre las principales destacan cuatro grandes grupos [8]:

- Regresión lineal
- Regresión polinómica
- Árboles de decisión
- Deep Learning

Este estudio se centra en este último campo, el Deep Learning, y concretamente, en las Redes Neuronales.

Otra posible agrupación de los algoritmos de Inteligencia Artificial está basada en el tipo de aprendizaje. Nos encontramos así cuatro grupos [9]:

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje semi supervisado
- Aprendizaje por refuerzo

En el caso que nos ocupa el tipo de aprendizaje que utiliza el modelo es supervisado, pues disponemos de un histórico de datos con los precios de las distintas acciones. Esto permite indicarle al sistema cuál es el valor correcto de una determinada predicción y calcular el error para saber cómo de bien o de mal funciona el modelo.

2.3.1 Algoritmos basados en Deep Learning

El Deep Learning es una disciplina comprendida dentro del Machine Learning. En esencia cuando se habla de Deep Learning se hace referencia a redes neuronales con varias capas [10]. Una de las principales ventajas en Deep Learning es que permite trabajar con datos no estructurados como imágenes, texto, video, etc.

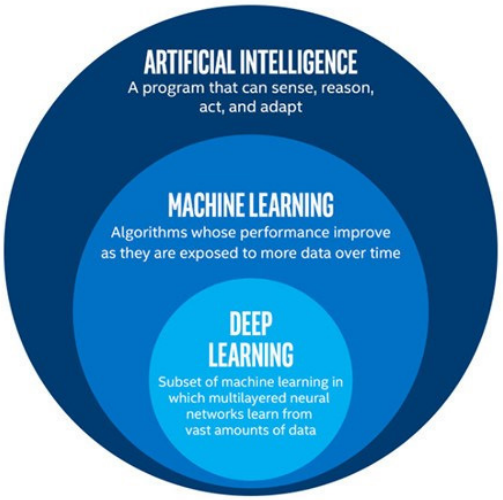


Figura 2.3. Categorización campos dentro de la Inteligencia Artificial. (Fuente: [11])

Existen distintas aproximaciones en el mundo del Deep Learning para tratar de predecir un valor, en este caso un precio de mercado a futuro. En todos los casos se trata de Redes Neuronales Artificiales (ANN) en sus variantes LSTM, GRU, BiLSTM, etc.

2.3.2 Redes Neuronales Recurrentes (RNN)

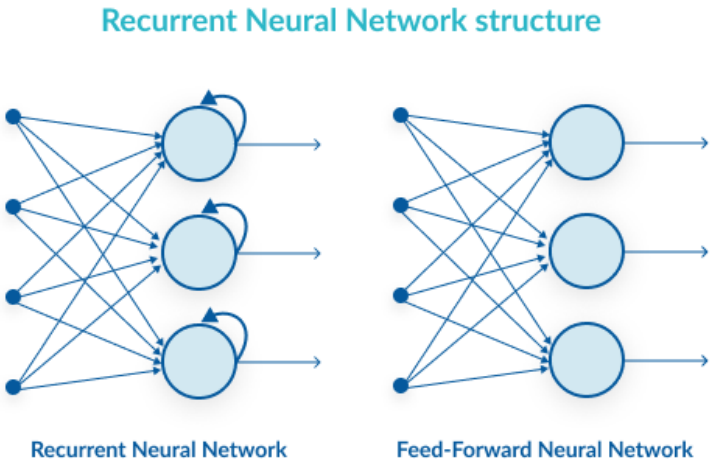


Figura 2.4 Comparación tipos de ANN. (Fuente: [12])

La figura 2.4 muestra los tipos de ANN que podemos encontrar, principalmente dos tipos de redes:

- **Feed-Forward Neural Networks (FNN)**: la información fluye en una única dirección, desde los nodos de entrada por las capas ocultas hacia los nodos de salida.
- **Recurrent Neural Networks (RNN):** la información avanza igual que en la estructura Feed-Forward pero se permite bucles en su interior.

Las RNN ofrecen la posibilidad de recordar salidas anteriores como entradas. Esto hace que sean muy útiles a la hora de trabajar con datos que presentan una secuencia temporal. Para este proyecto se presenta un modelo de predicción basado en RNN, esta decisión ha sido tomada teniendo en cuenta las ventajas y desventajas [13].

Entre las principales ventajas de las RNN encontramos:

- Permiten trabajar con entradas de gran tamaño. Además el tamaño del modelo es independiente del tamaño de la entrada, es decir, el número de capas apiladas no depende del tamaño de la entrada.
- Permite mantener los pesos de la red a lo largo del tiempo.
- Permiten lidiar con el ruido.

Sus principales desventajas serían:

- El entrenamiento es lento y computacionalmente costoso.
- Se puede producir desvanecimiento de gradiente.

2.3.3 Long Short-Term Memory (LSTM)

Dentro de las RNN encontramos un tipo de arquitectura llamadas Long Short-Term Memory (LSTM), estas redes permiten trabajar con datos en los que se establece una secuencia temporal, por ello son ampliamente utilizadas en campos relacionados con el lenguaje natural, lenguaje musical, bolsa, etc.

Este tipo de arquitectura se ajusta al problema de predicción en bolsa ya que se trabaja con datos que siguen una relación temporal y están correlacionados en base a su cercanía temporal.

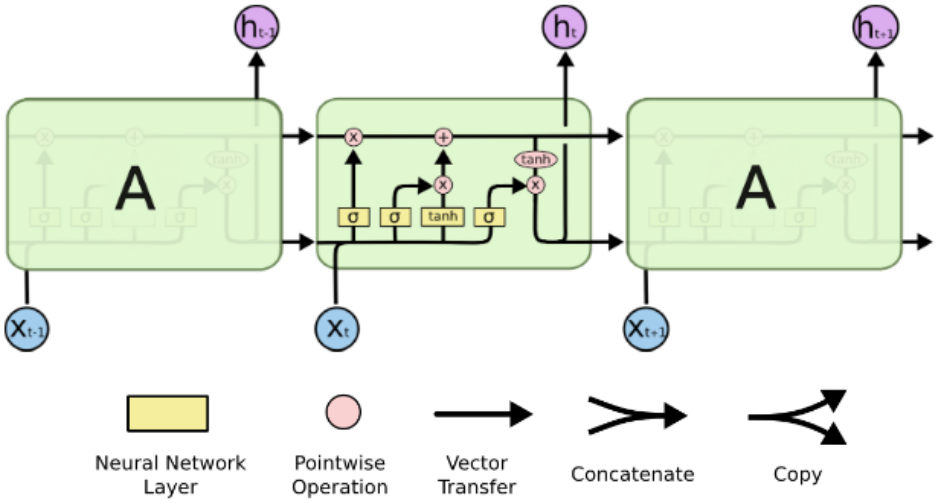


Figura 2.5: Comportamiento en el tiempo de una capa LSTM (Fuente [14])

En la figura 2.5, Xt representa la entrada para la red neuronal en t. Ht representa la salida en t. En esta misma figura se puede apreciar como la entrada a la capa LSTM en un instante t es Xt y también es una entrada la salida en t-1. A su vez, la salida de la capa en el instante t, supone la entrada en t+1.

Una capa LSTM presenta distintas partes que permiten llevar a cabo la tarea de olvidar o recordar determinada información en base a la relevancia que esta información tenga para una predicción.

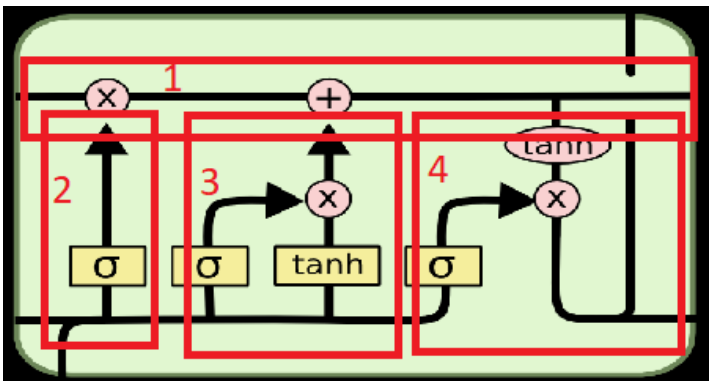


Figura 2.6: Detalle de partes de una capa LSTM (Fuente: Elaboración propia a partir de [14])

Viendo la figura 2.6, se distinguen cuatro partes en una capa LSTM de una red neuronal:

1. **Memory Cell**: su función principal es recordar y olvidar cosas según el contexto de la entrada (la salida del estado anterior y la salida de la Forget Gate).

2. **Forget Gate**: utiliza la función de activación sigmoide. La función sigmoide devuelve valores entre 0 y 1.
3. **Input Gate**: la operación tangente hiperbólica (tanh) devuelve valores entre -1 y 1. Por eso, la salida de la puerta I/O será un vector con valores entre -1 y 1.
4. **Output Gate**: salida para ser operada con la memory cell y producir la salida de la capa.

Generalmente, X (la entrada desde abajo) será un vector. La operación puntual con el símbolo de agregar adentro significa que agrega información a la celda de memoria.

2.4 Series temporales

Una serie temporal es una sucesión de observaciones de una variable realizadas a intervalos regulares de tiempo.

En el ámbito económico la mayor parte del material estadístico se corresponde con series temporales. Cuando se trabaja con este tipo de datos se añade una capa extra de complejidad respecto a trabajar en problemas de regresión tradicional.

Las principales diferencias frente a un problema de regresión son [15]:

- **Dependencia en el tiempo**: trabajando en un problema de regresión lineal las observaciones son independientes, con series temporales no.
- **Tendencia a la estacionalidad:** esto ocurre si, por ejemplo, se analiza la facturación de una empresa alquila material de esquí, la mayor parte de la facturación se centrará en los meses de invierno.

Además, cuando se trabaja con series temporales es necesario tener en cuenta los siguientes supuestos [16]:

- Se considera cierta estabilidad estructural el fenómeno estudiado, es decir, los periodos estudiados deben ser lo más homogéneos posibles.
- Se debe mantener la definición y la medición de la magnitud del objeto de estudio. Esto a veces no ocurre si, por ejemplo, en el periodo de tiempo estudiado se produce un cambio en el índice de producción, esto altera el indicador y dificulta la comparabilidad de la serie.

2.5 Medidas de calidad

En aprendizaje supervisado es imprescindible tener una métrica que indique cómo de bien o mal está funcionando el modelo. Al tratarse de un modelo de regresión con variables continuas los indicadores más utilizados son MAE y RMSE [17].

Mean Absolute Error (MAE)

MAE mide la magnitud media del error cometido en N predicciones. Calculado como el sumatorio del valor absoluto de todos los errores cometidos. Después se divide entre el tamaño de la muestra para obtener la media de todos los errores.

$$MAE = \frac{1}{n} \sum_{j=1}^{n} | Y_j - \widehat{Y}_j |$$

Donde:

$\sum_{j=1}^{n}$ = Sumatorio de los errores

$| Y_j - \widehat{Y}_j |$ = El error medido como la diferencia entre el valor real y la predicción.

n = Tamaño de la muestra.

Root Mean Square Error (RMSE)

RMSE consiste en calcular el sumatorio de todos los errores al cuadrado. Después se divide por el número de muestras para calcular la media. Por último se calcula la raíz cuadrada.

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^{n} (Y_j - \widehat{Y}_j)^2}$$

Donde:

$\sum_{j=1}^{n}$ = Sumatorio de los errores

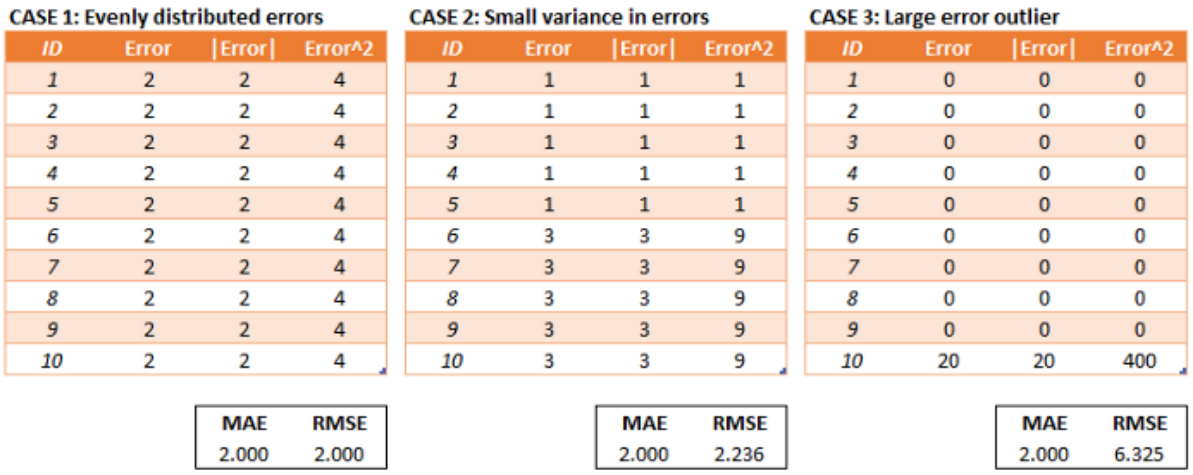
$(Y_j - \widehat{Y}_j)^2$ = Es el error elevado al cuadrado, el error es medido como la diferencia entre el valor real y la predicción realizada.

n = Tamaño de la muestra.

Dado que estos indicadores miden el error su valor estará en el intervalo [0, +∞). El objetivo a perseguir es que el modelo minimice este error hasta que idealmente este sea 0.

La principal diferencia entre estos dos estimadores es que RMSE eleva al cuadrado el error antes de hacer la media. Esto hace que los errores mayores tengan un mayor peso pues:

$e^2 <<< (5e)^2$ (siendo e el error cometido)



CASE 1: Evenly distributed errors

ID	Error	|Error|	Error^2
1	2	2	4
2	2	2	4
3	2	2	4
4	2	2	4
5	2	2	4
6	2	2	4
7	2	2	4
8	2	2	4
9	2	2	4
10	2	2	4

MAE	RMSE
2.000	2.000

CASE 2: Small variance in errors

ID	Error	|Error|	Error^2
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	3	3	9
7	3	3	9
8	3	3	9
9	3	3	9
10	3	3	9

MAE	RMSE
2.000	2.236

CASE 3: Large error outlier

ID	Error	|Error|	Error^2
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	20	20	400

MAE	RMSE
2.000	6.325

Figura 2.7: Comparación de MAE y RMSE ante un mismo error cometido (Fuente [17])

La figura 2.7 ayuda a comprender cómo puede variar el error en función de la medida de calidad utilizada para medir un mismo error. El caso más extremo sería el caso 3 en el que hay 10 mediciones y 9 de ellas tienen un error de 0, pero la última tiene un error de 20.

En este caso, al calcular el error con la métrica MAE el valor obtenido es 2. Este valor se obtiene al dividir la suma de los errores totales (20) entre el número de mediciones (10). Sin embargo al usar RMSE el error se eleva al cuadrado antes de hacer la media y tenemos un error de 400 (20^2), este error se divide entre el número de mediciones (10) y, por último, se aplica la raíz cuadrada a lo obtenido $(\sqrt{\frac{400}{10}})$, con esto se obtiene el valor de 6.325 que es más del triple que el valor obtenido al usar MAE.

La evaluación del modelo se realiza utilizando el estimador RMSE. Este estimador es el que integran por defecto la mayor parte de librerías y el que suele presentar un mejor resultado. Además, se considera que es beneficioso penalizar aquellos errores más grandes dado el problema que se pretende resolver. Por otro lado, la principal ventaja que ofrece MAE frente a RMSE es que al tratarse del error medio, permite tener una idea precisa de cuánto error está cometiendo el modelo.

3. Desarrollo

3.1 Caso de estudio

Se propone realizar una predicción a un día de las 100 empresas del NASDAQ-100 partiendo de los datos de cada empresa para los últimos 5 años. Los datos consisten en una serie temporal de precios, volumen de transacciones y otros datos recolectados desde una API externa.

Se proponen dos modelos alternativos y se evalúan los resultados obtenidos con cada uno de ellos:

- **Modelo 1 - Entrenamiento individual para cada empresa:** para el primer modelo se trata cada empresa de manera individual, se pre-procesan los datos de esa empresa, se construye una Red Neuronal Recurrente, se entrena y se realizan los test correspondientes. Este proceso es repetido por cada empresa y finalmente se evalúa el error medio cometido entre todas las empresas habiendo realizado una predicción individual para cada una de ellas.

- **Modelo 2 - Entrenamiento general con todas las empresas:** para el segundo modelo se propone crear una única red neuronal que tome los datos de entrenamiento y test correspondientes a todas las empresas en conjunto. Al igual que con el modelo 1 los datos son pre-procesados, se entrena y se evalúa el error medio cometido en el entrenamiento y las predicciones.

3.2 Recolección de datos.

En el repositorio del proyecto se dispone del fichero ‘Data_Collection.ipynb’.

Este fichero de tipo Jupyter notebook realiza la tarea de descargar los datos que se utilizaran en el modelo mediante la plataforma de finanzas Tiingo. Esta plataforma provee una API con la que se realiza la recolección de los datos. La API Tiingo se utiliza para descargar los datos de cada empresa como son precio de la acción al cierre, precio máximo diario, precio mínimo diario, etc.

El proceso requiere ser dividido en 2 partes, debido a que la API ofrece una limitación de un máximo de 50 peticiones/hora. Una vez hecho esto se recogen todos los datos en bruto y se almacenan en ficheros en formato .csv. con el nombre de cada empresa. De esta manera el modelo siempre va a trabajar con los mismos datos, esto ofrece la ventaja de que las posibles variaciones en el error nunca van a darse por cambios en los datos.

De esta manera se realiza la recolección de datos sobre las empresa del NASDAQ-100 (los ‘Ticker symbol’ se almacenan en el fichero ‘Symbols_Nasdaq100.txt’)

3.3 Análisis exploratorio de los datos

El Análisis exploratorio de los datos (EDA) consiste en un primer análisis para ver cuántos datos tenemos, si hay datos incompletos, si hay valores nulos, etc.

Este análisis también puede ayudar a detectar qué características son más relevantes, descartar las variables que no lo son, identificar qué distribución estadística siguen las variables, así como, empezar a trabajar sobre cuál será la posible salida del modelo (número de salidas, valores discretos o continuos, etc).

El NASDAQ-100 es un índice en el encontramos 100 compañías. Tras una primera exploración de los datos que nos ofrece la API utilizada vemos que de alguna de las empresas apenas se ofrecen datos de los últimos 200 días. Es por ello que se decide descartar aquellas compañías de las que no se tengan, al menos, 1000 días de información.

Por cada compañía disponemos de la siguiente información:

- **Symbol**: identificador de la compañía.
- **Date**: La fecha a la que pertenecen estos datos.
- **Close**: El precio de cierre del activo en la fecha indicada.
- **High:** El precio más alto alcanzado por el activo en la fecha dada.
- **Low:** El precio más bajo alcanzado por el activo en la fecha dada.
- **Open**: El precio de apertura del activo en la fecha indicada.
- **Volume**: El número de acciones negociadas por el activo.
- **adjClose**: El precio de cierre ajustado para el activo en la fecha dada.
- **adjHigh**: El alto precio ajustado para el activo en la fecha dada.
- **adjLow**: El precio bajo ajustado para el activo en la fecha dada.
- **adjOpen**: El precio de apertura ajustado para el activo en la fecha dada.
- **adjVolume**: El número de acciones ajustadas negociadas por el activo.
- **divCash**: El dividendo pagado en "fecha" (tenga en cuenta que "fecha" será la "exDate" para el dividendo).
- **splitFactor**: El factor utilizado para ajustar los precios cuando una empresa se divide, revierte o paga una distribución

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 1258 entries, ('AMZN', Timestamp('2017-05-18 00:00:00+0000', tz='UTC')) to ('AMZN',
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   close        1258 non-null   float64
 1   high         1258 non-null   float64
 2   low          1258 non-null   float64
 3   open         1258 non-null   float64
 4   volume       1258 non-null   int64
 5   adjClose     1258 non-null   float64
 6   adjHigh      1258 non-null   float64
 7   adjLow       1258 non-null   float64
 8   adjOpen      1258 non-null   float64
 9   adjVolume    1258 non-null   int64
 10  divCash      1258 non-null   float64
 11  splitFactor  1258 non-null   float64
dtypes: float64(10), int64(2)
memory usage: 164.0+ KB
```

Figura 3.1: Información de los datos disponibles para la empresa Amazon. (Fuente: elaboración propia)

De la figura 3.1 se puede observar que para la compañía Amazon se dispone de registros de los últimos 1258 días, entre los cuales no hay valores nulos para ninguno de los campos. Esto es lo deseable y no requiere ningún tratamiento por el momento.

El dato más importante sería el precio de cierre de la acción es de tipo float64. Es lo adecuado para representar números en coma flotante, por tanto, tampoco se realizará ningún cambio en el tipo de datos.

3.4 Preprocesamiento de datos

3.4.1 Reducción de dimensionalidad

La reducción de dimensionalidad es uno de los problemas que aparecen cuando trabajamos con gran cantidad de datos.

		close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash	splitFactor
symbol	date												
AMZN	2017-05-18 00:00:00+00:00	958.49	962.75	944.76	944.80	3876816	958.49	962.75	944.76	944.80	3876816	0.0	1.0
	2017-05-19 00:00:00+00:00	959.84	968.92	959.72	962.84	3846299	959.84	968.92	959.72	962.84	3846299	0.0	1.0
	2017-05-22 00:00:00+00:00	970.67	971.38	962.90	964.00	2613805	970.67	971.38	962.90	964.00	2613805	0.0	1.0
	2017-05-23 00:00:00+00:00	971.54	975.20	966.85	975.02	2395510	971.54	975.20	966.85	975.02	2395510	0.0	1.0
	2017-05-24 00:00:00+00:00	980.35	981.00	970.23	976.00	2407995	980.35	981.00	970.23	976.00	2407995	0.0	1.0

Figura 3.2: Primeras filas de datos para la empresa Amazon. (Fuente: elaboración propia)

Como se puede apreciar en la figura 3.2, la API utilizada provee gran cantidad de información sobre el precio de una acción.

Una posible aproximación al problema sería introducir toda la información a la entrada del modelo y esperar una respuesta. Pero si tenemos un cierto conocimiento sobre el campo que estamos tratando podemos tratar de identificar aquellas variables más relevantes con el objetivo de reducir el número de conexiones dentro de la red y, por tanto, la cantidad de parámetros a optimizar.

Con esto conseguiremos mejorar el rendimiento computacional y la complejidad. Como consecuencia de lo anterior, el entrenamiento y optimización de hiperparámetros de la red será más rápido. Además, será más sencillo visualizar la información. Como inconveniente, al realizar la reducción dimensional hay que tener cuidado para no perder datos que pudieran ser relevantes y generar un modelo que produzca un resultado no deseado.

Para seleccionar aquellas variables más importantes en Machine Learning se siguen dos criterios[18]:

- **Criterio de correlación**: existen métodos matemáticos para determinar si existe correlación entre distintas variables. Normalmente, interesa descartar aquellas variables que no tengan ninguna correlación con la variable objetivo. Hay que

prestar atención a que la demostración de que exista correlación entre dos variables no implica que exista causalidad.
- **Criterio de consistencia**: si existen variables redundantes no merece la pena introducir información duplicada.

Por simplicidad y dado el alcance del proyecto se decide seleccionar como entrada al modelo la variable objetivo, en este caso, el precio de cierre de la acción diario. Se mantiene también la fecha y para una mayor sencillez se va a utilizar el propio índice del Dataframe de la librería pandas para representar la fecha. Utilizando estos dos datos tenemos un modelo con una gran simpleza, que mantiene un alto nivel de correlación y sin redundancia.

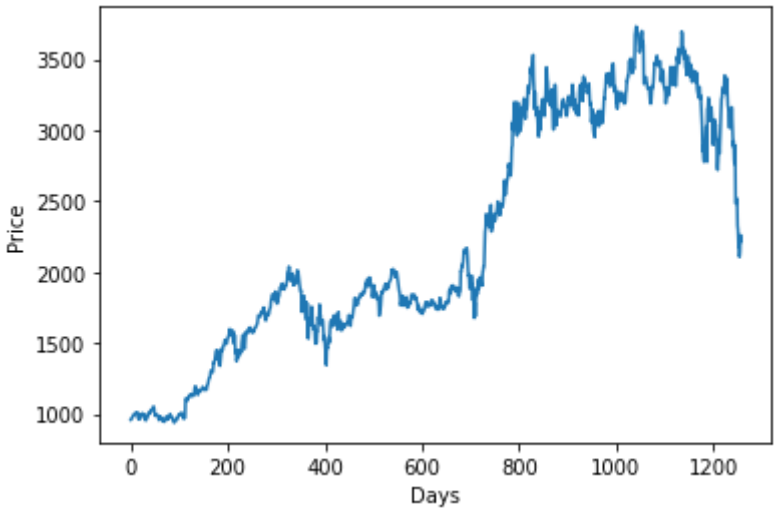


Figura 3.3: Precio de cierre de las acciones de Amazon en los últimos 1258 días. (Fuente: elaboración propia)

La figura 3.3 muestra los datos con los que va a trabajar el modelo, en este caso, la fecha y el precio de cierre de la acción, que es la variable objetivo.

3.4.2 Tipos de datos

Generalmente, podemos clasificar los datos en base a su organización:
- Datos **estructurados**: como hojas de cálculo, ficheros .csv, fichas estandarizadas, etc.
- Datos **no estructurados**: imágenes, archivos pdf, datos de redes sociales, videos, etc.
- Datos **semi-estructurados**: comúnmente suelen ser ficheros XML y de otros lenguajes de marcado, archivos comprimidos, etc.

En el caso del problema de predicción de bolsa los datos que vamos a tratar son exclusivamente estructurados ya que manejamos datos que nos provee una API y que se almacenan en ficheros .csv.

Desde la perspectiva de la Inteligencia artificial se suele clasificar los tipo de datos de la siguiente manera [19]:

- **Datos cuantitativos o numéricos**: se trata de números o cosas que pueden medirse. A su vez se subdividen en:
    - **Datos continuos**: generalmente mediciones. Se pueden definir con una precisión infinita. Ejemplo: precios, temperatura, etc.
    - **Datos discretos**: generalmente recuentos. Ejemplo: nº de personas (1, 2, 3, 4…)
- **Datos categóricos**: representan características. Ejemplo: posición, ciudad de origen. Dentro de esta categoría podríamos añadir los **datos ordinales**, estos datos se clasifican en categorías y esas categorías se ordenan. Ejemplo de datos ordinales: Pequeño / Mediano / Grande.
- **Datos de series temporales**: son datos numéricos recopilados en intervalos regulares. Ejemplo: los precios de cierre en bolsa diarios de una determinada empresa.
- **Datos de texto**: por ejemplo palabras o texto plano.

Para el caso de estudio que se trata en este trabajo dispondremos de series temporales que a su vez son datos numéricos. Una variable que será el índice nos indica el día, es un dato numérico discreto. La otra variable es el precio y es un tipo de dato numérico continuo.

3.4.3 Escalado de datos: Normalización y Estandarización

El escalado de datos es muy importante para ayudar a que la velocidad de convergencia sea baja y la precisión alta en modelos de ML y DL.

Las dos técnicas más populares a la hora de escalar datos numéricos son:

- **Normalización**: consiste en escalar cada variable de manera individual al rango [0,1].
- **Estandarización:** consiste en para cada variable restar la media (centrado) y dividir por la desviación típica. Con esto se obtiene una media de 0 y una desviación estándar de 1.

3.4.3.1 Preprocesamiento de datos continuos

Generalmente, a los datos continuos se les realiza una transformación con el fin de acotarlos a un rango de valores que permita compararlos entre sí independientemente de su naturaleza.

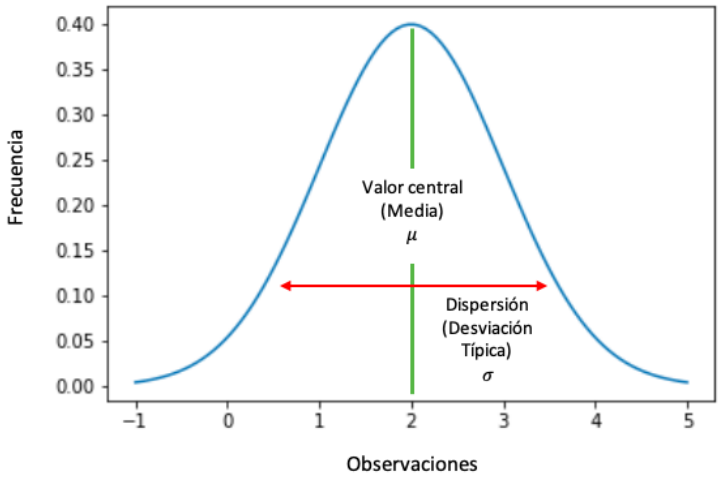


Figura 3.4: Ejemplo de distribución Gaussiana o Normal (Fuente: [20])

Es muy frecuente encontrar que los datos siguen una distribución normal con media μ y desviación típica σ (como los datos representados en el ejemplo de la figura 3.4)

Normalización - Feature Scaling o Min-Max Scaler

La normalización consiste en ajustar la variable a un rango predefinido, generalmente en el rango [0,1] aunque también puede hacerse al rango [-1,1]. Usualmente se utiliza cuando se tienen datos con una desviación típica muy pequeña, es decir, no hay muchos outliers.

Fórmula de normalización con Min-Max Scaler:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Este es un método que presenta un problema y es que dado que realiza una compresión de los datos al rango [0,1] puede amplificar el ruido. Por ejemplo si tenemos datos con una desviación típica baja, al aplicar Min-Max Scaler podemos amplificar el ruido y distorsionar los datos.
Existen alternativas para normalizar en datasets con outliers como Robust Scaler. Robust Scaler no sufre al escalar datos con outliers pues trabaja con percentiles. A diferencia de Min-Max Scaler es capaz de realizar una normalización en un rango mayor acorde con los datos disponibles. Se caracteriza por transformar los datos empleando estadísticos robustos a los outliers. Destacar que una vez realizado el escalado no suprime los outliers con lo que no se pierde información.

Estandarización - Standard Scaler

Existe otra alternativa para la estandarización de las características que suele ser útil aplicarla cuando los datos siguen una distribución normal.

La estandarización es el proceso, a partir del cual, un conjunto de datos que siguen una distribución normal, hecho que sucede con la mayoría de los datos empleados en machine learning, son transformados a una distribución normal con media 0 y desviación típica 1.

La operación de estandarización se lleva a cabo para cada muestra operando:

$$X_{normalized} = \frac{X - X_{mean}}{X_{stddev}}$$

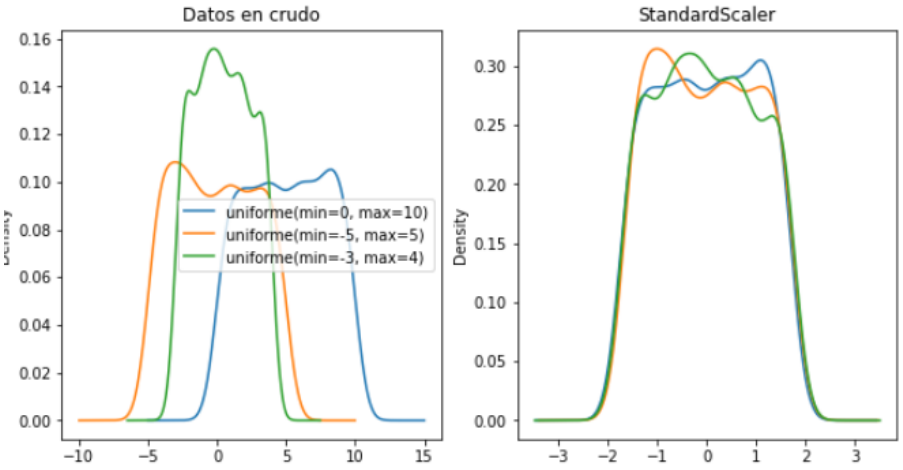


Figura 3.5: Ejemplo de datos sin estandarizar y datos estandarizados. (Fuente: elaboración propia)

En la figura 3.5 se puede observar como quedan los datos tras aplicar estandarización. Se puede observar claramente como la desviación típica se reduce y pasa a ser 1. Igualmente los datos se 'centran' haciendo que la media sea 0.

3.4.3.2 Preprocesamiento de datos discretos

Los modelos de Machine Learning requieren que tanto los datos de entrada como de salida sean numéricos. El principal problema se da cuando aparecen datos categóricos como, por ejemplo, podría ser el sexo que podría tomar los valores 'masculino'' o 'femenino' y eso no es tratable por un algoritmo de Machine Learning.

A continuación se detallan las principales técnicas de *encoding*, que ayudan a pasar de datos categóricos a datos numéricos [21].

Ordinal Encoding

Consiste en asignar un valor numérico a cada posible valor de una variable de entrada. Por ejemplo, si se dispone de la entrada 'color' y los posibles colores pueden ser 'blanco' o 'negro', se le asigna el valor 0 a 'blanco y 1 a 'negro'.

Entre las ventajas que ofrece Ordinal Encoding es que es sencillo codificar una variable y hacer el proceso inverso a la salida. El principal problema es que se puede estar imponiendo una relación de orden cuando esta no tiene por qué existir.

One-Hot Encoding

One-Hot Encoding soluciona el principal problema de Ordinal Encoding. La codificación One-Hot crea una nueva variable por cada posible valor de entrada. Asignando a esta nueva variable el valor 0 o 1.

Como en el ejemplo anterior, se dispone de una variable de entrada ‘color’ con posibles valores ‘blanco’ o ‘negro’. Al aplicar One-Hot se elimina la variable original ‘color’ y se crean dos nuevas variables ‘blanco’ y ‘negro’. El valor de estas nuevas variables será 0 o 1 en función del valor que tuviese originalmente la variable ‘color’.

3.4.3.3 Escalado sobre modelo 1 y modelo 2

	Ventajas	**Desventajas**
Normalización	● Acelera la velocidad a la hora de encontrar una solución óptima. ● Mejora la precisión del modelo. ● Útil cuando no conocemos qué tipo de distribución siguen los datos.	● Puede amplificar el ruido. ● Afectada por outliers.
Estandarización	● No se ve afectada por los outliers porque no define un rango para los datos. ● Asegura tener una media de 0 y desviación típica de 1. ● Útil cuando los datos siguen una distribución Gaussiana o Normal.	● Puede hacer que pasemos a tener valores negativos a la salida.

Tabla 1: Comparación entre técnicas de escalado de datos. (Fuente: elaboración propia a partir de [22])

En base a la tabla 1 se ha decidido llevar a cabo el siguiente escalado de datos para los modelos presentados:

- **Escalado para modelo 1 - Entrenamiento individual para cada empresa**: en este caso la desviación típica para el precio “Close” de la compañía oscila entre los 3 $ y 8 $ en la mayoría de compañías. Estamos ante una desviación típica baja, por lo que se opta por aplicar:
    - Normalización con Min-Max Scaler que dejará los valores entre 0 y 1.

- **Escalado para modelo 2 - Entrenamiento general con todas las empresas**: para este caso tenemos una desviación típica muy alta con un valor de 340,09 $. Es normal pues estamos introduciendo datos de muchas empresas de las cuales sus precios son dispares. En este caso se opta por:

- Estandarización mediante Standard Scaler, esto dejará los datos con una media de 0 y desviación típica de 1.

3.5 Partición de datos para entrenamiento y test

Disponemos de datos de los últimos 5 años. Por cada empresa tenemos aproximadamente 1250 precios de cierre para la acción.

Se procede a dividir el conjunto de datos. Tendremos dos conjuntos de datos para trabajar:

- **train_data**: usado para entrenar la red. Supone el 90% del total de datos disponibles.
- **test_data:** se utiliza para validar el modelo. Es el 10% de datos más recientes y suponen aproximadamente los 4 últimos meses de datos disponibles.

La decisión de seleccionar una partición 90-10 se ha realizado teniendo en cuenta que se necesitan tener suficientes casos de prueba tanto para entrenar como para validar el modelo. Se reserva el 10% de los datos para test y no menos debido a que con menos cantidad y un look_back de 100 apenas se generarían casos de test para el modelo. Con este tamaño de test_data y de look_back tenemos 1032 casos de entrenamiento y 25 casos de test.

Por cada conjunto de datos se crea una matriz en la que se especifique al modelo cuáles son los valores en los que tiene que fijarse a la hora de hacer una predicción y cuál es el valor que debería tener esa predicción. Esto se realiza mediante el método *create_dataset.* La variable *look_back* define cuántos días usará el modelo para predecir el día siguiente (por defecto se define con un valor de 100 pero se ajustará más adelante.).

Los datasets creados serán los que le pasaremos a la red para que entrene. A continuación se describen:

- **X_train**: se utilizará para entrenar la red neuronal.
- **y_train**: salida esperada para X_train. Se utilizará para entrenar la red neuronal. Esta matriz es el precio que debe predecir cuando entrene.
- **X_test**: supone la entrada a la red para validar su comportamiento.
- **y_tes**t: Esta matriz contiene el precio que debe predecir la red para la entrada X_test. Solo se usa para validar el modelo, no para entrenarlo.

A continuación se muestra como quedaría la matriz X_train e y_train en un caso de ejemplo.

train_data									
Día 1	Día 2	Día 3	Día 4	Día 5	Día 6	Día 7	Día 8	Día 9	Día 10
90	91	92	93	94	95	96	97	98	99

Tabla 2: Ejemplo de datos tomados para realizar un entrenamiento. (Fuente: elaboración propia)

Se toman los datos de los 3 últimos días para hacer una predicción (*look_back* = 3)

X_train			y_train
90	91	92	93
91	92	93	94
92	93	94	95
93	94	95	96
94	95	96	97
95	96	97	98
96	97	98	99

Tabla 3: conjunto de datos para entrenamiento tras realizar el *split. (*Fuente: elaboración propia)

Entiéndase que los ejemplos mostrados en la tabla 2 y tabla 3 son una simplificación del problema, pero en el modelo presentado, el funcionamiento es el mismo con la salvedad de que train_data tiene un tamaño mayor y la variable look_back toma también un valor mayor.

3.6 Arquitectura de la red

Cuando tratamos con arquitecturas de Deep Learning como es el caso de las ANN encontramos principalmente dos problemas [23]:

- **Overfitting**: la red aprende cada característica de los datos de entrenamiento y la precisión en entrenamiento es alta. Pero no generaliza bien y en test la precisión es baja.
- **Underfitting**: el modelo presenta un mal rendimiento, puede deberse al algoritmo de Machine Learning utilizado.

Se hace necesario encontrar un modelo equilibrado que no sufra de los problemas mencionados. Generalmente, el problema más recurrente es el overfitting y una de las posibles soluciones es utilizar Dropout.

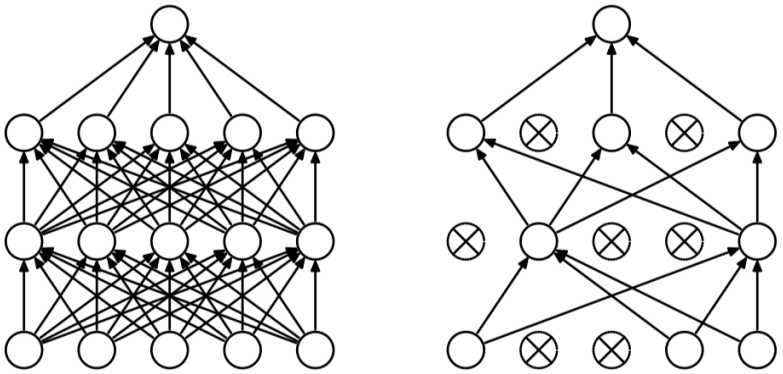


Figura 3.6: Ejemplo de ANN sin Dropout y con Dropout. (Fuente: [24])

Dropout es una técnica que desactiva algunas neuronas durante el entrenamiento (como se puede ver en la figura 3.6). En cada iteración se van desactivando un conjunto de neuronas, esto hace que no haya neuronas fuertes y neuronas débiles. De esta manera se reduce la posibilidad de que se produzca overfitting

Otra técnica utilizada en ambos modelos para intentar minimizar el overfitting es Early Stop. Early Stop funciona monitoreando el error cometido con los datos usados para validación (en el modelo construido se usan los datos de test también para validación en entrenamiento). Esta monitorización permite observar a partir de un cierto número de epoch definidos si el error en validación ha disminuido con cada nuevo epoch. Si no se da esa disminución permite detener el entrenamiento porque estamos ante el caso de que la red puede estar disminuyendo su error con los datos de entrenamiento pero no con los de test. En este caso el modelo no estaría generalizando e interesa detener el entrenamiento en ese momento para evitar que el modelo sufra de overfitting.

3.6.1 Optimización de arquitectura para modelo 1

Modelo 1 - Entrenamiento individual para cada empresa.

Para realizar pruebas tomamos los datos de 'AMZN' y vemos el RMSE obtenido en test. Un primer parámetro a determinar es el *look_back*, este parámetro no forma parte propiamente de la arquitectura de la red pero conviene fijarlo para poder comparar resultados.

Se llevan a cabo una serie de entrenamientos de la red para determinar un valor para la variable *look_back* teniendo en cuenta los siguientes parámetros:

- *look_back*: cuantos datos anteriores mira la red para realizar una predicción
- Número de capas LSTM: supone el número de capas apiladas (stacked) de tipo LSTM. La red además
- Número de neuronas por capa: determina qué cantidad de neuronas tiene cada capa LSTM

look_back	Nº de capas LSTM	Nº de neuronas por capa	Error entrenamiento	Error test
10	1	50	3269	3268
10	2	50	2277	3280
10	3	50	2274	3275
50	1	50	2303	3157
50	2	50	2293	3145
50	3	50	2286	3139
100	1	50	2356	3171
100	2	50	2356	3162
100	3	50	2350	3135

Tabla 4: Error del modelo 1 con diferentes arquitecturas. (Fuente: elaboración propia)

En base a la tabla 4 podemos observar que con un look_back de 50 y 100 el resultado en test es prácticamente el mismo. Por ello, en adelante, el valor para la variable look_back será 50. También en base a los datos de la tabla 4 se opta por una arquitectura de 2 capas LSTM apiladas, se considera que ofrece el mejor equilibrio entre complejidad y rendimiento para un *look_back* de 50.

El resto de hiperparametros de la red serán optimizados usando la librería Tuner de Keras. Los hiperparametros que está librería tiene que optimizar son:

- **hp_units:** supone el número de neuronas en cada capa
- **learning_rate:** es el factor de aprendizaje, importante para que la red converja hacia una solución rápidamente pero sin caer en mínimos locales.

```
INFO:tensorflow:Oracle triggered exit
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
LSTM1 (LSTM)                 (None, 50, 90)            33120
_________________________________________________________________
LSTM2 (LSTM)                 (None, 30)                14520
_________________________________________________________________
dropout_3 (Dropout)          (None, 30)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 31
=================================================================
Total params: 47,671
Trainable params: 47,671
Non-trainable params: 0
```

Figura 3.7. Arquitectura de la ANN para el modelo 1 con la compañía XRAY (Dentsply Sirona Inc). (Fuente: elaboración propia)

Se puede observar en el resumen del modelo (figura 3.7) que el número de neuronas que se considera óptimo para la empresa Dentsply Sirona Inc es de 90 neuronas para la primera capa LSTM y de 30 para la segunda capa.

El número de parámetros a optimizar durante el entrenamiento con esta empresa es de 47671. Esto es solo para la empresa Dentsply Sirona Inc, por cada empresa se llevará a cabo un entrenamiento con más o menos parámetros en función del tamaño de la ANN.

Es importante destacar que en este modelo cada empresa tiene una arquitectura propia, el número de capas se ha fijado en 2 capas LSTM apiladas, pero el número de neuronas será optimizado para cada empresa.

3.6.1 Optimización de arquitectura para modelo 2

Modelo 2 - Entrenamiento general con todas las empresas

El modelo 2 también utiliza un *look_back* de 50. Y al igual que el modelo 1, el número de neuronas por capa y el learning rate son optimizados con keras Tuner.

De esta manera, queda por definir para este modelo cuál es el número óptimo de capas LSTM apiladas. Se llevan a cabo pruebas sobre el modelo y se evalúa el resultado en la siguiente tabla.

Nº capas LSTM	Error entrenamiento	Error test
1	52.37	57.55
3	52.48	57.52
5	52.19	57.41

Tabla 5: Error cometido por el modelo 2 en función del número de capas LSTM. (Fuente: elaboración propia)

Con la información obtenida de la tabla 5 se decide optar por la estructura más sencilla para resolver el problema, esto es una RNN con una sola capa LSTM. Esto permite que el tiempo de entrenamiento sea menor dado que el número de conexiones dentro de la red será menor.

Una vez se ha generado el hipermodelo con el número de neuronas ajustado para el modelo 2 se puede ver el resumen de la arquitectura en la siguiente imagen.

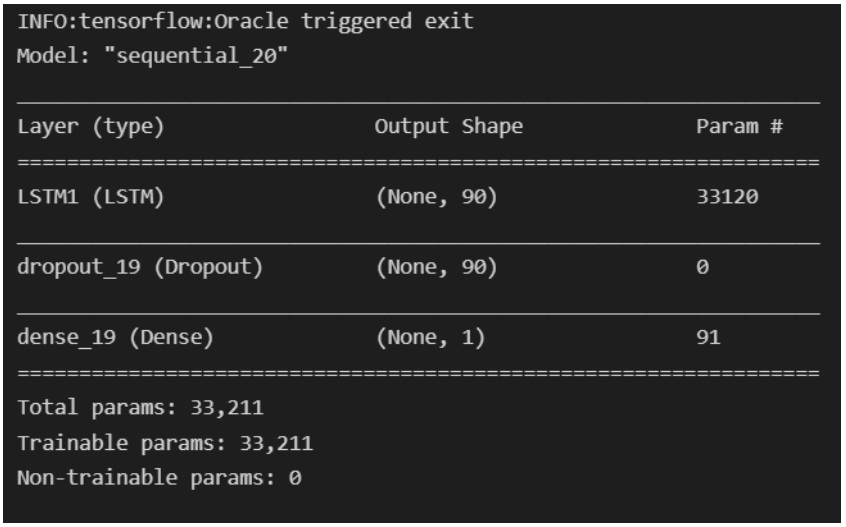


Figura 3.8. Arquitectura de la ANN para el modelo 2. (Fuente: elaboración propia)

En la figura 3.8 se puede apreciar que el total de parámetros a optimizar para el modelo 2 es 33211. A diferencia del modelo 1, en el modelo 2 está será la arquitectura única con la que se entrenará y realizarán las predicciones de todas las empresas. Cuando se genera el hipermodelo, se optimizan los hiperparámetros y se puede observar que se decide utilizar 90 neuronas para la capa LSTM.

3.7 Entrenamiento.

Cada uno de los modelos tiene sus pasos específicos, no obstante, a continuación se procede a detallar la estructura genérica del proceso de entrenamiento y predicción para ambos modelos:

1. Preprocesamiento de datos
2. Partición de datos para entrenamiento y test.
3. Generar hipermodelo
    a. Buscar mejores hiperparámetros para el hipermodelo
    b. Construir el hipermodelo
4. Entrenar hipermodelo
5. Realizar predicción
6. Calcular error entrenamiento y test

3.7.1 Entrenamiento para modelo 1

Este modelo realiza todos los pasos anteriormente indicados para cada compañía a analizar. Por cada compañía se crea una ANN y se optimiza específicamente para los datos de esa compañía.

Se realiza un entrenamiento por cada empresa definido con 50 epoch. Existe la posibilidad de que el entrenamiento se detenga antes gracias a la función implementada Early Stop pero esto no podrá suceder hasta pasados por lo menos 5 epoch.

Por cada empresa se guarda el error cometido tanto en entrenamiento como en test, posteriormente se evalúa el error medio cometido tanto en entrenamiento como en test para todas las empresas.

El tiempo de entrenamiento en el modelo 1 se establece en aproximadamente 10 minutos.

3.7.2 Entrenamiento para modelo 2

En el modelo en este caso sigue también las fases indicadas con la peculiaridad de que se crea un único dataset con la información de todas las compañías y se entrena una sola red neuronal con todos los datos.

El proceso seguido para crear el dataset de entrenamiento y test consiste en justo despues de hacer la partición de datos o split (paso 2) se concatena con numpy todos los valores de entrenamiento (X_train e y_train) y test (X_test e y_test). Con esto se obtiene un conjunto de datos para entrenar de 95849 filas y un conjunto de datos de test de 6626.

Al igual que en el modelo 1 se fija un número de epoch de 50 que podrá ser abortado por Early Stop si el error en los datos de validación no mejora durante el entrenamiento. No obstante, se fija un valor de *paciencia* para Early Stop de 10, esto quiere decir que, el entrenamiento no se podrá detener hasta pasados al menos 10 epoch.

```
Epoch 1/50
1052/1052 [==============================] - 11s 9ms/step - loss: 0.0226 - mse: 0.0226 - val_loss: 0.0127 - val_mse: 0.0127
Epoch 2/50
1052/1052 [==============================] - 9s 8ms/step - loss: 0.0180 - mse: 0.0180 - val_loss: 0.0107 - val_mse: 0.0107
Epoch 3/50
1052/1052 [==============================] - 8s 8ms/step - loss: 0.0178 - mse: 0.0178 - val_loss: 0.0108 - val_mse: 0.0108
Epoch 4/50
1052/1052 [==============================] - 9s 8ms/step - loss: 0.0177 - mse: 0.0177 - val_loss: 0.0172 - val_mse: 0.0172
...
Epoch 11/50
1052/1052 [==============================] - 8s 8ms/step - loss: 0.0172 - mse: 0.0172 - val_loss: 0.0115 - val_mse: 0.0115
Error entrenamiento: 52.4802544308028
Error test: 57.75397081803122
```

Figura 3.9: Evolución del entrenamiento para el modelo 2. (Fuente: elaboración propia)

El tiempo de entrenamiento que requiere el modelo 2 es de aproximadamente 2 minutos. Habiéndose detenido el entrenamiento en el epoch número 11. Esta gran diferencia de tiempo frente al modelo 1 se debe a que solo debe entrenar una red neuronal y no una red distinta para cada empresa.

3.8 Predicción.

3.8.1 Predicción para el modelo 1

Como ya se explicó en el análisis exploratorio de los datos, algunas compañías fueron descartadas por no disponer de suficientes datos. Es por ello que el modelo 1 entrena y realiza sus predicciones sobre 89 empresas del Nasdaq.

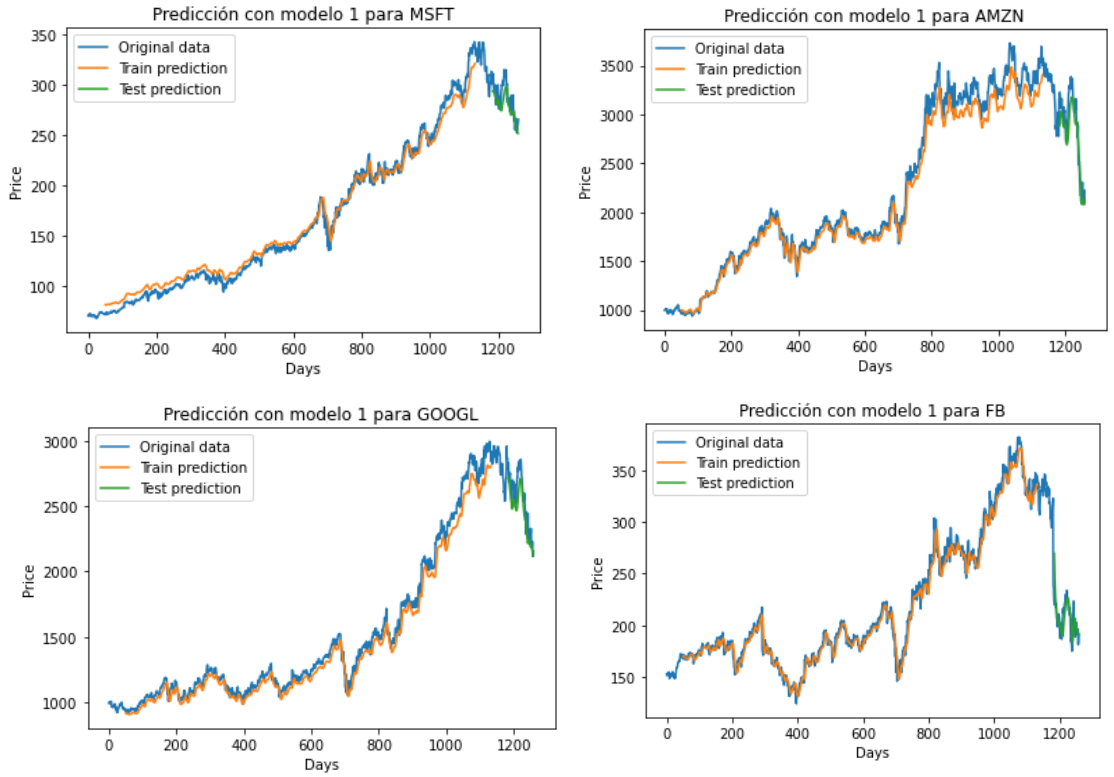


Figura 3.10: Predicción del modelo 1 para datos de entrenamiento y test de distintas empresas (Fuente: elaboración propia)

La figura 3.10 muestra la predicción para el día siguiente utilizando datos de los últimos 50 días de algunas de las empresas más importantes del Nasdaq.

El modelo 1 hace una predicción igual que la anterior para cada una de las 89 empresas. Esta predicción es guardada en un diccionario de python cuya clave es el nombre de la empresa y el valor es una tupla que contiene el error en entrenamiento y el error en test.

Posteriormente se procede al cálculo de la media aritmética del error cometido por cada empresa. Se muestra la media tanto del error de entrenamiento como de test. Obteniendo como resultado:

- Media RMSE entrenamiento = 184.98
- Media RMSE test = 224.92

En este caso, como disponemos de una predicción de test y entrenamiento podemos crear una estructura de datos con pandas que nos permita explorar ese error de manera individual para cada empresa.

Es importante indicar que el error en test puede variar ligeramente con el que se especifica anteriormente dado que el modelo se ha entrenado en varias ocasiones y los modelos de Machine Learning siempre cuentan con un cierto grado de aleatoriedad en cada entrenamiento.

	0
count	89.000000
mean	184.985078
std	306.154128
min	5.604452
25%	56.434111
50%	97.867089
75%	201.183631
max	2326.852298

Figura 3.11: Información sobre los errores de predicción con datos de test para las 89 empresas analizadas con el modelo 1 (Fuente: elaboración propia)

En la figura 3.11 se puede observar que la media de error en test es 184.98 y la desviación típica es 306.15. Una desviación típica tan alta indica que los errores cometidos son extremadamente dispersos, lo cual es lógico dado que las empresas tienen valores de cotización muy dispares y por tanto el error absoluto cometido entre ellas será muy distinto.

En la misma figura 3.11 vemos que el mínimo error cometido (medido como RMSE) es 5.6 y el máximo 2326.85. Este error no tiene porque indicar que una predicción es mejor que otra necesariamente, se debe tener en cuenta que hay empresas cuyo valor por acción supera los 2000$ como Alphabet Inc (GOOGL) y otras que no llegan a los 50$ como Comcast Corporation (CMCSA). Por tanto, es evidente que una tasa de error que suponga el mismo error en % respecto al precio de la acción será, en términos absolutos, un error mayor para GOOGL que para CMCSA.

De hecho, así sucede en este caso concreto ya que el error cometido en test para CMCSA es de 44.73 y el cometido para GOOGL es de 2589.78, lo cual representa en ambos casos un RMSE que, en términos relativos, se aproxima al 100% del valor de la acción pero en términos absolutos un error es aproximadamente 57 veces mayor que el otro.

3.8.2 Predicción para el modelo 2

Para hacer una predicción con el modelo 2 previamente se ha llevado a cabo un entrenamiento. Seguido de ello el modelo está listo para hacer predicciones.

Los resultados obtenidos medidos gracias al error cuadrático medio son:

- Media RMSE entrenamiento = 52.02

- Media RMSE test = 56.78

Los resultados obtenidos siguen lo esperado en un modelo de Machine Learning, esto es que el error en test sea superior al error en entrenamiento. Esto es así debido a que los datos de test son nuevos para el modelo y las predicciones realizadas sobre datos desconocidos siempre tendrán un error mayor que las realizadas con datos que se han visto varias veces durante el entrenamiento.

Por último se muestran algunas predicciones de los 6626 casos de test que ayuden a tener una visión de la precisión del modelo:

Nº caso de test	**Valor real**	**Valor predicción**
1	45.59	45.42
1000	56.02	49.47
2000	49.57	49.53
3000	50.59	50.18
4000	61.04	60.58

Tabla 6: Resultados del modelo 2 al realizar predicciones sobre datos de test (Fuente: elaboración propia)

Se puede observar que en general el error cometido por el modelo 2 es bajo, equivocándose en el precio de la acción en la mayoría de los casos en menos de 1 $.

4. Análisis de resultados

En base a los resultados obtenidos por los dos modelos propuestos es posible afirmar que los algoritmos de Deep Learning basados en redes LSTM pueden llevar a cabo predicciones sobre series temporales con un alto grado de precisión.

Las predicciones con mejor resultado son las obtenidas por el modelo 2. Unido a esto, el modelo 2 tiene una velocidad de entrenamiento mucho mayor lo que hace que sea un modelo superior en ambos aspectos.

Esta mayor precisión del modelo 2 se debe principalmente a que dispone de muchos más datos de entrada a la red y esto permite que el modelo generalize mejor. La mayor velocidad de entreno se debe a que en el modelo 2 se dispone de una única ANN sobre la que entrenar y ajustar los pesos, sin embargo, el modelo 1 tiene que crear múltiples ANN (una por cada empresa) y eso ralentiza mucho el entrenamiento.

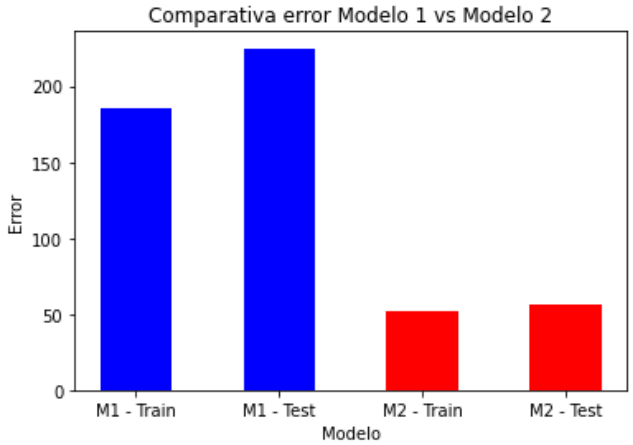


Figura 4.1: Comparativa error entre modelos (Fuente: elaboración propia)

De la figura 4.1 se puede apreciar que el error en test del modelo 2 es aproximadamente la quinta parte que el del modelo 1. Con esto podemos concluir que para el caso que se trata en este trabajo supone una mejor propuesta la creación de un modelo conjunto con datos de todas las empresas y entrenar una única ANN sobre ellos (modelo 2).

A continuación se analizan los resultados obtenidos con cada uno de los modelos sobre algunas de las empresas más importantes del Nasdaq.

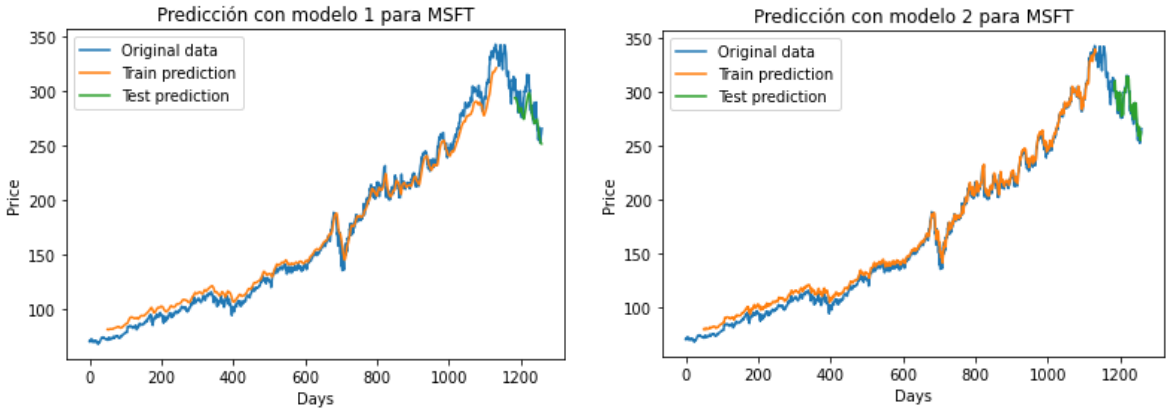


Figura 4.2: Comparación de error entre modelo 1 y modelos 2 para MSFT (Fuente: elaboración propia)

En la figura 4.2 se puede apreciar la diferencia de error entre los dos modelos presentados para la empresa Microsoft. El modelo 1 tiene un error de entrenamiento y test de, respectivamente, 174.84 y 281.62. Mientras que para el modelo 2 los errores son 179.04 en entrenamiento y 287.53. En este caso el modelo 1 ofrece un mejor resultado que el modelo 2, no obstante, este no es el comportamiento que más se repite ya que el modelo 2 en media ofrece un resultado superior al modelo 1.

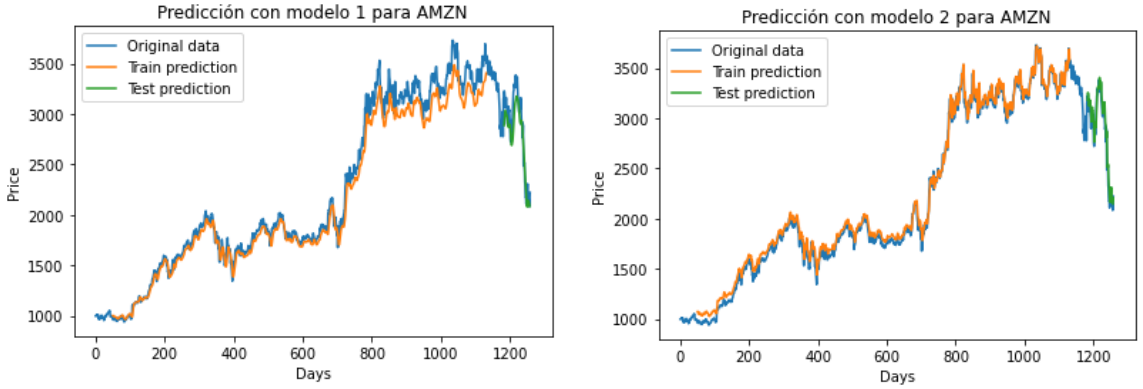


Figura 4.3: Comparación de error entre modelo 1 y modelos 2 para AMZN (Fuente: elaboración propia)

La figura 4.3 muestra la diferencia de comportamiento entre los modelos 1 y 2 al realizar predicciones sobre la empresa Amazon. El modelo 1 tiene un error de entrenamiento (medido con RMSE) de 2256.78 y en test el error es 2994.24. Para el modelo 2 los respectivos errores de entrenamiento y test son 2388.31 y 2742.48. Vemos en este caso como el error en test del modelo 2 es menor, lo que nos indica que a la hora de poner en funcionamiento alguno de los modelos los resultados que ofrecería el modelo 2 son mejores.

De las gráficas de la figura 4.3 se puede observar como el modelo 1 tiende a infravalorar los precios a la hora de predecir y el modelo 2 tiende a hacer lo contrario. Esto se ve claramente entre los días 800 y 1000 que corresponden a la fase de entrenamiento.

5. Conclusiones

En este capítulo se procede a analizar el impacto que ha tenido para el autor la realización del proyecto, así como el posible impacto de los resultados obtenidos. Además se analizarán las fortalezas y debilidades de los modelos propuestos.

Este proyecto ha servido al autor para profundizar en el área de la inteligencia artificial, un campo, que desde el inicio de sus estudios siempre le resultó apasionante. También le ha servido para entender mejor el campo de la economía y concretamente el sector financiero, habiendo adquirido los conocimientos necesarios para abordar el problema de la predicción en bolsa.

Además, todo lo aprendido en el ámbito de la IA y lo mucho que se puede llegar a profundizar en este campo ha animado al autor a continuar sus estudios en esta rama de las Tecnologías de la Información.

La principal fortaleza de los modelos presentados (especialmente el segundo modelo) es mostrar al autor que llevando a cabo un análisis técnico sobre la secuencia temporal de precios de distintas empresas se puede conseguir una tasa de error relativamente baja. Esto es algo que al inicio del proyecto el autor no tenía claro que se pudiera llegar a conseguir y supone un éxito ver hasta qué punto las tecnologías de Deep Learning actuales permiten abordar problemas reales, en concreto, predicción bursátil. No obstante, las posibilidades de aplicación de este tipo de tecnologías van más allá del sector financiero.

En lo que respecta a las debilidades del modelo se debe tener en cuenta que en el ámbito bursátil hay muchos factores que pueden afectar al precio al que cotiza una acción. Un ejemplo de lo anterior sería la inflación en EEUU, medida mediante el IPC, que en abril de 2022 se sitúa en el 8,3% [25]. Está inflación ha forzado a la Reserva Federal de Estados Unidos a llevar a cabo la mayor subida de tipos de los últimos 22 años, con una subida del tipo de interés en el mes de mayo de 2022 de 0,5 puntos [26]. Esta subida de tipos supone una restricción de dinero y crédito en la economía, además, al aumentar el coste de financiación a través de los bancos, las instituciones tienden a emitir bonos con mayor rentabilidad, lo que vuelve más atractiva la renta fija frente a la renta variable [27]. Todo ello empuja los precios en el mercado bursátil a la baja.

El ejemplo anterior muestra lo complicado que resulta determinar precios a futuro en un sistema económico tan complejo como el actual. Esto supone una debilidad en el modelo ya que este factor determinante para los precios de las acciones no está contemplado, si bien, esta información macroeconómica podría ser incorporada en un futuro.

Por otro lado, otro punto débil del modelo, sería que se basa únicamente en un análisis técnico y deja fuera aspectos tan importantes para determinar el precio de una acción como son los balances, cuentas de resultados, fondos de comercio, etc.

Entre las posibles aplicaciones de este trabajo (especialmente usando el modelo 2) estaría identificar aquellas compañías sobre las que se tiene un error menor en test y automatizar la recolección de datos y el entrenamiento de manera que se ejecute a diario y notifique el precio para el día siguiente. Esto permitiría tomar mejores decisiones de inversión a un día, siempre teniendo en cuenta que son predicciones basadas en un análisis técnico y teniendo en cuenta las limitaciones del modelo.

6. Aspectos sociales, ambientales, éticos y legales

En este capítulo se procede a la identificación, descripción y análisis de algunos de los principales impactos sociales, ambientales, éticos y legales.

En primer lugar se pretende dar respuesta a la pregunta de por qué se lleva a cabo este proyecto y qué problema pretende resolver. Para responder a ello es necesario explicar primero lo que representa el precio de cotización bursátil de una empresa. El precio representa un punto de equilibrio entre la oferta y la demanda de un determinado bien, en este caso, las acciones de una compañía. La cotización es, por tanto, un precio pero no necesariamente este precio se corresponde con el valor de la empresa.

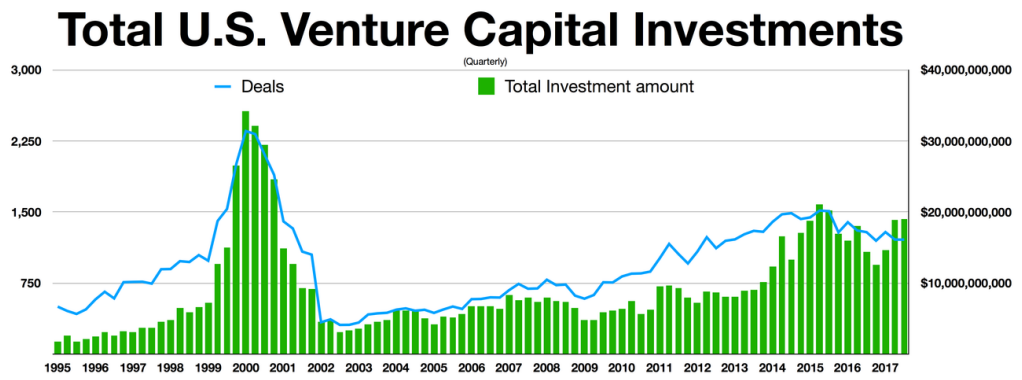


Figura 6.1: Evolución de las inversiones trimestrales de capital de riesgo en Estados Unidos, 1995–2017 (Fuente: [28])

La figura 6.1 muestra la burbuja ocurrida entre 1997 y 2001 conocida como burbuja de las puntocom. Este ejemplo es uno de los casos en los que se puede ver que el precio no representa correctamente el valor de las empresas. En este caso la gráfica muestra la inversión, si bien, a mayor inversión mayor demanda y por tanto mayores precios. En el año 2000 y 2001, los precios estaban inflados respecto al valor de las compañías y se produjo un ajuste de precios a la baja. Este fenómeno es fácil de detectar cuando uno dispone de toda la información y puede mirar al pasado con ella pero no es sencillo identificarlo cuando está ocurriendo en el presente.

Por todo lo anterior, uno de los aspectos sociales positivos que puede tener el modelo presentado es conseguir una rápida adaptación de las cotizaciones de las empresas al valor que estas tienen en un momento determinado. Esto evita burbujas y todos los efectos negativos que estas espirales de precios crecientes generan. Se podría pensar que en este proceso de ajuste a la baja de precios los accionistas de las empresas cuya cotización cae son los grandes perjudicados. Ante esta situación podemos recurrir a las palabras del economista J. Schumpeter para explicar por qué este proceso es necesario y hasta positivo para el conjunto de la economía:

“El proceso de mutación industrial … revoluciona incesantemente la estructura económica desde dentro, destruyendo constantemente las estructuras antiguas, creando constantemente nuevas… Este proceso de destrucción creativa es la esencia del capitalismo” - Joseph Schumpeter

El modelo presentado puede identificar acciones que están por encima de su valor pero también podría identificar el caso contrario: acciones que están por debajo de su valor real. Esto supone una oportunidad para los inversores y si invierten comprando por debajo de su valor real podrán obtener beneficios económicos. Otro agente que se beneficiaría en este caso son las empresas que reciben inversión, ya que esta entrada de capital puede permitirles crecer y generar más valor para sus clientes, crear empleo, rentabilidad para los accionistas, etc.

Respecto al aspecto legal de hacer predicciones de inversión, estas podrían ser interpretadas como recomendaciones de inversión. En último término, lo que indican los modelos presentados son acciones infravaloradas o sobrevaloradas y esto podría usarse como asesoramiento financiero.

En España el organismo encargado de la supervisión e inspección de los mercados de valores es la Comisión Nacional del Mercado de Valores (CNMV). La CNMV ejerce una supervisión prudencial, que garantiza la seguridad de sus transacciones y la solvencia del sistema [29]. La figura jurídica que, en España, permite dar un asesoramiento financiero son las EAFI (Empresa de Asesoramiento Financiero), avaladas por la CNMV. Las EAFI son sociedades, generalmente se acogen a ella agentes de bolsa, de banca privada, etc. Estas sociedades sí podrían, bajo regulación, dar recomendaciones de inversión o asesoramiento financiero.

Debido a lo anterior, los modelos presentados no deben considerarse bajo ningún concepto una recomendación de inversión sino una herramienta con fines académicos y cuya principal motivación es el aprendizaje.

7. Propuestas de futuro

A continuación se detallan algunas propuestas que se podrían implementar en los modelos propuestos.

El autor considera que una primera mejora sería buscar otro proveedor de datos que ofrezca más datos. La actual plataforma (Tiingo) ofrece de manera gratuita datos de los últimos 5 años aproximadamente y no provee los datos más recientes (los últimos días) con lo que no se podría poner el sistema a realizar predicciones en un entorno real a un día.

Aumentar el número de datos disponibles para un modelo de Deep Learning generalmente hace que el modelo tenga una mayor precisión [30]. No obstante, sería necesario repetir las fases de análisis exploratorio de los datos, preprocesamiento, escalado, etc.

Otra mejora que no ha sido implementada por falta de tiempo y que sí que sería factible con la cantidad de datos actual para el modelo 2 sería crear tres conjuntos de datos: entrenamiento, validación y test. De esta manera el modelo utilizará los datos de entrenamiento para entrenar, los datos de validación para realizar Early Stop durante el entreno y, por último, los datos de test para comprobar el rendimiento del modelo. Actualmente el modelo valida el entrenamiento con los datos de test y hace que se introduzca un cierto sesgo ya que el modelo antes de ponerse a prueba ha decidido parar el entreno cuando no mejora sus predicciones sobre los datos de test.

Por supuesto, la mejora más importante sería introducir análisis fundamental al modelo y datos macroeconómicos. Esto haría que el modelo tuviese mucha más información relevante para determinar el precio y seguro mejoraría su rendimiento.

Otra mejora posible y sencilla sería tomar más datos de los ya disponibles actualmente como el volumen, precio mínimo diario, precio máximo diario, etc. y ver si el modelo es capaz de mejorar sus predicciones con esa nueva información.

Por último, algunos datos que podrían ser introducidos al modelo y ser de utilidad sin pertenecer al ámbito económico serían búsquedas en Google (usando Google Trends) o publicación de hashtags en Twitter, etc. Esta información puede ayudar al modelo a detectar cuando se produce un movimiento en alguna empresa y determinar si afecta a su precio de cotización.

8. Entorno para despliegue

En este apartado se describe brevemente el entorno necesario para desplegar los modelos presentados.

Para este proyecto se ha utilizado el gestor de entornos y paquetes Conda. Esto permite establecer un entorno con una versión específica de python independientemente de la versión que se tenga instalada a nivel de sistema operativo. Con esto se consigue un entorno estable para cada proyecto con las versiones necesarias de python y las librerías específicas para dicho proyecto.

El entorno desplegado dispone de Python en su versión 3.9.11.

Las librerías necesarias para ejecutar el proyecto son:

- Tensorflow - 2.6.0: libreria con múltiples utilidades para Machine Learning
- Keras (disponible en tensorflow.keras): keras permite crear de manera rápida y sencilla modelos de Deep Learning mediante capas.
- Pandas: utilizado para el tratamiento de datos
- Pandas_datareader: para recolección de datos a través de la API Tiingo.
- Config: librería de python que utilizada para almacenar datos que no se quieren compartir (como la API_key personal)
- Time: librería de python usada para espaciar temporalmente la recolección de datos.
- os: librería de python, usada para leer directorios/ficheros.
- Matplotlib: librería utilizada para representación gráfica de datos
- Numpy: permite realizar fácilmente operaciones con matrices, calcular medias, concatenar arrays, etc.
- Sklearn / scikit-learn: librería con múltiples utilidades para Machine Learning. Usada para el escalado de datos, métricas como RMSE
- math: librería de python, es usada para operar matemáticamente, en este proyecto permite calcular raices cuadradas para el calculo del RMSE.
- keras_tuner: framework que permite la optimización de hiperparámetros en RNN.

9. Referencias

[1] Arias, A. S. (2021, March 9). Índices bursátiles del mundo. Economipedia. Retrieved May 14, 2022, from https://economipedia.com/ranking/indices-bursatiles-del-mundo.html

[2] Mokhtari, S., Yen, K. K., & Liu, J. (2021). Effectiveness of artificial intelligence in stock market prediction based on machine learning. *International Journal of Computer Applications*, *183*(7), 1–8. https://doi.org/10.5120/ijca2021921347

[3] Calixto, M. (2021, February 15). *Análisis técnico o fundamental, ¿Qué conviene más?* EL CEO. Retrieved May 11, 2022, from https://elceo.com/mercados/analisis-tecnico-o-fundamental-que-conviene-mas/#:~:text=El%20an%C3%A1lisis%20t%C3%A9cnico%20estudia%20el,de%20negocio%20y%20registros%20financieros

[4] July, G. T. · 29. (2020, November 15). *NASDAQ 100 - what is it and how to invest in this great index*. ToFinancialFreedom. Retrieved May 30, 2022, from https://tofinancialfreedom.co/en/nasdaq-100-index/

[5] *The evolution of forecasting*. ToolsGroup. (2021, June 17). Retrieved May 29, 2022, from https://www.toolsgroup.com/blog/supply-chain-innovation-the-evolution-of-forecasting/

[6] Piotroski, J. D. (2000). Value Investing: The Use of Historical Financial Statement Information to Separate Winners from Losers. Journal of Accounting Research, 38, 1–41. https://doi.org/10.2307/2672906

[7] Alvaro, S. el, Hola, & *, N. (2018, December 25). Problemas comunes en Aprendizaje Automático. MachineLearningParaTodos.com. Retrieved May 14, 2022, from https://machinelearningparatodos.com/problemas-comunes-en-aprendizaje-automatico/

[8] España, R. (2020, October 29). *Qué son Regresión y clasificación en machine learning*. Agencia B12. Retrieved May 14, 2022, from https://agenciab12.com/noticia/que-son-regresion-clasificacion-machine-learning

[9] Charte, F. (n.d.). *Qué tipos de problemas podemos resolver con técnicas de aprendizaje automático (machine learning)*. campusMVP.es. Retrieved May 14, 2022, from https://www.campusmvp.es/recursos/post/que-tipos-de-problemas-podemos-resolver-con-tecnicas-de-aprendizaje-automatico-machine-learning.aspx

[10] By: IBM Cloud Education. (n.d.). *What is deep learning?* IBM. Retrieved May 14, 2022, from https://www.ibm.com/cloud/learn/deep-learning#:~:text=Deep%20learning%20is%20a%20subset,from%20large%20amounts%20of%20data.

[11] *The difference between AI, Machine Learning and deep learning*. tipsmake.com. (n.d.). Retrieved May 30, 2022, from https://tipsmake.com/the-difference-between-ai-machine-learning-and-deep-learning

[12] *Home*. ASSOCIACIÓ CATALANA DE VEXIL·LOLOGIA. (n.d.). Retrieved May 19, 2022, from https://www.vexi.cat/tienda/tags/2132?ss=6_28_7_27_45&pp=recurrent%2Bneural%2Bnetwork%2Bimage&ii=4128396

[13] *Recurrent neural networks (RNN): Working: Steps: Advantages*. EDUCBA. (2022, April 11). Retrieved May 14, 2022, from https://www.educba.com/recurrent-neural-networks-rnn/

[14] *Understanding LSTM networks*. Understanding LSTM Networks -- colah's blog. (n.d.). Retrieved May 14, 2022, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[15] Pronóstico de Series Temporales con Redes Neuronales en Python. (2019, February 26). Aprende Machine Learning. Retrieved May 29, 2022, from https://www.aprendemachinelearning.com/pronostico-de-series-temporales-con-redes-neuronales-en-python/

[16] Parra, F. (n.d.). *Estadística y machine learning con r*. 8 Series Temporales. Retrieved May 29, 2022, from https://bookdown.org/content/2274/series-temporales.html

[17] Jj. (2016, March 23). *Mae and RMSE - which metric is better?* Medium. Retrieved May 14, 2022, from https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

[18] Reducción de la dimensionalidad (o por qué más datos no siempre es mejor). Aukera. (2018, May 22). Retrieved May 18, 2022, from https://aukera.es/blog/reduccion-dimensionalidad/

[19] *Tipos de Datos en Aprendizaje Automático*. sitiobigdata.com. (2021, June 15). Retrieved May 18, 2022, from https://sitiobigdata.com/2019/12/24/tipos-de-datos-de-aprendizaje-automatico-con-ejemplos/

[20] Londoño, N. V. R., López, J. F., Martinez, E., Henry, Luz, Morales, F. C., Kim, Eden, M., J. U., Rodó, P., Carlos, Evelyn, Changoleon, Westreicher, G., José, J., & danielgarcia.r94. (n.d.). *Distribución Normal*. Economipedia. Retrieved May 22, 2022, from https://economipedia.com/definiciones/distribucion-normal.html

[21] Brownlee, J. (2020, August 17). *Ordinal and one-hot encodings for Categorical Data*. Machine Learning Mastery. Retrieved May 19, 2022, from https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/

[22] *Normalization vs standardization*. GeeksforGeeks. (2021, November 12). Retrieved May 18, 2022, from https://www.geeksforgeeks.org/normalization-vs-standardization/

[23] Brownlee, J. (2019, August 12). *Overfitting and underfitting with machine learning algorithms*. Machine Learning Mastery. Retrieved May 19, 2022, from https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

[24] Ampadu, H. (2021, May 10). *Dropout in deep learning*. AI Pool. Retrieved May 19, 2022, from https://ai-pool.com/a/s/dropout-in-deep-learning

[25] Siguenos en. (2022, May 11). *IPC DE USA 2022*. datosmacro.com. Retrieved May 22, 2022, from https://datosmacro.expansion.com/ipc-paises/usa

[26] Jiménez, M. (2022, May 4). *La Reserva Federal de ee UU Aprueba la mayor Subida de tipos de interés en 22 años para contener la inflación*. El País. Retrieved May 22, 2022, from https://elpais.com/economia/2022-05-04/la-reserva-federal-de-ee-uu-aprueba-la-mayor-subida-de-tipos-de-interes-en-22-anos-para-contener-la-inflacion.html

[27] Abanades, P. M. F. (2022, February 10). *Las consecuencias de una subida de tipos sobre tu bolsillo*. Economist & Jurist. Retrieved May 22, 2022, from https://www.economistjurist.es/economia/las-consecuencias-de-una-subida-de-tipos-sobre-tu-bolsillo/

[28] Wikimedia Foundation. (2022, March 27). *Burbuja Puntocom*. Wikipedia. Retrieved May 28, 2022, from https://es.wikipedia.org/wiki/Burbuja_puntocom

[29] *Funciones*. CNMV. (n.d.). Retrieved May 29, 2022, from https://www.cnmv.es/portal/quees/Funciones/Funciones.aspx

[30] Chawla, V. (2021, June 9). *Is more data always better for building analytics models?* Analytics India Magazine. Retrieved May 22, 2022, from https://analyticsindiamag.com/is-more-data-always-better-for-building-analytics-models/