Trabajo de Fin de Grado

Herramienta de inteligencia artificial para detectar patrones en bolsa

Artificial intelligence tool to detect patterns in the stock market

Gabriel García Jaubert

La Laguna, 7 de julio de 2022

D. **Cándido Caballero Gil**, con N.I.F. 42.201.070-A profesor Ayudante Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

D. **Moisés Díaz Cabrera**, con N.I.F. 78.516-774-H profesor Contratado Doctor adscrito al Departamento de Física Aplicada del Departamento de Física de la Universidad de Las Palmas de Gran Canaria, como cotutor.

C E R T I F I C A (N)

Que la presente memoria titulada:

”*Herramienta de inteligencia artificial para detectar patrones en bolsa*”

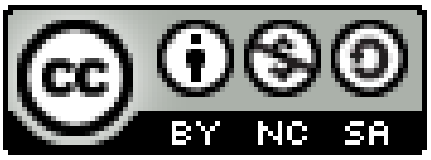
ha sido realizada bajo su dirección por D. **Gabriel García Jaubert**, con N.I.F. 42.240.610-Y.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de julio de 2022

Agradecimientos

Me gustaría comenzar agradeciendo a mi tutor Cándido Caballero Gil, junto a mi cotutor Moisés Díaz Cabrera, por su extraordinaria orientación y apoyo durante la realización de este trabajo. Ojalá una página fuera suficiente para agradecer a cada una de las personas que han hecho que este final de grado sea posible. Sin embargo, creo que mis amigos que me han acompañado durante la carrera merecen un lugar en esta página. No puedo asegurar que sin ellos llegar hasta aquí hubiera sido imposible, no obstante, sí puedo asegurar que sin ellos el camino no habría sido tan bonito.

Licencia



Resumen

En los últimos años, la inteligencia artificial ha experimentado un gran avance respecto a su funcionalidad. En la actualidad, la inteligencia artificial es capaz de procesar el lenguaje natural, producir textos de gran variedad de tipos e incluso detectar enfermedades. Por otro lado, la bolsa de valores es un mercado impredecible, en el que con pequeños detalles se puede intuir la dirección que tomará, pero nunca con certeza. Por consiguiente, aplicar la inteligencia artificial como herramienta de soporte en el momento de negociar puede ser de gran ayuda para tomar decisiones. Más concretamente, en este proyecto se pretende proveer al usuario con una herramienta que proporcione información precisa acerca de la probabilidad de que una acción alcance cierta dirección: alcista o bajista. Para este propósito se buscan patrones de bolsa dentro de un conjunto de empresas seleccionadas por el usuario para más tarde analizar sus valores alcanzados. De esta manera, es posible calcular la probabilidad que tiene un patrón de alcanzar cierta dirección. Para la búsqueda de patrones se emplea el algoritmo Dynamic Time Warping, que sirve para comparar series temporales entre sí, y detectar patrones de tipo relevante. Para evaluar la mejor configuración de los parámetros del sistema se realizó cross-validation, con un conjunto de datos formado por doscientas muestras. Estas muestras fueron escogidas por una persona, con la intención de aislar lo mejor posible patrones de tipo relevante. Los resultados obtenidos fueron la detección de los patrones de tipo doble techo y doble suelo, con un 76 % de exactitud. También se consiguió analizar la tendencia de los patrones y obtener la probabilidad de tomar cierta dirección. Por último, se añadió la opción de detectar patrones que se han terminado de formar en el momento de ejecución de la aplicación. Para facilitar el uso de la herramienta se creó una interfaz gráfica junto con un archivo ejecutable para Windows 10 y Linux. Esta aplicación puede ser una novedad respecto a los métodos existentes para la toma de decisiones efectuada por los inversores.

Palabras clave: bolsa de valores, *DTW*, mercados financieros, patrones chartistas, acciones.

Abstract

In recent years, artificial intelligence has made great strides in terms of functionality. Today, artificial intelligence is capable of processing natural language, producing texts of a wide variety of types and even detecting diseases. On the other hand, the stock market is an unpredictable market, where small details can give an intuitive sense of the direction it will take, but never with certainty. Therefore, applying artificial intelligence as a support tool when trading can be of great help in making decisions. More specifically, this project aims to provide the user with a tool that gives accurate information about the probability of a stock reaching a certain direction: bullish or bearish. For this purpose, stock market patterns are searched within a set of companies selected by the user in order to later analyse their values reached. In this way, it is possible to calculate the probability of a pattern reaching a certain direction. To search for patterns, the Dynamic Time Warping algorithm is applied, which has the purpose to compare time series with each other. This can be used to detect relevant patterns. To evaluate the configuration of the system parameters, cross-validation was carried out with a data set of two hundred samples. These samples were chosen by one person, with the intention of isolating relevant patterns as best as possible. The results obtained were the detection of double-top and double-bottom patterns, with 76 % accuracy. It was also possible to analyse the trend of the patterns and obtain the probability of taking a certain direction. Finally, the option of detecting patterns that have finished forming at the time the application was run was added. To facilitate the use of the tool, a graphical interface was created along with an executable file for Windows 10 and Linux. This application can be a novelty compared to existing methods of investor decision-making.

Keywords: stock market, *DTW*, financial markets, chart patterns, shares.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivo del proyecto

Este proyecto se enmarca en la era contemporánea, donde la inteligencia artificial tiene un gran impacto en el estilo de vida de las personas. Con el paso del tiempo la evolución y el desarrollo de este campo es cada vez mayor. Es por esto que surge la idea de aprovechar el potencial de estas herramientas para crear un sistema reconocedor de patrones en gráficos de bolsa que sea capaz de encontrar automáticamente la formación de patrones. Una vez encontrados estos patrones, se pretende que la herramienta compruebe si se ha alcanzado el valor esperado. De esta manera, habiendo analizado qué patrones alcanzaron su valor esperado, se puede calcular de forma precisa la probabilidad que tiene un patrón de alcanzar un valor para un conjunto de acciones bursátiles dadas. La finalidad de esta aplicación es que pueda servir como soporte inteligente en el análisis del comportamiento del mercado bursátil. Cumpliendo el propósito planteado, podría aumentar la eficiencia y maximizar los beneficios en las acciones tomadas, lo que puede llegar a reflejar una mejoría y reducción de incertidumbre en las operaciones realizadas por los usuarios que utilicen dicha herramienta.

Este proyecto puede servir también como investigación acerca de las técnicas sobre el reconocimiento de patrones, más allá del ámbito de gráficos de bolsa.

Culminar este proyecto con éxito supondría una aportación para los expertos que deseen tener un apoyo en sus operaciones en bolsa, así como una contribución al acercamiento de las tecnologías modernas a un área como la economía de los mercados financieros.

1.2. Objetivo

El objetivo de este proyecto es desarrollar en *Python* una herramienta con las siguientes características:

- Capacidad de reconocer y encontrar diferentes patrones en los gráficos del mercado bursátil.
- De dichos patrones, indicar el valor a alcanzar y la dirección que según el patrón encontrado debería tomar.
- Determinar la probabilidad de que dicho patrón bursátil alcance el valor esperado.

El conjunto de datos utilizado para el entrenamiento de la detección de los patrones consta de fragmentos de gráficos de bolsa, obtenidos a través de internet, que representan solamente los patrones que se pretenden detectar. Estos fragmentos son seleccionados a mano por una persona, y sirven como muestras para encontrar dichos patrones por aprendizaje supervisado.

Una vez creado el conjunto de datos, la herramienta recibe una o varias empresas por su ticket bursátil, y busca en la gráfica de los precios de las acciones en bolsa de dicha empresa los tipos de patrones que haya seleccionado. Una vez detectado un patrón, se analiza la dirección del precio de las acciones, es decir, si aumenta o disminuye el precio, y se miden las posibilidades de que ese objetivo sea cumplido. También, se le pedirá al usuario que introduzca un año en el que comenzar el análisis, de esta manera, puede elegir qué contexto económico prefiere estudiar, ya que dependiendo de las fechas escogidas de inicio, la probabilidad de las tendencias de formación de patrones podría variar drásticamente debido al comportamiento del mercado, como por ejemplo, en épocas de crisis o sobrecalentamiento económico.

1.3. Fases del proyecto

El proyecto se dividirá en las siguientes fases:

1. La primera parte consistirá en una investigación acerca de los objetivos que se desean cumplir. Se deberán estudiar los distintos métodos que existen en la actualidad para la detección de patrones en bolsa, así como buscar diferentes algoritmos para comparar secuencias de datos entre sí.

2. Después, se tendrá que realizar un minado de datos en busca de muestras de patrones para conformar así un conjunto de datos que será utilizado para el desarrollo y experimentación de la herramienta.

3. Después, se desarrollará el software encargado de la detección de patrones, empleando el método escogido tras la investigación realizada.

4. Una vez la herramienta sea capaz de detectar patrones, se planteará un algoritmo para el análisis del objetivo de cada patrón. Cuando se concluya este algoritmo, se implementará en forma de código.

5. Tras haber obtenido los resultados de los objetivos de cada patrón, será posible proporcionar a los usuarios las probabilidades de la tendencia de los patrones.

6. Por otro lado, utilizando algunos métodos que se emplean para la detección de patrones en el cálculo de probabilidades, se desarrollará e implementará un algoritmo capaz de detectar patrones que han terminado de formarse en el momento de la búsqueda.

7. Cuando se hayan terminado estas dos partes, se creará una interfaz gráfica junto con su documentación y archivos ejecutables para *Linux* y *Windows 10*.

8. Por último, se debe escribir una memoria que abarque el desarrollo de este proyecto y la investigación realizada para llevarlo a cabo.

Capítulo 2

Contextualización

2.1. Conceptos relevantes

En este capítulo se abordan los conceptos imprescindibles para entender el contexto del proyecto, como por ejemplo, ¿qué son los patrones en gráficos?, o algunos métodos para el reconocimiento de dichos patrones que existen en la actualidad.

2.2. Análisis técnico

El análisis técnico es un método de evaluación del mercado financiero basado en suponer que los precios se mueven en tendencias y que algo que ha pasado previamente, tiende a repetirse.

El análisis técnico intenta pronosticar la dirección que toman los precios de una determinada entidad en la bolsa de valores. Este pronóstico se genera a través de la recopilación de datos históricos acerca de la actividad del precio de las acciones. Una vez obtenidos estos datos, se pueden aplicar diferentes métodos estadísticos que resulten en una predicción de los cambios en los precios.

2.2.1. Análisis *chartista*

"El análisis chartista es la rama del análisis técnico que estudia los patrones que se forman en las cotizaciones bursátiles"(Francisco López, 2018). Por definición, ”un patrón de precios es una configuración reconocible del movimiento de los precios que se identifica mediante una serie de líneas de tendencia y/o curvas. Cuando un patrón de precios señala un cambio en la dirección de la tendencia, se conoce como patrón de inversión; un patrón de continuación se produce cuando la tendencia continúa en su dirección existente” (Hayes, 2022). En conclusión, ”un patrón se identifica por una línea que conecta los puntos de los valores del precio comunes, como los precios de cierre o los máximos o mínimos diarios, durante un periodo de tiempo específico”.

”En finanzas, una línea de tendencia, figura 2.1, es una línea que delimita el movimiento del precio de un valor. Se forma cuando se puede trazar una línea diagonal entre un mínimo de tres o más puntos de pivote del precio. Se puede trazar una línea entre dos puntos cualesquiera, pero no se considera una línea de tendencia hasta que se compruebe. De ahí la necesidad del tercer punto, la prueba” (Edwards and Magee, 1948). Dado que los patrones de precios se identifican utilizando una serie de líneas y/o curvas, resulta de

interés entender las líneas de tendencia y saber cómo dibujarlas. Las líneas de tendencia ayudan a los analistas técnicos a detectar zonas de soporte y resistencia en un gráfico de precios.

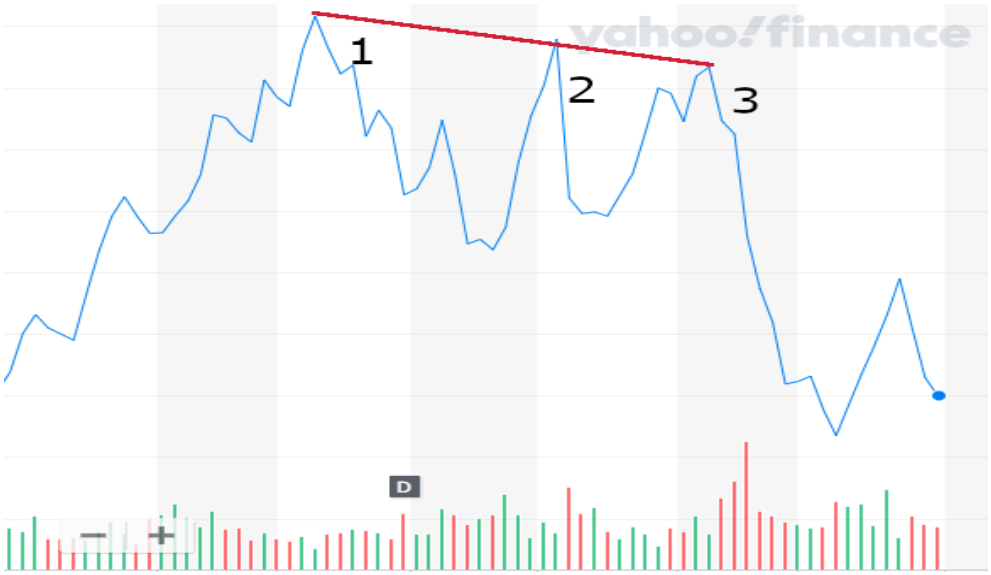


Figura 2.1: Ejemplo de línea de tendencia

En los gráficos de bolsa, el valor comúnmente utilizado es del precio de cierre. ”El precio de cierre es el valor bursátil más importante del día. Se utiliza para fijar el precio de las acciones de los fondos de inversión, para que los inversores institucionales informen a sus clientes de los resultados como en los índices S&P 500, el Dow Jones o el MSCI, el precio de los derivados, el valor de los activos de los fondos cotizados (ETF), el cálculo de los márgenes y la liquidación, entre otras cosas” (Bogousslavsky and Muravyev, 2019).

Algunos de los principales tipos de patrones que podemos encontrar son el doble techo, el doble suelo, el triple techo o el triple suelo entre muchos otros.

El patrón de tipo doble techo (2.2) o doble suelo (2.3) es formado en los momentos en el que el mercado intenta superar un nivel de resistencia o de soporte respectivamente dos veces, fallando en ambas ocasiones. ”Un doble techo se produce cuando los precios suben hasta un nivel de resistencia con volumen significativo, retroceden y posteriormente vuelven al nivel de resistencia con un volumen menor. A continuación, los precios descienden marcando el inicio de una nueva tendencia a la baja” (Achelis, 2001)

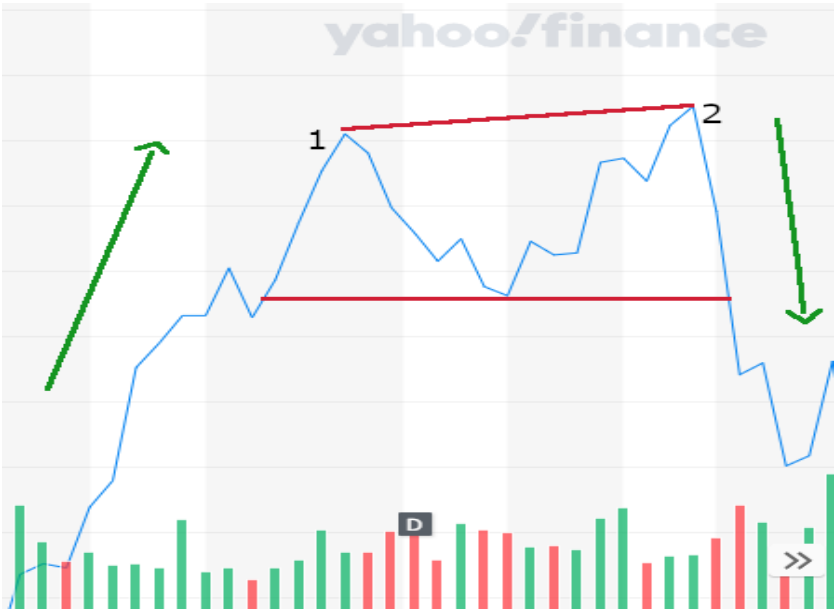


Figura 2.2: Ejemplo de doble techo

Un doble suelo, por otro lado, es similar a un doble techo, pero comenzando con una bajada de los precios hasta alcanzar un nivel de soporte, más tarde hay una subida, y de nuevo, una bajada hasta el ya mencionado soporte. A partir de aquí, continúa una nueva tendencia al alza.

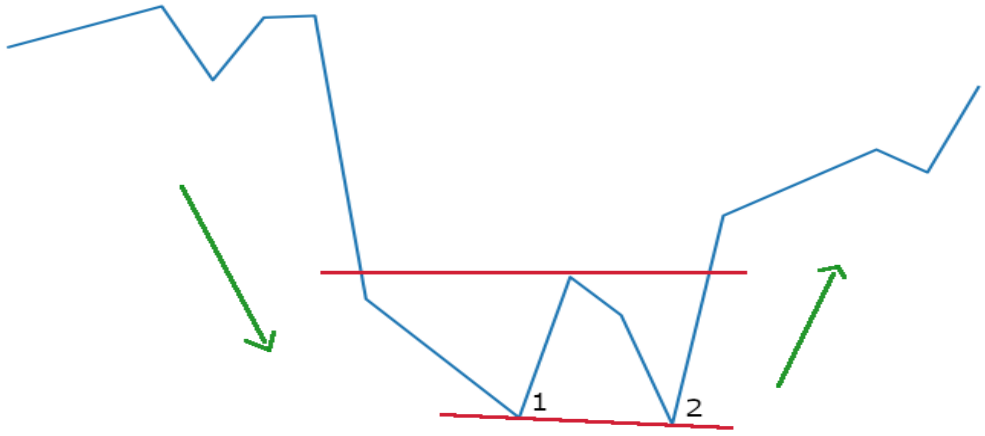


Figura 2.3: Ejemplo de doble suelo

Por último, el triple techo 2.4 o el triple suelo representa lo mismo que el doble techo y doble suelo, intentos fallidos de superar una resistencia o soporte, pero en este caso, son tres intentos, causando un cambio en la tendencia.

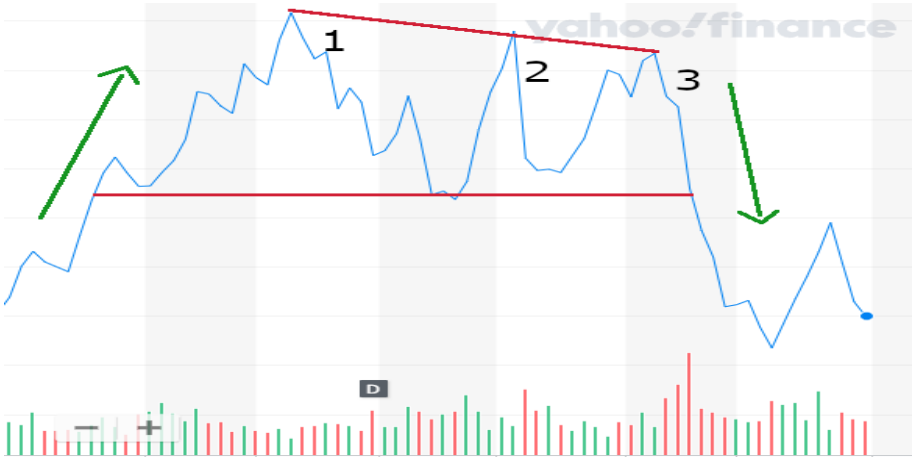


Figura 2.4: Ejemplo de triple techo

Tanto los doble suelo, doble techo, triple suelo, triple techo, así como un canal lateral, tienen el mismo objetivo, medir el ancho del canal o el tamaño desde el techo hasta el suelo, y proyectarlo por el lado que rompe.

2.3. Antecedentes y estado actual del tema

En el panorama tecnológico actual existen varias investigaciones con la misma temática que este trabajo, la detección de patrones chartistas. Por ejemplo, en Kim et al. (2018), los autores explican en su artículo cómo crear un sistema que reconoce patrones en los movimientos de la bolsa originados por la mañana, y según los patrones reconocidos, predecir el comportamiento del mercado por la tarde. Lo destacable de este artículo es el uso de *Dynamic Time Warping* para la detección de patrones, un algoritmo que también es utilizado en este proyecto y que se explicará más adelante. Otros investigadores confirman la utilidad de este algoritmo, como ocurre en el artículo de Grzejszczak et al. (2022) y Coelho (2012), en donde se ponen a prueba diferentes muestras de secuencias de valores del mercado financiero, comparándolas entre sí, con el fin de justificar la aplicabilidad de este algoritmo. Científicos que se suman a comprobar la utilidad del *Dynamic Time Warping* son Shirota and Murakami (2021), que concluyen que, tras agrupar las distancias obtenidas con *DTW* en diferentes *clusters*, los inversores tienden a invertir en compañías según su país y no según los resultados individuales que obtiene la empresa. También existen herramientas como *Finviz* (Duris, 2007) que muestran una gran variedad de patrones que ocurren en la actualidad. Esta plataforma fue utilizada con el fin de buscar muestras para la base de datos de este proyecto. Otras aproximaciones a la detección de patrones en bolsa han sido a través de aprendizaje automático. *VantagePoint* (Mendelsohn, 2021) es una empresa que, gracias al uso de redes neuronales artificiales, es capaz de predecir el comportamiento del mercado, según afirman, con un 87.4 % de precisión. Algunos investigadores también han sido cautivados por este fenómeno, como Umer et al. (2019), donde se explica como predecir el mercado de valores utilizando también aprendizaje automático. Sin embargo, dado el potencial de este tipo de herramientas, es probable que exista software muy potente con este mismo propósito que no sea conocido públicamente, ya que ocultar la posesión de programas de este tipo capaces de proporcionar resultados de alta calidad, puede conllevar una gran ventaja competitiva para la empresa que disponga de él.

Capítulo 3

Desarrollo

3.1. Conjunto de datos

Para el desarrollo de este trabajo, se debe crear un conjunto de datos obtenidos manualmente y que posteriormente será dividido en dos subconjuntos, una para entrenamiento y otro para evaluación. En el apartado ”Experimentación y resultados” se profundiza acerca de esta división. La finalidad de este conjunto es dotar a la herramienta de ejemplos que sirvan como referencia a la hora de tener que etiquetar una serie temporal como patrón especial o no.

3.1.1. Obtención de datos de la bolsa

Debido a que el propósito de la herramienta que se pretende crear es analizar los datos más recientes de la bolsa de valores, se necesita consultar estos datos en tiempo real y de manera rápida. Es por ello que, utilizar una *API* que permita obtener los datos de manera inmediata y fiable es una buena opción. Se decidió probar con dos *API*s diferentes: Alpha Vantage (Porté and Zheng, 2017) y Yahoo Finance (Aroussi, 2019). La elección final fue Yahoo Finance, ya que Alpha Vantage estaba más orientada a obtener datos por intervalos de tiempo de una hora, y el objetivo de este trabajo es el análisis con los precios de cierre de la bolsa, por lo que Alpha Vantage no es tan útil como Yahoo Finance, que sí obtiene dichos datos en el formato deseado.

3.1.2. Creación del conjunto de datos

La creación de los primeros conjuntos de datos se llevó a cabo desarrollando un pequeño programa que inicia un bucle donde el usuario introduce el nombre de la empresa, la fecha de inicio del patrón detectado y la fecha donde termina el patrón, y crea el *dataframe* correspondiente, que supuestamente representa un patrón especial. Más tarde, estos datos serán normalizados según se explica en el apartado 3.1.3. Tras generar el mencionado *dataframe*, se almacena su tamaño, denominado n, y se crea otro *dataframe* que no representa ningún patrón relevante del mismo tamaño, n. La intención de este tipo de series temporales que no representan nada será expuesta en apartados posteriores. Este *dataframe* corresponde a los datos inmediatamente anteriores a la fecha de comienzo del patrón relevante introducida por el usuario, de modo que se cogerán los n datos anteriores a dicha fecha. Con esto se pretende automatizar la creación de dos tipos de secuencias distintas con solamente introducir una fecha de inicio y una fecha de

fin. Debido a la poca escalabilidad de este programa para producir numerosos *dataframes* en poco tiempo, se decidió optar por otro programa que realizara un trabajo similar, pero, en vez de ser un bucle donde el usuario inserta los datos por terminal, el usuario crea una lista, en donde cada elemento contiene el nombre de la empresa, la fecha de inicio, la fecha final, y el tipo de patrón recogido. El programa lee uno a uno estos elementos y los almacena en el directorio correspondiente de cada tipo de patrón. La única diferencia de este programa con el anterior, es que esta vez no crea un *dataframe* que no represente ningún patrón relevante, sino que ahora este tipo de patrones deberán ser incluidos en dicha lista.

3.1.3. Estructura del conjunto de datos

Los patrones recogidos en el conjunto de datos se clasifican de la siguiente manera: doble techo, doble suelo, triple techo y falso positivo. El significado de los patrones de tipo ”falso positivo” o *Universal Background Model* será abordado en apartados posteriores 3.1.5. La cantidad de patrones por cada tipo fue cincuenta, haciendo un total de doscientos patrones en el conjunto de datos final. Este número fue elegido de manera heurística; esperando que fueran suficientes para poder dotar al experimento con un mínimo de rigurosidad y fiabilidad. El tamaño de las series temporales es variado, ya que lo cautivador de este proyecto es analizar series temporales comparándolas entre sí, aunque no tengan el mismo número de elementos.

Los valores de cada *dataframe* están alterados respecto a los originales debido a la aplicación de una normalización tipo Min-max (Munkhdalai et al., 2019) con la finalidad de que se encuentren en el rango entre 0 y 1. Esta normalización es necesaria, ya que los precios varían, por lo que estos deben ser transformados a una escala común, de este modo, las series temporales pueden ser comparadas conservando su forma original. Tras la normalización, algunos valores dentro de los *dataframes* fueron modificados a propósito para que su morfología se asemejara aún más a un tipo de patrón relevante.

3.1.4. Compañías y fechas seleccionadas

Los patrones elegidos para conformar el conjunto de datos, son los siguientes:

Doble techo			Doble Suelo		
Empresa	Fecha inicio	Fecha final	Empresa	Fecha inicio	Fecha final
AVA	2022-02-25	2022-04-13	CWAN	2022-03-01	2022-03-17
BMO	2022-01-10	2022-02-23	ECL	2021-11-22	2021-12-29
CMA	2021-12-31	2022-02-24	VIOT	2021-08-10	2021-09-2
CTO	2022-03-01	2022-04-06	FLXS	2021-10-22	2021-11-05
CTSH	2021-12-20	2022-02-23	SGBI	2021-08-12	2021-09-23
FWONK	2021-12-16	2022-03-01	BRAG	2021-07-12	2021-08-25
HLT	2021-12-16	2022-02-23	SKX	2021-12-08	2022-02-10
INVA	2022-02-10	2022-04-19	LNN	2021-11-24	2022-01-04
NBN	2022-01-04	2022-02-25	FKWL	2021-07-14	2021-08-02
OMFL	2021-10-27	2022-01-20	RADI	2021-09-16	2021-10-06

Doble techo		
Empresa	Fecha inicio	Fecha final
PRK	2021-11-02	2022-01-21
CTBI	2021-09-27	2022-02-25
MJIN	2022-01-07	2022-03-24
GL	2022-01-06	2022-02-24
CEPU	2021-10-13	2021-11-17
FXO	2021-12-20	2022-02-24
OXLC	2021-12-31	2022-02-14
GDEN	2022-03-15	2022-05-06
IMXI	2021-12-30	2022-01-21
CBTX	2021-12-22	2022-04-12
BLU	2021-10-26	2021-11-16
RJF	2022-03-23	2022-03-22
ISEE	2021-12-20	2022-01-04
PPI	2022-03-16	2022-04-25
DY	2021-11-22	2021-12-16
CPE	2021-11-04	2021-11-18
FLR	2022-03-15	2022-04-21
DDIV	2022-03-17	2022-04-25
DDIV	2021-12-23	2022-01-18
FJAN	2022-01-28	2022-02-11
ASH	2022-04-12	2022-05-09
ASH	2021-12-20	2022-01-18
SPYV	2022-03-23	2022-04-22
AG	2022-03-14	2022-04-21
OUT	2021-09-29	2021-10-25
OUT	2021-08-04	2021-09-10
SLG	2022-01-03	2022-01-20
VOOV	2022-03-23	2022-04-22
CAPL	2022-01-10	2022-02-03
DIN	2022-03-15	2022-04-06
OSW	2021-11-03	2021-11-16
SLB	2022-03-18	2022-04-25
PRU	2022-03-18	2022-04-25
HPE	2021-12-22	2022-03-01
TEQI	2022-03-16	2022-04-22
TSC	2022-03-23	2022-04-22
GWW	2021-10-28	2022-05-11
TEQI	2021-12-22	2022-02-24
DIN	2021-12-01	2022-01-28
PPI	2022-02-24	2022-03-15

Doble Suelo		
Empresa	Fecha inicio	Fecha final
IVLU	2021-09-15	2021-10-15
IVLU	2021-11-16	2022-01-04
BRKR	2022-04-08	2022-05-18
CHN	2022-04-14	2022-05-18
BIDU	2022-04-13	2022-05-20
HASI	2021-09-01	2021-10-14
GVAL	2021-10-18	2021-12-27
DCI	2022-11-24	2021-12-29
EWG	2021-11-23	2021-12-27
SGRP	2021-08-12	2021-09-03
GFL	2022-02-15	2022-03-18
GFL	2022-01-18	2022-02-01
SDIV	2021-12-10	2021-12-17
BCOV	2021-09-13	2021-11-11
KBH	2021-12-10	2022-01-13
WE	2022-03-02	2022-03-17
NOVA	2022-04-13	2022-05-19
ING	2022-04-21	2022-05-17
IAA	2021-11-17	2021-12-28
IDMO	2021-11-19	2021-12-27
FL	2021-07-26	2021-08-20
HUYA	2022-04-12	2022-05-19
EVR	2021-09-15	2021-10-05
CQQQ	2022-04-07	2022-05-19
ENIA	2021-07-30	2021-08-27
ENIA	2021-10-26	2021-11-23
DLB	2021-09-27	2021-10-19
APLE	2021-11-24	2021-12-22
CLDT	2021-07-29	2021-09-01
CLDT	2022-02-22	2022-03-10
PLAY	2021-07-27	2021-08-25
LAC	2022-02-28	2022-03-22
HESM	2022-03-11	2022-04-12
NVEE	2022-01-20	2022-02-09
NVEE	2022-01-14	2022-03-01
UUUU	2021-09-17	2021-10-13
DEF	2022-02-10	2022-03-18
IMBI	2021-12-07	2022-02-18
CF	2022-04-19	2022-05-17
WHR	2021-11-16	2022-01-04

Triple suelo		
Empresa	Fecha inicio	Fecha final
CORN	2022-04-06	2022-05-25
ACAD	2022-01-28	2022-04-22
TLK	2022-04-06	2022-05-09
HPE	2021-12-22	2022-04-06
CM	2022-01-05	2022-04-06
TEQI	2021-12-20	2022-04-29
WFG	2022-01-21	2022-03-31
FL	2021-12-20	2022-02-23
DIN	2021-12-01	2022-01-18
ECL	2021-10-28	2022-01-14
KBH	2021-10-29	2022-01-07
DIVZ	2021-12-22	2022-03-15
HSON	2022-01-26	2022-03-08
BXC	2022-02-23	2022-03-24
NVGS	2022-04-05	2022-05-25
RJI	2022-04-11	2022-05-19
LEGH	2021-11-10	2021-12-14
HPQ	2022-04-13	2022-05-20
JXN	2022-03-09	2022-05-10
CSGS	2022-03-01	2022-05-02
ESS	2021-11-04	2022-01-27
IIVI	2021-12-02	2022-04-14
AAPL	2021-12-03	2022-01-21
MSFT	2021-10-26	2022-01-05
AMZN	2018-08-17	2018-10-08

Triple suelo		
Empresa	Fecha inicio	Fecha final
GOOGL	2018-07-20	2018-09-05
GOOGL	2017-12-29	2018-03-27
GOOGL	2017-05-17	2017-08-10
GOOGL	2014-08-12	2014-10-01
GOOGL	2007-10-16	2008-01-08
UNH	2009-09-01	2009-09-23
JNJ	2021-12-03	2022-02-22
JNJ	2011-08-22	2011-09-09
JNJ	1997-07-18	1997-08-08
NVDA	2014-03-17	2014-04-11
NVDA	2011-10-07	2011-11-01
NVDA	2011-04-19	2011-06-09
FB	2021-06-25	2021-07-29
XOM	2018-09-17	2018-10-10
XOM	2014-10-30	2014-11-26
XOM	2012-03-14	2012-04-04
XOM	2010-04-16	2010-04-30
JPM	2018-10-29	2018-12-06
JPM	2013-06-05	2013-06-20
JPM	2009-08-04	2009-08-17
V	2021-09-21	2021-10-13
V	2015-12-02	2016-01-04
CVX	2019-11-01	2019-11-18
HD	2000-03-28	2000-04-13
MA	2012-06-29	2012-08-07

Tabla 3.1: Patrones relevantes que conforman el conjunto de datos

3.1.5. *Conjunto de falsos positivos*

Una vez creado un conjunto de datos con muestras de todos los tipos de patrones que serán clasificados en este trabajo, se plantea crear un conjunto de datos que no represente ningún tipo de patrón en específico. La finalidad de este conjunto de falsos positivos, es poder utilizarlo para saber si una secuencia es similar en mayor medida con el tipo falso positivo que con los tipos relevantes. En el caso de ser así, podemos concluir que la secuencia no conforma ningún tipo de patrón relevante. Es por ello, que el conjunto de datos de falsos positivos debe almacenar la mayor variedad de casos que no representen ningún patrón relevante. Al igual que en el resto de tipos de patrones, para este trabajo se almacenaron cincuenta muestras de patrones de tipo falso positivo.

3.2. *Dynamic Time Warping*

Existen múltiples métodos para comparar series temporales entre sí. Para el objetivo de este trabajo, se necesita un algoritmo que calcule la similitud entre dos series temporales, siendo ambas de distinto tamaño en la mayoría de los casos. De entre los principales algoritmos que miden la similitud entre series temporales, no hay ninguno significativamente mejor que *Dynamic Time Warping (DTW)* según concluyen los autores de Bagnall et al. (2016) en su artículo.

Como se explica en su artículo Alizadeh (2020), escrito para *towardsdatascience*, *Dynamic Time Warping* es un método para calcular el emparejamiento óptimo entre los puntos de dos series temporales.

Dadas dos secuencias:

$$\begin{aligned} X &= x[1], x[2], ..., x[i], ..., x[n] \\ Y &= y[1], y[2], ..., y[j], ..., y[m] \end{aligned} \tag{3.1}$$

Las secuencias *X* e *Y* pueden organizarse para formar una matriz *nxm* donde cada punto *(i, j)* es la alineación entre *x[i] e y[j]*. Un camino deformado *W* mapea los elementos de *X* e *Y* para minimizar la distancia entre ellos. *W* es una secuencia de puntos *(i, j)* de la matriz.

El camino óptimo entre *(i_k, j_k)* puede calcularse como:

$$D_{min}(i_k, j_k) = \min_{i_{k-1}, j_{k-1}} D_{min}(i_{k-1}, j_{k-1}) + d(i_k, j_k | i_{k-1}, j_{k-1}) \tag{3.2}$$

donde *d* es la distancia Euclídea. Entonces, el coste global de la ruta puede calcularse como

$$D = \sum_{k} d(i_k, j_k) \tag{3.3}$$

El algoritmo de *Dynamic Time Warping* utiliza un enfoque basado en programación dinámica para alinear dos secuencias. Recorrer todas las posibles rutas es tremendamente costoso (Berndt and Clifford, 1994). Por lo tanto, por razones de eficiencia, es importante limitar el número de posibles rutas deformadas, y de ahí que se destaquen las siguientes restricciones:

- **Condición límite**: Esta restricción asegura que la trayectoria de deformación comienza con los puntos de inicio de ambas secuencias y termina con los puntos finales.

$$i_1 = 1, i_k = n \text{ y } j_1 = 1, j_k = m \tag{3.4}$$

- **Condición de monotonicidad**: Esta restricción preserva el orden temporal de los puntos (no retroceder en el tiempo).

$$i_{t-1} \leq i_t \text{ y } j_{t-1} \leq j_t \tag{3.5}$$

- **Condición de continuidad (tamaño de los pasos)**: Esta restricción limita las transiciones de la ruta a puntos adyacentes en el tiempo (no a saltos en el tiempo).

$$i_t - i_{t-1} \leq 1 \text{ y } j_t - j_{t-1} \leq 1 \tag{3.6}$$

Además de las tres restricciones anteriores, existen otras condiciones menos frecuentes para una ruta deformada permitida:

- **Condición de ventana de deformación**: Los puntos permitidos pueden restringirse para que caigan dentro de una ventana de deformación dada de ancho ω (un número entero positivo).

$$\left|i_t - j_t\right| \leq \omega \tag{3.7}$$

- **Condición de pendiente**: La ruta deformada se puede limitar restringiendo la pendiente y, en consecuencia, evitando los movimientos extremos en una dirección.

$$\left|i_t - j_t\right| \leq \omega \tag{3.8}$$

Una ruta de deformación aceptable tiene combinaciones equivalentes a los movimientos del rey en el ajedrez:

- Movimientos horizontales: *(i, j)* $\rightarrow$ *(i, j + 1)*
- Movimientos verticales: *(i, j)* $\rightarrow$ *(i + 1, j)*
- Movimientos diagonales: *(i, j)* $\rightarrow$ *(i + 1, j + 1)*

3.2.1. Implementación

Actualmente existen multitud de librerías que permiten aplicar *Dynamic Time Warping*. Para este trabajo se utilizó la implementación creada por Giorgino (2009) debido a su sencillez de uso y las diversas posibilidades que ofrecía.

3.3. Búsqueda de patrones relevantes

El paso previo a calcular la probabilidad de que un patrón relevante cumpla su objetivo, es encontrar el mayor número posible de patrones de ese tipo, de modo que más tarde, una vez identificados, se analice si cumplen o no su objetivo individualmente para posteriormente poder realizar su media y conocer las probabilidades previamente mencionadas.

El algoritmo propuesto para identificar patrones relevantes en una empresa a partir de un año determinado es:

Algoritmo 1 Algoritmo para búsqueda de patrones relevantes

```
procedure buscarPatronesRelevantes(anchoeVentana, fechaInicial, fechaActual, patronesRelevantes)
    resultados[0..]
    indiceIzquierdo ← fechaInicial
    indiceDerecho ← indiceIzquierdo + anchoVentana
    while indiceDerecho ≤ fechaActual do
        ventana ← datosDeLaEmpresa.porcion(indiceIzquierdo, indiceDerecho)
        ventana ← normalizarConMinMax(ventana)
        tipoDePatron, distancia ← determinarTipoDePatron(ventana, patronesRelevantes)
        if tipoDePatron in patronesRelevantes then
            ventanaMejorada ← mejorarVentana(ventana, tipoDePatron)
            tendencia ← calcularTendencia(ventanaMejorada)
            if tendencia then
                patron ← new Patron(tipoDePatron, tendencia, indiceIzquierdo, indiceDerecho)
                resultados ← resultados + patron
                salto ← anchoVentana
            else
                salto ← 0
        indiceIzquierdo ← indiceIzquierdo + 1 + salto
        indiceDerecho ← indiceIzquierdo + anchoVentana
    return resultados
```

El propósito de este algoritmo (Algoritmo 1) es crear una lista de patrones, los cuales han sido detectados como relevantes. Los primeros pasos son asignar valores a las variables *anchoVentana, fechaInicial, indiceIzquierdo e indiceDerecho*. Las variables *indiceIzquierdo* e *IndiceDerecho* se usan para cortar los datos con los precios de la compañía y así crear una ventana. Después, un bucle *while* recorrerá todo el vector de precios y, por cada iteración, se buscará el tipo de patrón que encaja con la secuencia dentro de la ventana con el método *determinarTipoDePatrón*. Si el tipo de patrón determinado es

alguno de los patrones que se consideran relevantes, se llevará a cabo una mejora en la ventana gracias al método *mejorarVentana*. Una vez se mejora la ventana, se calculará la tendencia de ese patrón, es decir, si se cumple o no. Por último, si tiene una tendencia definida, es decir, se puede determinar si ha cumplido o no su objetivo, añadiremos el patrón a la lista de resultados. Una vez terminado este proceso moveremos la ventana. Cuando se encuentra un patrón, el salto que se debe efectuar es el del tamaño de la ventana del patrón encontrado, esto ocurre con el fin de evitar analizar una secuencia que ya se ha identificado como patrón. En cambio, cuando no se encuentra ningún patrón, el incremento equivale a 1. Una vez recorrido todo el histórico de precios de la empresa, devolveremos nuestra lista de patrones encontrados.

3.3.1. Determinación del tipo de patrón

Esta función es la encargada de, dada una secuencia, precisar el tipo de patrón más parecido. El método es de la siguiente manera:

Algoritmo 2 Algoritmo para determinar el tipo de patrón

```
procedure determinarTipoDePatron(secuencia, patronesRelevantes)
    distanciaMinima ← BIG_NUMBER
    tipoDePatronDeSecuencia
    for all tipoDePatron ∈ patronesRelevantes do
        for all ejemploDePatronRelevante ∈ tipoDePatron do
            distancia ← dtw(secuencia, ejemploDePatronRelevante)
            if distancia ≤ distanciaMinima then
                distanciaMinima ← distancia
                tipoDePatronDeSecuencia ← tipoDePatron
    return tipoDePatronSecuencia, distanciaMinima
```

La intención de este algoritmo (Algoritmo 2) es comparar la secuencia que recibe como parámetro con todos los patrones relevantes definidos que se encuentran en la base de datos. Por tanto, el primer bucle recorre los tipos de patrones relevantes que queremos encontrar. Es de interés recordar que, dentro de este conjunto de tipos, se encuentra el tipo de patrón *falso positivo*, por lo que si la secuencia que se está analizando, no pertenece a ninguno de los definidos como interesantes, encajará con el tipo *falso positivo*. Por cada tipo de patrón relevante, se recorrerán todos sus ejemplos que se encuentren en el conjunto de datos, y se calculará la distancia entre estas dos secuencias, utilizando *Dynamic Time Warping*. Se almacenará la menor distancia de entre todos los ejemplos y tipos, de esta manera, al terminar ambos bucles, obtendremos el tipo de patrón como resultado.

3.3.2. Mejora de ventana

Cuando encontramos una secuencia en donde se cree que hay patrones relevantes, no podemos afirmar que el tamaño de ese patrón coincide con el tamaño de la ventana. Esto ocurre porque el tamaño de la ventana es fijo. Para resolver este problema, y poder aislar el patrón de la mejor manera, se propone un algoritmo que busque, dentro de un tiempo razonable, los límites de ese patrón relevante encontrado. Este algoritmo es el siguiente:

Algoritmo 3 Algoritmo para mejorar la ventana

```
procedure mejorarVentana(ventana, tipoDePatron, listaDeDivisores)
    distanciaMinima ← ventana.distancia
    mejorVentana ← ventana
    for all divisor ∈ listaDeDivisores do
        ventanaCortada.ancho ← ventana.ancho/divisor
        ventanaCortada.indiceIzquiedo ← ventana.indiceIzquierdo
        while ventanaCortada.indiceDerecho ≤ ventana.indiceDerecho do
            tipoDePatron, distancia ←
            ← determinarTipoDePatron(ventanaCortada, tipoDePatron)
            if distancia ≤ distanciaMinima then
                distanciaMinima ← distancia
                mejorVentana ← ventanaCortada
            ventanaCortada ← moverVentana(ventanaCortada.ancho)
    return mejorVentana
```

Este algoritmo (Algoritmo 3) divide la ventana donde cree encontrar un patrón relevante, y la divide en partes iguales según los divisores que se introducen como lista por parámetro. Por cada divisor, se divide la ventana original, creando múltiples ventanas menores. Se recorre cada ventana menor una por una, aplicando *determinarTipoDePatrón*, de esta manera, si obtenemos una distancia menor a la que la ventana original tenía, significa que hemos encontrado una distancia menor respecto al mismo tipo de patrón que la ventana original. Se debe destacar que la función *determinarTipoDePatrón* no recibe todos los patrones relevantes como argumento, sino solamente el tipo de patrón que se detectó en la ventana original.

3.4. Búsqueda de patrones ocurriendo en el presente

Otro objetivo para este trabajo, es crear una opción para buscar patrones que están ocurriendo actualmente, en consecuencia, tras conocer la probabilidad de cumplimiento de la tendencia por cada tipo de patrón, el usuario podrá realizar las acciones que considere oportunas según dicha información. Para esta finalidad, se plantea un algoritmo que analice las empresas introducidas por el usuario en busca de patrones relevantes que finalicen en la fecha en la que se realiza la búsqueda. Para este algoritmo en concreto, se proponen dos opciones: la primera es una búsqueda en profundidad que dará un resultado preciso, pero exigirá más tiempo de cómputo, y la segunda opción, buscará algún tipo de patrón, y en el momento de dar con alguno parará y no intensificará la búsqueda para comprobar si hay alguna alternativa mejor, produciendo peores resultados que la primera opción pero ahorrando considerablemente el tiempo de cómputo. El algoritmo para la primera opción es el siguiente:

El algoritmo (Algoritmo 4) comienza por crear un diccionario que almacena, por cada tipo de patrón relevante, la distancia mínima encontrada y el índice por donde comienza la ventana comparada. Después comienza un bucle que altera el tamaño de la ventana, comenzando por un tamaño mayor que con el que termina. Para la opción de búsqueda

Algoritmo 4 Algoritmo para detectar patrones en la actualidad

```
procedure buscarPatronesActuales(indiceIzquierdoInicial, indiceIzquieroFinal, fechaActual, datosDeLaEmpresa, patronesRelevantes, nombreEmpresa)
    distancias{}
    resultados[0..]
    indiceIzquierdoVentana ← indiceIzquierdoInicial
    while indiceIzquierdoVentana ≤ anchoVentanaFinal do
        ventana ← datosDeLaEmpresa.cortar(indiceIzquierdoVentana, fechaActual)
        ventana ← normalizarConMinMax(ventana)
        tipoDePatron, distancia ← determinarTipoDePatron(ventana, patronesRelevantes)
        if tipoDePatron in patronesRelevantes then
            if distancia ≤ distancias[tipoDePatron].distancia then
                distancias[tipoDePatron].distancia ← distancia
                distancias[tipoDePatron].indice ← indiceIzquierdoVentana
        indiceIzquierdoVentana ← indiceIzquierdoVentana + 1
    for all tipo ∈ distancias do
        if distancias[tipo].indice ≠ −1 then
            patron ← new Patron(tipoDePatron, distancias[tipo].indice, fechaActual)
            resultados ← resultados + patron
    return resultados
```

en profundidad, comenzar con el mayor tamaño de ventana o el menor es irrelevante, sin embargo, cuando la búsqueda no es intensiva, sí que importa el orden. Para este trabajo, como se pretende dotar al usuario de la mayor cantidad de información posible, se comienza con el mayor tamaño de ventana. El siguiente paso a realizar es obtener los datos correspondientes a la posición de la ventana. Esto se lleva a cabo haciendo un *slice* del vector de todos los precios de la empresa, y aislando solamente los datos que se encuentren entre el índice izquierdo de la ventana y el índice derecho de la ventana, creando así un nuevo fragmento de datos. Se efectúa una normalización del nuevo fragmento de datos y se identifica el tipo de patrón que esta ventana representa utilizando el método *determinarTipoDePatrón*. Si el tipo de patrón encontrado se encuentra dentro de los patrones relevantes, se comprueba si la distancia encontrada es menor a la distancia más pequeña encontrada hasta el momento para ese tipo de patrón. Si es así, actualizamos el diccionario que almacena los valores de las distancias con el nuevo valor de distancia e índice de la ventana. Una vez terminado el bucle que recorre los distintos tamaños de ventana, recopilamos los datos almacenados en el diccionario de distancias, y los añadimos al vector de resultados para, como último paso, devolverlo.

3.4.1. Variación de la búsqueda de patrones actuales: segunda opción

Como se menciona previamente, existe una segunda opción para este algoritmo que no compara todos los tamaños posibles de ventana y almacena la menor distancia encontrada. Esta opción se detiene cuando encuentra un tipo de patrón que pertenece al conjunto de los tipos de patrón relevantes. La intención de esta opción, es poder recortar los tiempos de ejecución de la aplicación, ya que al devolver una ventana de datos, aunque no represente exclusivamente el patrón relevante y contenga ruido, el usuario es capaz de

identificarlo por sí mismo. Es por esto que el algoritmo sería similar a la primera opción, menos por la siguiente variación:

Algoritmo 5 Algoritmo para detectar patrones en la actualidad: opción 2

procedure buscarPatronesActualesSegundaOpcion(*indiceIzquierdoInicial, indiceIzquierdoFinal, fechaActual, datosDeLaEmpresa, patronesRelevantes, nombreEmpresa*)
  $distancias\{\}$
  $resultados[0..]$
  $indiceIzquierdoVentana \leftarrow indiceIzquierdoInicial$
  while $indiceIzquierdoVentana \leq anchoVentanaFinal$ **do**
    $ventana \leftarrow datosDeLaEmpresa.cortar(indiceIzquierdoVentana, fechaActual)$
    $ventana \leftarrow normalizarConMinMax(ventana)$
    $tipoDePatron,\ distancia \leftarrow determinarTipoDePatron(ventana, patronesRelevantes)$
    if $tipoDePatron$ in $patronesRelevantes$ **then**
      distancias[tipoDePatron].indice $\leftarrow$ **indiceIzquierdoVentana**
      patronesRelevante $\leftarrow$ **patronesRelevante - tipoDePatron**
    $indiceIzquierdoVentana \leftarrow indiceIzquierdoVentana + 1$
  for all $tipo \in distancias$ **do**
    if $distancias[tipo].indice \neq -1$ **then**
      $patron \leftarrow$ **new** $Patron(tipoDePatron, distancias[tipo].indice, fechaActual)$
      $resultados \leftarrow resultados + patron$
  return $resultados$

Nótese el cambio dentro de la condición que comprueba si el tipo de patrón encontrado es relevante. Para esta opción del algoritmo (Algoritmo 5), cuando esta condición es cumplida, solamente almacenamos el índice de la ventana que contiene el patrón encontrado, y eliminamos este tipo de patrón del conjunto de patrones relevantes, para que en futuras búsquedas, este tipo sea omitido.

3.5. Cálculo de la tendencia de los patrones

3.5.1. Objetivo para patrones tipo doble techo

Otra finalidad de este trabajo es calcular cuándo un patrón detectado cumple la dirección esperada según la teoría del análisis técnico del mercado. El análisis de la tendencia para el tipo de patrón ”doble techo” propuesto para este trabajo, se constituye de los siguientes componentes:

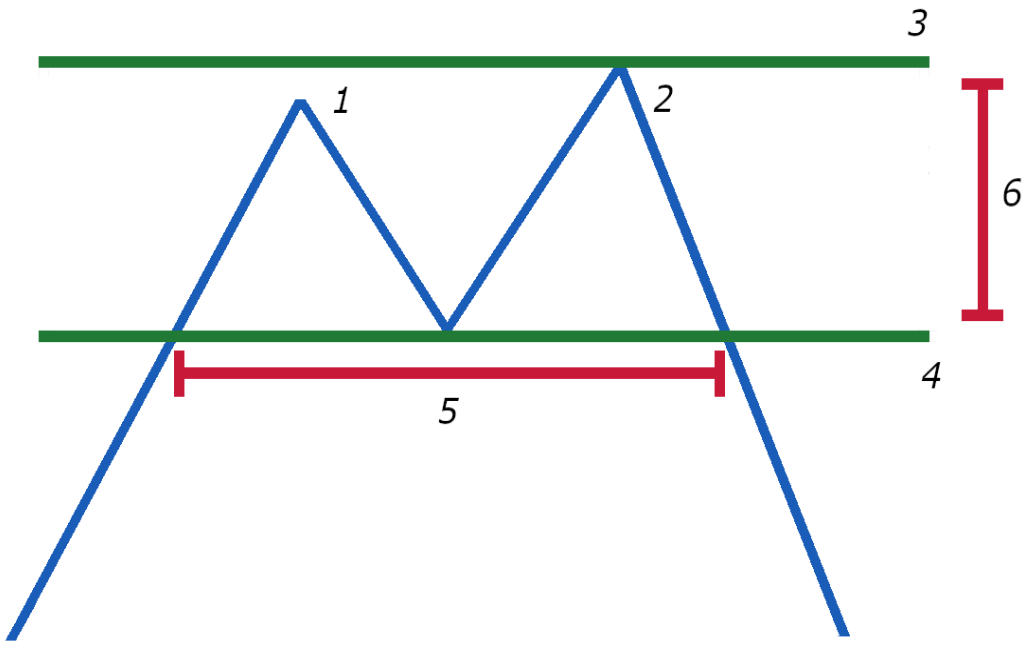


Figura 3.1: Componentes de un patrón doble techo

Primero debe ser identificado el techo (3.1, 6), este componente se calcula identificando los dos mayores máximos relativos o picos (3.1, 1 y 2). El techo (3.1, 3) se considera el máximo absoluto. El siguiente paso será encontrar la línea de soporte. Para esto debemos buscar el punto mínimo absoluto entre los dos picos. Una vez identificados los picos y la línea de soporte, se comprueba que la diferencia de altura entre ambos máximos no es elevada. Para este trabajo concretamente, se decide la altura de un máximo midiendo la diferencia entre él y la línea de soporte (3.1, 4). Para este análisis se decidió que la diferencia máxima decisiva es un tercio entre el pico menor y el máximo absoluto, es decir, si el pico menor, se encuentra a una altura menor o equivalente a un tercio de la altura del mayor pico, se considera que la diferencia es suficiente como para descartar el patrón. El próximo paso es medir el ancho y alto del patrón; el ancho (3.1, 5) corresponde a la distancia entre el punto en el que se superó la línea de apoyo por el lado izquierdo, y el punto donde se cruza la misma línea de apoyo por el lado derecho. Por último, la altura (3.1, 6) es la distancia entre la línea de soporte y la media entre las alturas del máximo absoluto (3.1, 2) y el pico menor (3.1, 1) previamente mencionadas.

Una vez determinados todos los componentes, queda determinar el objetivo que toma el patrón. El criterio escogido en este trabajo para saber si un patrón de tipo doble techo cumple el objetivo o no, es que tome una tendencia bajista equivalente o mayor a la altura del patrón en un tiempo limitado, comenzando desde que se cruza la línea de soporte. Un objetivo cumplido se vería de la siguiente manera:

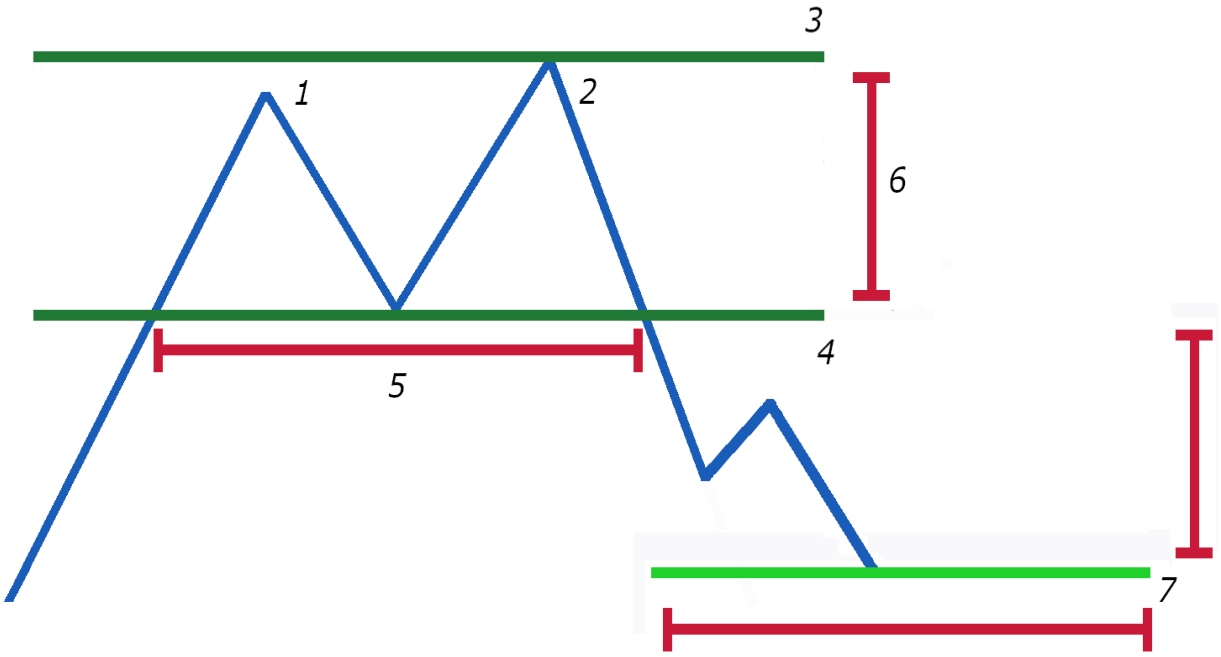


Figura 3.2: Doble techo con objetivo superado

La línea de color verde claro (3.2, 7) representa el valor que debe superar el patrón para concluir que ha cumplido su tendencia esperada. Sin embargo, puede ocurrir que nunca llegue a superar este valor, incluso, puede volver a cruzar la línea de soporte (3.3, 8), en este caso, cuando ocurre eso se etiqueta al objetivo como no cumplido. Este caso se vería así:

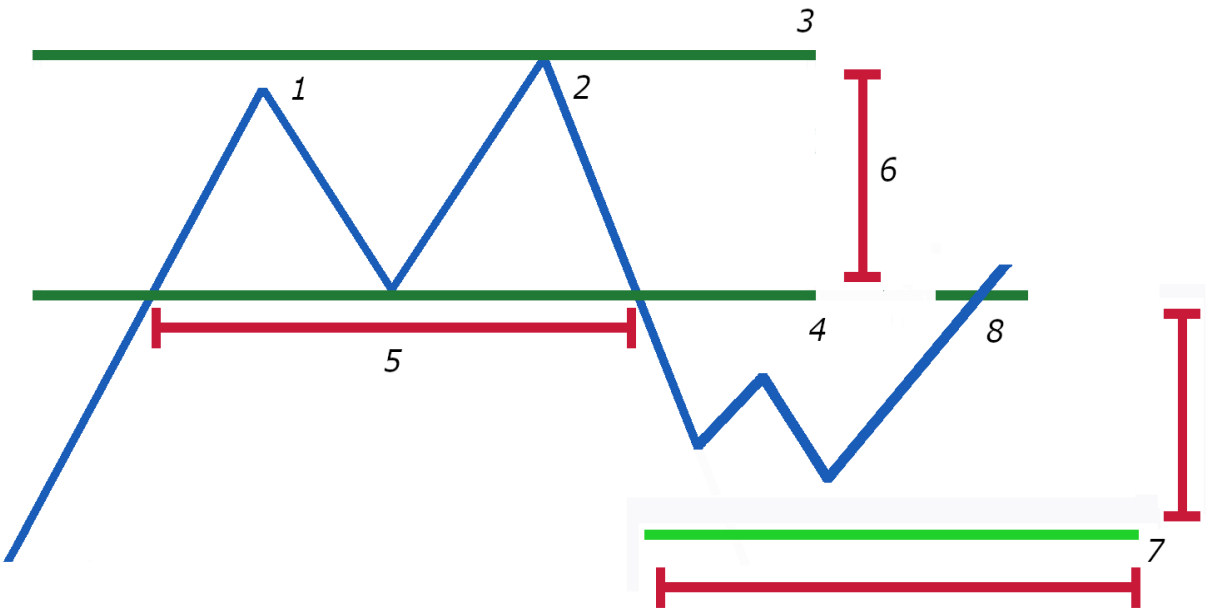


Figura 3.3: Doble techo con objetivo no superado

Por último, el ancho (3.3, 5) se utiliza para poner un límite de tiempo en el cual se

puede determinar el objetivo del patrón, este será el fragmento máximo a analizar, de modo que si no supera el valor esperado ni cruza la línea de soporte en ese tiempo establecido, se concluye como objetivo no superado.

3.5.2. Objetivo para patrones tipo doble suelo

La comprobación del cumplimiento de la dirección esperada de un patrón doble suelo es similar a la de un patrón doble techo, pero invertido. Es por ello que, en vez de buscar dos máximos relativos, se tendrá que obtener dos mínimos relativos (3.4, 1 y 2), y entre ellos dos, el máximo absoluto (3.4, 3). El cálculo del objetivo se efectuaría de igual manera, con la altura (3.4, 6) y anchura (3.4, 5) del patrón, en cambio, la tendencia para un doble suelo es ascendente, por lo que el objetivo (3.4, 7) deberá situarse por encima de la línea de resistencia (3.4, 4).

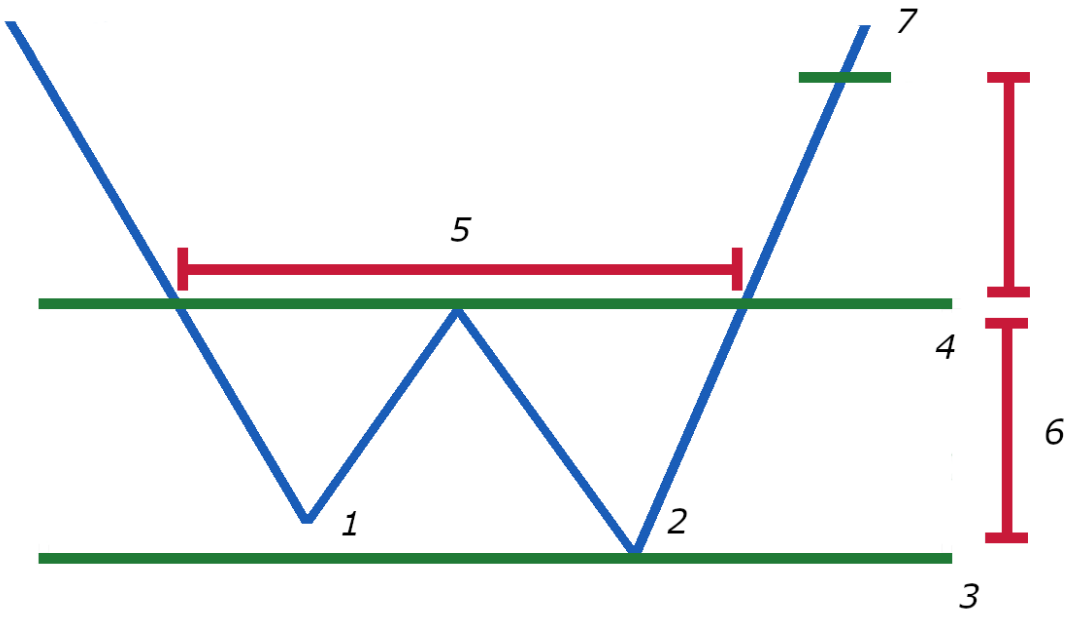


Figura 3.4: Componentes de un patrón tipo doble suelo

3.5.3. Suavizado para el cálculo del objetivo

Como propuesta para el cálculo del objetivo de un patrón encontrado, se realizó aplicando un suavizado denominado ”Media móvil” (Raudys et al., 2013). El objetivo que se pretendía de este suavizado era eliminar algunos picos que pudieran llevar a confusión durante la búsqueda de los dos máximos o techos. Los resultados no fueron los esperados debido a que, en algunos casos específicos, se eliminaban los techos que se pretendían encontrar, provocando un análisis del objetivo erróneo. Esto es un ejemplo de dicho caso:

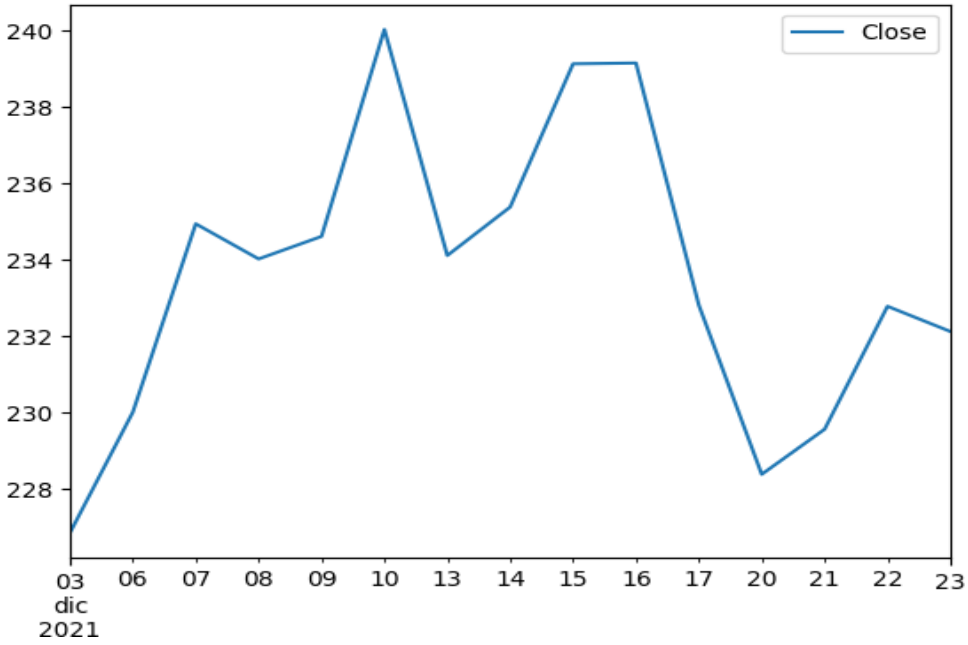


Figura 3.5: Doble techo sin suavizado

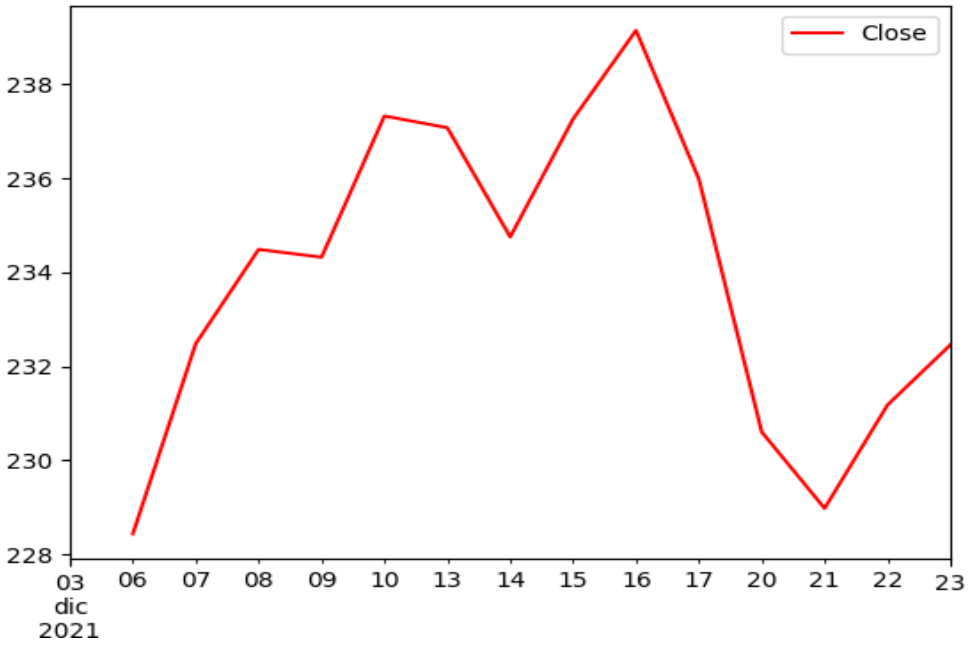


Figura 3.6: Doble techo con suavizado

3.5.4. Calcular probabilidad de cumplir el objetivo

Una vez creado un método para saber si un patrón cumple o no su objetivo, lo aplicamos a todos los patrones encontrados con los algoritmos de búsqueda de patrones, y contamos

cuántos patrones lo cumplen y cuántos no. La probabilidad de la tendencia se calcula por cada tipo de patrón de manera independiente. Se divide la cantidad de objetivos cumplidos entre el total de patrones de ese tipo encontrados.

$$\frac{cantidad\ de\ objetivos\ cumplidos\ de\ tipo\ A}{total\ de\ patrones\ de\ tipo\ A}$$

3.6. Interfaz gráfica de la aplicación

Una vez conseguida la detección de patrones y el cálculo de la tendencia, se presentó la creación de una interfaz gráfica para que el usuario pudiera utilizar la aplicación de manera sencilla e intuitiva. La librería que se empleó para la creación de dicha interfaz fue *Tkinter* para *Python*. Esta interfaz se ejecuta de manera local en el ordenador del usuario. Una vez iniciada la aplicación, el usuario puede navegar por las diferentes ventanas, ofreciendo las diferentes prestaciones que esta ofrece: cálculo de tendencia, búsqueda de patrones antiguos y búsqueda de patrones actuales.

La aplicación comienza con una página de inicio en la que se muestra el nombre de la aplicación y un botón de iniciar.

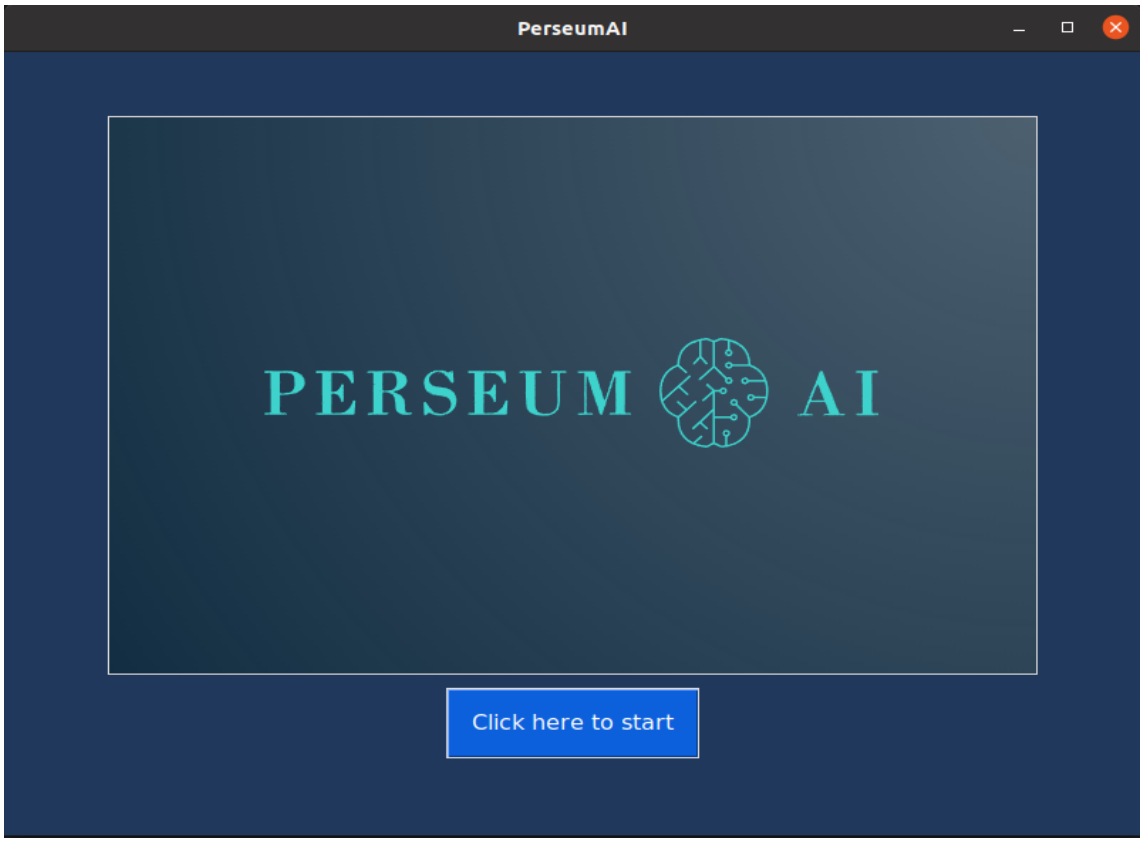


Figura 3.7: Página de inicio de PerseumAI

Cuando el usuario pulse el botón de iniciar, la ventana 3.7 se transformará a un menú 3.8 para poder introducir los datos de las empresas y fechas deseadas para ejecutar la aplicación. Estos datos son: los tipos de patrones que desea estudiar, las abreviaturas de los nombres de las empresas que desea analizar escritas en un fichero ".txt" y separadas entre ellas por un retorno de carro, el año desde el que se desea comenzar el estudio y la

opción para activar el modo de búsqueda intensiva para patrones actuales. La explicación de este modo se encuentra en la sección ” 3.4 Búsqueda de patrones” ocurriendo en el presente. Una vez introducidos estos datos, con el botón *Run* comenzará la ejecución del programa.

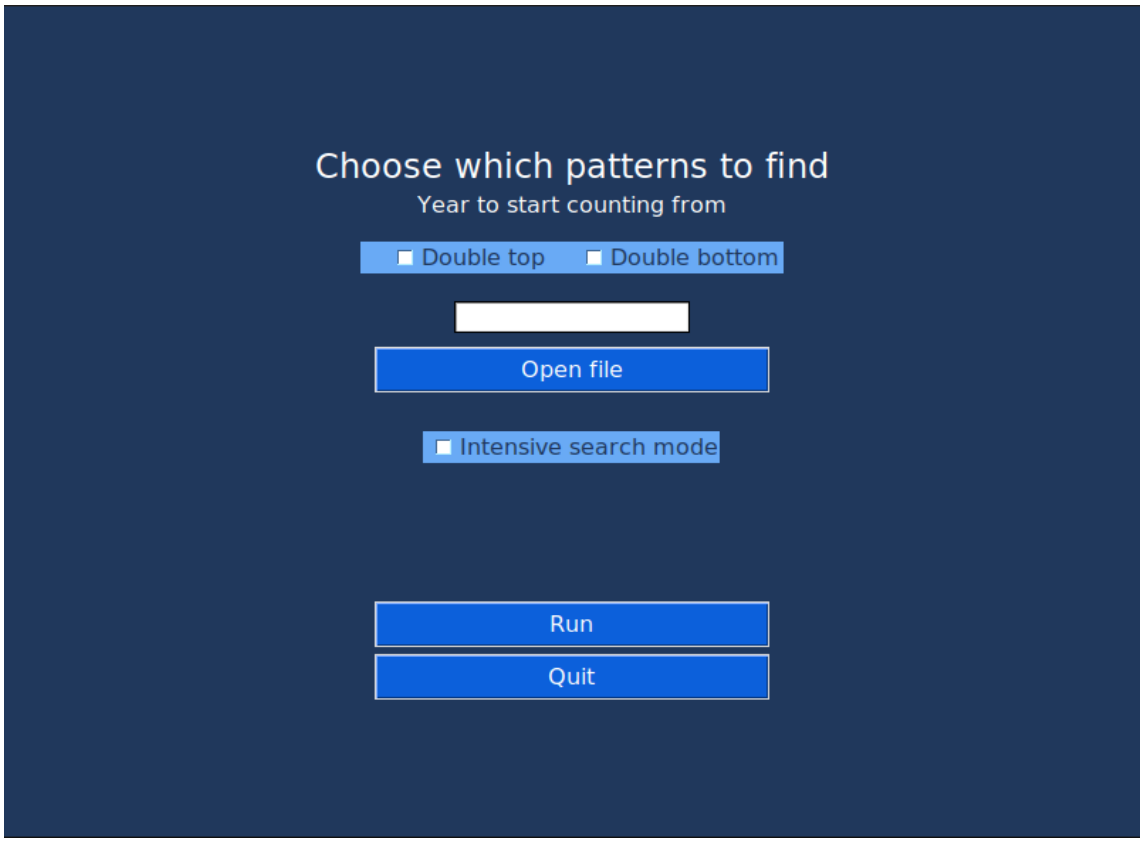


Figura 3.8: Menú principal de PerseumAI

Una vez se termine de calcular la tendencia de los tipos de patrones seleccionados, se abrirá una nueva ventana 3.9 en la que se mostrarán las probabilidades resultantes, así como dos botones, el primero con la opción de ver los patrones antiguos detectados que se usaron para el cálculo de dichas tendencias, y otro botón para mostrar los patrones que se han encontrado actualmente en el mercado de valores.

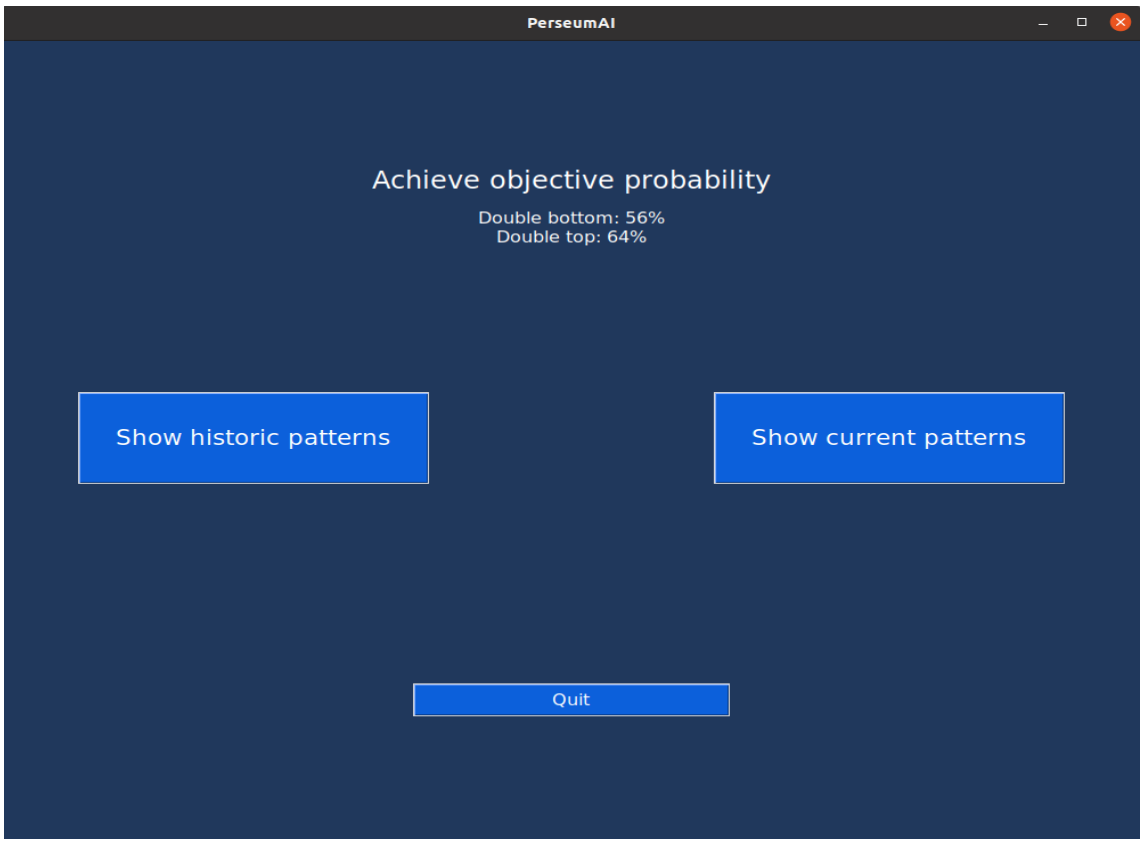


Figura 3.9: Resultados de PerseumAI

En la ventana de resultados de patrones antiguos 3.10, se muestran los patrones encontrados y utilizados para el cálculo de la tendencia, junto con una indicación de cumplimiento o no del objetivo esperado. Esta indicación se manifiesta en forma de cuadrado verde o rojo, verde para indicar que se ha cumplido, y rojo para indicar que no se ha cumplido. La visualización del patrón termina en el momento en el que se determinó su objetivo o no, es decir, termina cuando supera el objetivo o cruza la línea de soporte. Esta ventana tiene un botón para desplazarse verticalmente debido a que se imposibilita poder mostrar todos los patrones dentro de una misma ventana fija.

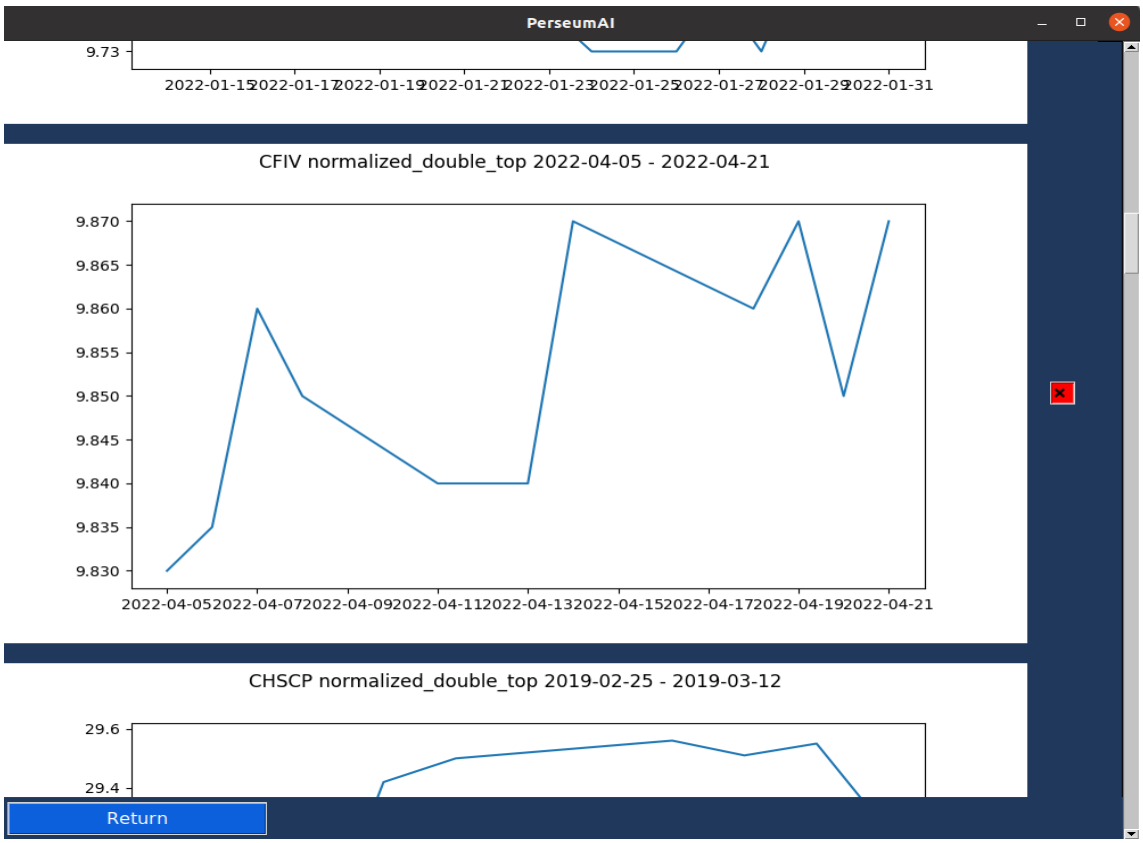


Figura 3.10: Ventana de patrones antiguos

Otra opción que puede elegir el usuario es mostrar los patrones que ocurren en el momento de ejecución de la aplicación. Esta ventana 3.11 muestra, al igual que la ventana de patrones antiguos, una ventana desplazable con varios gráficos de patrones que han sido encontrados en la fecha de realización de la búsqueda. Sin embargo, en esta ocasión, no es necesario indicar la tendencia porque en ese instante de tiempo no se puede saber si el objetivo se cumple o no.

Figura 3.11: Ventana de patrones actuales

3.7. Distribución de la aplicación

Debido a que Python requiere de la instalación de numerosas librerías en el ordenador del usuario para poder ejecutar esta aplicación, se planteó la creación de un archivo ejecutable, para facilitar su distribución, para que, tras descargarlo, el usuario pueda utilizarlo en cualquier equipo informático con sistema operativo *Linux* o *Windows 10*. La librería utilizada para este propósito fue *pyinstaller*. Con esta librería, simplemente añadiendo las opciones *–onefile* para que elimine el directorio TEMP tras la ejecución del empaquetado, la opción *–icon=PerseumAI.icon* para dotar al ejecutable de un icono, y *–hidden-import='PIL._tkinter_finder'*, para aportar las librerías necesarias para poder empaquetar Tkinter, se crea el directorio que contiene el ejecutable junto a los demás archivos necesarios para su correcto funcionamiento.

Los archivos ejecutables se encuentran en el repositorio principal de la aplicación y tienen un peso total de 538 MB. Debido al gran peso de los archivos compilados, fue necesaria la utilización de *GitHub LFS* para el almacenamiento de grandes archivos. La dirección de dicho repositorio es "`https://github.com/GJaubert/PerseumAI.git`".

3.8. Documentación de la aplicación

Como herramienta de software que puede seguir siendo mantenida y desarrollada por otras personas en un futuro, se documentó para proporcionar la información necesaria

para entender el código y el funcionamiento de la misma, de modo que se muestran todos los aspectos del código necesarios para estos propósitos. La documentación fue llevada a cabo con la librería *pdoc*. Este generador de documentación analiza los ficheros buscando bloques de texto en formato *pdoc* y los recopila y formatea.

Un ejemplo de bloque de documentación puede verse en la figura 3.12

```
def findCommonPattern(normalized_vector, all_patterns_dictionary):
    """Find the type of pattern for a given vector
        Args:
            normalized_vector (List[]): previous normalized
                vector containing prices
            all_patterns_dictionary (Dict{}): dictionary
                containing pattern types and prices
        Return:
            common_pattern_type (str): type of the pattern
            minimum_distance (float): minimum distance
                found between the best match and the vector
    """
    minimun_distance = BIG_NUMBER
    common_pattern_type = 'rest_normalized'
    for pattern_type in all_patterns_dictionary.keys():
      for single_pattern in all_patterns_dictionary[pattern_type]:
        current_distance = dtw_applier
          .comparePatterns(normalized_vector, single_pattern)
        if current_distance < minimun_distance:
          common_pattern_type = pattern_type
          minimun_distance = current_distance

    return common_pattern_type, minimun_distance
```

Figura 3.12: Código comentado con *pdoc*

Una vez recopilados todos los bloques de texto con este formato, crea varios ficheros *html* y vuelca esta información en ellos. Por último, crea un fichero *index.html* que sirve como inicio para la web de documentación.

3.8.1. Despliegue de la web con la documentación

La documentación fue desplegada utilizando *Github Pages*, ya que este servicio es gratuito. La dirección donde se puede encontrar la página web alojada es: `https://alu0101240374.github.io/PerseumAIDoc/`

Capítulo 4

Experimentación y resultados

Durante el desarrollo del proyecto fue necesario realizar numerosas pruebas explorando los diferentes valores para los parámetros del sistema con el fin de ofrecer los mejores resultados posibles. En este capítulo se mostrarán las pruebas realizadas junto a los resultados para justificar las decisiones tomadas respecto a la aplicación final.

4.1. Métricas usadas

Con el objetivo de calcular los resultados obtenidos según las distintas opciones y parámetros que pueden afectar al rendimiento de la herramienta, se creó un programa con el propósito de comparar patrones etiquetados previamente, para más tarde recopilar los resultados de dichas comparaciones, y obtener las respectivas métricas como precisión y exactitud. Para esta comparación, primero se deben obtener varias muestras de cada clase que se desee estudiar, en este caso, por cada tipo de patrón, doble techo, doble suelo, y triple techo, se escogieron 50 muestras. Para las pruebas se decidió utilizar *cross-validation* (Berrar, 2019). Este método consiste en realizar una prueba conformada por, en este caso, 25 iteraciones, escogiendo por cada iteración muestras de manera arbitraria para el conjunto de datos de entrenamiento, y el resto de muestras, para el conjunto de datos de prueba. Para este trabajo, la elección del número de muestras para el conjunto de entrenamiento fue de 15 muestras, por lo que el conjunto de prueba se conformó de 35 muestras. Una vez creados ambos conjuntos de datos, se comienzan a clasificar los patrones del conjunto de datos de prueba, utilizando el método *determinarTipoDePatrón*. Este método devuelve el tipo del patrón introducido por parámetro, así que el tipo de patrón devuelto se compara con la etiqueta original del mismo. Si es un acierto, se contabiliza. Cuando todos los patrones del conjunto de prueba hayan sido clasificados, se calculan las ya mencionadas métricas de precisión y exactitud. Para la precisión se utilizó la siguiente fórmula:

$$\frac{Verdaderos\ positivos\ de\ tipo\ T}{Verdaderos\ positivos\ de\ tipo\ T + Falsos\ positivos\ de\ tipo\ T}$$

Y para la exactitud se usó esta fórmula:

$$\frac{Verdaderos\ positivos\ de\ cualquier\ tipo}{Numero\ de\ pruebas\ total}$$

Debido a que se efectuaron varias iteraciones, y por cada iteración se escogieron conjuntos de datos de entrenamiento y de prueba diferentes, se realizó una media de los

resultados de precisión y exactitud.

4.2. Tamaño de ventana óptimo para la búsqueda de patrones antiguos

Tras desarrollar el sistema de pruebas para la herramienta, es posible comenzar con la experimentación de los diferentes parámetros. En este caso, el tamaño de ventana con el que recorreremos el histórico de precios, afecta de manera significativa a la capacidad de detectar patrones correctamente. Es por ello que un tamaño de ventana óptimo ayudará a obtener resultados más precisos.

El estudio del tamaño de ventana se realizó comenzando por probar el tamaño de ventana equivalente al tamaño del patrón más pequeño de la base de datos, 10, hasta terminar con una ventana equivalente al mayor tamaño de patrón, 136. Es importante recordar, que por cada tamaño de ventana a comprobar se realizaron 25 iteraciones de *cross-validation*. Estos fueron los resultados obtenidos de la exactitud con los tres tipos de patrones (doble techo, doble suelo y triple techo):

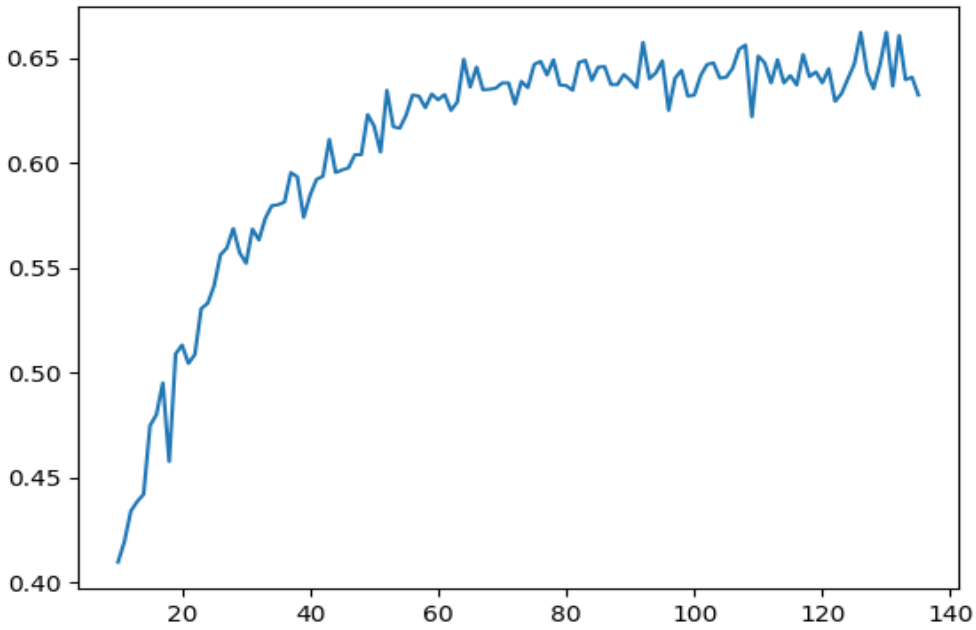


Figura 4.1: Resultados de la exactitud con todos los tipos de patrones

En el eje vertical de la figura 4.1 se representa la exactitud, y en el eje horizontal el tamaño de la ventana. Se observa cómo, a partir de un tamaño de ventana equivalente a 91, la exactitud no aumenta, incluso, en ocasiones disminuye. Si analizamos la precisión para cada tipo de patrón con los tamaños de ventana con mayor exactitud obtenemos lo siguiente:

Precisión obtenida				
Tamaño de ventana	**Exactitud**	**Doble T.**	**Doble S.**	**Triple T.**
91	63.5 %	76.7 %	88.5 %	45.6 %
108	65.6 %	76.4 %	89.7 %	46.9 %
126	66.2 %	77.7 %	85 %	55.3 %

Tabla 4.1: Precisión obtenida con triple techo como patrón relevante

Tras observar los resultados obtenidos de estas pruebas 4.1, se decidió realizar otro test, pero eliminando ahora el tipo de patrón ”triple techo”, así que ahora solo se sometieron a las pruebas los ”doble techo” y ”doble suelo”. La exactitud obtenida fue la siguiente:

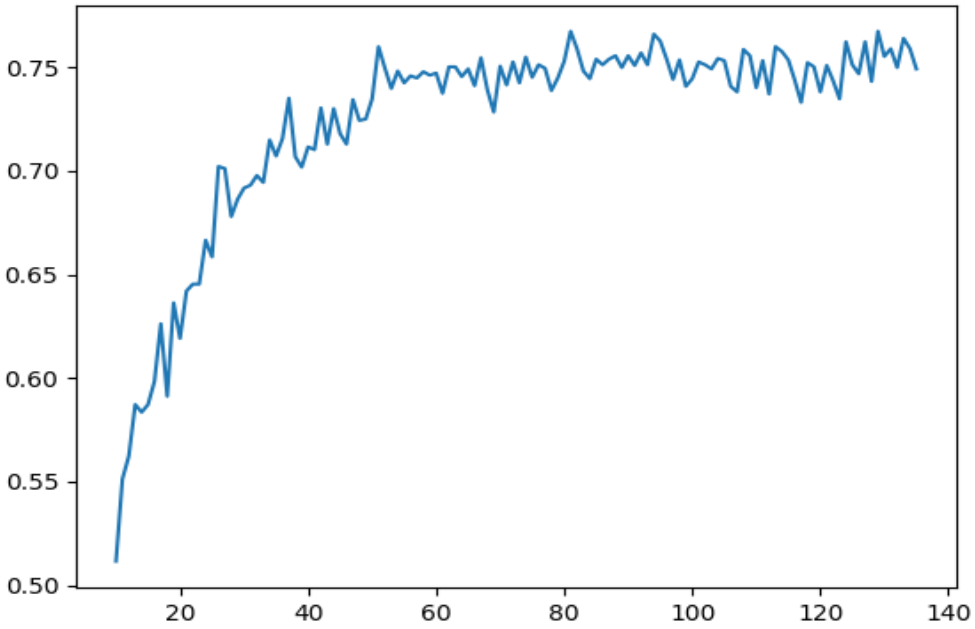


Figura 4.2: Resultados de la exactitud sin triples techos

Ahora, si analizamos la precisión en los casos con mayor exactitud, obtendremos los siguientes datos:

Precisión obtenida				
Tamaño ventana	**Exactitud**	**Doble Techo**	**Doble suelo**	**Falsos positivos**
51	76 %	88.9 %	90.2 %	48.2 %
81	76.7 %	87.8 %	92.5 %	49.4 %
94	76.6 %	94.4 %	86.1 %	46.6 %

Tabla 4.2: Precisión obtenida sin triple techo como patrón relevante

Se observa cómo, en ambos casos, la precisión por cada tipo de patrón es similar, sin embargo, la exactitud de la búsqueda que incluye todo tipo de patrones, es menor

que la búsqueda que excluye a los triples techos. Es por ello que, para este trabajo, con la finalidad de poder ofrecer unos datos probabilísticos más precisos, se optó por eliminar el tipo de patrón triple techo de la herramienta final. De este análisis obtenemos también el tamaño de ventana. Se observa que, tras haber eliminado los triples techos del análisis, cuando se supera el tamaño de ventana equivalente a 51 la exactitud aumenta y disminuye dentro de un rango fijo. Tras observar esto, se decidió escoger este tamaño para la herramienta final, ya que se pretende escoger un número lo menor posible, para eliminar todo el ruido posible dentro de la comparación entre dos series temporales. Como se observa, con este tamaño de ventana casi no se sacrifica rendimiento en cuanto a la detección de patrones de mayor tamaño.

4.3. Ventana para la búsqueda de patrones actuales

Como se menciona en la sección "Búsqueda patrones actuales", se necesita una ventana de tamaño variable para una correcta búsqueda. El resultado de esta operación es más preciso que utilizar el método mejorarVentana. En este caso, debido a que la búsqueda queda restringida a solamente patrones que ocurren en el momento en el que se efectúa la búsqueda, el tiempo requerido para dicha acción no es problemático. Algo que sí es relevante, es el tamaño inicial y final de esta ventana dinámica. Para este proyecto se propone asignar un valor para el tamaño de la ventana inicial equivalente al tamaño del mayor patrón que se encuentre dentro del conjunto de datos escogido para dicha búsqueda. El tamaño de la ventana disminuirá hasta se sea igual al tamaño del patrón más pequeño del ya mencionado conjunto de datos. Es importante recordar que por cada ejecución del programa, el conjunto de datos es diferente, por lo que el tamaño inicial y final pueden variar.

4.4. Divisores para la mejora de ventana

Para el método *mejorarVentana*, se necesita especificar los divisores que se quieren utilizar para fragmentar la ventana. La propuesta para este proyecto es dividir la ventana en 2, 3, 4 y 5 partes iguales. Dividir la ventana en más de cinco partes crearía patrones de tamaño menores a 11, ya que el tamaño escogido para la ventana original es 55, y según el análisis del tamaño de ventana óptimo, cuando el tamaño de la ventana es menor a 10, los resultados de la exactitud son muy bajos, alrededor del 27 %, por lo que dividir la ventana en más de 5 partes no proporcionaría buenos resultados.

4.5. Número de muestras en el conjunto de datos de búsqueda

Como se alude previamente en la subsección "Determinación del tipo de patrón", debemos tener un conjunto de datos conformado con patrones que sirvan como muestras para comparar con las series temporales a analizar. Para esta finalidad, se debe definir un parámetro del sistema que será el número de muestras a escoger para este conjunto

de datos. Para este trabajó se concluyó heurísticamente que 15 muestras es un buen compromiso tecnológico entre prestaciones y tiempo de cómputo.

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

El principal objetivo de este trabajo era desarrollar una herramienta que, utilizando búsqueda de patrones, fuera capaz de analizar automáticamente y calcular la probabilidad de que un patrón cumpliese su objetivo. Tras desarrollar la herramienta se confirma que los objetivos iniciales fueron cumplidos con éxito. La herramienta es capaz de detectar dos tipos de patrones, doble techo y doble suelo, y analizar sus valores alcanzados. Para esta tarea, se empleó el algoritmo *Dynamic Time Warping*, dando la posibilidad de comparar secuencias entre sí para encontrar patrones de tipo relevante. El rendimiento se evaluó con el método de *cross-validation* utilizando doscientas muestras de patrones previamente etiquetados, alcanzando una exactitud del setenta y seis por ciento. Gracias a esto es posible ofrecer al usuario la probabilidad exacta de que cada uno de los dos tipos de patrones reconocidos por la aplicación, alcancen los valores esperados. También se consiguió detectar patrones de tipo doble techo y doble suelo que acaban de formarse en el momento de la ejecución del programa. Conociendo estos patrones, el usuario de esta herramienta puede aprovechar las probabilidades previamente obtenidas de que el objetivo para los distintos tipos de patrones se cumple. Además, para poder llevar a cabo estas tareas, fue necesario crear un conjunto de datos para el cual se buscaron doscientas muestras, que podrán ser utilizadas por cualquier investigador que quiera seguir desarrollando y perfeccionando este proyecto. Estas muestras se pueden encontrar en el repositorio oficial de la herramienta: `https://github.com/gjaubert/PerseumAI`. Por otro lado, la creación de una interfaz gráfica, así como la documentación y creación de archivos ejecutables para *Linux* y *Windows 10*, facilitan el uso y distribución de la aplicación. Tras haber concluido este trabajo, dar soporte inteligente a los inversores y analistas para reducir la incertidumbre respecto al objetivo de ciertos tipos de patrones en bolsa, se vuelve una realidad.

5.2. Líneas de trabajo futuras

Como puede ocurrir en algunos proyectos que emplean la inteligencia artificial, experimentar cada parámetro del sistema, así como poseer un conjunto de datos lo suficientemente grande para obtener resultados altamente eficientes, es complejo. Para este trabajo en concreto, una línea futura podría ser la ampliación del conjunto de datos, para así poder obtener resultados más representativos en cuanto las pruebas realizadas o para

el conjunto de patrones utilizado en la búsqueda en sí. Por otra parte, se invita a quienes quieran ampliar el alcance de este proyecto, añadir más tipos de patrones al motor de detección de esta herramienta. De este modo, es posible nutrir mejor a los usuarios que utilicen la aplicación con más información acerca del mercado bursátil. Asimismo, una línea de investigación futura podría abordar diferentes extensiones del algoritmo *Dynamic Time Warping*. Un ejemplo de mejora podría ser la que proponen Yang et al. (2022). Esta mejora se basa en utilizar estrategias de paralelización para las secuencias largas, permitiendo así estudiar ventanas de longitud mayor optimizando los tiempos de computación. Por último, sería positivo realizar distintas pruebas en profundidad acerca de parámetros del sistema mencionados a lo largo de la memoria, ya que algunos fueron estudiados de manera heurística, como por ejemplo, el número de muestras empleadas para la búsqueda de la herramienta. En referencia a este último punto, aunque en el estado actual de la herramienta no se observó que fuera ningún problema, una optimización del tiempo de ejecución de la herramienta conllevaría un efecto indudablemente práctico para los usuarios.

Capítulo 6

Summary and Conclusions

The main purpose of this work was the development of a tool that, using pattern search, was able to analyse and probabilistically calculate the direction of these patterns. After developing the tool, it is confirmed that the initial objectives were successfully achieved. The tool is able to detect two types of patterns, double-top and double-bottom, and analyse their values reached. For this task, the Dynamic Time Warping algorithm was used, giving the possibility to compare sequences with each other to find patterns of relevant type. The performance was evaluated with the cross-validation method using two hundred samples of previously labelled patterns, achieving an accuracy of seventy-six percent. Thanks to this, it is possible to offer the user the probability that the two types of patterns recognised by the application reach the expected values. It was also possible to detect double-top and double-bottom patterns that have finished forming at the time of the programme's execution. Knowing these patterns, the user can take advantage of the probabilities previously obtained of the possible directions that the different types of patterns can take, for their own benefit. In addition, in order to carry out these tasks, it was necessary to create a dataset of two hundred samples, which can be used by any researcher who wants to further develop and refine this project. These samples can be found in the official repository of the tool: `https://github.com/gjaubert/PerseumAI`. On the other hand, the creation of a graphical interface, as well as the documentation and creation of executable files for *Linux* and *Windows 10*, facilitate the use and distribution of the application. After having completed this work, providing intelligent support to investors and analysts to reduce uncertainty regarding the targeting of certain types of stock market patterns becomes a reality.

Capítulo 7

Presupuesto

En la elaboración de este trabajo están incluidos los siguientes costes.

7.1. Costes materiales

Tipos	Descripción	Coste
Equipo informático utilizado para el desarrollo	Ordenador, pantalla, teclado, ratón...	1500€
GitHub LFS	Almacenamiento	5€/mes
Total		1505€

Tabla 7.1: Resumen de costes materiales

7.2. Costes de horas de trabajo

Tipos	Horas	Coste	Total
Investigación	45	35	1575€
Creación del conjunto de datos	75	30	2250€
Detección de patrones	120	35	4200€
Front-end	45	32	1440€
Documentación	15	30	450
Total			9915€

Tabla 7.2: Resumen de horas trabajadas y sus costes

Bibliografía

Achelis, S. B. (2001). Technical analysis from a to z.

Alizadeh, E. (2020). An illustrative explanation to dynamic time warping. https://towardsdatascience.com/an-illustrative-introduction-to-dynamic-time-warping-36aa98513b98, Last accessed on 2022-06-03.

Aroussi, R. (2019). Yfinance. https://pypi.org/project/yfinance/, Last Accessed on 2022-06-03.

Bagnall, A. J., Bostrom, A., Large, J., and Lines, J. (2016). The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version. *CoRR*, abs/1602.01711.

Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press.

Berrar, D. (2019). Cross-validation.

Bogousslavsky, V. and Muravyev, D. (2019). Should we use closing prices? institutional price pressure at the close. *Work. Pap., Carroll School Manag., Boston*.

Coelho, M. S. (2012). *Patterns in financial markets: Dynamic time warping*. PhD thesis, NSBE-UNL.

Duris, J. (2007). Finviz.com. https://finviz.com/, Last accessed 2022-06-17.

Edwards, R. D. and Magee, J. (1948). *Análisis técnico de tendencias bursátiles*. John Magee Inc., Springfield, MA, EE.UU ., 11 edition.

Francisco López, J. (2018). Analisis chartista.

Giorgino, T. (2009). Computing and visualizing dynamic time warping alignments in r: The dtw package. *Journal of Statistical Software*, 31(7):1–24.

Grzejszczak, T., Probierz, E., Galuszka, A., Simek, K., and Jędrasiak, K. (2022). Dynamic time warping in financial data – modification of algorithm in context of stock market similarity analysis. *EUROPEAN RESEARCH STUDIES JOURNAL*, XXV:967–979.

Hayes, A. (2022). Introduction to technical analysis price patterns. https://www.investopedia.com/articles/technical/112601.asp, Last accessed 2022-06-03.

Kim, S. H., Lee, H. S., Ko, H. J., Jeong, S. H., Byun, H. W., and Oh, K. J. (2018). Pattern matching trading system based on the dynamic time warping algorithm. *Sustainability*, 10(12):4641.

Mendelsohn, L. B. (2021). Artificial intelligence trading software. `https://www.vantagepointsoftware.com/`, Last accessed 2022-06-03.

Munkhdalai, L., Munkhdalai, T., Park, K. H., Lee, H. G., Li, M., and Ryu, K. H. (2019). Mixture of activation functions with extended min-max normalization for forex market prediction. *IEEE Access*, 7:183680–183691.

Porté, O. and Zheng, S. (2017). Alpha vantage api documentation. `https://www.alphavantage.co/documentation/`, Last accessed on 2022-06-03.

Raudys, A., Lenčiauskas, V., and Malčius, E. (2013). Moving averages for financial data smoothing. In *International conference on information and software technologies*, pages 34–45. Springer.

Shirota, Y. and Murakami, A. (2021). Long-term time series data clustering of stock prices for portfolio selection. In *2021 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, page 1–6. IEEE Press.

Umer, M., Awais, M., and Muzammul, M. (2019). Stock market prediction using machine learning (ml) algorithms. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 8(4):97–116.

Yang, D., Shaw, T., and Tsai, T. (2022). A study of parallelizable alternatives to dynamic time warping for aligning long sequences. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:2117–2127.