Máster en Matemática Computacional

Departamento de Matemáticas.

Universidad Jaume I.

Trabajo Fin de Máster

Inteligencia Artificial aplicada a la Microestructura de los Mercados Financieros

Realizado por Francisco J. Luguera Parúas y dirigido por Pablo Gregori Huerta

Septiembre 2023

Agradecimientos

Quisiera transmitir mi más sincero agradecimiento a todos los que me han ayudado durante esta etapa y han colaborado de una u otra forma en esta investigación.

En especial, a mi mujer, quién ha vivido día a día el esfuerzo y dedicación sobre este trabajo, apoyando en todo momento y haciendo que cada palabra cuente.

En segundo lugar, a mi familia, quien sin ella, nunca hubiera adoptado los valores de perseverancia y esfuerzo que me han inculcado desde pequeño.

A mi tutor, Pablo Gregori Huerta, por su tiempo y confianza sobre la temática de dicho trabajo.

A todos ellos, mil gracias.

Índice general

Lista de figuras **V**

Lista de tablas **IX**

1. Introducción **1**

2. Aprendizaje Automático **5**

2.1. Máquinas de Soporte Vectorial 5

2.1.1. Margen y Hard-SVM 6

2.1.2. Soft-SVM y regularización de las normas 9

2.1.3. Condiciones de optimización y Vectores de Soporte . . 10

2.1.4. Dualidad 10

2.1.5. Método Kernel 12

2.1.5.1. Incrustaciones en espacios característicos . . . 12

2.1.5.2. El truco del Kernel 13

2.2. Redes Neuronales 16

2.2.1. El Perceptron 16

2.2.2. Estructura de la Red Neuronal 18

2.2.3. Entrenamiento de la Red Neuronal 21

2.2.3.1. Funciones de Coste 22
2.2.3.2. Algoritmo de Descenso de Gradiente 22
2.3. Algoritmo de Bosque Aleatorio 23
2.3.1. Algoritmo de Árbol de Decisión 24
2.3.1.1. Entropía 25
2.3.1.2. Ganancia de Información 26
2.3.2. Clasificación final con Bosque Aleatorio 27

3. Caso de Estudio **29**
3.1. Datos 29
3.2. Pre-Procesado de datos 31
3.2.1. Selección de Características y Creación de las Clases a Predecir 34
3.3. Análisis Exploratorio de Datos 37
3.4. Reducción de la Dimensión 45
3.5. Aprendizaje Automático 48
3.5.1. Máquina de Soporte Vectorial (SVM) 52
3.5.2. Red Neuronal 55
3.5.3. Bosque Aleatorio 58
3.6. Conclusión Final 60

A. Rendimiento en la Clasificación **65**
A.1. SVM 65
A.2. Red Neuronal 68
A.3. Bosque Aleatorio 71

Bibliografía **75**

Índice de figuras

2.1. Diagrama de una neurona. 17

2.2. Estructura de una red neuronal. Fuente Wikipedia. 19

2.3. Árbol de decisión. Fuente: https://www.javatpoint.com. 24

2.4. Random forest. Fuente: https://www.javatpoint.com. 27

3.1. Representación de 1000 Precios Bid/Ask agregados a 60 segundos. . . . 31

3.2. Representación de 1000 Volúmenes Bid/Ask agregados a 60 segundos. . . 32

3.3. Representación del Número de Transacciones Bid/Ask agregados a 60 segundos. 33

3.4. Porcentaje de precios vacíos (Bid/Ask) por día de la semana. 34

3.5. Representación del desbalanceo entre las distintas clases. 38

3.6. Diagrama de cajas e Histograma de la variable de retorno en diferentes periodos. 40

3.7. Diagrama de cajas e Histograma de la variable de transacciones en diferentes periodos. 42

3.8. Diagrama de cajas e Histograma de la variable de volúmenes en diferentes periodos. 43

3.9. Matriz de correlación entre cada una de las variables Bid. 44

3.10. Número de componentes según la información de la varianza explicada. . 46
3.11. Número de componentes según la información de los autovalores. 46
3.12. Validación cruzada K=5. Fuente: www.scikit-learn.com. 49
3.13. Representación de una matriz de confusión. 50
3.14. Tiempos medios de entrenamiento para cada clasificador en segundos. . . 63

A.1. Resultados de la clasificación de la SVM para tiempos de agregación de 60 segundos y ventanas de predicción de 300 y 120 segundos respectivamente. 65
A.2. Resultados de la clasificación de la SVM para tiempos de agregación de 60 y 30 segundos y ventanas de predicción de 60 y 150 segundos respectivamente. 66
A.3. Resultados de la clasificación de la SVM para tiempos de agregación de 30 segundos y ventanas de predicción de 60 y 30 segundos respectivamente. 66
A.4. Resultados de la clasificación de la SVM para tiempos de agregación de 15 segundos y ventanas de predicción de 75 y 30 segundos respectivamente. 67
A.5. Resultados de la clasificación de la SVM para tiempos de agregación de 15 y 5 segundos y ventanas de predicción de 15 y 25 segundos respectivamente. 67
A.6. Resultados de la clasificación de la SVM para tiempos de agregación de 5 segundos y ventanas de predicción de 10 y 5 segundos respectivamente. 68
A.7. Resultados de la clasificación de la ANN para tiempos de agregación de 60 segundos y ventanas de predicción de 300 y 120 segundos respectivamente. 68
A.8. Resultados de la clasificación de la ANN para tiempos de agregación de 60 y 30 segundos y ventanas de predicción de 60 y 150 segundos respectivamente. 69
A.9. Resultados de la clasificación de la ANN para tiempos de agregación de 30 segundos y ventanas de predicción de 60 y 30 segundos respectivamente. 69

A.10. Resultados de la clasificación de la ANN para tiempos de agregación de 15 segundos y ventanas de predicción de 75 y 30 segundos respectivamente. 69
A.11. Resultados de la clasificación de la ANN para tiempos de agregación de 15 y 5 segundos y ventanas de predicción de 15 y 25 segundos respectivamente. 70
A.12. Resultados de la clasificación de la ANN para tiempos de agregación de 5 segundos y ventanas de predicción de 10 y 5 segundos respectivamente. 70
A.13. Resultados de la clasificación de RF para tiempos de agregación de 60 segundos y ventanas de predicción de 300 y 120 segundos respectivamente. 71
A.14. Resultados de la clasificación de RF para tiempos de agregación de 60 y 30 segundos y ventanas de predicción de 60 y 150 segundos respectivamente. 71
A.15. Resultados de la clasificación de RF para tiempos de agregación de 30 segundos y ventanas de predicción de 60 y 30 segundos respectivamente. 72
A.16. Resultados de la clasificación de RF para tiempos de agregación de 15 segundos y ventanas de predicción de 75 y 30 segundos respectivamente. 72
A.17. Resultados de la clasificación de RF para tiempos de agregación de 15 y 5 segundos y ventanas de predicción de 15 y 25 segundos respectivamente. 73
A.18. Resultados de la clasificación de RF para tiempos de agregación de 5 segundos y ventanas de predicción de 10 y 5 segundos respectivamente. . 73

Índice de tablas

3.1. Evolución del número de componentes a través de las diferentes agregaciones temporales y ventanas de predicción 47
3.2. Evolución de los hiperparámetros de la SVM a través de las diferentes agregaciones temporales y ventanas de predicción . 54
3.3. Evolución de los hiperparámetros del perceptron multicapa a través de las diferentes agregaciones temporales y ventanas de predicción 57
3.4. Evolución de los hiperparámetros del RF a través de las diferentes agregaciones temporales y ventanas de predicción 60
3.5. Evolución de las precisiones de cada clasificador a través de las diferentes agregaciones temporales y ventanas de predicción 62

Capítulo 1

Introducción

El mercado de divisas se destaca como el mercado financiero más extenso a nivel mundial, con un volumen de transacciones diarias que supera los 5 trillones de dólares. En su mayoría, estas transacciones se efectúan a través de sistemas de comunicación electrónica (ECNs),los cuales permiten la ejecución de órdenes de compra y venta de instrumentos financieros al brindar a los participantes acceso directo a los precios y a las condiciones de mercado en tiempo real. Esto implica que los comerciantes pueden observar las mejores ofertas disponibles y tomar decisiones comerciales informadas.

El campo del aprendizaje automático y la inteligencia artificial (IA) ha avanzado significativamente en la última década. Durante este periodo, ha habido progresos notables en varios aspectos clave como por ejemplo; se han desarrollado algoritmos de aprendizaje automático más avanzados y eficientes impulsando mejoras en diversas tareas (procesamiento de lenguaje natural, visión por computadora, reconocimiento de voz), el aumento en la potencia de cómputo, incluyendo el desarrollo de unidades de procesamiento gráfico

(GPU) especializadas para tareas de IA, la disponibilidad de grandes conjuntos de datos etiquetados, siendo fundamental para el éxito de muchos modelos de IA, empresas e instituciones han invertido considerablemente en investigación y desarrollo de IA, lo que ha llevado a una adopción de estas tecnologías en diversas industrias, entre ellas, el sector financiero (predicción de mercados, riesgos, composición de carteras de valores...).

Una teoría ampliamente aceptada en la economía y las finanzas es la Hipótesis del Mercado Eficiente (HME), desarrollada en 1970 por Eugene Fama [4]. Esta hipótesis sostiene que los precios de los activos financieros ya incorporan toda la información pública y, por lo tanto, reflejan su valor real en todo momento. Sin embargo, hay estudios que contradicen esta teoría. El estudio de Christopher Neely et al.[11] encuentra rendimientos positivos tras utilizar técnicas de programación genética para encontrar reglas de comercio técnico en seis intercambios de divisas utilizando una muestra de datos desde 1981 hasta 1995.

Otro estudio de Tsong-Wuu Yu et al.[15] crea una estrategia de comercio utilizando redes neuronales artificiales (ANN) sobre datos diarios del índice S&P500. Los rendimientos conseguidos fueron de 11.98 % anualizado por lo que refuta considerablemente la hipótesis.

Wei Huang et al. [7] utiliza las Máquinas de Soporte Vectorial (SVM's) para estudiar el grado de predicción de los movimientos de mercado financieros utilizando datos semanales del índice Nikkei 225, comparando los resultados con un análisis discriminante lineal, análisis discriminante cuadrático y una red neuronal Elman de retropropagación. El experimento muestra que la SVM tiene mejores resultados que los otros métodos de clasificación.

En el estudio realizado por Tristan Fletcher et al.[5] fueron construidos un conjunto de kernels, individualmente y simultáneamente sobre el tipo de cambio EURUSD, para entrenar una SVM multiclase con la intención de predecir la dirección de movimientos de precios. Los métodos kernel utilizados superan los resultados económicos de una estrategia de seguimiento de tendencia.

Artículos más recientes como Yue Zhang et al.[2] sugieren la utilización de redes neuronales profundas junto con datos procedentes de noticias económicas para medir su influencia en los mercados financieros, tanto para el corto como largo plazo, sobre los movimientos de acciones bursátiles. El estudio logra un 6 % de mejoras sobre las rentabilidades del índice S&P500.

En resumen, estos estudios son solo algunos ejemplos que sugieren la presencia de patrones en los mercados financieros, lo que cuestiona la validez de la Hipótesis de Mercado Eficiente (HME).

El propósito principal de este trabajo no se limita a la predicción de un valor, sino que se enfoca en el seguimiento de las predicciones a lo largo de diferentes intervalos de tiempo. En consecuencia, el objetivo fundamental es identificar patrones y características específicas del activo financiero bajo estudio. Para llevar a cabo este análisis, nos basaremos en un estudio previo realizado por Aleksandar Palikuca et al.[12] donde emplearon una combinación de múltiples redes neuronales y máquinas de soporte vectorial para realizar una clasificación múltiple en intervalos de predicción de 5, 20 y 60 segundos utilizando datos de un segundo del mercado de divisas EURUSD.

El propósito central de dicha investigación es evaluar la precisión de cada una de las combinaciones de modelos. Los resultados resaltan que un modelo

compuesto exhibe niveles prometedores de precisión. Además, se lleva a cabo una simulación de operaciones en el mercado financiero, sin tener en cuenta los costos de transacción, lo que arroja un rendimiento anualizado que alcanza hasta el 53 %.

Nuestra metodología, como se mencionó anteriormente, tiene como objetivo descubrir las características subyacentes del activo que estamos analizando. Llevamos a cabo un análisis exhaustivo de manera individual, evaluando la precisión de varios algoritmos de clasificación, teniendo en cuenta que trabajamos con datos de alta frecuencia y mantenemos un enfoque pragmático en cuanto a la eficiencia computacional necesaria para su uso. Por ello, utilizamos diferentes metodologías sobre los datos que nos permitan liberar exigencias computacionales como la reducción de la dimensionalidad, el tratamiento de datos vacíos, el estudio de la distribución de las variables en cada agregación temporal creada, el tratamiento de valores atípicos y las relaciones de las características con la variable objetivo. Hemos de notar, que algunas variables iniciales no las obtenemos de forma directa, siendo necesaria su creación precisa, dado el impacto que tienen sobre cada uno de los modelos. Por todo ello, se ha decidido analizar tres algoritmos de aprendizaje automático: máquina de soporte vectorial, red neuronal y random forest con datos de 60, 30, 15, y 5 segundos y de esta forma, evaluar su precisión entre diferentes horizontes temporales. De esta forma, extraemos la información sobre la complejidad de cada uno de los modelos en diferentes datos, gracias a la evolución de sus parámetros.

Capítulo 2

Aprendizaje Automático

En este capítulo vamos a esbozar teóricamente los distintos algoritmos de Aprendizaje Automática que aplicaremos en el caso de estudio. Cabe destacar que nos centraremos en la Máquina de Soporte Vectorial (SVM), la Red Neuronal y el Bosque Aleatorio (RF).

2.1. Máquinas de Soporte Vectorial

Para la elaboración de esta sección se tuvo en cuenta el libro de Robert Tibshirani[8], libro de Shai Shalev[14], el artículo de Vapnik et al.[1] y Wikipedia principalmente.

Una técnica de aprendizaje automático ampliamente reconocida, desarrollada principalmente por Vladimir Vapnik y su colaborador Corinna Cortes [1], es la Máquina de Soporte Vectorial (SVM). Vladimir Vapnik es un destacado científico de la informática y estadístico que trabajó en AT&Bell Labs y posteriormente en otras instituciones académicas y de investigación.

La Máquina de Soporte Vectorial (SVM) es una técnica de aprendizaje supervisado ampliamente empleada para llevar a cabo tareas de clasificación y regresión. Su objetivo principal consiste en identificar un hiperplano óptimo de separación (en el caso de la clasificación binaria) o un hiperplano óptimo de regresión (en el caso de la regresión) que maximice la distancia entre las diferentes clases o puntos de datos. Los puntos de datos que se encuentran próximos a este hiperplano se denominan *vectores de soporte*.

Las SVM originalmente se diseñaron para la clasificación binaria, pero con el tiempo se han adaptado a resolver problemas de clasificación multiclase y regresión. En términos generales, funcionan mediante la introducción de una matriz $\mathcal{X}$ de dimensiones $N \times D$ que contienen datos y una matriz $\mathcal{Y}$ de dimensiones $N \times 1$ con valores $+1$ y -1 que representan las dos clases. Estas SVM buscan separar estas clases utilizando un hiperplano, donde N es el número de muestras y D es el número de características en los datos.

La obtención del hiperplano se realiza mediante la resolución de un problema de optimización convexa, asegurando que cualquier óptimo local sea también un óptimo global. Una vez obtenido, este hiperplano se utiliza para clasificar nuevas entradas x^* de dimensión $1 \times D$.

2.1.1. Margen y Hard-SVM

Sea $S = (x_1, y_1), ..., (x_m, y_m)$ un conjunto de entrenamiento donde cada $x_i \in \mathbb{R}^d$ y $y_i \in \{\pm 1\}$. Decimos que este conjunto de entrenamiento es linealmente separable si existe un semiespacio, (w, b), tal que $y_i = sign(\langle w, x_i \rangle + b)$ para todo i. Alternativamente, esta condición puede ser escrita como

$$\forall i \in [m]\,, \;\; y_i(\langle w, x_i\rangle + b) > 0. \tag{2.1}$$

Todos los semiespacios (w, b) que satisfacen esta condición son hipótesis de Minimización del Riesgo Empírico (ERM - el error de clasificación es cero). Para cualquier conjunto de entrenamiento en la muestra, hay muchos semiespacios de ERM.

Para seleccionar un semiespacio, evaluamos el margen del hiperplano. El margen se define como la distancia más corta entre un punto en el conjunto de entrenamiento y el hiperplano.

El Hard-SVM es una regla de aprendizaje que busca devolver un hiperplano ERM que logre la máxima separación con el conjunto de entrenamiento. Para definir formalmente, comenzamos por expresar la distancia entre un punto x y un hiperplano utilizando los parámetros que caracterizan el semiespacio.

Definición. La distancia entre un punto x y el hiperplano definido por (w, b) donde $||w|| = 1$ es $|\langle w, x\rangle + b|$.

Según la definición anterior, el punto más cercano del conjunto de entrenamiento al hiperplano de separación es $min_{i\in[m]}|\langle w, x_i\rangle + b|$ Por lo tanto, la regla Hard-SVM es

$$\underset{(w,b):||w||=1}{\operatorname{argmax}} \; \underset{i\in[m]}{\operatorname{mín}} |\langle w, x_i\rangle + b| \;\; \forall i, \; y_i(\langle w, x_i\rangle + b) > 0. \tag{2.2}$$

Cuando haya una solución disponible (en el caso de que los datos sean separables), podemos formular un problema equivalente de la siguiente ma-

nera:

$$\underset{(w,b):||w||=1}{\operatorname{argmax}} \underset{i\in[m]}{\operatorname{mín}} y_i(|\langle w, x_i\rangle + b|). \tag{2.3}$$

A continuación, vamos a dar una formulación equivalente de Hard-SVM como un problema de optimización cuadrático:

1. Las entradas al modelo son $(x_1, y_1), ..., (x_m, y_m)$.

2. La ecuación es:

$$(w_0, b_0) = \underset{(w,b)}{\operatorname{argmin}} ||w||^2, \ \ \forall i, y_i(\langle w, x_i\rangle + b) \geq 1. \tag{2.4}$$

3. Como salida obtenemos $\hat{w} = \frac{w_0}{||w_0||}$, $\hat{b} = \frac{b_0}{||w_0||}$.

El siguiente lema demuestra que la salida Hard-SVM es el hiperplano separador con el margen más grande. En términos simples, Hard-SVM busca un vector w que minimice su norma, asegurando que $|\langle w, x_i\rangle| \geq 1$ para todos los puntos de datos x_i. En esencia, estamos escalando el margen a una unidad de uno, y la tarea de encontrar el semiespacio de margen más grande se reduce a encontrar el vector w con la norma mínima. Formalmente:

Lema 1. La salida de Hard-SVM es una solución de la ecuación

$$\underset{(w,b):||w||=1}{\operatorname{argmax}} \underset{i\in[m]}{\operatorname{mín}} y_i(\langle w, x_i\rangle + b)$$

.

2.1.2. Soft-SVM y regularización de las normas

La formulación Hard-SVM asume que el conjunto de entrenamiento es linealmente separable, la cual es una suposición fuerte. Soft-SVM puede verse como una relajación de la regla Hard-SVM, que se puede aplicar incluso si el conjunto de entrenamiento no es linealmente separable.

Soft-SVM es una versión más flexible de Hard-SVM, diseñada para lidiar con conjuntos de entrenamiento que no son necesariamente linealmente separables. mientras que Hard-SVM asume la separabilidad lineal, Soft-SVM permite ciertas violaciones de esta suposición.

En lugar de imponer restricciones estrictas, como en Hard-SVM, que requiere que $y_i\left(\langle w, x_i\rangle + b\right) \geq 1$ para todos los puntos de datos, Soft-SVM introduce variables de holgura no negativas $\xi_1, ..., \xi_m$. Estas variables miden cuánto se violan las restricciones. En Soft-SVM, se busca minimizar tanto la norma de w (correspondiente al margen) como el promedio de ξ_i (correspondiente a las violaciones de las restricciones). La compensación entre estos dos términos se controla mediante el parámetro λ. Esto da lugar a la formulación de Soft-SVM:

1. Las entradas $(x_1, y_1), ..., (x_m, y_m)$.

2. Parámetro $\lambda > 0$.

3. Ecuación:

$$\min_{w,b,\xi}\left(\lambda||w||^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i\right) \forall i,\ \ y_1\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i. \tag{2.5}$$

4. Como salidas obtenemos w, b

2.1.3. Condiciones de optimización y Vectores de Soporte

La denominación *Máquina de Soporte Vectorial* proviene del hecho de que la solución de Hard-SVM, representada como w_0, está relacionada con los ejemplos que se encuentran exactamente a una distancia de $1/||w_0||$ del hiperplano separador. Estos ejemplos se llaman *vectores de soporte*. Esta relación se fundamenta en las condiciones de optimización de Fritz John.

Teorema. Sea w_0 como se define en la ecuación (2.7) y sea $I = \{i : | \langle w_0, x_i \rangle | = 1\}$. Entonces, existen coeficientes $\alpha_1, ..., \alpha_m$ tal que

$$w_0 = \sum_{i \in I} \alpha_i x_i. \tag{2.6}$$

Los ejemplos $\{x_i : i \in I\}$ son llamados *vectores de soporte*.

2.1.4. Dualidad

Históricamente, las propiedades de la Máquina de Soporte Vectorial se han investigado principalmente a través del análisis del dual de la ecuación,

$$\min_w ||w||^2, \ \ \forall i, y_i \langle w, x_i \rangle \geq 1 \tag{2.7}$$

Para completar, mostramos como derivar el dual de la ecuación (2.7).

Comenzamos reescribiendo el problema en una forma equivalente de la siguiente forma. Considerar la función

$$g(w) = \max_{\alpha \in \mathbb{R}^m : \alpha \geq 0} \sum_{i=1}^{m} \alpha_i (1 - y_i \langle w, x_i \rangle) = \begin{cases} 0 & \text{si } \forall i, y_i \langle w, x_i \rangle \geq 1 \\ \infty & \text{otro caso} \end{cases} \tag{2.8}$$

Así que podemos reescribir la ecuación (2.7) como,

$$\min_{w} \left(||w||^2 + g(w) \right). \tag{2.9}$$

Reordenando lo anterior, obtenemos que la ecuación (2.7) puede ser reescrita como el problema

$$\min_{w} \max_{\alpha \in \mathbb{R}^m : \alpha \geq 0} \left(\frac{1}{2} ||w||^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i \langle w, x_i \rangle\right) \right). \tag{2.10}$$

Ahora supongamos que invertimos el orden de mínimo y máximo en la ecuación anterior. Esto sólo puede hacer decrecer el valor objetivo, y nosotros tenemos,

$$\begin{aligned} &\min_{w} \max_{\alpha \in \mathbb{R}^m : \alpha \geq 0} \left(\frac{1}{2} ||w||^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i \langle w, x_i \rangle\right) \right) \\ \geq &\max_{\alpha \in \mathbb{R}^m : \alpha \geq 0} \min_{w} \left(\frac{1}{2} ||w||^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i \langle w, x_i \rangle\right) \right) \end{aligned}$$

La desigualdad anterior se llama dualidad débil. Resulta que en nuestro caso, también se mantiene una fuerte dualidad, es decir, la desigualdad se cumple con la igualdad. Por lo tanto, el problema dual es

$$\max_{\alpha \in \mathbb{R}^m : \alpha \geq 0} \min_{w} \left(\frac{1}{2} ||w||^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i \langle w, x_i \rangle\right) \right). \tag{2.11}$$

Podemos simplificar el problema dual notando que una vez que α es fijada, el problema de optimización con respecto a w no tiene restricciones y el objetivo es diferenciable; por lo tanto, en el punto óptimo, el gradiente es igual a cero:

$$w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \Longrightarrow w = \sum_{i=1}^{m} \alpha_i y_i x_i. \tag{2.12}$$

Esto nos muestra que la solución debe estar en el lapso lineal de los ejemplos. Conectando lo anterior a la ecuación (2.11) obtenemos que el problema dual puede ser reescrito como

$$\max_{\alpha \in \mathbb{R}^m : \alpha \geq 0} \left(\frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i x_i \right\|^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i \left\langle \sum_j \alpha_j y_j x_j, x_i \right\rangle \right) \right). \tag{2.13}$$

Reorganizando términos obtenemos el problema dual

$$\max_{\alpha \in \mathbb{R}^m : \alpha \geq 0} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \left\langle x_j, x_i \right\rangle \right). \tag{2.14}$$

2.1.5. Método Kernel

Un kernel es una medida de similitud entre instancias que se pueden interpretar como un producto interno en un espacio de Hilbert donde las instancias se incorporan de manera virtual.

2.1.5.1. Incrustaciones en espacios característicos

Para hacer que los semiespacios sean más efectivos en la captura y representación de información, se puede relacionar el espacio original de las instancias con otro espacio, posiblemente de mayor dimensión, y luego aprender

un semiespacio en ese nuevo espacio. En lugar de aprender directamente en la representación original, se introduce una función $\psi : \mathbb{R} \longrightarrow \mathbb{R}^2$,

$$\psi(x) = (x, x^2).$$

Utilizamos el término de espacio característico para denotar el rango de ψ. Después de aplicar ψ, los datos se puede explicar fácilmente utilizando el semiespacio $h(x) = sign(\langle w, \psi(x)\rangle - b)$.

El paradigma básico sería:

1. Dado algún conjunto de dominio χ y una tarea de aprendizaje, elegimos una función $\psi : \chi \longrightarrow \mathcal{F}$ para un espacio característico $\mathcal{F}$ que normalmente será $\mathbb{R}^n$ para algún n.

2. Dada una secuencia de ejemplos etiquetados, $S = (x_1, y_1), ..., (x_m, y_m)$, crear la secuencia de imagen $\hat{S} = (\psi(x_1), y_1), ..., (\psi(x_m), y_m)$.

3. Entrenar un predictor lineal h sobre $\hat{S}$.

4. Predecir la etiqueta de un punto test, x, a ser $h(\psi(x))$.

El éxito de este paradigma de aprendizaje depende de escoger un buen ψ.

2.1.5.2. El truco del Kernel

Al aumentar la dimensión del espacio característico, los semiespacios se vuelven más efectivos en la captura y representación de la información, pero también aumenta la complejidad computacional. Calcular separadores lineales en espacios de alta dimensión puede ser costoso. Para abordar este

problema, se utiliza el aprendizaje basado en kernel, donde el término *kernel* se refiere a productos internos en el espacio característico. Esto permite implementar separadores lineales en espacios de alta dimensión sin necesidad de especificar puntos en ese espacio o definir la incrustación ψ de manera explícita.

Además, hemos observado anteriormente que regularizar la norma de w puede ayudar a lidiar con la complejidad de muestras incluso en espacios de alta dimensión. Sorprendentemente, la regularización de la norma de w también puede ser beneficiosa para abordar problemas computacionales. Todas las formulaciones del problema SVM que hemos discutido hasta ahora son instancias de un problema general:

$$\min_{w} \left(f\left(\langle w, \psi(x_1) \rangle, ..., \langle w, \psi(x_m) \rangle \right) + R(||w||) \right) \tag{2.15}$$

donde $f : \mathbb{R}^m \longrightarrow \mathbb{R}$ es una función arbitraria y $R : \mathbb{R}_+ \longrightarrow \mathbb{R}$ es una función monótona no decreciente.

El siguiente teorema muestra que existe una solución óptima de la ecuación (2.15) que radica en el intervalo de $\{\psi(x_1), ..., \psi(x_m)\}$.

Teorema (Teorema del Representante). Asume que ψ es una función desde χ a un espacio de Hilbert. Entonces, existe un vector $\alpha \in \mathbb{R}^m$ tal que $w = \sum_{i=1}^{m} \alpha_i \psi(x_i)$ es una solución óptima de la ecuación (2.15).

Sobre la base del teorema del Representante podemos optimizar la ecuación (2.15) con respecto a los coeficientes α en lugar de los coeficientes w como sigue. Escribiendo $w = \sum_{j=1}^{m} \alpha_j \psi(x_j)$, tenemos que para todo i

$$\langle w, \psi(x_i)\rangle = \left\langle \sum_j \alpha_j \psi(x_j), \psi(x_i) \right\rangle = \sum_{j=1}^{m} \alpha_j \langle \psi(x_j), \psi(x_i)\rangle . \tag{2.16}$$

Similarmente

$$||w||^2 = \left\langle \sum_j \alpha_j \psi(x_j), \sum_j \alpha_j \psi(x_j) \right\rangle = \sum_{i,j=1}^{m} \alpha_i \alpha_j \langle \psi(x_i), \psi(x_j)\rangle . \tag{2.17}$$

Ahora sea $K(x, x') = \langle \psi(x), \psi(x')\rangle$ una función que implementa la función kernel con respecto a la incrustación ψ. En lugar de solucionar la ecuación (2.15), podemos solucionar el problema equivalente

$$\min_{\alpha \in \mathbb{R}^m} f\left(\sum_{j=1}^{m} \alpha_j K(x_j, x_1), ..., \sum_{j=1}^{m} \alpha_j K(x_j, x_m) \right) + R\left(\sqrt{\sum_{i,j=1}^{m} \alpha_i \alpha_j K(x_j, x_i)} \right) . \tag{2.18}$$

Para solucionar el problema de optimización dado en la ecuación (2.18), no necesitamos tener acceso directo a los elementos del espacio característico. Lo único que necesitamos conocer es cómo calcular productos internos en el espacio característico, o equivalentemente, cómo calcular la función kernel. De hecho, para solucionar la ecuación (2.18) necesitamos conocer el valor de la matriz G de dimensiones $m \times m$ tal que $G_{i,j} = K(x_i, x_j)$, la cual, es llamada con frecuencia la matriz Gram.

En particular, especificando lo anterior a la Soft-SVM, podemos reescribir el problema como

$$\min_{\alpha \in \mathbb{R}^m} \left(\lambda \alpha^T G \alpha + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i (G\alpha)_i\} \right) . \tag{2.19}$$

Una vez que aprendemos los coeficientes α, podemos calcular la predicción en una nueva instancia aplicando,

$$\langle w, \psi(x) \rangle = \sum_{j=1}^{m} \alpha_j \langle \psi(x_j), \psi(x) \rangle = \sum_{j=1}^{m} \alpha_j K(x_j, x). \tag{2.20}$$

2.2. Redes Neuronales

Para esta sección tenemos en cuenta diferentes fuentes de información donde destacamos la tesis de Aleksandar Palituka et al.[12], el artículo de Shiv Ram Dubey et al.[3], el artículo de Roberto García [6], el artículo de Nasr et al.[10] y Wikipedia principalmente.

Las redes neuronales tienen su raíz en el esfuerzo por emular el funcionamiento de las sinapsis cerebrales. El perceptron, que constituye la forma más elemental de una red neuronal, fue concebido en la década de 1950 por Frank Rosenblatt, en un desarrollo que se apoyó en investigaciones previas realizadas por Warren McCulloch y Walter Pitts en la década de 1940.

Las redes neuronales tienen la capacidad de aproximarse a cualquier función medible, siempre y cuando se cumplan ciertas condiciones. En particular, cuando se dispone de una red neuronal con una arquitectura lo suficientemente compleja, esta puede aproximar con precisión cualquier función continua en un conjunto compacto.

2.2.1. El Perceptron

El equivalente artificial de una neurona biológica se representa mediante un nodo o neurona artificial, cuyo esquema se ilustra en la Figura 2.1. Este

componente es una unidad de procesamiento que lleva a cabo una combinación lineal de señales de entrada x_i. Cada señal de entrada se multiplica por un peso sináptico correspondiente w_i, lo que da como resultado una entrada neta, que se indica mediante el símbolo sumatoria. Si la entrada neta alcanza o supera un cierto umbral predefinido, la neurona emite un potencial de activación y con un valor de 1. Por otro lado, si la entrada neta no cumple con el umbral, la neurona permanece en estado inactivo y la activación se establece en 0.

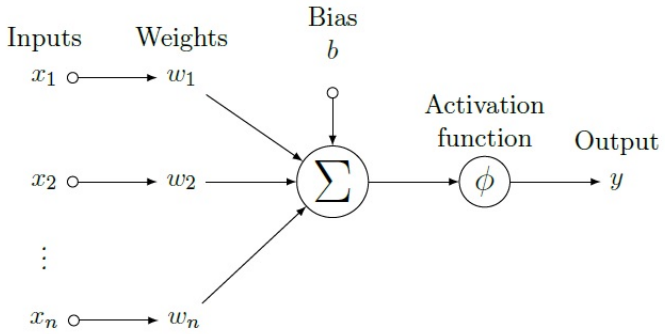

Figura 2.1: Diagrama de una neurona.

Este comportamiento se asemeja al modelo de una neurona binaria y se describe matemáticamente mediante una función de activación en forma de escalón,

$$y = \begin{cases} 1 & si\ w_1x_1 + w_2x_2...w_nx_n + b > 0 \\ 0 & en\ otro\ caso \end{cases} \tag{2.21}$$

En desarrollos posteriores, se han introducido diversas funciones de activación con el fin de habilitar modelos más complejos y flexibles. Estas funcio-

nes de activación permiten a las neuronas artificiales capturar relaciones no lineales en los datos y aprender patrones más sofisticados. Las más comunes son la función de activación logística,

$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{2.22}$$

o la función de activación de tangente hiperbólica

$$\phi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.23}$$

Estas funciones de activación son muy populares debido a varias razones, entre ellas la diferenciabilidad en su dominio, rangos acotados y la facilidad de interpretación.

2.2.2. Estructura de la Red Neuronal

La estructura de una red neuronal puede variar significativamente según la tarea específica que se esté abordando y el diseño deseado. Sin embargo, en términos generales, una red neuronal se compone de varias capas (capa de entrada, capas ocultas y capa de salida), cada una de las cuales contiene un conjunto de neuronas artificiales o nodos. Las capas se organizan en una secuencia, y la información fluye desde la capa de entrada hasta la capa de salida.

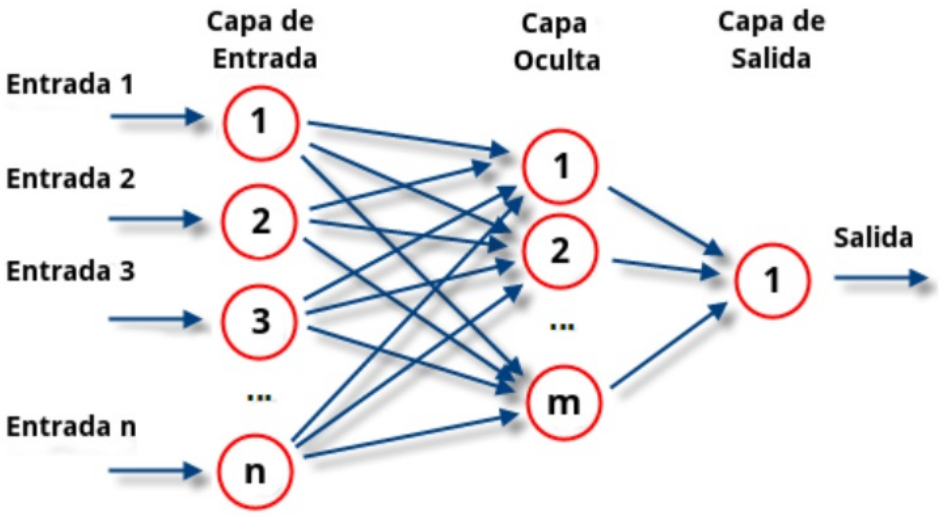

Figura 2.2: Estructura de una red neuronal. Fuente Wikipedia.

A continuación nos basamos en la tesis de Aleksandar Palikuca et al.[12] para sintetizar el flujo de cálculo en una red neuronal con una capa de entrada, una capa oculta y la capa de salida.

Denotamos por (x^i, y^i) el i-ésimo dato con $x \in \mathbb{R}^n$ siendo las entradas en la capa de entrada y y el valor objetivo en la capa de salida. El flujo a través de la red neuronal puede ser estructurado como sigue:

Para ser consistentes con las notaciones, sea,

$$\begin{cases} a_1^{(1)} = x_1^i, \\ a_2^{(1)} = x_2^i, \\ \quad \vdots \\ a_n^{(1)} = x_n^i, \end{cases} \tag{2.24}$$

donde $a^{(1)}$ es el vector de activaciones de la primera capa. Las activaciones de la segunda (capa oculta), se calculan como

$$\begin{cases} a_1^{(2)} = \phi\left(w_{10}^{(1)}a_0^{(1)} + w_{11}^{(1)}a_1^{(1)} + ... + w_{1n}^{(1)}a_n^{(1)}\right), \\ a_2^{(2)} = \phi\left(w_{20}^{(1)}a_0^{(1)} + w_{21}^{(1)}a_1^{(1)} + ... + w_{2n}^{(1)}a_n^{(1)}\right), \\ \vdots \\ a_m^{(2)} = \phi\left(w_{m0}^{(1)}a_0^{(1)} + w_{m1}^{(1)}a_1^{(1)} + ... + w_{mn}^{(1)}a_n^{(1)}\right), \end{cases} \tag{2.25}$$

donde $a_0^{(1)}$ es un término de sesgo añadido y m denota el número de neuronas en la capa oculta. Finalmente, la capa de salida se calcula como

$$\begin{cases} a_1^{(3)} = \phi\left(w_{10}^{(2)}a_0^{(2)} + w_{11}^{(2)}a_1^{(2)} + ... + w_{1m}^{(2)}a_m^{(2)}\right), \\ a_2^{(3)} = \phi\left(w_{20}^{(2)}a_0^{(2)} + w_{21}^{(2)}a_1^{(2)} + ... + w_{2m}^{(2)}a_m^{(2)}\right), \\ \vdots \\ a_k^{(3)} = \phi\left(w_{k0}^{(2)}a_0^{(2)} + w_{k1}^{(2)}a_1^{(2)} + ... + w_{km}^{(2)}a_m^{(2)}\right), \end{cases} \tag{2.26}$$

donde $a_0^{(2)}$ es un término de sesgo añadido y k denota el número de neuronas en la capa de salida. Con

$$x^i = \begin{bmatrix} x_0^i \\ x_1^i \\ x_2^i \\ \vdots \\ x_n^i \end{bmatrix}, \qquad W^{(1)} = \begin{bmatrix} w_{10}^{(1)} & w_{11}^{(1)} & \cdots & w_{1n}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & \cdots & w_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m0}^{(1)} & w_{m1}^{(1)} & \cdots & w_{mn}^{(1)} \end{bmatrix}, \qquad W^{(2)} = \begin{bmatrix} w_{10}^{(2)} & w_{11}^{(2)} & \cdots & w_{1m}^{(2)} \\ w_{20}^{(2)} & w_{21}^{(2)} & \cdots & w_{2m}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k0}^{(2)} & w_{k1}^{(2)} & \cdots & w_{km}^{(2)} \end{bmatrix} \tag{2.27}$$

Las ecuaciones (2.24), (2.25) y (2.26) pueden ser escritas como $a^{(1)} = x^i$, $a^{(2)} = \phi(z^{(2)})$ y $a^{(3)} = \phi(z^{(3)})$, donde $z^{(2)} = W^{(1)}a^{(1)}$ y $z^{(3)} = W^{(2)}a^{(2)}$ y la función de activación ϕ es aplicada por elementos.

En conclusión, dada una muestra de datos de entrada x^i, la red neuronal

produce la siguiente función de aproximación

$$h_W(x^i) = a^{(3)} = \phi\left(W^{(2)}a^{(2)}\right) = \phi\left(W^{(2)}\phi\left(W^{(1)}a^{(1)}\right)\right) = \phi\left(W^{(2)}\phi\left(W^{(1)}x^i\right)\right). \tag{2.28}$$

Este proceso de cálculo se conoce como propagación hacia delante. Para conseguir que los resultados obtenidos mediante la función de aproximación sean aproximados a los valores objetivo, la red neuronal necesita ser entrenada. Este proceso se lleva a cabo en parte mediante un procedimiento conocido como retropropagación.

2.2.3. Entrenamiento de la Red Neuronal

Para entrenar la red neuronal es necesaria una función de coste (también llamada función de pérdida o función objetivo). Dicha función desempeña un papel fundamental durante el entrenamiento de la red. Su función principal es cuantificar la discrepancia entre las predicciones realizadas por la red neuronal y los valores reales o etiquetas de los datos de entrenamiento. La función de coste proporciona una medida de cuán equivocada es la red en sus predicciones.

El objetivo del entrenamiento de una red neuronal es minimizar la función de coste. Para hacerlo, se utilizan técnicas de optimización como el descenso de gradiente para ajustar los pesos y los sesgos de la red de manera que las predicciones se acerquen lo más posible a los valores reales.

2.2.3.1. Funciones de Coste

Las funciones de coste dependen del tipo de problema que se está resolviendo. Una de las funciones de coste más común es la función de coste cuadrática. Es una función efectiva y ampliamente utilizada en problemas de regresión debido a su facilidad de cálculo, pero es importante tener en cuenta que puede ser sensible a valores atípicos en los datos,

$$J_i\left(W, x^i, y^i\right) = \frac{1}{2}\sum_{k=1}^{K}\left(h_W(x^i)_k - y_k^i\right)^2 \tag{2.29}$$

y la función de entropía cruzada, que es comúnmente utilizada en problemas de clasificación en el aprendizaje automático y las redes neuronales. También se la conoce como pérdida logarítmica. Su fórmula matemática varía según el contexto de clasificación binaria o multiclase. Aquí definimos la versión multiclase,

$$J_i\left(W.x^i, y^i\right) = -\sum_{k=1}^{K} y_k^i log\left(h_W(x^i)_k\right) + (1 - y_k^i) log(1 - h_W(x^i)_k), \tag{2.30}$$

donde K denota la dimensión de la capa de salida, y el coste total sobre el total de muestras $\mathcal{S}$ es

$$J(W) = \frac{1}{S}\sum_{i=1}^{S} J_i\left(W, x^i, y^i\right). \tag{2.31}$$

2.2.3.2. Algoritmo de Descenso de Gradiente

El algoritmo de Descenso de Gradiente (SGD) es un método de optimización ampliamente utilizado en el aprendizaje automático para ajustar

los parámetros de un modelo con el objetivo de minimizar una función de coste. Su funcionamiento se basa en la idea de encontrar los valores de los parámetros que hacen que la función de coste sea lo más pequeña posible, lo que equivale a encontrar el mínimo de la función de coste en el espacio de parámetros. En el contexto de la función de coste de una red neuronal, el algoritmo de descenso de gradiente estocástico se desplaza de forma iterativa hacia un conjunto de pesos en el espacio de parámetros que minimiza la función de coste. Este proceso se define como:

$$w_{kj}^{l}(n+1) = w_{kj}^{l}(n) - \eta \frac{\partial J(n)}{\partial w_{kj}^{l}(n)}. \tag{2.32}$$

donde η es el ratio de aprendizaje, controlando la magnitud de los cambios de pesos en la red neuronal. Este método actualiza los pesos después de cada observación y en cada iteración selecciona una muestra aleatoria, haciendo que el procesamiento sea menos costoso.

2.3. Algoritmo de Bosque Aleatorio

Para la elaboración de esta sección hemos tenido en cuenta la información de la página web www.scikit-learn.org, Wikipedia, el artículo de Lior Rokach et al.[13], y los libros [8] [14] principalmente.

El algoritmo Bosque Aleatorio es un método avanzado para analizar una colección de muchas clasificaciones diferentes de árboles de decisión. El nombre de bosque aleatorio describe con precisión el algoritmo de clasificación en el sentido que es el análisis de una infinidad de árboles de decisión generados al azar a partir de un conjunto finito de datos conocidos y clasificados.

2.3.1. Algoritmo de Árbol de Decisión

Un Árbol de Decisión es una técnica de modelado predictivo que se emplea para descomponer un problema de clasificación en una jerarquía de decisiones que se toman a nivel de variables. El objetivo principal de un árbol de decisión es ofrecer una comprensión explicativa de los patrones y las variables que influyen en la clasificación. Los árboles se componen de tres elementos fundamentales: la raíz, los nodos y las hojas.

La raíz de cada árbol simboliza la variable inicial de decisión que proporciona la información más crucial para que el árbol continúe dividiéndose. Cada nodo crea un nuevo punto de decisión a lo largo del árbol, y cada ruta de nodos finaliza en una hoja que representa la clasificación de salida.

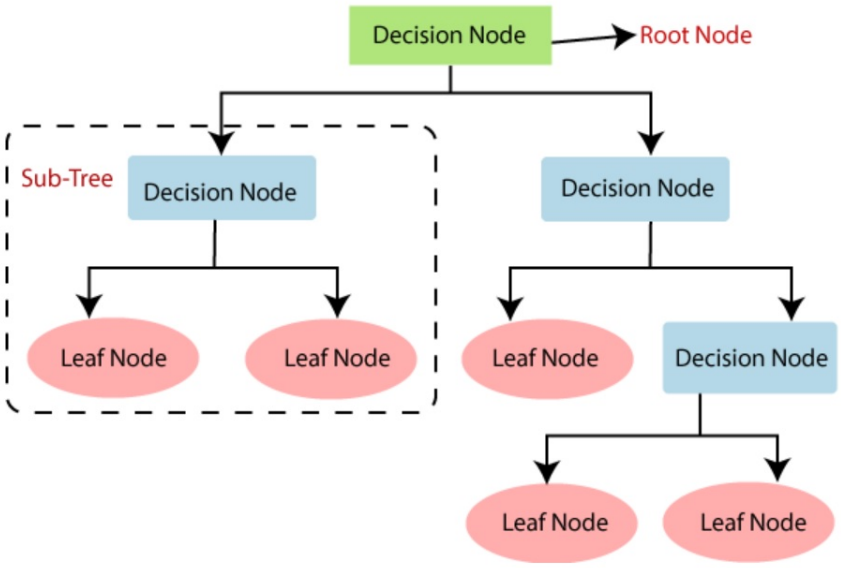

Figura 2.3: Árbol de decisión. Fuente: https://www.javatpoint.com.

A continuación explicamos la métrica de la entropía, utilizada en árboles

de decisión y bosques aleatorios para medir la impureza de un conjunto de datos y ayudar en la toma de decisiones sobre cómo dividir los datos en nodos durante la construcción del árbol. Ambas métricas se utilizan para determinar la calidad de una división y se esfuerzan por crear divisiones que sean más homogéneas en términos de clases o categorías objetivo. En la práctica también utilizaremos la métrica Gini, aunque el objetivo ahora es obtener una visión general del funcionamiento de un algoritmo de bosque aleatorio.

2.3.1.1. Entropía

La entropía se describe por Shannon como una medida de cuantos datos son producidos y cuanta incertidumbre se presenta en la producción. Para calcular la entropía, necesitamos calcular la probabilidad de todos los eventos posibles, eso es, no podemos calcular la entropía de un sistema desconocido.

Shannon define el valor de la entropía de un sistema discreto a ser,

$$H = -\sum p_i log_2 p_i \tag{2.33}$$

donde p_i es la probabilidad de cada posible evento.

Podemos extender un poco más definiendo el significado de entropía condicional tal que para dos eventos X e Y, deseamos considerar la entropía de Y dado un evento X. En términos prácticos, esto nos dice la incertidumbre y predictibilidad de un evento Y asumiendo que el evento X ocurra con anterioridad al evento Y

$$H_X(Y) = -\sum p(i)p(j)log_2 p_i(j) \tag{2.34}$$

donde:

1. $p(i)$ es la probabilidad de que el evento X ocurra

2. $p(j)$ es el evento Y

3. $p_i(j)$ es la probabilidad de que el evento Y ocurra después del evento X.

2.3.1.2. Ganancia de Información

La ganancia de información en un árbol de decisión es una métrica que se utiliza para evaluar la calidad de una división potencial en un nodo del árbol. En el contexto de los árboles de decisión, el objetivo principal es dividir los datos de entrenamiento en subconjuntos más homogéneos en términos de la variable objetivo para mejorar la capacidad predictiva del árbol.

La ganancia de información se basa en el concepto de entropía y se utiliza para medir cuánto se reduce la incertidumbre o la impureza en los datos al realizar una división particular en un nodo del árbol.

La fórmula general para calcular la ganancia de información es,

$$IG = \text{Entropia Inicial} - \text{Entropia despues de la division.} \tag{2.35}$$

Una vez que obtenemos la entropía y la ganancia de información, ya es posible tomar decisiones sobre cómo dividir los datos en los nodos del árbol para construir un árbol de decisión efectivo.

2.3.2. Clasificación final con Bosque Aleatorio

Dado que el algoritmo Bosque Aleatorio requiere la construcción de múltiples árboles de decisión, la fase final implica tomar una decisión de clasificación definitiva. Existen varias maneras de abordar este proceso, pero mencionaremos dos de los enfoques más comunes.

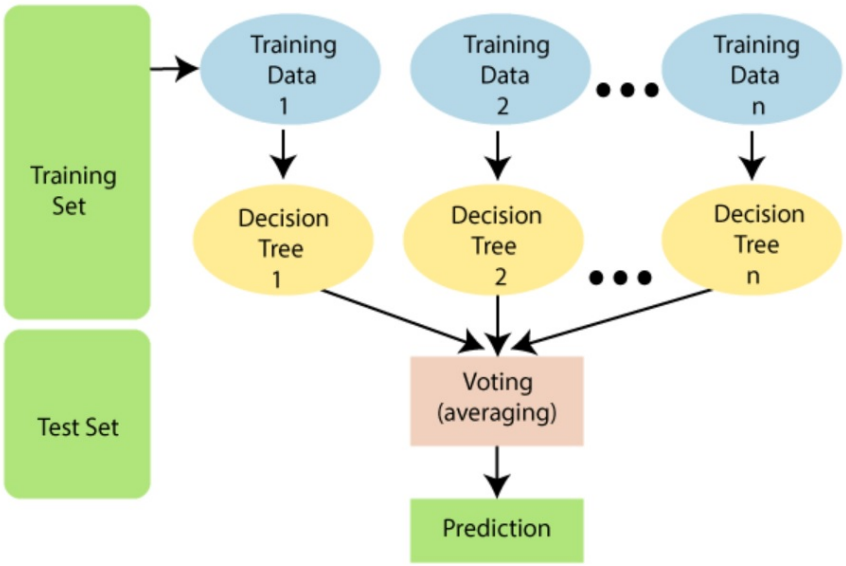

Figura 2.4: Random forest. Fuente: https://www.javatpoint.com.

El método más común es llamado *majority voting*, es decir, si tenemos 10 árboles y 4 resultados en una clasificación de A mientras que 6 resultados en la clasificación de B, la clasificación final será B.

El otro método es llamado *performance voting*. En este método, cada árbol tiene un voto ponderado a su precisión en la clasificación de un conjunto conocido como *validation set*.

Capítulo 3

Caso de Estudio

A continuación, presentaremos la metodología utilizada para llevar a cabo la implementación práctica.

Los datos utilizados provienen de la página web *Tickstory.com*, cuya fuente principal de información pertenece al intermediario bursátil suizo Dukascopy, obteniendo de esta manera la descarga de datos sobre mercados de divisas, materiales, indices, bonos, etc...

El procesamiento se llevó a cabo mediante la creación de una máquina virtual en la plataforma de servicios Cloud de Amazon. Se configuró una máquina con Sistema Operativo Windows, equipada con 16 GB de memoria RAM y un procesador Intel I7 de dos núcleos.

3.1. Datos

Los datos utilizados en este trabajo corresponden al Libro de Órdenes Limitadas o Profundidad de Mercado del par de divisas EUR/USD. Estos

datos difieren de las cotizaciones observadas en marcos temporales más amplios, como mensuales, semanales o diarios, ya que consideran tanto el precio de compra (Bid) como el precio de venta (Ask). Además, debido a que se basan en transacciones realizadas en tiempo real (Ticks) por diversos participantes en el mercado de valores, no siguen una regularidad en la serie temporal.

En consecuencia, estamos utilizando datos recopilados a una escala de milisegundos, que abarcan desde el 1 de enero de 2023 hasta el 30 de abril de 2023 y que incluyen un total de 10.521.204 transacciones en dicho periodo.

El esquema de los datos extraídos es el siguiente:

1. Time stamp: Fecha y tiempo en el que se realiza la transacción (compra o venta)

2. Precios Bid: Cada precio para el cual existe un participante de mercado dispuesto a comprar EUR

3. Precios Ask: Cada precio para el cual existe un participante de mercado dispuesto a vender EUR

4. Volúmenes Bid: El volumen total en cada precio Bid

5. Volúmenes Ask: El volumen total en cada precio Ask

El interés en este tipo de datos se basa tanto en su complejidad computacional como en la riqueza de la información que proporciona. Requiere una velocidad extrema y ofrece la capacidad de observar no solo las operaciones de compra o venta, sino también las posiciones en espera para alcanzar ciertos

niveles y las operaciones que gestionan los riesgos de una operativa específica. Por lo tanto, la calidad de esta información se convierte en una fuente fundamental para la creación de modelos de riesgo y el diseño de estrategias de comercio, en su mayoría.

3.2. Pre-Procesado de datos

Debido a que estamos utilizando datos transaccionales, es necesario mitigar los efectos de ruido en el mercado construyendo marcos temporales que ademas nos establezca una serie temporal con puntos equidistantes.

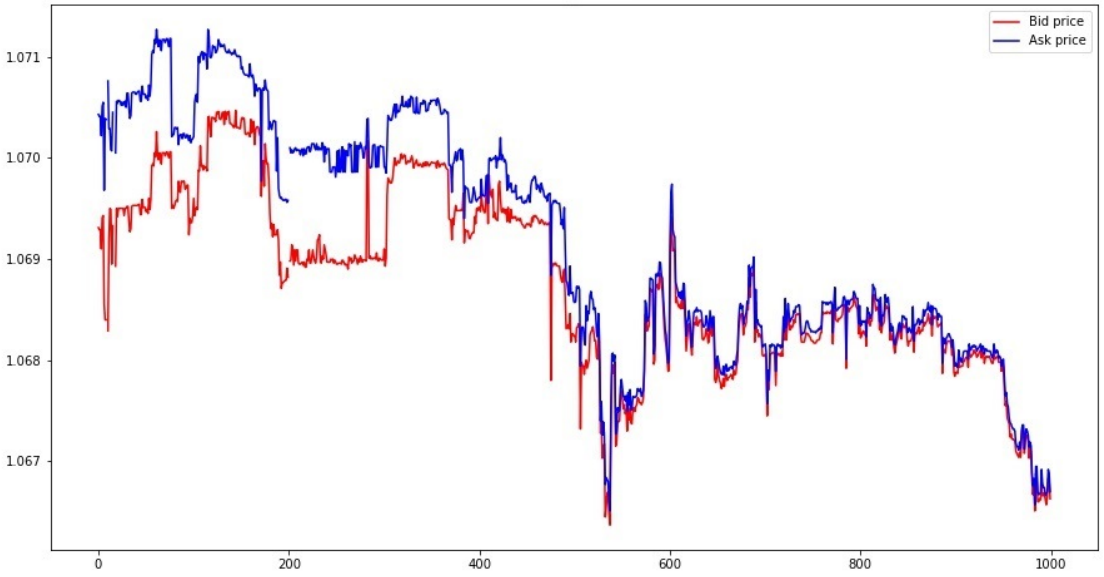

Figura 3.1: Representación de 1000 Precios Bid/Ask agregados a 60 segundos.

Es por ello, que en nuestro trabajo, vamos a crear las temporalidades

de 60, 30, 15 y 5 segundos, seleccionando en cada agregación el último tick observado tanto para el precio Bid como para el precio Ask.

Vemos en la Figura 3.1 como existe una separación entre la oferta y la demanda al inicio de la serie indicando un periodo de baja liquidez en el mercado.

Por otro lado, los volúmenes son agregados como una suma, dentro de cada intervalo de agregación.

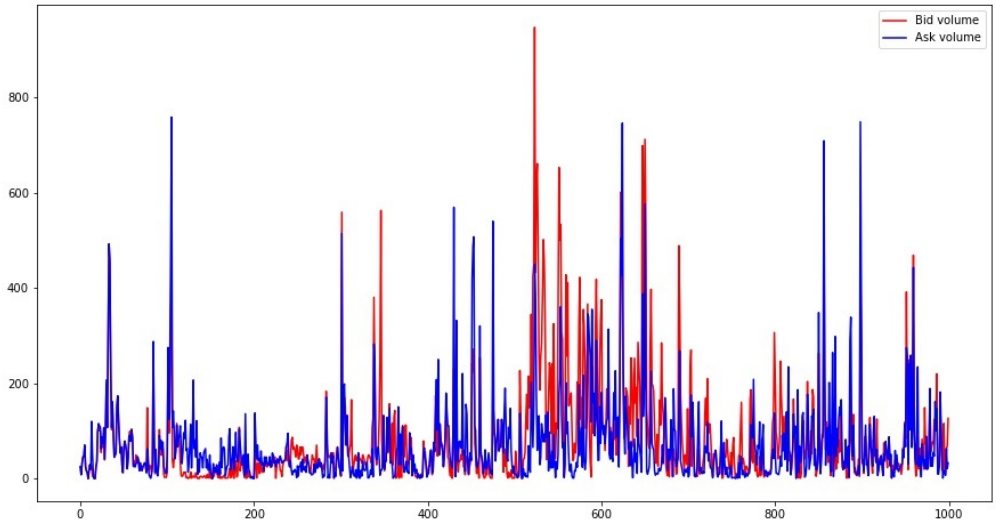

Figura 3.2: Representación de 1000 Volúmenes Bid/Ask agregados a 60 segundos.

También se tienen en cuenta diferentes ventanas de predicción definiendo tres parámetros sobre los periodos de agregación:

$$Horizonte\ Prediccion(seg) = Agregacion(seg) \cdot \Delta t \tag{3.1}$$

para $\Delta t = \{1, 2, 5\}$.

Por último, se crean dos variables adicionales: *número de transacciones*, donde se realiza un conteo del número de ticks tanto para los precios Bid como para los precios Ask y se agregan según la agregación temporal analizada y los *retornos* de precio de un solo paso.

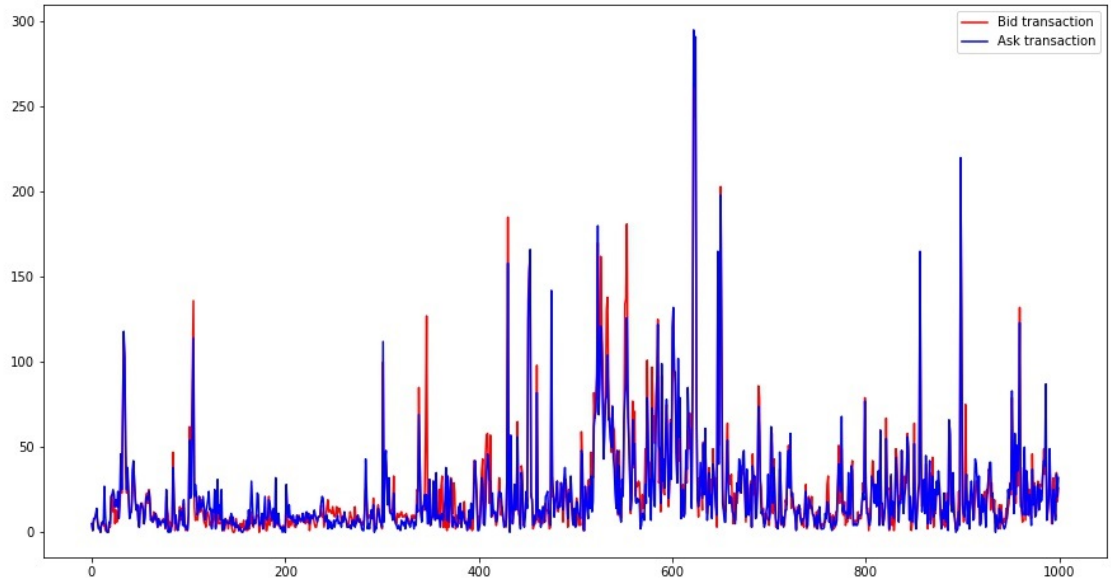

Figura 3.3: Representación del Número de Transacciones Bid/Ask agregados a 60 segundos.

Por otro lado, eliminamos datos de sábados y domingos dado que durante el fin de semana no existen cotizaciones, y la apertura de mercado del Domingo a las 23:00 suele causar cierto sesgo por movimientos poco representativos.

Para el resto de datos vacíos, aplicamos el precio del último precio disponible, tanto para el Bid como para el Ask. De esa forma mantenemos la lógica de que si no hubo cambio de cotización, es que no hubo ningún tipo

de transacción.

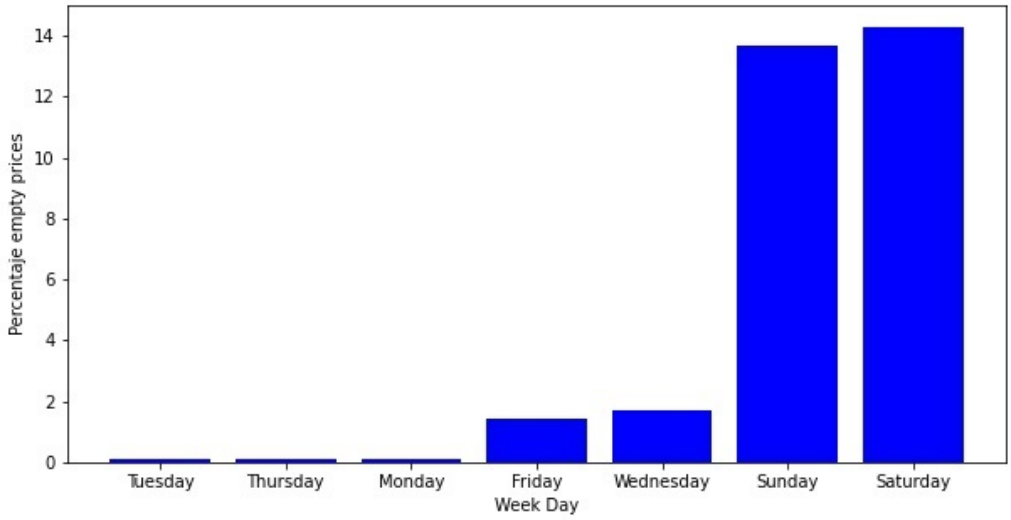

Figura 3.4: Porcentaje de precios vacíos (Bid/Ask) por día de la semana.

Para finalizar este apartado, se crean dos variables adicionales:

1. Total Volume: es la suma del Bid Volume y el Ask Volume.
2. Total Transactions: es la suma del Bid Transactions y el Ask Transactions.

3.2.1. Selección de Características y Creación de las Clases a Predecir

Los datos recopilados permiten la creación de conjuntos de características que se utilizan como entradas en varios algoritmos de aprendizaje automático. Estos conjuntos se aplican a variables relacionadas con el volumen, el

precio y las transacciones con el propósito de identificar patrones en diferentes dinámicas del mercado de divisas (EURUSD). Hemos definido tres tipos de medias móviles: una de corto plazo, que corresponde a un múltiplo de cinco en términos de la agregación temporal; una de mediano plazo, que es un múltiplo de veinte en la agregación temporal; y una de largo plazo, que se basa en un múltiplo de cien en la agregación temporal utilizada

La razón detrás de la utilización de diversas medias móviles en diferentes intervalos de tiempo radica en la búsqueda de suavizar la variabilidad de cada variable, con el objetivo de identificar con mayor nitidez los patrones que puedan conducir a mayores rendimientos.

Este enfoque sigue la línea de investigación presentada en la tesis de Aleksandar Palikuca et al.[12], con la distinción de que empleamos datos de profundidad de mercado de un paso y no de sesenta pasos, queriendo decir, que para un instante t, tenemos la información perteneciente a un precio de la oferta y otro precio de la demanda, mientras que en la tesis, para un instante t, se obtiene la información de los 60 precios de la oferta más cercanos y los 60 precios de la demanda más cercanos. Además, introducimos un factor multiplicador en las ventanas temporales analizadas para mantener una cierta equivalencia entre cada intervalo de tiempo definido. Esto nos permite realizar comparaciones incluso en los casos en los que no se haya logrado optimizar los diversos algoritmos de aprendizaje automático desarrollados.

Las características creadas son las siguientes:

1. $\mathcal{F}_1$: Media móvil de corto periodo sobre volúmenes Bid y Ask, aplicando la regla $agregacion(seg) \cdot 5$.

2. $\mathcal{F}_2$: Media móvil de medio periodo sobre volúmenes Bid y Ask, apli-

cando la regla $agregacion(seg) \cdot 20$.

3. $\mathcal{F}_3$: Media móvil de largo periodo sobre volúmenes Bid y Ask, aplicando la regla $agregacion(seg) \cdot 100$.

4. $\mathcal{F}_4$: Media móvil de largo periodo sobre el total del volumen aplicando la regla $agregacion(seg) \cdot 100$.

5. $\mathcal{F}_5$: Media móvil de corto periodo sobre retornos Bid y Ask, aplicando la regla $agregacion(seg) \cdot 5$.

6. $\mathcal{F}_6$: Media móvil de medio periodo sobre retornos Bid y Ask, aplicando la regla $agregacion(seg) \cdot 20$.

7. $\mathcal{F}_7$: Media móvil de largo periodo sobre retornos Bid y Ask, aplicando la regla $agregacion(seg) \cdot 100$.

8. $\mathcal{F}_8$: Media móvil de corto periodo sobre transacciones Bid y Ask, aplicando la regla $agregacion(seg) \cdot 5$.

9. $\mathcal{F}_9$: Media móvil de medio periodo sobre transacciones Bid y Ask, aplicando la regla $agregacion(seg) \cdot 20$.

10. $\mathcal{F}_{10}$: Media móvil de largo periodo sobre transacciones Bid y Ask, aplicando la regla $agregacion(seg) \cdot 100$.

11. $\mathcal{F}_{11}$: Media móvil de largo periodo sobre el total de transacciones aplicando la regla $agregacion(seg) \cdot 100$.

Para la creación de las diferentes clases a predecir, tenemos en cuenta el artículo de Alec Kercheval et al.[9], donde asumiendo que el precio de la oferta es inferior al precio de la demanda en el mismo instante t, podemos definir tres escenarios: el primero, un escenario de compra, cuando el mejor precio de la oferta en un instante futuro $t + \Delta t$ supera al mejor precio de la demanda en el instante t; un escenario de venta, cuando el mejor precio de la demanda en un instante futuro $t + \Delta t$ es inferior al mejor precio de la oferta en un instante t; un escenario de no comprar ni vender, si no existe ninguna de las dos situaciones anteriores.

Por lo tanto, las tres clases $\{C_{+1}, C_0, C_{-1}\}$ son definidas matemáticamente de la siguiente manera,

1. C_{+1}: $P^{Bid}_{t+\Delta t} > P^{Ask}_t$.

2. C_0: $P^{Bid}_{t+\Delta t} \leq P^{Ask}_t$ y $P^{Ask}_{t+\Delta t} \geq P^{Bid}_t$.

3. C_{-1}: $P^{Ask}_{t+\Delta t} < P^{Bid}_t$.

Donde P^{Ask}_t y P^{Bid}_t denotan el mejor precio de la demanda (Ask) y el mejor precios de la oferta (Bid) en el tiempo t. Similarmente, $P^{Ask}_{t+\Delta t}$ y $P^{Bid}_{t+\Delta t}$ denotan el mejor precio Ask y Bid al final del intervalo de predicción $t + \Delta t$.

3.3. Análisis Exploratorio de Datos

Realizamos un primer análisis de los datos para evaluar principalmente el desbalanceo de las clases en la muestra, la relación y la distribución de cada una de las variables. Por simplicidad tomaré el ejemplo de la agregación

temporal a 60 segundos, ya que el análisis de todas las distintas agregaciones temporales se hace de una forma similar y de forma automática.

En primer lugar, observamos cierto desbalanceo en la muestra entre las diferentes clases, por lo que aplicamos la técnica del submuestreo para mitigar dicho efecto. Buscamos generalizar al máximo el modelo, ya que como sabemos, en los mercados financieros existen tendencias y dependiendo de la ventana de datos utilizada, puede existir cierto sesgo en una determinada dirección. Por otro lado, nos inclinamos por el método del submuestreo, dado que computacionalmente es menos costoso, existe un histórico de datos abundante y podemos de esta forma también, mitigar la acción de ciertos valores atípicos (outliers).

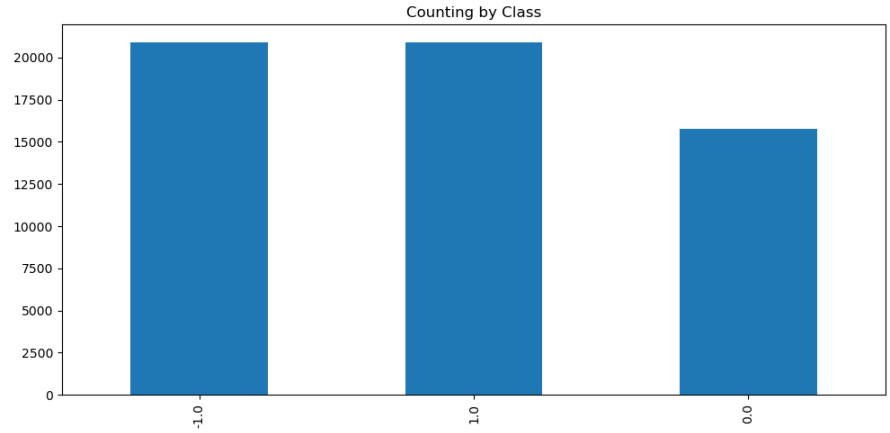

Figura 3.5: Representación del desbalanceo entre las distintas clases.

Para el estudio de valores atípicos utilizamos tanto el gráfico de cajas como la gráfica de distribución de densidad para ver la dinámica de una determinada variable.

Podemos apreciar en la Figura 3.6 que, aparentemente, los retornos, tanto

en el precio Bid como en el precio Ask, mantienen una distribución aproximadamente normal, que se ve distorsionado cuanto más grande es el periodo de la media utilizada. No obstante, se aprecian también ciertos valores atípicos en cada uno de los periodos, obligándonos de esta forma a tomar ciertas reglas para mitigar dicho efecto en distribuciones no normales.

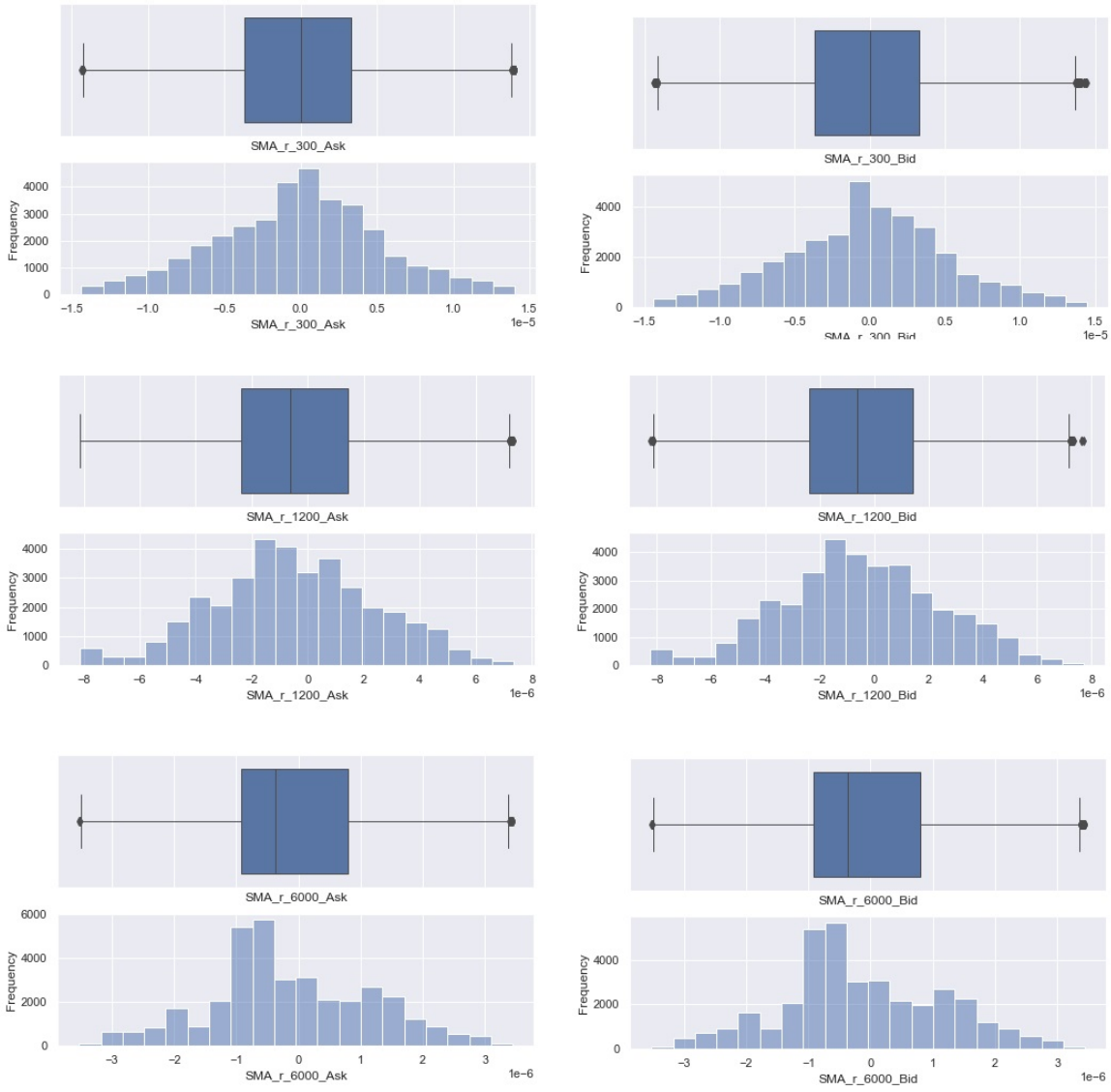

Figura 3.6: Diagrama de cajas e Histograma de la variable de retorno en diferentes periodos.

En la Figura 3.7, donde se tienen en cuenta las transacciones, podemos

ver como en periodos cortos, no existe normalidad y se aprecia la presencia de valores atípicos, tanto en el precio Bid como en el precio Ask. En el medio periodo se aprecia un poco de desplazamiento con respecto a la distribución normal y la presencia de valores atípicos. En el periodo largo no se aprecia normalidad y sí la presencia de valores atípicos.

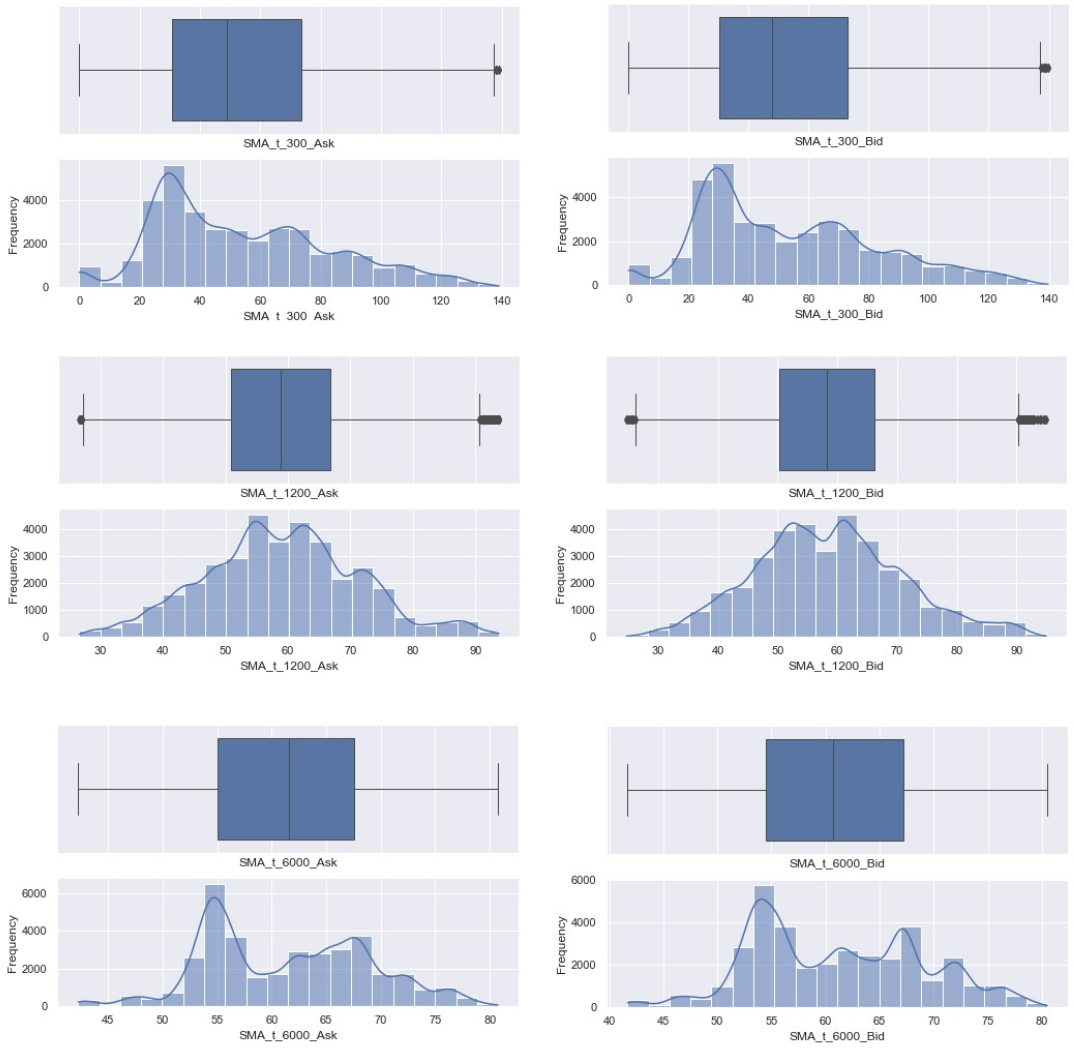

Figura 3.7: Diagrama de cajas e Histograma de la variable de transacciones en diferentes periodos.

En la Figura 3.8, donde se tienen en cuenta volúmenes, se aprecia bastante desplazamiento y presencia de valores atípicos en todos los periodos tanto

para el precio Bid como para el precio Ask.

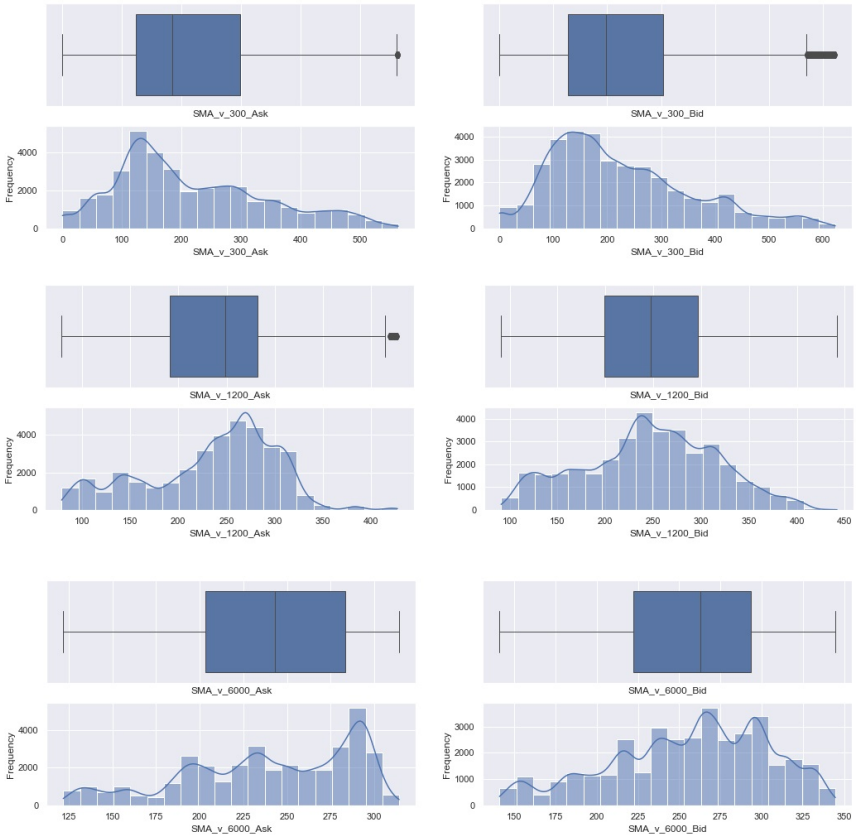

Figura 3.8: Diagrama de cajas e Histograma de la variable de volúmenes en diferentes periodos.

Para el control de los valores atípicos, excluimos todos aquellos valores que superan los límites calculados:

1. $outlier\ minimo < Q1 - 1.5 \cdot IQR$

2. $outlier\ maximo > Q3 + 1.5 \cdot IQR$

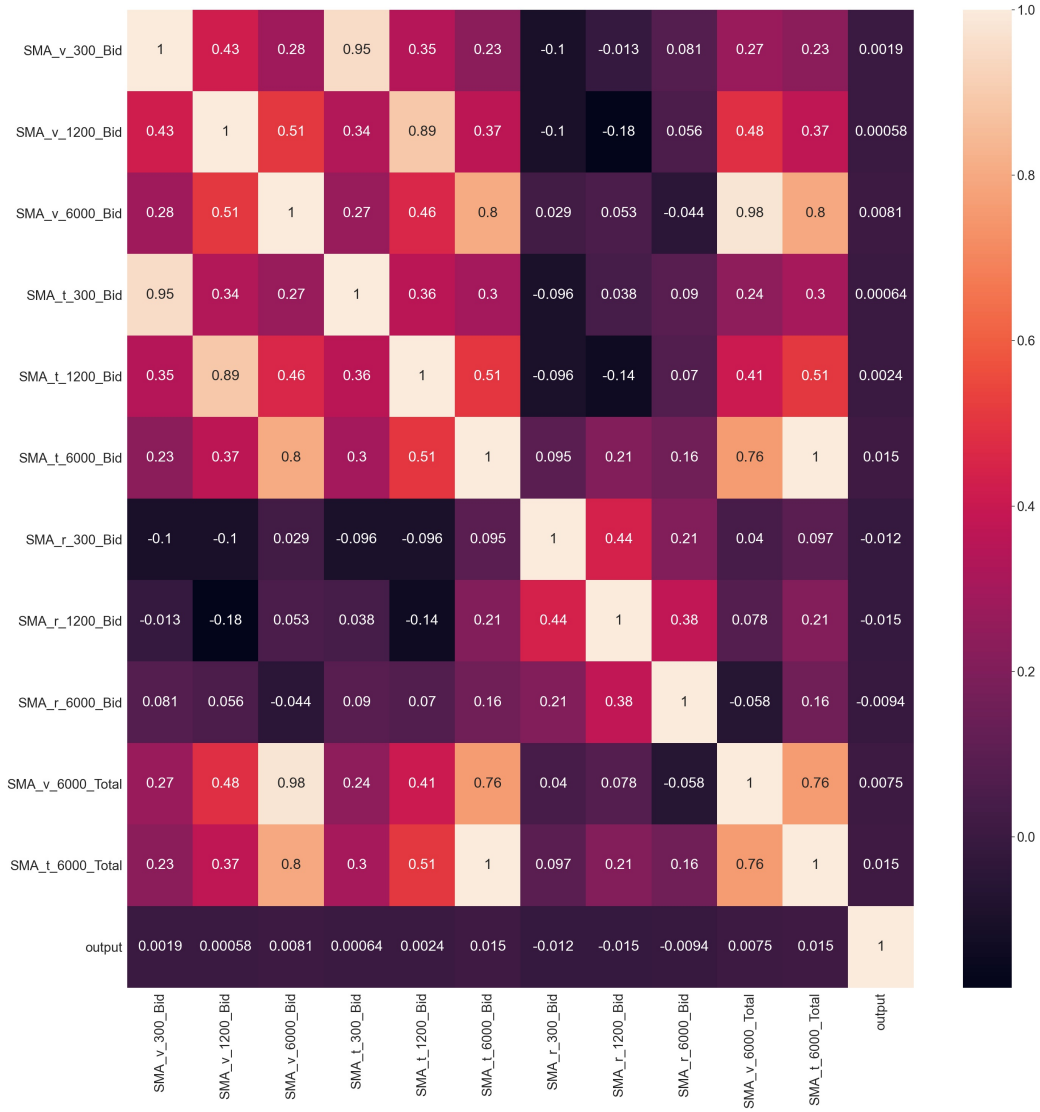

Figura 3.9: Matriz de correlación entre cada una de las variables Bid.

Después de llevar a cabo el submuestreo inicial para equilibrar los datos y aplicar la regla para eliminar los valores atípicos, procedemos a realizar un segundo submuestreo con el objetivo de mantener la muestra completa-

mente balanceada. Nuestra intención es generalizar el modelo al máximo, evitando el sobre-entrenamiento en la utilización de algoritmos de aprendizaje automático.

En la Figura 3.9, se muestra la matriz de correlación de las variables relacionadas con la información de la oferta (Bid), para una mejor visualización, reduciendo así la cantidad de variables representadas. Es importante destacar que la dinámica en relación a la información de los precios de la demanda (Ask) es similar a la de los precios de la oferta (Bid). En cuando a la variable objetivo, no presenta relaciones fuertes con las características.

3.4. Reducción de la Dimensión

Para este apartado hemos utilizado diversas fuentes de internet dado que no existe un método determinista para calcular el número de componentes de forma precisa. Entre las fuentes destaca https://support.minitab.com, www.scikit-learn.org y Wikipedia.

En esta sección evaluamos la posibilidad de reducir la cantidad de variables de entrada, que actualmente ascienden a once. Es crucial recordar que estamos trabajando con intervalos temporales muy cortos, lo que implica que la eficiencia computacional es un factor de gran importancia. Para abordar este objetivo, llevamos a cabo un análisis con el fin de determinar cuántas variables son necesarias para explicar el 90 % de la varianza acumulada. Además, consideramos la información proporcionada por los autovalores, identificando cualquier autovalor menor a 1 como insignificante. El número máximo de componentes resultante de la combinación de ambos análisis, ya

sea a partir del análisis de autovalores o del análisis de la varianza acumulada, nos indicará la cantidad óptima de componentes a utilizar en nuestro modelo.

En la Figura 3.10 podemos ver que el número de componentes que explican el 90 por ciento de la varianza son 6.

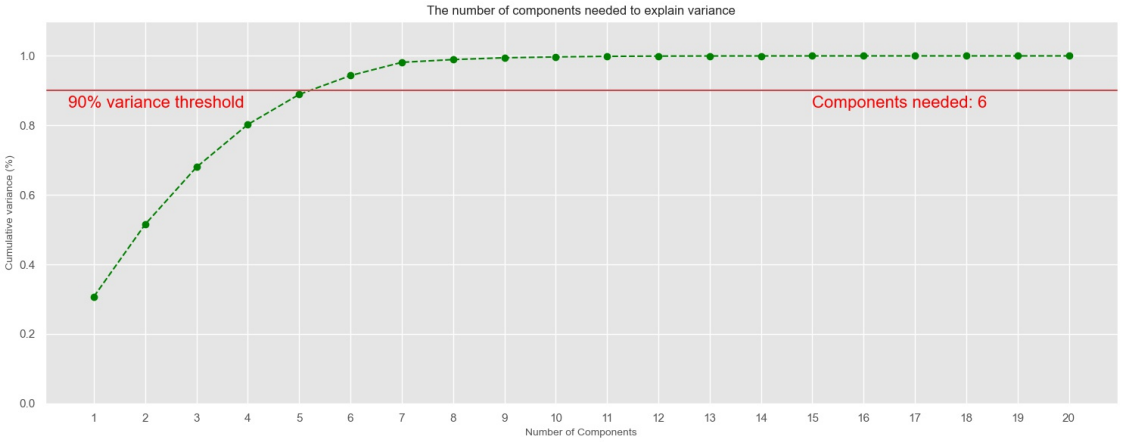

Figura 3.10: Número de componentes según la información de la varianza explicada.

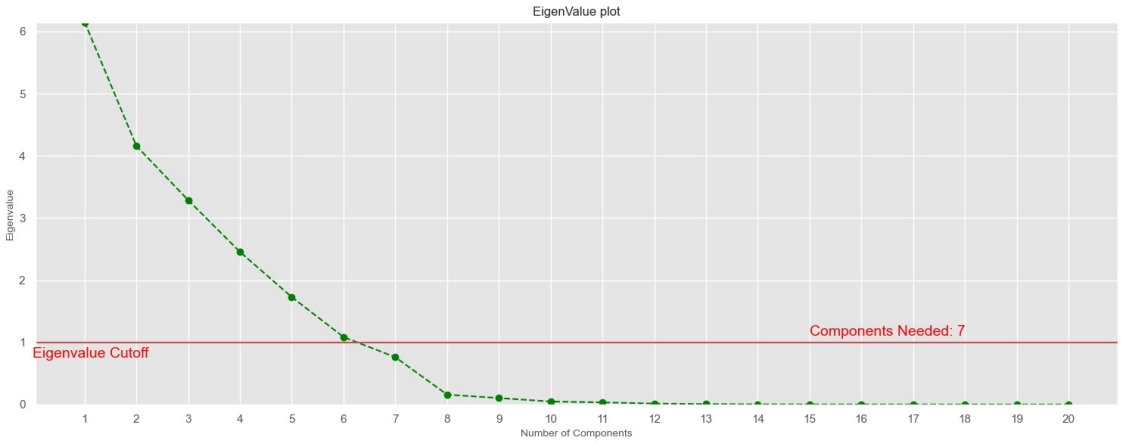

Figura 3.11: Número de componentes según la información de los autovalores.

En la Figura 3.11 se realiza un análisis parecido pero como hemos dicho, utilizando la información de los autovalores. En este caso, necesitamos 7 componentes, ya que cae por debajo del valor de 1 y por tanto, todo lo demás es insignificante para el modelo. El valor máximo entre la utilización de la varianza o los autovalores indica el número de componentes a elegir para el modelo, que en este caso es 7.

Agregación(seg)	**Pred.(seg)**	**Num.Comp.(Var)**	**Num. Comp.(Auto)**
60	300	6	7
60	120	6	7
60	60	6	7
30	150	6	7
30	60	6	7
30	30	6	7
15	75	5	6
15	30	5	6
15	15	5	6
5	25	5	6
5	10	5	5
5	5	5	5

Tabla 3.1: Evolución del número de componentes a través de las diferentes agregaciones temporales y ventanas de predicción

La Tabla 3.1 representa la variación del número de componentes con cada uno de los métodos a través de cada agregación y ventana de predicción,

notando que el número de componentes que explican la mayor parte de la información disminuyen conforme disminuyen los tiempos de agregación.

3.5. Aprendizaje Automático

En este módulo vamos a aplicar diferentes clasificadores múltiples para evaluar no sólo su predicción en diferentes horizontes sino también la búsqueda de características para poder desarrollar un algoritmo de *trading* que sitúe las probabilidades a nuestro favor.

Para la aplicación de los distintos algoritmos de clasificación vamos a utilizar librerías en el lenguaje de programación Python, en concreto *Sklearn*, que nos ofrece una amplia variedad de herramientas para, no sólo entrenar el algoritmo, sino evaluar su precisión, división de los datos en entrenamiento y prueba, transformaciones en los datos de entrada (estandarización) y la utilización de validaciones cruzadas que nos permitan optimizar la búsqueda de hiperparámetros que consigan la mejor precisión en el modelo.

Tenemos una serie de consideraciones generales para la aplicación de cada uno de los clasificadores que citamos a continuación.

Validación Cruzada

La validación cruzada o *cross-validation* es una técnica utilizada para evaluar los resultados de un análisis estadístico y hacerla algo más independiente de la partición entre datos de entrenamiento y prueba. En nuestro caso, la validación cruzada es utilizada para encontrar los hiperparámetros óptimos

del modelo.

En la Figura 3.12 podemos observar un método de validación cruzada K de cinco bloques e iteraciones. Uno de los subconjuntos se utiliza como datos de prueba y el resto $(K-1) = 4$ como datos de entrenamiento. El proceso de validación cruzada es repetido durante $K = 5$ iteraciones, con cada uno de los posibles subconjuntos de datos de prueba. Finalmente se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. Este método es muy preciso puesto que evaluamos a partir de $K = 5$ combinaciones de datos de entrenamiento y de prueba, pero aún así tiene una desventaja, y es que, es lento desde el punto de vista computacional.

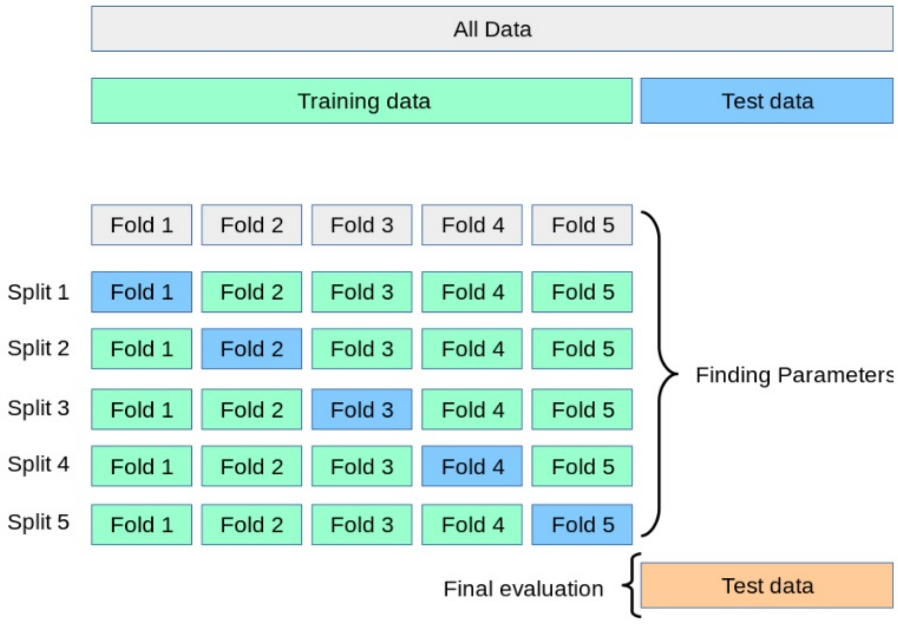

Figura 3.12: Validación cruzada K=5. Fuente: www.scikit-learn.com.

Medición de la precisión del Modelo

Para la medición de la precisión del modelo sobre datos de prueba, utilizaremos cuatro métricas: Precision, Recall, F1-Score y Accuracy, siendo la más utilizada Accuracy, aunque en los anexos adjuntamos las tablas con el desglose de cada una de las métricas.

Para comenzar, necesitamos tener conocimiento de lo que representa una matriz de confusión,

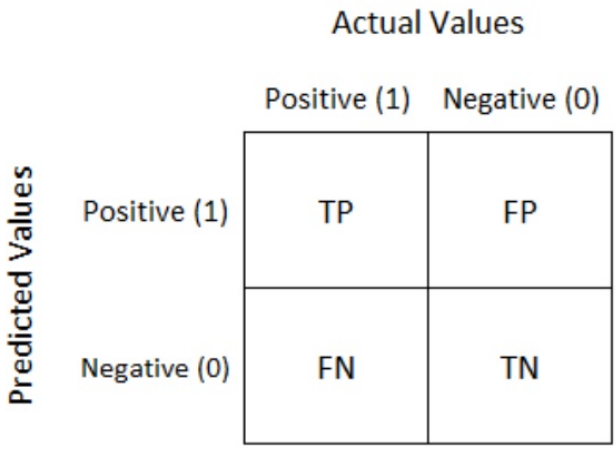

Figura 3.13: Representación de una matriz de confusión.

En la Figura 3.13 se representa una matriz de confusión donde los valores de la diagonal $\{TP, TN\}$ reflejan valores correctamente clasificados, mientras que $\{FP, FN\}$ reflejan valores que no han sido clasificados correctamente.

1. **Precision**: con la métrica de precisión podemos medir la calidad del modelo en las tareas de clasificación. Responde a la pregunta, ¿qué proporción de predicciones positivas es correcta?. Para calcular la precisión

utilizaremos la fórmula,

$$precision = \frac{TP}{TP + FP}. \tag{3.2}$$

2. **Recall**: nos informa sobre la cantidad que el modelo es capaz de identificar. Responde a la pregunta ¿qué proporción de positivos reales se han predicho correctamente?. Utilizamos la expresión,

$$recall = \frac{TP}{TP + FN}. \tag{3.3}$$

3. **F1-Score**: se utiliza para combinar las medidas de precisión y recall en un solo valor,

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{3.4}$$

4. **Accuracy**: mide el porcentaje de casos que el modelo ha acertado,

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{3.5}$$

Etapas Generales

Principalmente, se respetan los mismos pasos para cada desarrollo de cada clasificador, a excepción de la especificación de los hiperparámetros, dado que en cada caso son diferentes,

1. Importamos las librerías necesarias en el entorno de Python.

2. Dividimos los datos en entrenamiento y prueba. Utilizamos una relación del 70 por ciento de la muestra para realizar el entrenamiento. Los datos restantes se utilizan para evaluar la precisión del algoritmo en datos de prueba. Esta proporción se ha calculado realizando varios muestreos de datos reducidos.

3. Aplicamos a cada columna la estandarización de los datos $Z = \frac{x-\mu}{\sigma}$, donde μ es la media de la muestra de entrenamiento y σ es la desviación estándar de la muestra de entrenamiento.

4. Aplicamos el número de componentes calculadas en la Sección 3.4 sobre los datos de entrada, tanto en la partición de entrenamiento como en la de prueba.

5. Definimos los hiperparámetros para conseguir optimizar el modelo.

6. Entrenamos el modelo empleando una validación cruzada de cinco bloques.

7. Se emplean los hiperparámetros óptimos obtenidos en el paso anterior para realizar la clasificación sobre los datos de prueba.

8. Se extrae la precisión del modelo desarrollado sobre los datos de prueba, junto con un reporte de cada una de las medidas de clasificación (Precisión, Recall, F1-Score) y la matriz de confusión.

3.5.1. Máquina de Soporte Vectorial (SVM)

Para la aplicación de la SVM a través de Python seguimos los pasos citados en **Etapas Generales** a excepción de la definición específica de los

hiperparámetros de dicho algoritmo.

1. **El parámetro C**: Ayuda a controlar el equilibrio entre el error de entrenamiento y el margen, ya que puede determinar la penalización por puntos de datos mal clasificados durante el proceso de entrenamiento. Por ello, definimos una lista de valores $C = \{0.1, 1, 10, 100, 500, 1000\}$. Para la definición de dicha lista, hemos realizado muestreos de datos reducidos.

2. **Kernel**: El kernel que vamos a utilizar es el gaussiano,

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right) \tag{3.6}$$

   donde l es la longitud de escala del kernel y $d(\cdot, \cdot)$ es la distancia euclidiana. Este kernel es infinitamente diferenciable.

3. **Gamma**: El parámetro gamma define hasta qué punto llega la influencia de un único ejemplo de entrenamiento, donde los valores bajos significan "lejos" y los valores altos significan "cerca". Los parámetros gamma pueden verse como la inversa del radio de influencia de las muestras seleccionadas por el modelo como vectores de soporte. Por ello definimos la siguiente lista de valores $gamma = \{0.1, 1, 10, 100, 500, 1000\}$. Para la definición de dicha lista, hemos realizado muestreos de datos reducidos.

En la Tabla 3.2 se presenta una tabla con los parámetros calculados en cada agregación temporal para cada ventana de predicción junto con su precisión (Accuracy).

Vemos como mejora la precisión si disminuimos nuestros tiempos de agregación y definimos ventanas de predicción 5 veces superiores a la temporalidad de agregación. También se observa una nítida tendencia en los valores del parámetro C conforme disminuimos la temporalidad de agregación, lo que nos indica que se fomenta un mayor margen y por tanto, una función de decisión más simple.

Agreg.(seg)	**Pred.(seg)**	**C**	**Gamma**	**Acc(Test)**
60	300	10	10	0.6273
60	120	500	1	0.5266
60	60	500	0.1	0.4638
30	150	10	10	0.6144
30	60	100	1	0.5125
30	30	500	0.1	0.4613
15	75	1	10	0.6618
15	30	10	1	0.5159
15	15	1	0.1	0.4525
5	25	10	10	0.7083
5	10	10	10	0.5542
5	5	10	0.1	0.4701

Tabla 3.2: Evolución de los hiperparámetros de la SVM a través de las diferentes agregaciones temporales y ventanas de predicción

3.5.2. Red Neuronal

En este apartado vamos a realizar una red neuronal (perceptron multicapa MLP).

Como anteriormente, seguimos los pasos citados en **Etapas Generales** definiendo específicamente los hiperparámetros del percetrón multicapa,

1. **Función de activación**: Tenemos en cuenta cinco funciones de activación que actúan sobre las capas ocultas. Al realizar un problema de clasificación múltiple, la función de activación de la capa de salida es la función Softmax:

   a) Tangente hiperbólica: $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

   b) Logística: $g(z) = 1/(1 + e^{-z})$

   c) Softmax: $softmax(z)_i = \frac{e^{z_i}}{\sum_{l=1}^{k} e^{z_l}}$

   d) Lineal rectificada (ReLu): $g(z) = \text{máx}\,[0, z]$

2. **Número de capas ocultas**: Se representa no sólo el número de capas ocultas sino también el número de neuronas, es decir, $(5, 10)$ significan dos capas ocultas donde una de ellas tiene 5 neuronas y la otra 10. En nuestro caso vamos a tener en cuenta la siguiente lista: $\{(5, 10), (8, 15), (10, 21), (15, 33), (21, 57)\}$. Dicha configuración de capas y neuronas no se ha elegido de una forma aleatoria sino que se ha hecho un muestreo con un número reducido de datos.

3. **Solver**: donde tenemos en cuenta,

a) *Descenso Estocástico del Gradiente (SGD)*:

$$w \leftarrow w - \eta \left(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w} \right)$$

donde η es el ratio de aprendizaje que controla el tamaño de paso en el espacio de búsqueda de parámetros. *Loss* es la función de pérdida utilizada en la red.

b) *Adam*: Es similar a SGD en el sentido que es un optimizador estocástico, pero puede ajustarse a la cantidad de parámetros actualizados basados en función de estimaciones adaptativas de momentos de orden inferior.

c) *Memoria limitada BFGS (LBFGS)*: es un solver que se aproxima a la matriz de Hessiana la cual representa la derivada parcial de segundo orden de una función. Además se aproxima a la inversa de la matriz de Hessiana para realizar la actualización de parámetros.

Al igual que en la sección anterior, en la Tabla 3.3 tenemos la evolución de los hiperparámetros y la precisión conseguida con dichos hiperparámetros.

Podemos observar que no existe una tendencia clara en la utilización de unos hiperparámetros conforme cambian las ventanas de agregación y predicción. Simplemente observamos que la configuración de capas y neuronas que mejor funciona es (21,57), es decir dos capas ocultas con 21 neuronas en la primera capa oculta y 57 neuronas en la segunda capa oculta. A diferencia de la SVM, no aumenta mucho la precisión al disminuir los tiempos de agregación, aunque si se aprecia un aumento de la precisión en intervalos

de predicción de al menos 5 veces los intervalos de agregación. Si conviene mencionar que el coste computacional del entrenamiento de dicho algoritmo ha sido muy exigente, con unos tiempos de entrenamiento altos.

Agreg.(seg)	**Pred.(seg)**	**activ.**	**Capas**	**Solver**	**Acc(Test)**
60	300	Tanh	(21,57)	LBFGS	0.5334
60	120	Logistic	(21,57)	LBFGS	0.4708
60	60	ReLu	(21,57)	Adam	0.4548
30	150	Tanh	(21,57)	LBFGS	0.5232
30	60	ReLu	(21,57)	LBFGS	0.4742
30	30	Tanh	(21,57)	Adam	0.4430
15	75	Tanh	(21,57)	LBFGS	0.5546
15	30	ReLu	(21,57)	LBFGS	0.4672
15	15	ReLu	(10,21)	SGD	0.4446
5	25	Tanh	(21,57)	Adam	0.5467
5	10	ReLu	(21,57)	Adam	0.4937
5	5	Tanh	(5,10)	SGD	0.4591

Tabla 3.3: Evolución de los hiperparámetros del perceptron multicapa a través de las diferentes agregaciones temporales y ventanas de predicción

Una primera apreciación sobre las redes neuronales es la complejidad de optimización dado que son muy sensibles a hiperparámetros que a su vez están relacionados entre sí entre las distintas capas que conforman la red. También parece tener bastante sensibilidad al dato subyacente, a la limpieza y la dinámica que éste mantiene. En este caso concreto, las precisiones con-

seguidas en la muestra de prueba no son realmente llamativas en ninguno de los tiempos de agregación ni en ningún intervalo de predicción. Se destaca el gran coste computacional realizado para la búsqueda de hiperparámetros y el entrenamiento de la red neuronal, cuestión que debemos tener en cuenta ante la ejecución de algoritmos de alta frecuencia en los mercados financieros, teniendo el riesgo de incurrir en pérdidas si se efectúa una mala modelización de costes en las transacciones.

3.5.3. Bosque Aleatorio

En este apartado vamos a desarrollar un algoritmo de Bosque Aleatorio (RF). Un RF es un meta-estimador que ajusta una serie de clasificadores de árboles de decisión en varias submuestras del conjunto de datos y utiliza promedios para mejorar la precisión predictiva y controlar el sobre ajuste.

Similarmente a los apartados anteriores, y con la utilización de Python, seguimos los pasos citados en **Etapas Generales** y definimos los hiperparámetros para su optimización.

1. **Cantidad de árboles**: definimos la lista $\{300, 500\}$. Esta lista no ha sido elegida al azar, sino que se ha realizado un muestreo con un menor tamaño.

2. **Máximo de características**: es la cantidad de características a considerar al buscar la mejor división,

   a) sqrt: entonces el máximo de características es $\sqrt{n\ caracteristicas}$

   b) log2: entonces el máximo de características es $log_2(n\ caracteristicas)$

c) auto

3. **Máxima Profundidad**: la profundidad máxima del árbol. Definimos la lista $\{4, 5, 6, 7, 8\}$. Esta lista no ha sido elegida de forma aleatoria, sino que se ha realizado varios muestreos con tamaño de muestra pequeños.

4. **Criterio**: la función para medir la calidad de una división. los criterios utilizados son $\{gini, entropy\}$. Si un objetivo es un resultado de clasificación que toma valores $0, 1, ..., K-1$, para el nodo m, entonces sea $p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$ la proporción de observaciones de clase k en el nodo m. Las medidas de impureza, por tanto son,

   a) Gini: $H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$

   b) Entropy: $H(Q_m) = -\sum_k p_{mk} log(p_{mk})$

A continuación la Tabla 3.4 muestra los parámetros y la precisión conseguida con dichos hiperparámetros.

Vemos que la precisión mejora conforme disminuimos los tiempos de la agregación y ampliamos las ventanas de predicción al igual que observamos de forma notable con la SVM. Por otra parte, no se aprecia una tendencia clara en cuanto a la evolución de los parámetros específicos. El número máximo de profundidad permanece prácticamente constante en 8. El criterio sobre la medida de impureza o desorden en los datos se alterna entre *Gini* y *Entropy*. Las precisiones alcanzadas son inferiores a las de la SVM y del orden del algoritmo perceptron multicapa.

Agreg.(seg)	**Pred.(seg)**	**Num.Estim**	**Max.Caract**	**Max.Prof.**	**criterio**	**Acc(Test)**
60	300	500	sqrt	8	Gini	0.5142
60	120	500	sqrt	8	Gini	0.4747
60	60	500	sqrt	8	Entropy	0.4466
30	150	300	sqrt	8	Gini	0.5153
30	60	300	sqrt	8	Gini	0.4774
30	30	500	sqrt	8	Gini	0.4475
15	75	500	sqrt	8	Gini	0.5396
15	30	300	sqrt	8	Entropy	0.4710
15	15	300	sqrt	8	Entropy	0.4417
5	25	300	sqrt	8	Gini	0.5418
5	10	500	sqrt	8	Gini	0.5022
5	5	300	sqrt	6	Entropy	0.4737

Tabla 3.4: Evolución de los hiperparámetros del RF a través de las diferentes agregaciones temporales y ventanas de predicción

3.6. Conclusión Final

En este trabajo podemos observar cierto paralelismo con respecto a la tesis de Aleksandar Palikuca[12], pero existen diferencias sustanciales, tanto desde el punto de vista de ejecución como el propósito del trabajo, haciendo incomparables los resultados entre dichos trabajos. Por un lado, en la tesis si utilizan datos cuyo formato y significado son diferentes a los tratados en este trabajo, es decir, se utilizan datos de la profundidad de mercado obteniendo información de cada transacción en un radio de 60 precios circundantes

al precio de la oferta y la demanda en el instante t. En nuestro caso, sólo tenemos los precios de la oferta y la demanda en el instante t, sin la información que rodea dichos precios. En nuestra base de datos, hemos tenido que construir variables, como el conteo de transacciones, a diferencia de la tesis. El propósito de la tesis es la creación de distintos modelos combinados para medir su precisión en la predicción en intervalos de tiempo de 5, 20 y 60 segundos con la utilización de datos en intervalo de segundo, mientras que en nuestro caso, utilizamos más intervalos de tiempo (60, 30, 15 y 5 segundos) con diferentes ventanas de predicción múltiplos $\Delta t = \{1, 2, 5\}$ de los intervalos de tiempo creados. En la tesis se menciona brevemente alguna exploración de los datos, mientras que en nuestro caso se tienen en cuenta todos los procedimientos necesarios para asegurar en lo posible un tratamiento correcto de los datos con el propósito de alcanzar una generalización de cada uno de los modelos ejecutados (análisis exploratorio, tratamiento de valores atípicos, datos vacíos, tratamiento del desbalanceo de la muestra, reducción de la dimensionalidad,...). Por último, la base de las combinaciones de los algoritmos en la tesis es la red neuronal y la máquina de soporte vectorial, mientras que en nuestro caso no sólo son estos dos últimos sino que introducimos un algoritmo adicional, el bosque aleatorio (RF).

Dicho esto, hemos observado que existe una ventana de predicción de unos segundos, incluso pocos minutos, de una precisión considerable usando datos de intervalos de unos pocos segundos, por lo que nuestra primera conclusión es la utilización de datos en intervalos de tiempo muy pequeños para aumentar la probabilidad en la predicción. También hemos observado, que en la SVM, el parámetro C decae conforme utilizamos datos de intervalos más pequeños,

significando que el modelo se simplifica considerablemente, encajando con la conclusión extraída en el número de componentes principales.

Así mismo, a continuación se muestra la Tabla 3.5 con la comparación de precisiones entre los diferentes clasificadores desarrollados para cada uno de los tiempos de agregación y tiempo de predicción,

Agreg.(seg)	**Pred.(seg)**	**SVM**	**MLP**	**RF**
60	300	0.6273	0.5334	0.5142
60	120	0.5266	0.4708	0.4747
60	60	0.4638	0.4548	0.4466
30	150	0.6144	0.5232	0.5153
30	60	0.5125	0.4742	0.4774
30	30	0.4613	0.4430	0.4475
15	75	0.6618	0.5546	0.5396
15	30	0.5159	0.4672	0.4710
15	15	0.4525	0.4446	0.4417
5	25	0.7083	0.5467	0.5418
5	10	0.5542	0.4937	0.5022
5	5	0.4701	0.4591	0.4737

Tabla 3.5: Evolución de las precisiones de cada clasificador a través de las diferentes agregaciones temporales y ventanas de predicción

Podemos observar que la SVM marca una diferencia sustancial con respecto a los otros dos clasificadores, destacando la precisión alcanzada con datos en intervalos de tiempo de 5 segundos y una ventana de predicción de 25 segundos.

Figura 3.14: Tiempos medios de entrenamiento para cada clasificador en segundos.

Dado que dichos algoritmos se ejecutan en intervalos de tiempo muy cortos, es importante observar los tiempos de entrenamiento medio de cada uno de los algoritmos, mostrados en la Figura 3.14.

Notamos que el perceptron multicapa tiene un coste computacional muy elevado comparado con la SVM y el RF, por lo que lo hace poco idóneo para algoritmos de alta frecuencia.

Apéndice A

Rendimiento en la Clasificación

A.1. SVM

	precision	recall	f1-score
-1.0	0.64	0.63	0.63
0.0	0.60	0.62	0.61
1.0	0.65	0.63	0.64
accuracy			0.63
macro avg	0.63	0.63	0.63
weighted avg	0.63	0.63	0.63

	precision	recall	f1-score
-1.0	0.50	0.53	0.52
0.0	0.55	0.55	0.55
1.0	0.53	0.50	0.52
accuracy			0.53
macro avg	0.53	0.53	0.53
weighted avg	0.53	0.53	0.53

Figura A.1: Resultados de la clasificación de la SVM para tiempos de agregación de 60 segundos y ventanas de predicción de 300 y 120 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.41	0.42	0.41
0.0	0.57	0.59	0.58
1.0	0.41	0.39	0.40
accuracy			0.46
macro avg	0.46	0.46	0.46
weighted avg	0.46	0.46	0.46

	precision	recall	f1-score
-1.0	0.63	0.64	0.63
0.0	0.56	0.57	0.56
1.0	0.66	0.63	0.65
accuracy			0.61
macro avg	0.62	0.61	0.61
weighted avg	0.62	0.61	0.61

Figura A.2: Resultados de la clasificación de la SVM para tiempos de agregación de 60 y 30 segundos y ventanas de predicción de 60 y 150 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.51	0.52	0.52
0.0	0.50	0.54	0.52
1.0	0.53	0.48	0.50
accuracy			0.51
macro avg	0.51	0.51	0.51
weighted avg	0.51	0.51	0.51

	precision	recall	f1-score
-1.0	0.42	0.41	0.41
0.0	0.54	0.58	0.56
1.0	0.42	0.40	0.41
accuracy			0.46
macro avg	0.46	0.46	0.46
weighted avg	0.46	0.46	0.46

Figura A.3: Resultados de la clasificación de la SVM para tiempos de agregación de 30 segundos y ventanas de predicción de 60 y 30 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.67	0.70	0.69
0.0	0.65	0.57	0.61
1.0	0.66	0.71	0.68
accuracy			0.66
macro avg	0.66	0.66	0.66
weighted avg	0.66	0.66	0.66

	precision	recall	f1-score
-1.0	0.52	0.53	0.52
0.0	0.53	0.52	0.52
1.0	0.50	0.50	0.50
accuracy			0.52
macro avg	0.52	0.52	0.52
weighted avg	0.52	0.52	0.52

Figura A.4: Resultados de la clasificación de la SVM para tiempos de agregación de 15 segundos y ventanas de predicción de 75 y 30 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.39	0.47	0.43
0.0	0.54	0.63	0.58
1.0	0.40	0.26	0.32
accuracy			0.45
macro avg	0.44	0.45	0.44
weighted avg	0.45	0.45	0.44

	precision	recall	f1-score
-1.0	0.67	0.80	0.73
0.0	0.73	0.57	0.64
1.0	0.74	0.76	0.75
accuracy			0.71
macro avg	0.71	0.71	0.71
weighted avg	0.71	0.71	0.70

Figura A.5: Resultados de la clasificación de la SVM para tiempos de agregación de 15 y 5 segundos y ventanas de predicción de 15 y 25 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.50	0.65	0.56
0.0	0.63	0.48	0.55
1.0	0.57	0.54	0.55
accuracy			0.55
macro avg	0.57	0.56	0.55
weighted avg	0.57	0.55	0.55

	precision	recall	f1-score
-1.0	0.37	0.49	0.42
0.0	0.60	0.59	0.59
1.0	0.41	0.30	0.35
accuracy			0.46
macro avg	0.46	0.46	0.46
weighted avg	0.46	0.46	0.46

Figura A.6: Resultados de la clasificación de la SVM para tiempos de agregación de 5 segundos y ventanas de predicción de 10 y 5 segundos respectivamente.

A.2. Red Neuronal

	precision	recall	f1-score
-1.0	0.51	0.53	0.52
0.0	0.60	0.54	0.57
1.0	0.50	0.53	0.52
accuracy			0.53
macro avg	0.54	0.53	0.53
weighted avg	0.54	0.53	0.53

	precision	recall	f1-score
-1.0	0.42	0.43	0.43
0.0	0.57	0.53	0.55
1.0	0.43	0.45	0.44
accuracy			0.47
macro avg	0.47	0.47	0.47
weighted avg	0.47	0.47	0.47

Figura A.7: Resultados de la clasificación de la ANN para tiempos de agregación de 60 segundos y ventanas de predicción de 300 y 120 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.39	0.49	0.44
0.0	0.59	0.55	0.57
1.0	0.39	0.32	0.35
accuracy			0.45
macro avg	0.46	0.45	0.45
weighted avg	0.46	0.45	0.45

	precision	recall	f1-score
-1.0	0.50	0.55	0.52
0.0	0.56	0.48	0.52
1.0	0.52	0.54	0.53
accuracy			0.52
macro avg	0.53	0.52	0.52
weighted avg	0.53	0.52	0.52

Figura A.8: Resultados de la clasificación de la ANN para tiempos de agregación de 60 y 30 segundos y ventanas de predicción de 60 y 150 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.44	0.47	0.45
0.0	0.55	0.51	0.53
1.0	0.44	0.45	0.44
accuracy			0.47
macro avg	0.48	0.47	0.48
weighted avg	0.48	0.47	0.48

	precision	recall	f1-score
-1.0	0.40	0.43	0.42
0.0	0.52	0.57	0.55
1.0	0.39	0.33	0.36
accuracy			0.44
macro avg	0.44	0.44	0.44
weighted avg	0.44	0.44	0.44

Figura A.9: Resultados de la clasificación de la ANN para tiempos de agregación de 30 segundos y ventanas de predicción de 60 y 30 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.54	0.59	0.56
0.0	0.60	0.49	0.54
1.0	0.53	0.58	0.56
accuracy			0.55
macro avg	0.56	0.55	0.55
weighted avg	0.56	0.55	0.55

	precision	recall	f1-score
-1.0	0.44	0.47	0.45
0.0	0.56	0.50	0.52
1.0	0.42	0.44	0.43
accuracy			0.47
macro avg	0.47	0.47	0.47
weighted avg	0.47	0.47	0.47

Figura A.10: Resultados de la clasificación de la ANN para tiempos de agregación de 15 segundos y ventanas de predicción de 75 y 30 segundos respectivamente.

	precision	recall	f1-score		precision	recall	f1-score
-1.0	0.38	0.50	0.43	-1.0	0.52	0.49	0.50
0.0	0.55	0.60	0.57	0.0	0.63	0.56	0.59
1.0	0.40	0.24	0.30	1.0	0.50	0.59	0.55
accuracy			0.44	accuracy			0.55
macro avg	0.44	0.45	0.43	macro avg	0.55	0.55	0.55
weighted avg	0.44	0.44	0.43	weighted avg	0.55	0.55	0.55

Figura A.11: Resultados de la clasificación de la ANN para tiempos de agregación de 15 y 5 segundos y ventanas de predicción de 15 y 25 segundos respectivamente.

	precision	recall	f1-score		precision	recall	f1-score
-1.0	0.43	0.54	0.48	-1.0	0.37	0.49	0.42
0.0	0.61	0.61	0.61	0.0	0.60	0.59	0.59
1.0	0.44	0.33	0.38	1.0	0.41	0.30	0.35
accuracy			0.49	accuracy			0.46
macro avg	0.49	0.49	0.49	macro avg	0.46	0.46	0.46
weighted avg	0.50	0.49	0.49	weighted avg	0.46	0.46	0.46

Figura A.12: Resultados de la clasificación de la ANN para tiempos de agregación de 5 segundos y ventanas de predicción de 10 y 5 segundos respectivamente.

A.3. Bosque Aleatorio

	precision	recall	f1-score
-1.0	0.49	0.47	0.48
0.0	0.52	0.70	0.60
1.0	0.53	0.36	0.43
accuracy			0.51
macro avg	0.51	0.51	0.50
weighted avg	0.52	0.51	0.50

	precision	recall	f1-score
-1.0	0.44	0.40	0.42
0.0	0.51	0.68	0.58
1.0	0.46	0.34	0.39
accuracy			0.47
macro avg	0.47	0.47	0.46
weighted avg	0.47	0.47	0.46

Figura A.13: Resultados de la clasificación de RF para tiempos de agregación de 60 segundos y ventanas de predicción de 300 y 120 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.39	0.39	0.39
0.0	0.51	0.66	0.58
1.0	0.40	0.28	0.33
accuracy			0.45
macro avg	0.44	0.45	0.43
weighted avg	0.44	0.45	0.43

	precision	recall	f1-score
-1.0	0.50	0.49	0.49
0.0	0.55	0.57	0.56
1.0	0.50	0.49	0.49
accuracy			0.52
macro avg	0.51	0.52	0.51
weighted avg	0.51	0.52	0.51

Figura A.14: Resultados de la clasificación de RF para tiempos de agregación de 60 y 30 segundos y ventanas de predicción de 60 y 150 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.41	0.37	0.39
0.0	0.54	0.58	0.56
1.0	0.39	0.39	0.39
accuracy			0.45
macro avg	0.44	0.45	0.45
weighted avg	0.44	0.45	0.45

	precision	recall	f1-score
-1.0	0.41	0.37	0.39
0.0	0.54	0.58	0.56
1.0	0.39	0.39	0.39
accuracy			0.45
macro avg	0.44	0.45	0.45
weighted avg	0.44	0.45	0.45

Figura A.15: Resultados de la clasificación de RF para tiempos de agregación de 30 segundos y ventanas de predicción de 60 y 30 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.50	0.58	0.54
0.0	0.60	0.54	0.57
1.0	0.53	0.50	0.51
accuracy			0.54
macro avg	0.54	0.54	0.54
weighted avg	0.54	0.54	0.54

	precision	recall	f1-score
-1.0	0.44	0.45	0.44
0.0	0.55	0.56	0.55
1.0	0.42	0.40	0.41
accuracy			0.47
macro avg	0.47	0.47	0.47
weighted avg	0.47	0.47	0.47

Figura A.16: Resultados de la clasificación de RF para tiempos de agregación de 15 segundos y ventanas de predicción de 75 y 30 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.38	0.42	0.40
0.0	0.54	0.61	0.57
1.0	0.38	0.30	0.34
accuracy			0.44
macro avg	0.44	0.44	0.44
weighted avg	0.44	0.44	0.44

	precision	recall	f1-score
-1.0	0.51	0.47	0.49
0.0	0.62	0.61	0.62
1.0	0.50	0.54	0.52
accuracy			0.54
macro avg	0.54	0.54	0.54
weighted avg	0.54	0.54	0.54

Figura A.17: Resultados de la clasificación de RF para tiempos de agregación de 15 y 5 segundos y ventanas de predicción de 15 y 25 segundos respectivamente.

	precision	recall	f1-score
-1.0	0.44	0.44	0.44
0.0	0.62	0.61	0.61
1.0	0.45	0.46	0.45
accuracy			0.50
macro avg	0.50	0.50	0.50
weighted avg	0.50	0.50	0.50

	precision	recall	f1-score
-1.0	0.38	0.38	0.38
0.0	0.63	0.56	0.59
1.0	0.42	0.48	0.45
accuracy			0.47
macro avg	0.48	0.47	0.47
weighted avg	0.48	0.47	0.48

Figura A.18: Resultados de la clasificación de RF para tiempos de agregación de 5 segundos y ventanas de predicción de 10 y 5 segundos respectivamente.

Bibliografía

[1] Cortes, C., and Vapnik, V. Support vector networks. *Machine Learning 20* (1995), 273–297.

[2] Ding, X., Zhang, Y., Liu, T., and Duan, J. Deep learning for event-driven stock prediction.

[3] Dubey, S. R., Singh, S. K., and Chaudhuri, B. B. Activation functions in deep learning: A comprehensive survey and benchmark.

[4] Fama, E. Efficient capital markets: A review of theory and empirical work. *Annual Meeting of the American Finance Association New York* (1970).

[5] Fletcher, T., Hussain, Z., and Shawe-Taylor, J. Multiple kernel learning on the limit order book. *Workshop on Applications of Pattern Analysis* (2010).

[6] García, R. El perceptrón: Una red neuronal artificial para clasificar datos.

[7] Huang, W., Nakamori, Y., and Wang, S.-Y. Forecasting stock market movement direction with support vector machine. *Computers and Operations Research 32* (2005), 2513–2522.

[8] James, G., Witten, D., Hastie, T., and Tibshirani, R. *An Introduction to Statistical Learning with Applications in R*. 2021.

[9] Kercheval, A., and Zhang, Y. Modeling high-frequency limit order book dynamics with support vector machines.

[10] Nasr, G., and adn E.A.Badr, C. Cross entropy error function in neural networks: Forecasting gasoline demand.

[11] Neely, C., Weller, P., and Dittmar, R. Is technical analysis in the foreign exchange market profitable? a genetic programming approach. *The Journal of Financial and Quantitative Analysis 32* (1997), 405–426.

[12] Palikuca, A., and Seidl, T. *Predicting High Frequency Exchange Rates using Machine Learning*. 2016.

[13] Rokach, L., and Maimon, O. Decision trees.

[14] Shalev-Shwartz, S., and Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. 2014.

[15] Yu, T.-W., and Yu, C.-C. Forecasting stock market with neural networks. *Statist. Soc B* (2009).