ESTUDIO DE TÉCNICAS DE MODELIZACIÓN ANALÍTICA EN EL MERCADO DE DIVISAS

SERGIO MORENO GONZÁLEZ

GRADO EN MATEMÁTICAS Y ESTADÍSTICA. FACULTAD DE MATEMÁTICAS
UNIVERSIDAD COMPLUTENSE DE MADRID

Madrid, 2 de agosto de 2022

Director:

Daniel Vélez Serrano

Autorización de difusión

Sergio Moreno González

Madrid, 2 de agosto de 2022

El arriba firmante, matriculado en el Grado de Matemáticas y Estadística de la Facultad de Matemáticas, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Grado: "ESTUDIO DE TÉCNICAS DE MODELIZACIÓN ANALÍTICA EN EL MERCADO DE DIVISAS", realizado durante el curso académico 2021-2022 bajo la dirección de Daniel Vélez Serrano en el Departamento de Estadística e Investigación Operativa, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen

En este trabajo se pretende estudiar las fluctuaciones del euro con respecto al dólar, entrenar varios modelos de Machine Learning con los correspondientes datos del mercado para predecir el tipo de cambio en el futuro y crear algunos algoritmos de inversión que permitan obtener beneficios de esas predicciones. Los datos del mercado serán analizados, estudiando los más atípicos y los posibles errores que se hayan podido cometer en los mismos. También observaremos si existen correlaciones entre las variables utilizadas además de introducir nuevas variables que ofrezcan más información para el estudio. Los modelos de Machine Learning que utilizaremos para este trabajo serán el de regresión logística múltiple, el adaptative boosting, el random forest y la red neuronal, cuatro tipos de aprendizaje supervisado que presentaremos matemáticamente al principio del trabajo y optimizaremos según la conveniencia del estudio. Finalmente, se hará una comparativa entre los modelos enfocándonos en las ventajas y desventajas de cada uno en base a sus resultados. También se compararán los algoritmos generados para identificar el que mayores beneficios reporte.

Por un lado nos centraremos en los resultados de cada modelo para obtener conclusiones sobre su rendimiento en la predicción del mercado de divisas. Para ello utilizaremos diferentes medidas estadísticas como la curva ROC, el F1-score o la matriz de confusión. Por otro lado, crearemos algoritmos basados en las predicciones hechas por los modelos y compararemos sus resultados para ver si es posible generar uno que ofrezca buenas predicciones. Serán útiles conceptos como la estrategia Benchmarking o los indicadores FOREX.

Palabras clave

Machine Learning, FOREX, Euro-Dólar, algoritmo, modelización, predicción

Abstract

The purpose of this project is to study the fluctuations of the euro with the dollar, train several Machine Learning models with the corresponding market data capables of predicting the type of change in the future and create some investment algorithms that allow to obtain profits of those predictions. The market data will be analyzed, studying atypical observations and possible errors in them. We will also observe if there is variable correlation in addition to introducing new variables that will offer more information for the study. The Machine Learning models that we will use for this work will be multiple logistic regression, adaptive boosting, random forest and neural network, four types of supervised learning that we will optimize according to the convenience of the study. Finally, a comparison will be made between the models, focusing on the advantages and disadvantages of each one based on their results. The generated algorithms will also be compared to identify the one that reports the greatest benefits.

On the one hand we will focus on the results of each model to obtain conclusions about their performance in forex market prediction. For this we will use different statistic measures such as the ROC curve, the F1-score or the confusion matrix. On the other hand, we will create algorithms based on the predictions made by the models and we will compare their results to see if it is possible to generate one that offers good predictions. They will be useful concepts such as Benchmarking strategy or FOREX indicators.

Keywords

Machine Learning, FOREX, Euro-Dollar, modelling, prediction

Índice general

Agradecimientos

Agradecimientos.....

Capítulo 1

Introducción

1.1. Motivación

El origen de este trabajo surge del creciente uso de la tecnología y modelos de predicción basados en el aprendizaje automático. Es verdad que son muy útiles para realizar clasificaciones y predicciones de conjuntos de datos cada vez más complejos pero queremos ver su verdadero alcance con algo tan volátil como es el mercado de divisas. A lo largo de la carrera siempre hemos trabajado con datasets muy sencillos o que estaban preparados para obtener muy buenos resultados con los modelos que aplicábamos de Machine Learning. Por esto yo siempre tenía la duda de qué pasaría si se tuviera un conjunto de datos en el estos modelos no pudieran dar mejores resultados. ¿Se podría hacer algo diferente para conseguir buenas predicciones a pesar de que los modelos por sí solos no lo consiguieran? O, sin embargo, ¿había que conformarse con esos resultados y dejar el problema como imposible? De aquí que decidiera intentar predecir los tipos de cambio entre el euro y el dólar, pues es bien sabido que pocas cosas hay más difíciles de predecir. Es un mercado que mueve mucho capital y con el que se puede ganar mucho dinero. Por ello ha habido tantos esfuerzos y hay tanta gente que trabaja continuamente para conseguir buenas predicciones. Si se consigue predecir de manera constante y fiable se va a poder obtener grandes beneficios económicos de estas predicciones.

1.2. Objetivos

El primer objetivo de este trabajo es el de crear varios modelos de Machine Learning basados en la regresión logística múltiple, el adaptative boosting, el random forest y la red neuronal que sean capaces de realizar las mejores predicciones posibles en el conjunto de datos financieros proporcionado. También compararemos los resultados de los modelos entre sí para encontrar y describir las limitaciones de cada uno. Por ello hemos seleccionado un modelo de cada uno de los sectores más importantes del Machine Learning. La regresión logística múltiple es el ejemplo por excelencia de un modelo lineal, el adaptative boosting es uno de los modelos que utilizan la técnica de creación boosting, mientras que el random forest utiliza la de bagging y, por último, la red neuronal que es un modelo no lineal. Más tarde, y utilizando estos modelos querremos

estudiar si es posible usar todos los resultados para desarrollar un algoritmo que identifique de manera fiable oportunidades de inversión. Este será el segundo objetivo y el más importante.

Por lo tanto, por un lado presentaremos matemáticamente los modelos y las métricas que vamos a utilizar para evaluarlos. Desarrollaremos estos modelos y ajustaremos lo mejor posible sus parámetros para optimizarlos al máximo y así intentar obtener un alto porcentaje de aciertos en las predicciones. Por otro lado, utilizaremos las predicciones ofrecidas por los modelos para desarrollar diferentes algoritmos que encuentre puntos interesantes de compra y de venta y evaluaremos sus rendimientos para elegir el que más beneficios pueda producir. Éste, lo compararemos con algunas medidas económicas, comola estrategia Benchmark, para evaluar si hemos conseguido un algoritmo útil.

Así pues, el fin último de este proyecto es el de ser capaces de analizar un conjunto de datos complicado, ajustarle modelos de Machine Learning y, aunque no den resultados muy positivos, mostrar una forma de tratar los datos obtenidos de los modelos para obtener, aún así, buenas predicciones.

1.3. Estructura del documento

Primero, presentaremos algunas definiciones clave del mercado de divisas para poder seguir sin problema todo el trabajo. Posteriormente definiremos matemáticamente las métricas que vamos a utilizar así como los cuatro modelos presentados anteriormente. Después, analizaremos los datos comprobando que estos sean coherentes y no presenten inconsis- tencias. En caso de no serlo, estudiaremos cada caso, inputando nuevos valores o manteniendo los que hay según el caso. Entonces haremos una primera prueba con una red neuronal para encontrar fallos en nuestro dataset que podamos solucionar antes de evaluar los demás métodos. Si encontramos alguno lo corregiremos, definiremos nuevas variables y a continuación, desarrollaremos nuestros modelos optimizándolos para obtener las mejores predicciones. Por último, tras evaluar cada uno de los modelos, crearemos varios algoritmos con las predicciones obtenidas para identificar oportunidades de inversión y los compararemos con la estrategia Benchmarking para ver si hemos conseguido algún algoritmo útil que genere buenos beneficios.

Capítulo 2

Estado del arte

Desde que se observó el potencial de los modelos de Machine Learning para la predicción de eventos, ya sea a través de modelos de regresión, modelos de k vecinos más cercanos, o los árboles de decisión, han sido muchos los entornos en los que se han implantado. Sin embargo hasta ahora los resultados obtenidos en conjuntos de datos muy volátiles o que dependían de muchos factores (la bolsa, el mercado de futuros, *etc*) han sido poco eficaces. Debido al desarrollo de modelos cada vez más potentes como las redes neuronales y al avance de los ordenadores, nuevas líneas de investigación se han abierto para intentar mejorar las predicciones sobre esta clase de problemas.

Capítulo 3

Conceptos teóricos

Antes de presentar el caso práctico se van a explicar ciertos conceptos teóricos. Definiremos los tres tipos de análisis de mercado, las fórmulas de las métricas que se utilizarán y su interpretación, los desarrollos matemáticos de los cuatro modelos que generaremos: regresión logística, adaptative boosting, random forest y red neuronal, y los algoritmos que identificarán las oportunidades de inversión a partir de las predicciones realizadas por los modelos.

3.1. Tipos de análisis en FOREX

Es importante para entender este trabajo saber los tres tipos de análisis que existen en el mercado de FOREX.

Definición 3.1 El análisis técnico[21] es el proceso por el que se estudian los gráficos de las pautas de precios pasadas en busca de pistas sobre la dirección de los futuros movimientos de los precios.

La teoría es que una persona puede observar los movimientos de precios históricos y determinar las condiciones comerciales actuales y el movimiento de precios potencial. La principal evidencia para usar el análisis técnico es que toda la información actual del mercado se refleja en el precio. Esto simplemente significa que toda la información fundamental conocida tiene un precio de mercado actual.

Si el precio refleja toda la información que existe, entonces la acción del precio es todo lo que realmente se necesita para realizar una operación. El análisis técnico analiza el ritmo, el flujo y las tendencias en la acción del precio fijándose en los datos pasados.

Por todo esto el análisis técnico será el tipo de análisis que utilicemos en la construcción de los modelos. Utilizaremos los datos del tipo de cambio entre el euro y el dólar, así como diferentes indicadores muy utilizados por los analistas técnicos para crear un conjunto de datos con el que entrenar los modelos.

Definición 3.2 El análisis fundamental[14] es la interpretación y análisis de reportes estadísticos e indicadores económicos. Los sucesos como los cambios a las tasas de interés, los reportes de

empleo y los indicadores de la inflación son solo algunos ejemplos delo que engloba el análisis fundamental. El Análisis Fundamental en el Forex se refiere al mismo tipo de estudio, pero con un enfoque más marcado a los factores que puedan afectar las paridades entre divisas.

Es por esto que los inversionistas Forex deben prestar mucha atención a los indicadores económicos que puedan tener un impacto en las economías de los países, ya que ultimadamente afectarán a sus monedas. Dada la importancia del Análisis Fundamental Forex, los inversionistas deben saber con anticipación cuando se llevarán a cabo los sucesos.

La diferencia entre el análisis fundamental tradicional y el Análisis Fundamental Forex es que en el tradicional es más fácil estudiar los factores que puedan afectar una empresa para tratar de predecir la variación en el precio de sus acciones a futuro. Pero en el Análisis Fundamental Forex se están analizando dos economías a nivel nacional.

Por este nivel tan alto de interpretación y por el hecho de que lo ideal sería saber los sucesos con anticipación y esa es una tarea que no podemos implementar en los modelos, este tipo de análisis no lo tendremos en cuenta.

Definición 3.3 El análisis de sentimiento[7] del mercado es el proceso de conocer la opinión de los mercados expresada por los traders que controlan los mayores volúmenes de negociación.

Como conseguir conocer con precisión esto es una tarea imposible de hacer consistentemente tampoco lo tendremos en cuenta en nuestros modelos.

3.2. Métricas

Para evaluar de manera objetiva los resultados de los diferentes modelos y poder hacer comparaciones entre ellos se utilizarán una serie de métricas, unas para el caso binario y otras para el caso continuo.

3.2.1. Caso binario

Para definir correctamente las métricas del caso binario, denotaremos:

$$\begin{aligned} tp &= cantidad\ de\ valores\ que\ el\ modelo\ predice\ 1\ y\ son\ 1 \\ fp &= cantidad\ de\ valores\ que\ el\ modelo\ predice\ 1\ y\ son\ 0 \\ fn &= cantidad\ de\ valores\ que\ el\ modelo\ predice\ 0\ y\ son\ 1 \\ tn &= cantidad\ de\ valores\ que\ el\ modelo\ predice\ 0\ y\ son\ 0 \end{aligned}$$

Definición 3.4 La métrica exactitud (accuracy) mide el porcentaje de casos que el modelo ha acertado en total.

$$acc = \frac{tp + tn}{tn + fp + fn + tp} \tag{3.1}$$

Definición 3.5 La métrica sensibilidad (sensitivity), también conocida como recall o TPR (tasa positiva real), mide el porcentaje de unos que es capaz de identificar el modelo. En nuestro caso, la cantidad de veces que el mercado está alcista.

$$sen = \frac{tp}{fn + tp} \tag{3.2}$$

Definición 3.6 La métrica especificidad (specificty) o TNR (tasa negativa real) es homóloga a sensitivity pero para ceros. Mide la capacidad de un modelo para dar con las veces en las que el mercado está bajista.

$$spe = \frac{tn}{fp + tn} \tag{3.3}$$

Definición 3.7 La métrica precisión (precision) mide la calidad del modelo a la hora de clasificar, en nuestro caso, la cantidad de veces en las que el mercado es alcista entre las veces en las que realmente el mercado sube con respecto al día anterior.

$$ppv = \frac{tp}{fp + tp} \tag{3.4}$$

Definición 3.8 El valor F1 se utiliza para combinar las medidas de precision y sensitivity en un solo valor. Esto es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones, aunque tiene una mala interpretabilidad. F1 alcanza su máximo en 1, y su mínimo en 0, donde el primer valor simboliza un modelo infalible (un ideal imposible), y 0 significa que el algoritmo falla todo el tiempo.

$$F1 = 2\frac{sen * ppv}{sen + ppv} \tag{3.5}$$

Definición 3.9 La matriz de confusión es una matriz que indica por grupos el número de aciertos y fallos de los modelos de clasificación.

Definición 3.10 La curva ROC (curva de funcionamiento del receptor) es una representación gráfica de la sensibilidad frente a la especificidad para un sistema clasificador binario según se varía el umbral de discriminación.

Definición 3.11 El área bajo la curva ROC posee un valor comprendido entre 0,5 y 1, donde 1 representa un valor diagnóstico perfecto y 0,5 es una prueba sin capacidad discriminatoria diagnóstica.

3.2.2. Caso continuo

Las métricas a las que recurriremos en el caso continuo no se pueden basar en el conteo de predicciones acertadas y fallidas. Para definir correctamente las métricas del caso continuo, definiremos.

$$y_j = valores\ reales\ a\ predecir$$
$$\hat{y}_j = predicciones\ realizadas\ por\ el\ modelo$$

Definición 3.12 El error absoluto medio (mean absolute error) es la diferencia promedio entre las observaciones (valores verdaderos) y la salida del modelo (predicciones). El signo de estas diferencias se ignora para que no se produzcan cancelaciones entre valores positivos y negativos.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \tag{3.6}$$

Definición 3.13 El error cuadrático medio (MSE) mide la cantidad de error en los modelos estadísticos. Evalúa la diferencia cuadrática promedio entre los valores observados y predichos. Cuando un modelo no tiene error, el MSE es igual a cero. A medida que aumenta el error del modelo, aumenta su valor, penalizando de forma más grave que con el MAE las predicciones que se alejan de forma extrema de los valores reales. El error cuadrático medio también se conoce como desviación cuadrática media (MSD).

$$\mathrm{MSE} = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2 \tag{3.7}$$

Definición 3.14 El coeficiente de determinación, también denominado puntuación R2 ($R^2\ score$), se utiliza para evaluar el rendimiento de un modelo de regresión lineal. Es la cantidad de variación en el atributo dependiente de salida que es predecible a partir de las variables independientes de entrada. Se utiliza para comprobar cómo de buenas son las predicciones hechas por el modelo, dependiendo de la proporción de desviación total de los resultados descritos por el modelo.

$$R^2 = 1 - \frac{\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}{\sum_{j=1}^{n}(y_j - \bar{y})^2} \tag{3.8}$$

Siendo $\bar{y}$ la media de las observaciones reales.

También desarrollamos unas métricas que medían la precisión del modelo mostrando el porcentaje de aciertos exactos, con un margen de 5, 10, 25 y 50 *pips*. El código utilizado para ello al igual que para generar el resto de métricas se puede observar en el apéndice **??**.

3.3. Modelos

3.3.1. Regresión logística

La Regresión Logística Simple, desarrollada por David Cox en 1958[8], es un método de regresión que permite estimar la probabilidad de una variable cualitativa binaria en función de una variable cuantitativa. Una de las principales aplicaciones de la regresión logística es la de clasificación binaria, en el que las observaciones se clasifican en un grupo u otro dependiendo del valor que tome la variable empleada como predictor.

En consecuencia, buscándose que el modelo devuelva esa probabilidad que te digo: La idea en la regresión logística[11,12] es buscar un modelo de regresión que devuelva un valor probabilístico (entre 0 y 1), pero un modelo de regresión lineal $Y = a + bX$ no se puede plantear porque este modelo asume que la Y puede tomar cualquier valor de la recta real. Por lo tanto, para obtener la probabilidad

$$\pi(x) = p(y \mid X = x) = p(Y = 1 \mid X = x)$$

lo que se hace es linealizar dicha probabilidad utilizando a tal fin la siguiente ecuación, denominada la ecuación de la función logística:

$$\text{logit}(\pi(x)) = \log\left(\frac{\pi(x)}{1-\pi(x)}\right) \tag{3.9}$$

Esta nueva cantidad sí se mueve en toda la recta real, porque cuando $\pi(\text{x}) = 0$ vale $-\infty$ y cuando $\pi(\text{x}) = 1$ vale ∞. Con esta nueva Y, se puede plantear la regresión lineal:

$$\text{logit}(\pi(x)) = \alpha + \beta x + \varepsilon' \tag{3.10}$$

y haciendo unas cuentas sencillas en 3.10 se obtiene:

$$\frac{\pi(x)}{1-\pi(x)} = e^{\alpha+\beta x+\varepsilon'} \Leftrightarrow \frac{1-\pi(x)}{\pi(x)} = e^{-\alpha-\beta x+\varepsilon} \Leftrightarrow$$

$$\Leftrightarrow \frac{1}{\pi(x)} = 1 + e^{-\alpha-\beta x+\varepsilon} \Leftrightarrow \pi(x) = \frac{1}{1+e^{-\alpha-\beta x+\varepsilon}} \; siendo\, \varepsilon = -\varepsilon' \tag{3.11}$$

que es la ecuación del modelo de regresión logística cuando la función de linkaje es la logística. Es importante tener en cuenta que, aunque la regresión logistica permite clasificar, se trata de un modelo de regresión que modela el logaritmo de la probabilidad de pertenecer a cada grupo. La asignación final se hace en función de las probabilidades predichas.

La función de pérdida de regresión logística (es decir, la probabilidad logarítmica negativa)es esencialmente una regresión en las probabilidades logarítmicas. Cambiar eso a una función de pérdida por mínimos cuadrados hará que sea una regresión lineal, que perderá tres cosas:

1. La interpretación de los coeficientes de regresión en términos de probabilidades logarítmi-

cas.

2. interpretación de las predicciones del modelo como probabilidades logarítmicas (o cuando exponenciadas), como probabilidades.

3. que la regresión lineal no limita las predicciones del modelo entre 0 y 1, por lo que fácilmente puede hacer predicciones menores que 0 o mayores que 1.

Por lo que para obtener el β^T óptimo (el que minimiza la función de pérdida de la regresión logística) se recurrirá al método de máxima verosimilitud.

Partiremos de la probabilidad conjunta,

$$P(X, \vec{y}|\beta) = [\sigma(f(X))]^{\vec{y}}[1 - \sigma(f(X))]^{1-\vec{y}} \tag{3.12}$$

Si desarrollamos, se tiene:

$$P(\vec{x_1}, ..., \vec{x_N}; y_1, ..., y_N|\beta) = \prod_{i=1}^{N}[\sigma(f(\vec{x_i}))]^{y_i}[1 - \sigma(f(\vec{x_i}))]^{1-y_i}] \tag{3.13}$$

Para simplicar denotamos $D = \vec{x_1}, ..., \vec{x_N}; y_1, ..., y_N$ y $\mu_i = \sigma(f(\vec{x_i}))$ en 3.13 y obtenemos:

$$P(D|\beta) = \prod_{i=1}^{N}[\mu_i]^{y_i}[1 - \mu_i]^{1-y_i}] \tag{3.14}$$

Maximizar 3.14 es como minimizar $-logP(D|\beta)$. Para ello debemos presentar la siguiente definición:

Definición 3.15 Se define la función negative log-likelihood como:

$$NLL(\beta) = -\sum_{i=1}^{N} y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i) \tag{3.15}$$

El gradiente de 3.15 queda entonces como,

$$\nabla_\beta NLL(\beta) = \sum_{i=1}^{N}(\mu_i - y_i)\vec{x_i} \tag{3.16}$$

Como $\nabla_\beta \nabla_\beta^T NLL(\beta)$ es una matriz semidefinida positiva, la función es convexa y si encontramos un mínimo sabemos que es un mínimo global. Pero $\nabla_\beta NLL(\beta) = 0$ no tiene solución directa, por lo que se precisará de un método de optimización para aproximar.[20]

Algoritmo 3.1 El algoritmo del descenso de gradiente consta de los siguientes pasos:

1. Se parte de un w_0 arbitrario.

2. Se calcula el gradiente y se avanza en la dirección indicada a w_1 en función de $\vec{w_{t+1}} = \vec{w_t} - \eta_t \nabla_\beta NLL(\beta)$

3.3.2. Adaptative boosting

La idea intuitiva detrás del algoritmo de adaptative boosting[5,15], desarrollado por Freund Y y Schapire R en 1995 en su artículo *A decision-theoretic generalization of on-line learning and an application to boosting*[27] reside en:

- En vez de entrenar modelos paralelos (como se hace con las técnicas de bagging), se entrenan los modelos secuencialmente.
- Cada modelo debe centrarse en dónde desempeñó mal el clasificador anterior.

Esta idea descrita arriba se puede describir como:

1. Entrenar el modelo h_1 en todo el conjunto de datos.
2. Entrenar el modelo h_2 con mayor cantidad de datos de la región en la que h_1 desempeña mal.
3. Entrenar el modelo h_3 con mayor cantidad de datos en la región donde $h_1 \neq h_2$.

Entonces, se puede resumir la idea clave del algoritmo de adaptative boosting como que el algoritmo se basa en una sequencia de modelos donde cada modelo da unos pesos mayores a los errores cometidos por el modelo anterior.

Los algoritmos de boosting entrenan una serie de algoritmos de bajo rendimiento, llamados aprendices débiles, ajustando la métrica de error con el tiempo. Los aprendices débiles son algoritmos cuya tasa de error está ligeramente por debajo del 50 %.

Con estas ideas ya claras podemos describir los pasos del algoritmo adaptative boosting.

Paso 1: Sea $w_t(i) = \frac{1}{N}$ donde N denota el número de muestras de entrenamiento, y sea T el número de iteraciones elegidas.

Paso 2: Para cada t en T:
a. Se elije h^t el clasificador débil que minimiza ϵ_t

$$\epsilon_t = \sum_{i=1}^{m} w_t(i) \left[y_i \neq h\left(x_i\right) \right] \tag{3.17}$$

b. Se calcula el peso del clasificador elegido

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t} \tag{3.18}$$

c. Se actualizan los pesos de las muestras de entrenamiento $w_{t+1}(i)$ y se regresa al paso *a.*

Paso 3: Se obtiene el modelo final mediante la fórmula

$$H(x) = \text{sign}\left(\alpha^1 h^1(x) + \alpha^2 h^2(x) + \ldots + \alpha^T h^T(x)\right) \tag{3.19}$$

3.3.3. Random Forest

El random forest[5] es un algoritmo desarrollado por Leo Breiman y Adele Cutler[6] de aprendizaje supervisado que utiliza la técnica de aprendizaje por conjuntos o bagging (la técnica de bagging utiliza varios algoritmos a la vez o un único algoritmo varias veces para hacer que un modelo sea más potente) para construir varios árboles de decisión en puntos de datos aleatorios. Luego se promedian sus predicciones. Tomando el valor promedio de las predicciones realizadas por varios árboles de decisión y luego prediciendo el resultado final. Este tipo de modelos está basado en el teorema central del límite (TCL)

Teorema 3.1 (Teorema Central del Límite) Sea $\{X_n\}_{n=1}^{\infty}$ una sucesión de variables aleatorias independientes e idénticamente distribuidas con $E(X_n) = \mu$ y $\text{Var}(X_n) = \sigma^2$ para todo n. Entonces:

$$\frac{\sum_{i=1}^{n} X_i - n\mu}{\sigma\sqrt{n}} \xrightarrow{\mathcal{L}} N(0,1) \tag{3.20}$$

En particular, si dividimos el numerador y el denominador de 3.20 por n, se obtiene el siguiente corolario.

Corolario 3.1 Una condición equivalente al teorema del Teorema Central del Límite es:

$$\frac{\bar{X}_n - \mu}{\frac{\sigma}{\sqrt{n}}} \xrightarrow{\mathcal{L}} N(0,1) \tag{3.21}$$

Es decir, que si se tienen n variables aleatorias independientes e idénticamente distribuidas con $E(X_n) = \mu$ y $\text{Var}(X_n) = \sigma^2$ para todo n, se tiene que la media de esas variables converge en ley a una normal de media μ y varianza σ^2/n. El algoritmo de random forest selecciona aleatoriamente un conjunto de árboles de decisión del conjunto de entrenamiento. Cada árbol es generado con una muestra aleatoria del conjunto de entrenamiento, y realiza una predicción. El conjunto de esas predicciones actúa como la sucesión de variables aleatorias independientes idénticamente distribuidas. La media de esas predicciones cumple el corolario 3.21.

De esta manera el algoritmo de random forest crea árboles de decisión en muestras de datos, luego obtiene la predicción de cada uno de ellos y finalmente selecciona la mejor solución mediante votación.

Con esto explicado podemos describir los pasos del algoritmo de random forest.

Paso 1: Se selecciona un conjunto de muestras aleatorias del dataset de entrenamiento.

Paso 2: Se construye un árbol de decisión para cada muestra. Posteriormente, al introduir una observación de la muestra de test se obtendrá el resultado de la predicción de cada árbol de decisión.

Paso 3: Se clasifican todas las predicciones y se hace un recuento de la cantidad de predicciones que caen en cada grupo.

Paso 4: Se escoge la predicción que más repeticiones tiene consiguiendo así un método robusto a partir de muchos otros que no lo son tanto.

3.3.4. Red neuronal

El primer algoritmo que presentaba una red neuronal simple se llamó Perceptrón, creado por Frank Rosenblatt en 1958[24]

Las redes neuronales[19,25] se refieren a un amplio tipo de modelos/parametrizaciones no lineales $h_\theta(x)$ que implican combinaciones de multiplicaciones de matrices y otras operaciones no lineales de entrada. Las redes neuronales que se utilizarán en este trabajo y, por tanto, las que definiremos aquí son las redes neuronales multicapa totalmente conectadas.

Sea $h_\theta(x)$ un modelo abstracto no lineal. Supongamos que $\left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$ son las muestras de entrenamiento. Por simplicidad, comenzaremos por el caso en el que $y^{(i)} \in \mathbb{R}$ y $h_\theta(x) \in \mathbb{R}$.

Definición 3.21 (Función de pérdida) Definimos la función de mínimos cuadrados para el i-ésimo ejemplo $\left(x^{(i)}, y^{(i)}\right)$ como

$$J^{(i)}(\theta) = \frac{1}{2}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 \tag{3.22}$$

y definimos la función de pérdida por mínimos cudrados para el conjunto de datos completo como

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} J^{(i)}(\theta) \tag{3.23}$$

que es la misma que la de la regresión lineal salvo porque introducimos la constante $1/n$ al principio de a función de coste para ser consistente con la notación. Pa

Comúnmente, las personas usan el descenso de gradiente (GD), el gradiente estocástico (SGD) o sus variantes para optimizar la función de pérdida $J(\theta)$. La regla de actualización de GD se puede escribir como

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$

donde $\alpha > 0$ a menudo se denomina tasa de aprendizaje o tamaño de paso. A continuación, presentamos una versión del SGD.

Stochastic Gradient Descent

1: Hiperparámetros: ratio de aprendizaje α, número total de iteraciones n_{iter} .
2: Inicializar θ aleatoriamente.
3: para cada $i = 1$ hasta n_{iter} hacer
4: Remuestrear j uniformemente desde $\{1, \ldots, n\}$, y actualizar θ mediante

$$\theta := \theta - \alpha \nabla_\theta J^{(j)}(\theta)$$

Con este algoritmos genérico, se entrena un modelo típico de aprendizaje profundo con los siguientes pasos.

1. Definir una parametrización de red neuronal $h_\theta(x)$

2. escribir el algoritmo de retropropagación para calcular el gradiente de la función de pérdida $J^{(j)}(\theta)$ eficientemente.

3. ejecutar SGD (u otros optimizadores basados en gradientes) con la función de pérdida $J(\theta)$.

Para terminar de definir lo necesario para desarrollar la red neuronal definimos una función parametrizada $h_\theta(x)$ con entrada x, parametrizada por θ, que genera la variable de salida y. Formalmente, $h_\theta : x \to y$. Quizás una de las parametrizaciones más sencillas sería

$$h_\theta(x) = \text{máx}(wx + b, 0), \text{donde} \theta = (w, b) \in \mathbb{R}^2$$

Aquí $h_\theta(x)$ devuelve un único valor: $(wx + b)$ o cero, el que sea mayor. En el contexto de las redes neuronales, la función $\text{máx}\{t, 0\}$ se llama ReLU, o unidad lineal rectificada, y a menudo se denota por $\text{ReLU}(t) \triangleq \text{máx}\{t, 0\}$

Generalmente, una función no lineal unidimensional que asigna $\mathbb{R}$ a $\mathbb{R}$, como ReLU, a menudo se denomina **función de activación**.

Con todo lo anterior ya explicado, podemos definir por fin una red neuronal multicapa. Sea r el número de capas (matrices de pesos). Sean $W^{[1]}, \ldots, W^{[r]}, b^{[1]}, \ldots, b^{[r]}$ las matrices de pesos y los errores de todas las capas. Sea ReLU una función de activación no lineal, definida como $max(a^{[i]}, 0)$. Entonces una red neuronal multicapa puede escribirse como

$$\begin{aligned} a^{[1]} &= \text{ReLU}\left(W^{[1]}x + b^{[1]}\right) \\ a^{[2]} &= \text{ReLU}\left(W^{[2]}a^{[1]} + b^{[2]}\right) \\ &\cdots \\ a^{[r-1]} &= \text{ReLU}\left(W^{[r-1]}a^{[r-2]} + b^{[r-1]}\right) \end{aligned}$$

$$h_\theta(x) = W^{[r]}a^{[r-1]} + b^{[r]} \tag{3.24}$$

Se hace notar que las matrices de pesos y los errores tienen que tener dimensiones compatibles para que las ecuaciones 3.24 tengan sentido. Si $a^{[k]}$ tiene dimensión m_k, entonces la matriz de pesos $W^{[k]}$ tiene que tener dimensión $m_k \times m_{k-1}$, y el error $b^{[k]} \in \mathbb{R}^{m_k}$.Es más, $W^{[1]} \in \mathbb{R}^{m_1 \times d}$ y

$W^{[r]} \in \mathbb{R}^{1\times m_{r-1}}$.

El número total de neuronas en la red es $m_1 + \cdots + m_r$, y el número total de parámetros en esta red es $(d+1)m_1 + (m_1+1)\,m_2 + \cdots + (m_{r-1}+1)\,m_r$.

Algunas veces, por consistencia de la notación, también se escribe $a^{[0]} = x$, and $a^{[r]} = h_\theta(x)$. Entonces tenemos recursividad simple que

$$a^{[k]} = \mathrm{ReLU}\left(W^{[k]}a^{[k-1]} + b^{[k]}\right), \forall k = 1, \ldots, r-1 \tag{3.25}$$

La función de activación ReLu puede ser reemplazada por muchas otras funciones no lineales $\sigma(\cdot)$ que van de $\mathbb{R}$ a $\mathbb{R}$ como

$$\begin{aligned} \sigma(z) &= \tfrac{1}{1+e^{-z}} \quad (\text{ sigmoid }) \\ \sigma(z) &= \tfrac{e^z - e^{-z}}{e^z + e^{-z}} \quad (\tanh) \end{aligned}$$

Para obtener los pesos de las neuronas que minimizan la función de pérdida optimizando así los resultados se utiliza el siguiente algoritmo.

Algoritmo 3.2 Algoritmo de retropropagación para redes neuronales multicapa.

1. Se calculan y se guardan los valores de $a^{[k]}$'s y $z^{[k]}$'s para $k = 1, \ldots, r$, y J. Esto a menudo se llama el 'paso hacia adelante'.

2. Ahora se realiza el 'paso hacia atrás' en el que para todo valor de k = r hasta 1 hacer:

   a. Si k = r, entonces se calcula $\delta^{[r]} \triangleq \frac{\partial J}{\partial z^{[r]}}$

   b. En cualquier otro caso se calcula $\delta^{[k]} \triangleq \frac{\partial J}{\partial z^{[k]}} = \left(W^{[k+1]^\top}\delta^{[k+1]}\right) \odot \mathrm{ReLU}'\left(z^{[k]}\right)$

3. Por último se halla el valor final como

$$\begin{aligned} \frac{\partial J}{\partial W^{[k]}} &= \delta^{[k]}a^{[k-1]^\top} \\ \frac{\partial J}{\partial b^{[k]}} &= \delta^{[k]} \end{aligned} \tag{3.26}$$

3.4. Algoritmos de inversión

Aunque las predicciones que se obtengan de los modelos no puntúen muy alto en las métricas definidas, otros autores consiguen definir algoritmos que utilizan esas predicciones de manera inteligente para obtener resultados muy positivos en las inversiones. Aquí se van a presentar cuatro algoritmos que hemos diseñado y que son los que utilizaremos en las predicciones obtenidas y evaluaremos en la sección 4.3.3. El objetivo de estos algoritmos no será predecir exactamente cada cambio en el mercado, sino identificar momentos en los que las predicciones hechas por los modelos tienen más probabilidad de acertar. De esta manera se identificarán entradas muy favorables para invertir.

Para definir correctamente estos algoritmos denotaremos:

$$a_i = prediccion\ binaria\ hecha\ por\ regresion\ logistica$$
$$b_i = prediccion\ binaria\ hecha\ por\ adaptative\ boosting$$
$$c_i = prediccion\ binaria\ hecha\ por\ random\ forest$$
$$d_i = prediccion\ binaria\ hecha\ por\ red\ neuronal$$

$$w_i = \ confianza\ de\ la\ prediccion\ a_i$$
$$x_i = \ confianza\ de\ la\ prediccion\ b_i$$
$$y_i = \ confianza\ de\ la\ prediccion\ c_i$$
$$z_i = \ confianza\ de\ la\ prediccion\ d_i$$

Sea también s_i una señal en el día i, que indique si compramos (1), vendemos (-1) o nos mantenemos como estamos (0) y p_i una variable binaria que representa nuestra posición (1) si tenemos una posición alcista, (0) si la tenemos bajista. Ahora ya podemos presentar los algoritmos.

Algoritmo 3.3 (conf3) Este algoritmo busca los días en los que tres modelos coinciden en su predicción para realizar su inversión con un nivel de confianza superior a unos valores que definiremos dentro del algoritmo. Como seguirá habiendo un modelo que prediga lo contrario a los demás lo consideraremos de alto riesgo.

Paso 1: Se calcula la confianza de las predicciones: w_0, x_0, y_0 y z_0.

a. Si al menos tres de las predicciones cumplen que $w_0 > 0{,}55$, $x_0 > 0{,}501$, $y_0 > 0{,}55$ o $z_0 > 0{,}51$ entonces definir $s_0 = 1$ y $p_0 = 1$.

b. Si al menos tres de las predicciones cumplen que $w_0 < 0{,}45$, $x_0 < 0{,}498$, $y_0 < 0{,}45$ o $z_0 < 0{,}49$ entonces definir $s_0 = -1$ y $p_0 = 0$.

c. En cualquier otro caso definir únicamente $s_0 = 0$

Paso 2: Repetir comenzando con i=1 y haciendo i=i+1 con cada iteración hasta que se llegue al día presente.

a. Si al menos tres de las predicciones cumplen que $w_i > 0{,}55$, $x_i > 0{,}501$, $y_i > 0{,}55$ o $z_i > 0{,}51$ y se cumple que $s_{i-1} \neq 1$ entonces definir $s_i = 1$ y $p_i = 1$.

b. Si al menos tres de las predicciones cumplen que $w_i < 0{,}45$, $x_i < 0{,}498$, $y_i < 0{,}45$ o $z_i < 0{,}49$ y se cumple que $s_{i-1} \neq -1$ entonces definir $s_0 = -1$ y $p_0 = 0$.

c. En cualquier otro caso definir $s_i = 0$ y $p_i = p_{i-1}$

Algoritmo 3.4 (conf4) Este algoritmo busca los días en los que los cuatro modelos coinciden en su predicción para realizar su inversión con un nivel de confianza superior a unos valores que definiremos dentro del algoritmo. Como en este algoritmo se buscan únicamente los días en los que las predicciones de todos los modelos sean iguales se tendrán menos puntos de entrada que en el anterior, pero serán puntos de entrada más seguros. Por esto consideraremos este algoritmo

como el de bajo riesgo.

Paso 1: Se calcula la confianza de las predicciones: w_0, x_0, y_0 y z_0.

a. Si las cuatro predicciones cumplen que $w_0 > 0{,}55$, $x_0 > 0{,}501$, $y_0 > 0{,}55$ y $z_0 > 0{,}51$ entonces definir $s_0 = 1$ y $p_0 = 1$.

b. Si las cuatro predicciones cumplen que $w_0 < 0{,}45$, $x_0 < 0{,}498$, $y_0 < 0{,}45$ y $z_0 < 0{,}49$ entonces definir $s_0 = -1$ y $p_0 = 0$.

c. En cualquier otro caso definir únicamente $s_0 = 0$

Paso 2: Repetir comenzando con i=1 y haciendo i=i+1 con cada iteración hasta que se llegue al día presente.

a. Si las cuatro predicciones cumplen que $w_i > 0{,}55$, $x_i > 0{,}501$, $y_i > 0{,}55$ o $z_i > 0{,}51$ y se cumple que $s_{i-1} \neq 1$ entonces definir $s_i = 1$ y $p_i = 1$.

b. Si las cuatro predicciones cumplen que $w_i < 0{,}45$, $x_i < 0{,}498$, $y_i < 0{,}45$ o $z_i < 0{,}49$ y se cumple que $s_{i-1} \neq -1$ entonces definir $s_0 = -1$ y $p_0 = 0$.

c. En cualquier otro caso definir $s_i = 0$ y $p_i = p_{i-1}$

Algoritmo 3.5 (bina3) Este algoritmo busca los días en los que tres modelos coinciden en su predicción. Al no introducir el nivel de confianza este algoritmo identificará más señales de entrada que los basados en la confianza de las predicciones. Por el mismo motivo que en el algoritmo conf3, éste lo consideraremos de alto riesgo.

Paso 1: Se calculan las predicciones: a_0, b_0, c_0 y d_0.

a. Si al menos tres de las predicciones cumplen que $a_0 = 1$, $b_0 = 1$, $c_0 = 1$ o $d_0 = 1$ entonces definir $s_0 = 1$ y $p_0 = 1$.

b. Si al menos tres de las predicciones cumplen que $a_0 = 0$, $b_0 = 0$, $c_0 = 0$ o $d_0 = 0$ entonces definir $s_0 = -1$ y $p_0 = 0$.

c. En cualquier otro caso definir únicamente $s_0 = 0$

Paso 2: Repetir comenzando con i=1 y haciendo i=i+1 con cada iteración hasta que se llegue al día presente.

a. Si al menos tres de las predicciones cumplen que $a_i = 1$, $b_i = 1$, $c_i = 1$ o $d_i = 1$ y se cumple que $s_{i-1} \neq 1$ entonces definir $s_i = 1$ y $p_i = 1$.

b. Si al menos tres de las predicciones cumplen que $a_i = 0$, $b_i = 0$, $c_i = 0$ o $d_i = 0$ y se cumple que $s_{i-1} \neq -1$ entonces definir $s_0 = -1$ y $p_0 = 0$.

c. En cualquier otro caso definir $s_i = 0$ y $p_i = p_{i-1}$

Algoritmo 3.6 (bina4) Por último, este algoritmo busca los días en los que los cuatro modelos coinciden en su predicción. Al no introducir el nivel de confianza este algoritmo identificará más señales de entrada que los basados en la confianza de las predicciones. Sin embargo, a diferencia del anterior, los cuatro modelos predicen lo mismo, creando señales de entrada mucho más fiables. Se considerará este modelo como de bajo riesgo.

Paso 1: Se calculan las predicciones: a_0, b_0, c_0 y d_0.

a. Si se cumple que $a_0 = b_0 = c_0 = d_0 = 1$ entonces definir $s_0 = 1$ y $p_0 = 1$.

b. Si se cumple que $a_0 = b_0 = c_0 = d_0 = 0$ entonces definir $s_0 = -1$ y $p_0 = 0$.

c. En cualquier otro caso definir únicamente $s_0 = 0$

Paso 2: Repetir comenzando con i=1 y haciendo i=i+1 con cada iteración hasta que se llegue al día presente.

a. Si se cumple que $a_0 = b_0 = c_0 = d_0 = 1$ y $s_{i-1} \neq 1$ entonces definir $s_i = 1$ y $p_i = 1$.

b. Si se cumple que $a_0 = b_0 = c_0 = d_0 = 0$ y $s_{i-1} \neq -1$ entonces definir $s_0 = -1$ y $p_0 = 0$.

c. En cualquier otro caso definir $s_i = 0$ y $p_i = p_{i-1}$

Capítulo 4

Caso práctico

Ahora que ya hemos presentado matemáticamente los algoritmos de machine learning que se van a utilizar, vamos a implementarlos y evaluarlos en el mercado de divisas. Más específicamente en la tabla del Euro-Dólar (EURUSD).

4.1. Introducción del proyecto

Lo primero que se hizo fue buscar un dataset que recogiera los datos requeridos. Desde la página https://es.investing.com/ pude hallar uno que recogía la apertura, el cierre, el máximo, el mínimo y el VaR desde el año 1979 hasta la actualidad (en el momento de realizar esta primera prueba a 27 de junio de 2022).

A pesar de que el Euro no fue lanzado hasta el 1 de enero de 1999 como moneda electrónica y puesto en circulación hasta el 1 de enero de 2002[16], la idea de una moneda única europea ya venía de mucho antes. De hecho en marzo de 1979 se acordó constituir el Sistema Monetario Europeo (SME o ERM en inglés). El objetivo era reducir la alta volatilidad de los tipos de cambio de las divisas europeas entre sí y crear, por tanto, una zona de estabilidad monetaria que ayudase a aumentar la integración económica europea, promoviendo un incremento de las transacciones comerciales[9]. El Euro se introdujo con un tipo de cambio 1:1 con el SME y la tabla recoge los movimientos, primero del SME y, posteriormente del Euro, con respecto al dólar. Permitiéndonos tener una mayor cantidad de datos para trabajar.

	Fecha	Ultimo	Apertura	Maximo	Minimo	var
0	27/06/2022	1.0604	1.0567	1.0615	1.0550	0.35
1	26/06/2022	1.0567	1.0555	1.0575	1.0552	0.10
2	24/06/2022	1.0556	1.0526	1.0572	1.0511	0.31
3	23/06/2022	1.0523	1.0567	1.0582	1.0482	-0.40
4	22/06/2022	1.0565	1.0527	1.0606	1.0468	0.38
...	... ...	...	...	...	...	
11019	03/01/1980	1.5177	1.5177	1.5177	1.5177	0.18
11020	02/01/1980	1.5149	1.5149	1.5149	1.5149	0.45
11021	31/12/1979	1.5081	1.5081	1.5081	1.5081	0.03
11022	28/12/1979	1.5076	1.5076	1.5076	1.5076	0.01
11023	27/12/1979	1.5074	1.5074	1.5074	1.5074	0.41

11024 rows x 6 columns

La librería matplolib previamente importada nos permite visualizar la forma que va a tener esta serie temporal. En la figura 4.1 podemos observar todos los precios de cierre diarios del par euro dólar desde 1979.

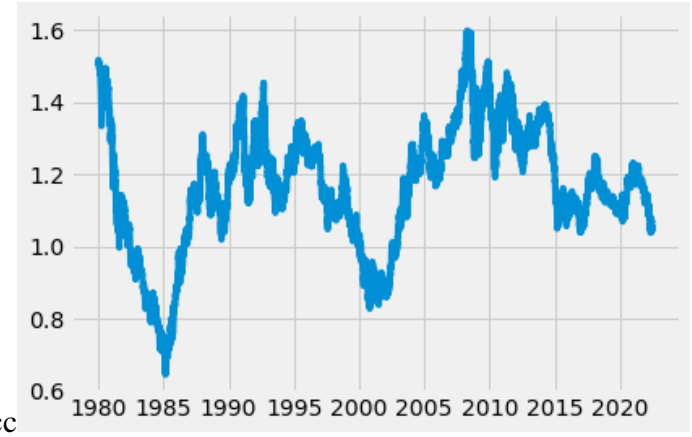
cc 

Figura 4.1: Precios de cierre desde 1979

Una regla general en el análisis de series temporales dice que si no se es capaz de trazar intuitivamente el recorrido de la serie en el futuro, predecirla va a ser casi imposible mediante los métodos convencionales. Como se puede ver esta serie temporal parece completamente impredecible. Ya podemos ir observando algunos de los problemas a los que nos enfrentamos en este proyecto.

El objetivo principal será desarrollar lo más eficientemente posible los algoritmos presentados en el Capítulo 3, estudiar su desempeño y la calidad de sus predicciones. Y, posteriormente, ver si utilizando esas predicciones y un poco de pensamiento matemático podemos crear un algoritmo de inversión efectivo.

4.2. Análisis estadístico

Para ir analizando más profundamente los datos recogidos se hizo una pequeña exploración de la calidad de éstos.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11024 entries, 11023 to 0
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Fecha     11024 non-null  datetime64[ns]
 1   Ultimo    11024 non-null  float64
 2   Apertura  11024 non-null  float64
 3   Maximo    11024 non-null  float64
 4   Minimo    11024 non-null  float64
 5   var       11024 non-null  float64
dtypes: datetime64[ns](1), float64(5)
memory usage: 602.9 KB
```

Observemos que, al menos, el dataset no presenta nulos y todas las variables son de tipo float64. Esto nos va a reducir notablemente el preprocessing puesto que no se va a tener que hacer la imputación de missings ni la transformación de variables categóricas.

4.2.1. Creación de nuevas variables explicativas

A partir de los datos que tenemos, podemos crear ciertas variables que van a ofrecer algo más de información sobre las variaciones del mercado y que los modelos pueden utilizar para hacer las predicciones.

La primera variable serán los *pips*. El término *pip* es la abreviatura en inglés de «point in percentage» o punto porcentual. Es la medida del movimiento más pequeño del tipo de cambio de un par de divisas en el mercado cambiario. Un *pip* es una unidad estándar que permite medir cuánto puede variar la cotización de un activo durante una transacción. Esta unidad permite calcular el riesgo y proteger a los inversores de variaciones grandes en el tipo de cambio que implicarían grandes pérdidas[4]. Un *pip* es diferente según el par de monedas que se esté valorando, pero en el EURUSD, que es el que a nosotros nos interesa, un *pip* es una unidad del cuarto decimal. EUR/USD = 1,1850 ->1,1851

	Fecha	Ultimo	Apertura	Maximo	Minimo	var	Pips
4	2022-06-22	1.0565	1.0527	1.0606	1.0468	0.38	38.0
3	2022-06-23	1.0523	1.0567	1.0582	1.0482	-0.40	-44.0
2	2022-06-24	1.0556	1.0526	1.0572	1.0511	0.31	30.0
1	2022-06-26	1.0567	1.0555	1.0575	1.0552	0.10	12.0
0	2022-06-27	1.0604	1.0567	1.0615	1.0550	0.35	37.0

La segunda variable que crearemos será la variación. Ésta representa la máxima diferencia entre los cambios entre el euro y el dólar de cada día. Siempre es positiva y suele representar el volumen de cambio de divisas que se ha producido a lo largo de la jornada.

	Fecha	Ultimo	Apertura	Maximo	Minimo	var	Pips	Variacion
4	2022-06-22	1.0565	1.0527	1.0606	1.0468	0.38	38.0	138.0
3	2022-06-23	1.0523	1.0567	1.0582	1.0482	-0.40	-44.0	100.0
2	2022-06-24	1.0556	1.0526	1.0572	1.0511	0.31	30.0	61.0
1	2022-06-26	1.0567	1.0555	1.0575	1.0552	0.10	12.0	23.0
0	2022-06-27	1.0604	1.0567	1.0615	1.0550	0.35	37.0	65.0

4.2.2. Análisis de datos extremos (outliers)

Con las variables que vamos a utilizar para entrenar a nuestro primer modelo de prueba creadas, tenemos todo lo necesario para poder explorar más a fondo los datos. Se comenzará por la representación de los boxplots de todas las variables. 'Figura 4.2'. De esta manera podremos analizar outliers y ver si es necesario corregirlos o eliminarlos. Observemos que en las variables Último, Apertura, Máximo y Mínimo los outliers que se presentan son pocos y están muy cerca de los cuantiles 1 y 3, así que no les prestaremos mucha atención.

Sin embargo, en la variable var, podemos observar 3 datos extremos que superan el valor de 3'5. Recordemos que el VaR, más conocido como Valor en Riesgo o Value at Risk es una técnica

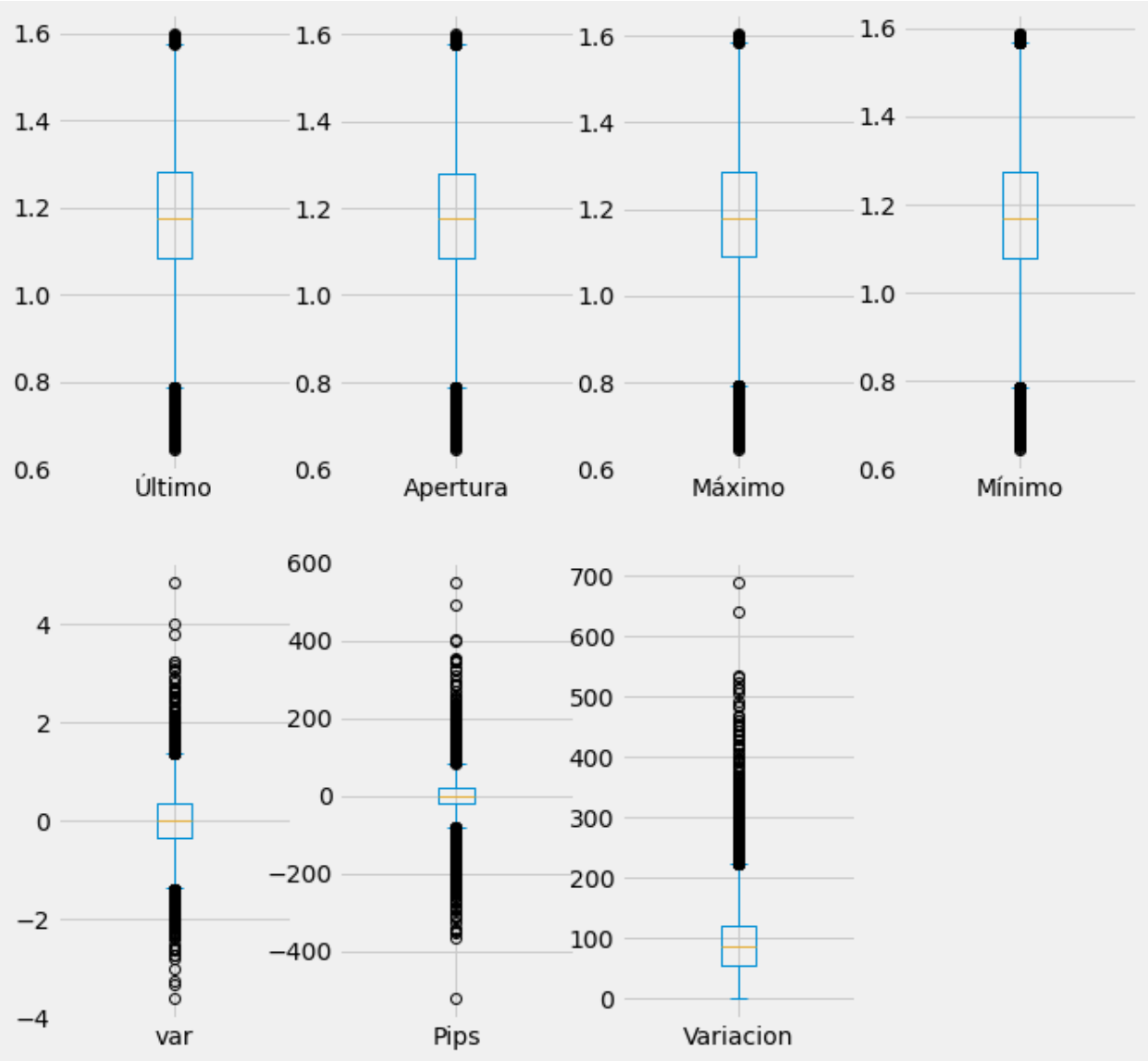


Figura 4.2: Gráficos boxplot de las variables desde 1979

estadística para medir el riesgo financiero de una inversión. Indica la probabilidad de sufrir una determinada pérdida durante un periodo de tiempo (normalmente 1 día, 1 semana o 1 mes).[3] Por lo que un VaR con un valor muy alto representa que esos días en el mercado había una alta volatilidad, y los precios en el mercado podían variar mucho. Echemos un vistazo a esos días.

	Fecha	Ultimo	Apertura	Maximo	Minimo	var	Pips	Variacion
9716	1985-02-27	0.6703	0.6703	0.6703	0.6703	4.02	0.0	0.0
9573	1985-09-23	0.8064	0.8064	0.8064	0.8064	4.84	0.0	0.0
3464	2009-03-18	1.3508	1.3015	1.3520	1.2987	3.80	493.0	533.0

En 1985 todavía los precios de la moneda no variaban en un sólo día y esos valores del var

representan momentos económicos inciertos. Por ejemplo, en el 23 de septiembre de 1985, Ronald Reagan impuso el cambio de política económica en Estados Unidos provocando una fuerte caída del dólar estadounidense.

El otro valor extremo se presenta el 18 de marzo de 2009. En ese momento europa (en especial España e Italia) estaban sufriendo más fuerte las consecuencias de la crisis de 2008 y ese mismo día París y Berlín pidieron una mayor regulación bancaria por lo que se realizó un consenso entre los países europeos para dotar de más fondos al FMI. Esto provocó una confianza en el Euro, explicando la subida de casi 500 *pips* de ese día.

Estos acontecimientos reflejan mejor lo expuesto en el capítulo 3. Para realizar inversiones correctas no es necesario el análisis fundamental, pues todo lo que ocurre en el mundo se ve reflejado en las tablas y en los datos. Por ello en este trabajo, el análisis técnico va a ser la base de nuestras predicciones.

Estos valores extremos del VaR, tienen, por tanto, significado y no los eliminaremos ni cambiaremos.

También podemos observar dos outliers muy notorios en la variable variación. Estudiemoslos más a fondo.

```
      Fecha         Ultimo  Apertura  Maximo  Minimo  var     Pips    Variacion
7764  1992-09-16     1.298    1.3503  1.3523  1.2883  -3.62   -523.0  640.0
7763  1992-09-17     1.327    1.2722  1.3341  1.2653  2.23    548.0   688.0
```

Ambos outliers corresponden a la dramática fecha del conocido miércoles negro de 1992. En el cual, Gran Bretaña (junto con España e Italia), por no poder mantener los tipos de cambio impuestos por la SME suspendía su pertenencia al SME y su obligación de gastar mas dinero defendiendo su divisa.

Previamente algunos inversores con mucha visión vieron que aunque la libra, la lira o la peseta estuviesen en el rango mínimo inferior, la realidad acabaría haciendo que tuviesen que devaluar. George Soros y Stanley Druckenmiller, uno de sus cerebros en el Hedge Fund Quantum, decidieron empezar a comprar marcos alemanes y vender libras en el punto límite. En teoría no habia recorrido posible ya que la libra no podía caer ni el marco apreciarse.

Así que cuando al final terminó ocurriendo lo inevitable y Gran Bretaña anunció su salida del SME, la libra esterlina cayó en el mercado un 15 %. Gran Bretaña pudo bajar los tipos de interés que ahogaban a los ingleses y George Soros y Druckenmiller ganaron 1.100 millones de libras para sus inversores. Desde aquel día George Soros es conocido como “the man who broke the Bank of England”.[9]

Todo esto se puede apreciar en una sencilla línea de nuestro dataset y es la que refleja esta variación tan alta. Por lo que estos outliers también tienen significado (y mucho) y los dejaremos en la base de datos.

4.2.3. Creación de variables a predecir

Como ya se ha explicado, nuestro enfoque en este proyecto va a ser utilizar las diferentes técnicas de modelización analítica para predecir lo que va a ocurrir mañana con los tipos de cambio

entre el dólar y el euro en función de los datos que tenemos hasta hoy. Para ello debemos crear las variables a predecir. UpDownPred que va a ser binaria y sólo nos dirá si al día siguiente va a ser superior el precio de cierre que el de hoy (1) o si va a ser inferior (0).

También queremos evaluar nuestros algoritmos en un entorno de predicción de una variable continua, por lo que necesitaremos otra variable para entrenarlos que refleje el precio de cierre del día siguiente. Se llamará UltimoPred.

Generadas así las dos variables que utilizaremos para entrenar a nuestros modelos de aprendizaje supervisado, deberemos eliminar la última observación del dataset pues de ella no conocemos el dato del día siguiente. Veamos cómo queda.

```
       Fecha  Ultimo  Apertura  Maximo  Minimo  var  Pips Variacion  UpDown  UpDownPred
    UltimoPred
5      2022-06-21  1.0525  1.0510  1.0583  1.0508  0.15    15.0    75.0    1       1
    1.0565
4      2022-06-22  1.0565  1.0527  1.0606  1.0468  0.38    38.0    138.0   1       0
    1.0523
3      2022-06-23  1.0523  1.0567  1.0582  1.0482  -0.40   -44.0   100.0   0       1
    1.0556
2      2022-06-24  1.0556  1.0526  1.0572  1.0511  0.31    30.0    61.0    1       1
    1.0567
1      2022-06-26  1.0567  1.0555  1.0575  1.0552  0.10    12.0    23.0    1       1
    1.0604
```

Observemos que cuando en UpDownPred aparece un 1, al día siguiente en UpDown aparece un 1 y que UltimoPred, refleja lo que va a aparecer al día siguiente en Ultimo.

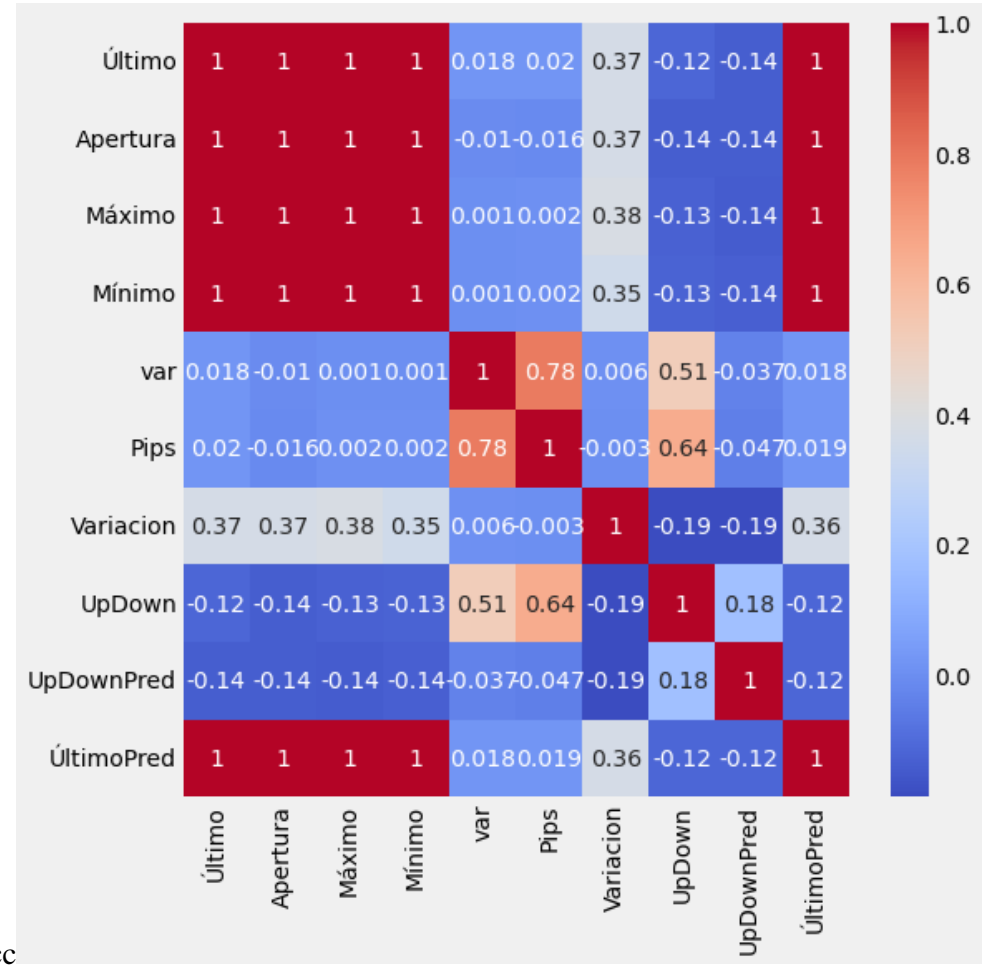
4.2.4. Análisis de correlaciones

Con todas las variables ya creadas podemos hacer un último análisis. Comenzaremos por el gráfico de correlaciones.

Bajo la máxima *'menos es más'*, realizaremos un estudio de las correlaciones sobre todas las variables para observar si alguna de ellas puede ser explicada por otra por su fuerte correlación. Tratamos con un conjunto de variables que a priori consideramos no paramétricas, es decir, no siguen nigún tipo de distribución. Por otra parte estamos interesados en todo tipo de correlaciones, lineales y no lineales. Es por ello que descartamos el uso de la correlación de Pearson, y por su robustez y conveniencia utilizaremos la correlación de Kendall. 'Figura 4.3'

También, prestaremos especial atención en las correlaciones de las variables explicativas (Apertura, Ultimo, *Pips*...) con las variables a predecir (UpDownPred y UltimoPred). Una fuerte correlación será algo muy positivo pues nos está indicando que nuestras variables explicativas nos están ofreciendo una clara información sobre los tipos de cambio al día siguiente. Sería muy sorprendente que estas correlaciones fueran altas pero estaremos atentos a ellas por si acaso.

1. Se puede observar una correlación de 1 entre Último, Apertura, Máximo, Mínimo y ÚltimoPred. Pero como cada una de estas variables refleja un valor distinto de los datos de las inversiones cada día no eliminaremos ninguna de ellas.



cc

Figura 4.3: Gráfico de correlaciones de Kendall (1979)

2. También presentan correlaciones altas las variables pips y VaR. Esto es así pues ambas reflejan cierta volatilidad del mercado. Cuantos más *pips* haya habido entre la apertura y el último valor del tipo de cambio más dinero se podría haber perdido y el VaR, por lo tanto es mayor. Como los *pips* reflejan un valor real entre la apertura y el cierre del día y el VaR representa un potente indicador estadístico no eliminaremos ninguna de las dos de nuestro dataset.

3. Por desgracia y como preveíamos anteriormente, las correlaciones entre las variables objetivo y el resto de las variables son muy bajas. Esto nos indica que las predicciones que intentemos hacer probablemente den porcentajes de acierto muy bajos.

4.2.5. Estudio de normalidad

Como sabemos, que haya normalidad en la variable a predecir es muy positivo y nos permitiría aplicar algoritmos que tienen como hipótesis inicial la condición de normalidad (un ejemplo de

esto sería el análisis discriminante gaussiano). Por ello vamos a analizar la variable ClosePred, que es continua. Aparentemente, por la figura 4.4 nuestra variable no va a presentar similitudes

cc

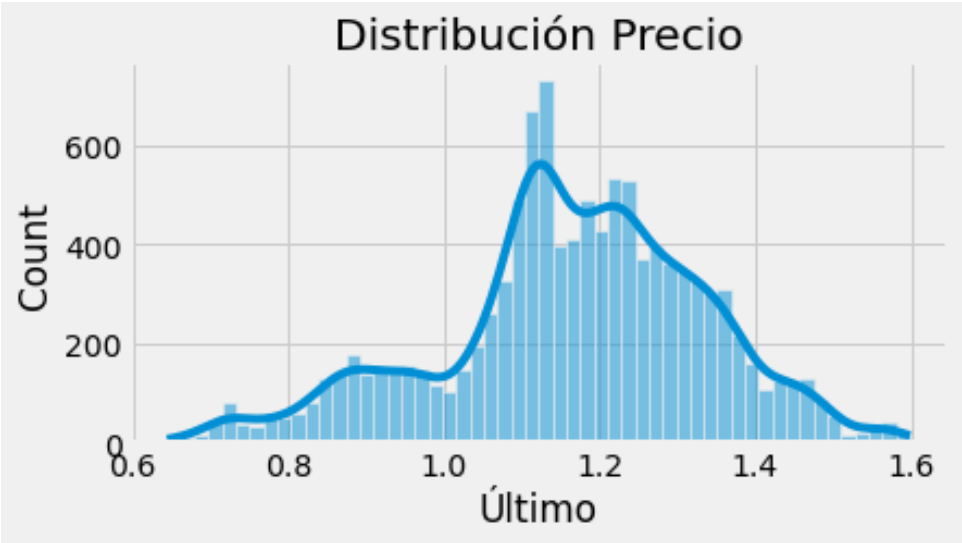


Figura 4.4: Histograma de la variable ClosePred

con ninguna de las distribuciones más famosas. Aún así hagamos el test de Saphiro para confirmar que no hay normalidad.

```
ShapiroResult(statistic=0.9845324158668518, pvalue=7.141309648185805e-33)
```

Como el valor del p-valor es claramente menor que 0.05 podemos rechazar la hipótesis de normalidad.

4.2.6. Conteo de observaciones y umbral de referencia

En la variable binaria UpDownPred, podemos tener el problema de que la variable se encuentre desbalanceada. Esto puede generar modelos que aunque tengan un alto porcentaje de aciertos sean modelos nulos. Como se puede observar en la figura 4.5, en nuestra base de datos hay un 64,89 % de unos. Por lo que para que un modelo se considere válido deberá superar ese umbral de referencia, puesto que un modelo que siempre diga que va a subir ya acertaría casi el 65 % de las veces.

4.3. Preprocesamiento de los datos

4.3.1. Estandarización

El algoritmo que vamos a generar en esta primera prueba será una red neuronal. Como ya hemos eplicado en el capítulo 3 es conveniente estandarizar las variables de entrada para el correcto

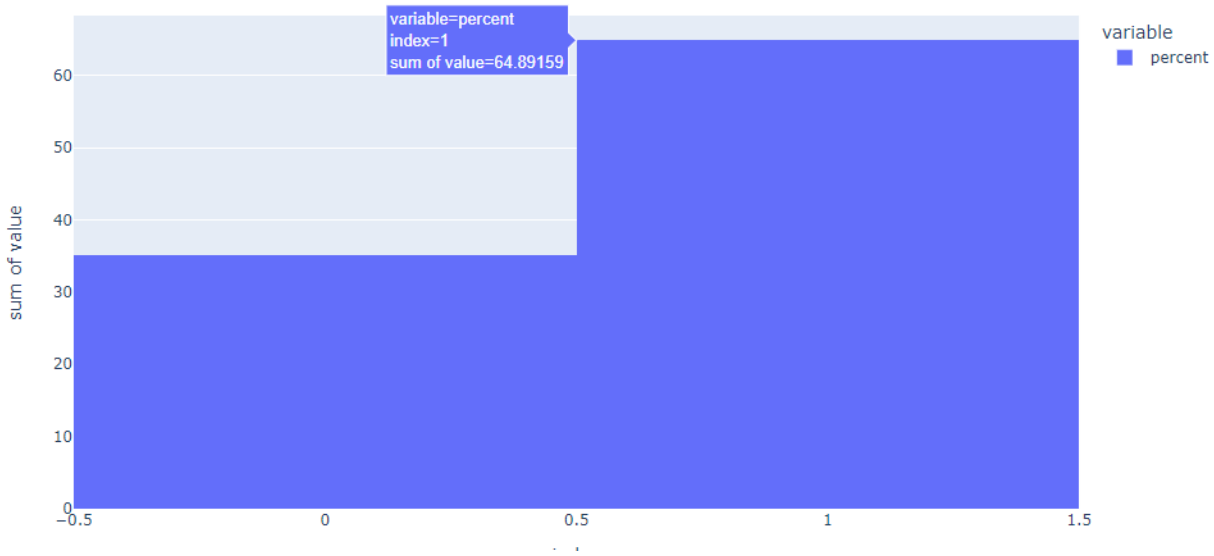


Figura 4.5: Conteo de observaciones en UpDownPred

funcionamiento de la red. Crearemos un dataset a parte con los datos estandarizados llamado datastd.

Observemos que las variables explicativas ya tienen todas media 0 y desviación típica 1.

	mean	std
Ultimo	-3.294612e-16	1.000045
Apertura	-1.981138e-17	1.000045
Maximo	-1.593824e-15	1.000045
Minimo	1.065584e-15	1.000045
var	1.450602e-17	1.000045
Pips	2.926888e-17	1.000045
Variacion	4.902587e-15	1.000045
UpDown	1.332413e-14	1.000045
UpDownPred	6.489159e-01	0.477331
UltimoPred	1.166034e+00	0.171099

4.3.2. Train y test

Ahora que ya tenemos el dataset estandarizado podemos generar las particiones de entrenamiento y test que utilizaremos para generar nuestra red neuronal. Aunque pueda generar overfitting tomaremos como test únicamente las observaciones de los años 2021 y 2022, que corresponde con el 5 % de los datos. De esta manera podremos comprobar lo eficiente que es nuestro modelo con observaciones recientes y no de hace 40 años.

Obtenemos, de esta manera 10471 observaciones en la partición de entrenamiento y 552 en la de test.

4.3.3. Prueba con una red neuronal

Para poder analizar mejor nuestro dataset y hallar así algunos problemas que todavía podemos arreglar en el preprocesamiento de los datos antes de generar, optimizar y evaluar nuestros algoritmos realizaremos una primera prueba de modelado con una red neuronal.

Explicaremos la creación de esta red neuronal aquí y. Se utilizará el mismo procedimiento en el resto de redes neuronales del trabajo aunque variaremos algunos hiperparámetros dependiendo de las características que precisemos. Todo ello se especificará en cada una de ellas pero dejaremos el código para los interesados en la sección de apéndices.

Como hemos explicado en el capítulo 3, utilizaremos el método de validación cruzada para el entrenamiento de la red. Siempre utilizaremos 5 bloques que aleatorizaremos en cada iteración. Además, para optimizar la red neuronal al máximo y hallar los hiperparámetros que mejor resultados den, utilizaremos la librería RandomizedSearchCV que nos permitirá introducir el valor de los hiperparámetros que estamos interesados en probar y buscará la combinación de ellos que mejores resultados obtenga en función de la medida que le introduzcamos.

En el caso de esta primera red neuronal los hiperparámetros que buscará serán: el número de capas ocultas de la red, en este caso le hemos dado tres opciones diferentes; el hiperparámetro alpha, explicado en el capítulo 3, y el criterio de convergencia inicial, que elegirá entre dos valores (0.001 o 0.01).

A parte de eso definimos la función de activación de las capas ocultas, como queremos predecir una variable binaria (UpDownPred) utilizaremos el método logistic que que devuelve la función sigmoide.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

Definimos también la función de regresión para la obtención de los pesos de las neuronas. En este caso, al tener miles de observaciones nos podemos permitir aplicar el optimizador estocástico basado en el gradiente propuesto por Kingma, Diederik y Jimmy Ba. Se aplica en la red neuronal con el nombre de adam.'Véase capítulo 3'.

Con ello, obtenemos la red neuronal que da mejores resultados.

```
{'activation': 'logistic',
 'alpha': 0.001,
 'batch_size': 'auto',
 'beta_1': 0.9,
 'beta_2': 0.999,
 'early_stopping': False,
 'epsilon': 1e-08,
 'hidden_layer_sizes': (10, 10, 10, 10),
 'learning_rate': 'constant',
 'learning_rate_init': 0.01,
 'max_fun': 15000,
 'max_iter': 200,
 'momentum': 0.9,
 'n_iter_no_change': 10,
 'nesterovs_momentum': True,
 'power_t': 0.5,
 'random_state': None,
 'shuffle': True,
```

```
 'solver': 'adam',
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': False,
 'warm_start': False}
```

Podemos observar que tenemos una red neuronal de 4 capas ocultas, con un parámetro alpha de 0.001 y un criterio de convergencia inicial de 0.01.

Con la red neuronal ya generada, utilizamos la función accuracy que definimos en la sección 3.2 para calcular el porcentaje de aciertos que tiene en la partición de test.

```
Acertamos exactamente el 50.181% de las predicciones.
```

Lanzando una moneda al aire casi obtendríamos mejores resultados. Es claro que éste modelo no es válido. Nuestro objetivo ahora será solucionar los problemas que se presentan con nuestros datos, añadir más información en forma de variables para que las técnicas de predicción tengan más información para trabajar e intentar optimizar estos modelos al máximo. Pero, primero, haremos unas observaciones que nos ayudarán en el desarrollo de estas soluciones.

1. Podemos obtener la confianza con la que la red neuronal nos ofrece una predicción (cuanto más cercano a 0,5 sea, menos segura está) y filtrar los datos para ver qué porcentaje de ellos acierta cuando nos ofrece predicciones más seguras. Por ejemplo, si filtramos las que nos ofrecen un nivel de confianza de 0,9 obtenemos:

   ```
   Acertamos exactamente el 98.37251356238698 % de las predicciones.
   ```

2. Sin embargo, notamos que este increíble porcentaje de aciertos se debe a que desde el año 1979 hasta 1990 los precios de apertura y de cierre eran los mismos, por lo que a la red neuronal se ha sobreajustado a esos datos y acierta casi siempre con esos datos de entrenamiento. Se puede ver que, si hacemos el mismo filtrado en la partición de test obtenemos:

   ```
   Acertamos exactamente el 50% de las predicciones.
   ```

   Podemos solucionar esto si eliminamos los datos más antiguos y nos quedamos con los más recientes. De esta manera reducimos el tamaño de la partición de test y reducimos la posibilidad de overfitting. Decidimos quedarnos sólo con los datos del 2009 hacia delante, evitando así que la crisis de 2008 pueda introducir ruido en nuestros modelos.

3. Por último, observamos que cada vez que queramos introducir un nuevo dato al sistema tendríamos que hacerlo manualmente descargándolo de internet. Vamos a resolver esto generando un API que recoja automáticamente todos los datos y genere un csv que podamos utilizar con ellos.

Antes de hacer todo esto, vamos a ajustar una red neuronal a los precios de cierre para obtener unos valores de referencia para el resto de modelos.

Y evaluamos los resultados con las métricas explicadas en la sección 3.2

```
Acertamos exactamente el 1.087 % de las predicciones.
Acertamos con un margen de 5 pips el 7.79 % de las predicciones.
Acertamos con un margen de 10 pips el 15.58 % de las predicciones.
Acertamos con un margen de 25 pips el 39.13 % de las predicciones.
Acertamos con un margen de 50 pips el 63.768 % de las predicciones.
```

Observamos que con un margen de 50 pips se acierta casi el 65 % de las predicciones en un dataset sin refinar y que de forma binaria no acertaba más del 51 % de predicciones. Mantendremos estos resultados de umbral de referencia para cuando hagamos la evaluación del resto de modelos.

Resolvamos ahora estos problemas y realicemos un último análisis del dataset para, posteriormente, poder hacer el estudio completo de los modelos y desarrollar los algoritmos de inversión.

4.3.4. Creación del API

Habiendo observado los problemas que se nos presentaban al modelar con el dataset escogido al principio, se nos ocurrieron algunas soluciones para optimizar el rendimiento de nuestros modelos. Comenzaremos por desarrollar un programa que obtenga los datos directamente de una URL en internet para evitar el tedioso trabajo de descargarlos a mano y poder tener siempre que queramos datos actualizados con los que trabajar.

Para obtener estos datos actualizados y con el menor número de errores posibles recurriremos a la conocida web de inversión https://tradermade.com/. De esta manera podremos importar la librería que han creado y obtener los datos actualizados desde 2009.

```
        date            close  open    high    low     var
3391    2009-08-11      1.4146  1.4146  1.4186  1.4110  0.00
3390    2009-08-12      1.4203  1.4148  1.4248  1.4084  0.40
3389    2009-08-13      1.4289  1.4204  1.4330  1.4202  0.61
3388    2009-08-14      1.4205  1.4283  1.4309  1.4161 -0.59
3387    2009-08-17      1.4078  1.4186  1.4194  1.4045 -0.89
...     ...     ...     ...     ...     ...     ...
4       2022-08-04      1.0243  1.0162  1.0254  1.0154  0.78
3       2022-08-05      1.0181  1.0247  1.0253  1.0141 -0.61
2       2022-08-08      1.0195  1.0179  1.0223  1.0158  0.14
1       2022-08-09      1.0211  1.0198  1.0248  1.0188  0.16
0       2022-08-10      1.0297  1.0214  1.0369  1.0201  0.84
```

De esta manera, bastaría cambiar la fecha final a la del día actual (a la fecha de escribir esto estamos a 10 de agosto de 2022) y obtendríamos todos los datos actualizados hasta la fecha. Podemos obtener una representación visual de estos datos gracias a la librería matplotlib. 'Figura 4.6'.

4.3.5. Compleción del dataset

Al igual que hicimos en la sección 4.2, debemos generar las variables explicativas *pips*, variation y updown y las variables objetivo updownpred y closepred. Una vez generadas obtenemos el siguiente dataset con el que seguir trabajando.

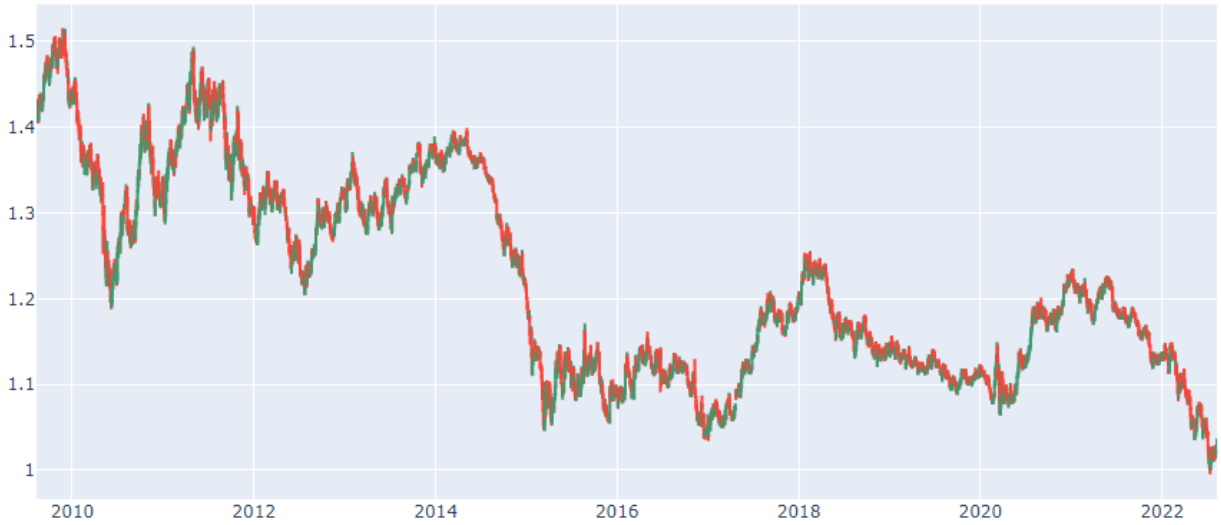


Figura 4.6: Gráfico de velas desde 2009

date	close	open	high	low	var pips	variation	updown	updownpred	closepred
2009-08-11	1.4146	1.4146	1.4186	1.4110	0.00	0.0	76.0	1	1	1.4203
2009-08-12	1.4203	1.4148	1.4248	1.4084	0.40	55.0	164.0	1	1	1.4289
2009-08-13	1.4289	1.4204	1.4330	1.4202	0.61	85.0	128.0	1	0	1.4205
2009-08-14	1.4205	1.4283	1.4309	1.4161	-0.59	-78.0	148.0	0	0	1.4078
2009-08-17	1.4078	1.4186	1.4194	1.4045	-0.89	-108.0	149.0	0	1	1.4133
...	...	...	...	...	...	...	...	...	...	...
2022-08-03	1.0164	1.0166	1.0210	1.0122	0.00	-2.0	88.0	0	1	1.0243
2022-08-04	1.0243	1.0162	1.0254	1.0154	0.78	81.0	100.0	1	0	1.0181
2022-08-05	1.0181	1.0247	1.0253	1.0141	-0.61	-66.0	112.0	0	1	1.0195
2022-08-08	1.0195	1.0179	1.0223	1.0158	0.14	16.0	65.0	1	1	1.0211
2022-08-09	1.0211	1.0198	1.0248	1.0188	0.16	13.0	60.0	1	1	1.0297

4.3.6. Creación de nuevas variables explicativas

Lo que haremos para proporcionar más información a nuestros métodos de predicción será generar los indicadores más utilizados por los grandes inversores [18]. Para no sobrecargar el trabajo ofreceré únicamente una breve explicación matemática de cada uno y un gráfico final de cómo se representa sobre nuestros datos para su mejor compresión.

Medias móviles

Como su propio nombre indica la Media Móvil Simple (SMA, por sus siglas en inglés) [19], también conocida como 'n-SMA' es simplemente la media de los precios del precio de cierre de las últimas n velas. El parámetro n es configurable: cuando se establece en un número grande, digamos 20, la línea tiende a cambiar muy lentamente: esto tiene la ventaja de detectar bien la tendencia general, pero no tiene mucho en cuenta las velas más recientes.

Estas medias se calculan según la expresión matemática

$$\bar{x}_p = \frac{1}{n}\sum_{i-1}^{n} x_{p-i} \tag{4.2}$$

Siendo n el número de periodos que se quieren tener en cuenta, y p el día desde el que se quiere calcular la media móvil.

Las medias móviles que vamos a generar son conocidas como las medias móviles de Guppy, GMMA (Guppy Multiple Moving Average), también conocidas sólo como Guppy[23]. Es un indicador técnico que identifica cambios en las tendencia. Lo que significa que ofrece un método objetivo para saber cuándo entrar en una inversión y cuándo salir. Esta técnica utiliza un grupo de 12 medias móviles, 6 pensadas para el corto plazo y 6 diseñadas para el largo. Las doce medias utilizadas son 3, 5, 8, 10, 12, 15, 30, 35, 40, 45, 50, and 60. Vamos a crearlas, a añadirlas al dataset y representarlas.y figura 4.7'

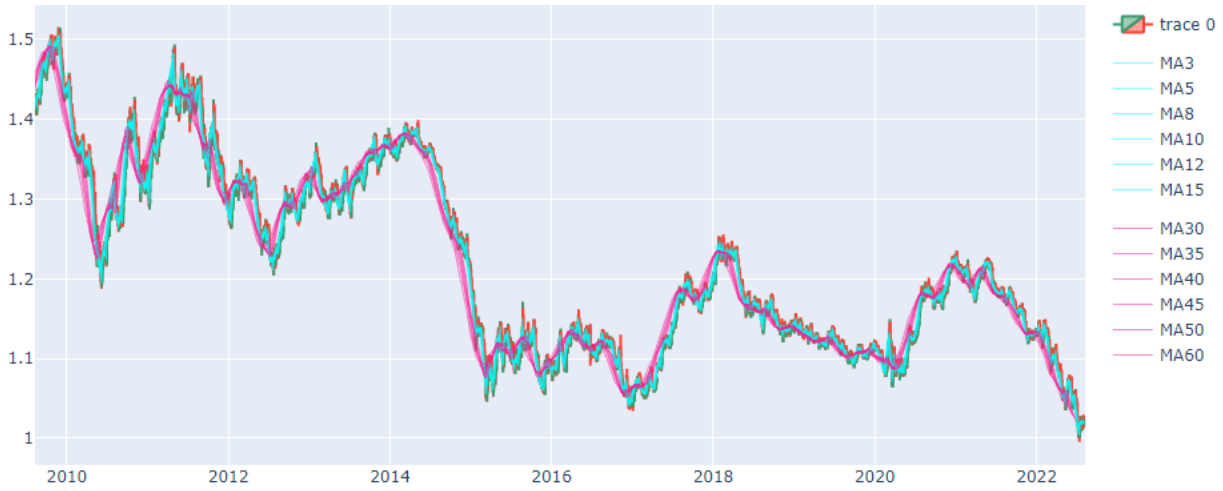


Figura 4.7: Medias móviles de Guppy

Bandas de Bollinger

Las Bandas de Bollinger (BB)[10] son un indicador técnico popular creado por John Bollinger en 1980, que ayuda a determinar si los precios son altos o bajos en términos relativos.

El indicador llenó la necesidad de visualizar cambios en la volatilidad.

Los inversores generalmente usan las Bandas de Bollinger para determinar las zonas de sobrecompra y sobreventa, para confirmar las divergencias entre los precios y los indicadores, y para proyectar los objetivos de precios.

Consta de tres bandas que se calculan de la siguiente forma:

La banda media $\bar{x}_p$ es una media móvil simple de 15 periodos.

$$\bar{x}_p = \frac{1}{15}\sum_{i-1}^{15} x_{p-i} \tag{4.3}$$

La banda superior B_p es la $\bar{x}_p$ más el doble de la desviación típica de los últimos 15 períodos.

$$B_p = \frac{1}{15}\sum_{i-1}^{15} x_{p-i} + 2\sqrt{\frac{1}{15}\sum_{i-1}^{15}(x_{p-i} - \bar{x}_{p-i})^2} \tag{4.4}$$

La banda inferior b_p es la $\bar{x}_p$ menos el doble de la desviación típica de los últimos 15 períodos.

$$b_p = \frac{1}{15}\sum_{i-1}^{15} x_{p-i} - 2\sqrt{\frac{1}{15}\sum_{i-1}^{15}(x_{p-i} - \bar{x}_{p-i})^2} \tag{4.5}$$

Las calculamos en python, y las representamos, como se puede ver en la figura 4.8.

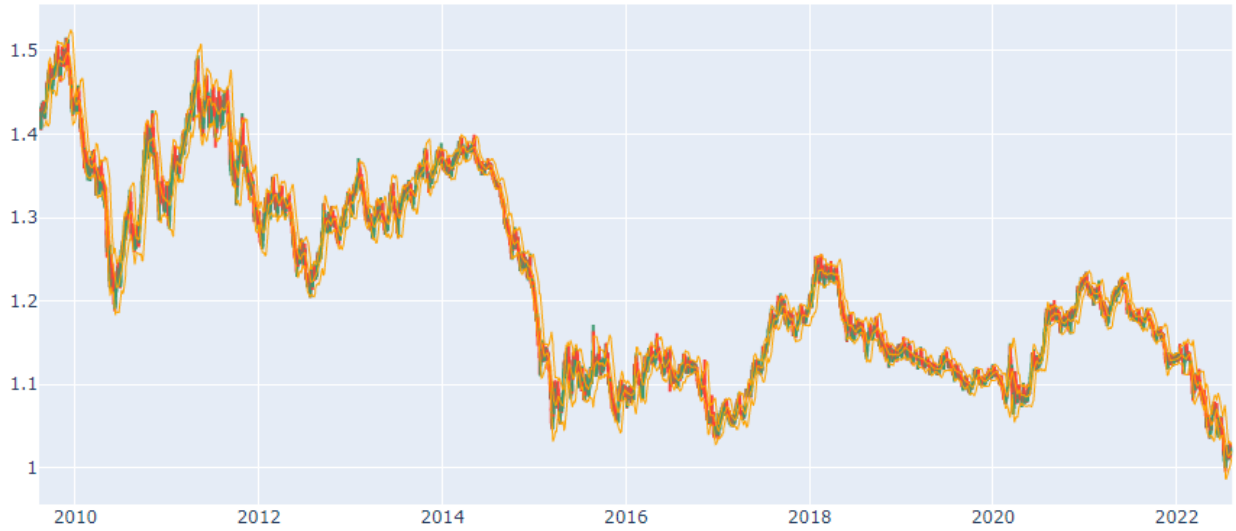


Figura 4.8: Bandas de Bollinger

MACD

MACD es un acrónimo para Moving Average Convergence Divergence[2]. Este indicador técnico es una herramienta que se utiliza para identificar medias móviles que indican una nueva tendencia, ya sea alcista o bajista.

El MACD representa 3 valores, cada uno de los cuales está interconectado. A continuación se muestran las principales señales primarias del MACD.

- MACD: el valor de una media móvil exponencial (EMA) restado de otro EMA con un período retrospectivo más corto. Los valores comunes son 26 días para la EMA más larga y 12 para la más corta. Esto se conoce como la línea de señal 'lenta' del MACD.
- Señal: la EMA del MACD de un período más corto que el período más corto utilizado para calcular el MACD. Por lo general, se utiliza un período de 9 días. Esto se conoce como la línea de señal 'rápida'.
- Diferencia: la diferencia entre el MACD y la línea de activación se utiliza para representar las presiones de venta actuales en el mercado. Este valor se representa comúnmente como un histograma superpuesto a las señales MACD + Trigger. Un valor positivo de esta señal representa una tendencia alcista, mientras que un valor negativo indica una tendencia bajista[26].

Veamos cómo queda.

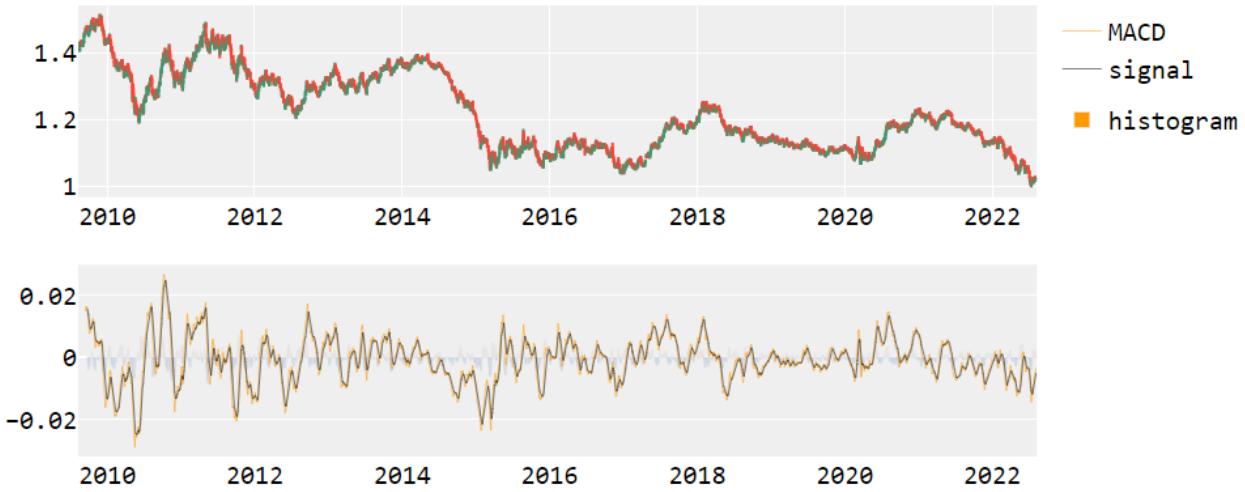


Figura 4.9: MACD

Oscilador estocástico

El oscilador estocástico es otro indicador técnico que ayuda a los inversores a determinar dónde podría estar terminando una tendencia. Este oscilador de impulso simple fue creado por George Lane a fines de la década de 1950[17]. El indicador funciona según la siguiente teoría:

Durante una tendencia alcista, los precios permanecerán iguales o por encima del precio de cierre anterior. Durante una tendencia bajista, es probable que los precios permanezcan iguales o por debajo del precio de cierre anterior.

El oscilador estocástico está compuesto por dos medias móviles $\bar{x}_p$ y $\bar{x}_k$, que se obtienen utilizando las siguientes fórmulas:

$$\bar{x}_p = 100\frac{close - low}{high - low} \tag{4.6}$$

$$\bar{x}_k = \frac{1}{14}\sum_{i-1}^{14} \bar{x}_{p-i} \tag{4.7}$$

Nótese que $\bar{x}_k$ es la media móvil simple de 14 periodos centrada en $\bar{x}_p$.
Veamos el resultado final.

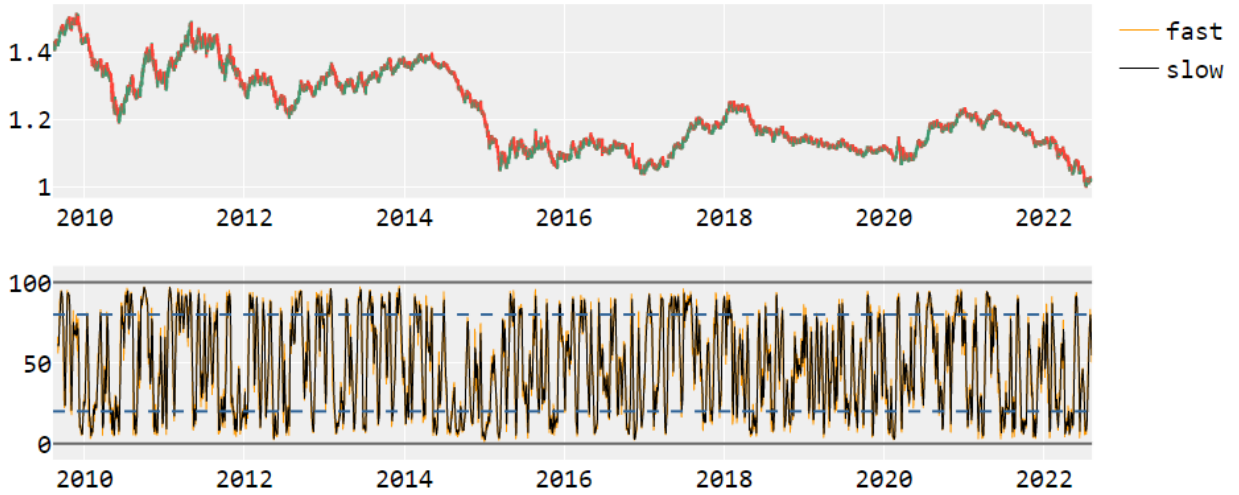


Figura 4.10: Oscilador estocástico

ADX

El ADX (Average Directional Index)[1] es un indicador técnico que se usa comúnmente para medir la fuerza de la tendencia del mercado. Sin embargo, el ADX no mide la dirección de la tendencia, ya sea alcista o bajista, sino que solo representa cómo de fuerte es la tendencia. Por lo que, para identificar la dirección de la tendencia, el ADX se combina con un índice direccional positivo (+ DI) y un índice direccional negativo (- DI). Como sugiere el nombre, el + DI mide la tendencia alcista o positiva del mercado, de manera similar, el - DI mide la tendencia bajista o negativa del mercado. Los valores de todos los componentes están limitados entre 0 y 100, por lo que actúan como un oscilador. La fórmula para calcular tanto el + DI como el - DI se puede

representar de la siguiente manera:

$$\bar{x}_p = \frac{1}{14}\sum_{i-1}^{14} x_{p-i} \tag{4.8}$$

$$\bar{x}_h = \frac{1}{14}\sum_{i-1}^{14} (x_{h-i} - x_{h-i-1}) \tag{4.9}$$

$$\bar{x}_l = \frac{1}{14}\sum_{i-1}^{14} (x_{l-i} - x_{l-i-1}) \tag{4.10}$$

$$+DI14 = 100\frac{\bar{x}_h}{\bar{x}_p} \tag{4.11}$$

$$-DI14 = 100\frac{\bar{x}_l}{\bar{x}_p} \tag{4.12}$$

Siendo x_h el máximo valor alcanzado en el período de hoy y x_l el mínimo. Con esto podemos ya calcular en python el valor del indicador y añadirlo a nuestro dataset.

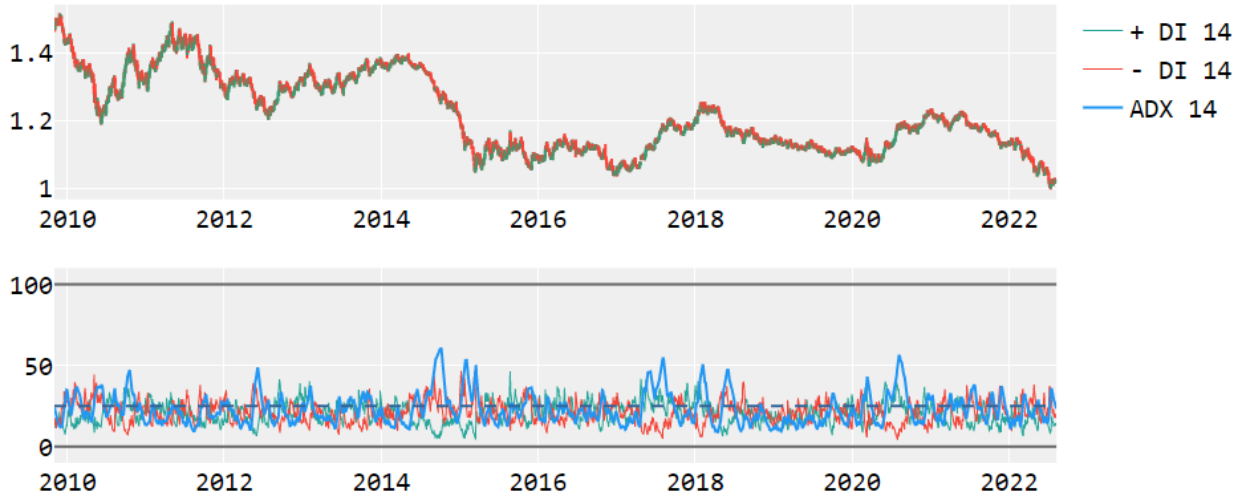


Figura 4.11: ADX

4.3.7. Análisis estadístico

Como ya realizamos un análisis estadístico del dataset de 1979, y este dataset contiene los mismos datos que el anteriormente mencionado, dejaremos el desarrollo completo del análisis en el apéndice A.1 para el interesado. Aunque casi todos los resultados son idénticos, sí cabe destacar lo siguiente:

1. Al crear las nuevas variables explicativas hemos generado nulos en nuestro dataset. Como tenemos una gran cantidad de observaciones y el porcentaje de nulos es muy pequeño, decidimos eliminar las observaciones con observaciones faltantes de nuestro dataset. Lo único que le ocurre al dataset es que perdemos las observaciones de agosto de 2009 a octubre de ese mismo año. Por lo que esta decisión no afectará de forma notable al modelado.

2. En el gráfico de correlaciones 'Ver figura A.2' podemos ver que las medias móviles del grupo 'corto' de las medias móviles de GUPPY presentan correlación 1. Al igual que las del grupo 'largo'. Esto representa que, en realidad, nos están ofreciendo la misma información. Por ello decidimos quedarnos con una sóla media móvil de cada grupo. Del grupo corto escogemos la MA8 y del largo, la MA40, por ser las medias móviles centrales de cada grupo. El gráfico de correlaciones ya con las variables eliminadas se puede ver en A.3 y en él observamos que las correlaciones de las variables objetivo updownpred y closepred son muy pequeñas lo que va a dificultar obtener modelos muy buenos. Al menos, existen algunas correlaciones superiores a 0,2 con las variables explicativas introducidas en la sección 4.3.5, cosa que no ocurría en el otro dataset con ninguna variable.

3. Por último, en la figura 4.12, podemos observar que esta vez tenemos los datos balanceados, lo que va a facilitar la creación de nuestros modelos. Ahora, por tanto, el umbral de referencia para que nuestros modelos se consideren mínimamente válidos es 50,42 %.

4.3.8. Estandarización y train-test

Unimos estas dos secciones en una pues se va a realizar de la misma forma que en el capítulo 4. Únicamente notar que como vamos a predecir dos variables objetivo generaremos 4 conjuntos diferentes de entrenamiento y test para poder generar posteriormente los modelos de la forma más limpia posible.

1. Para las predicciones de updownpred, eliminaremos closepred y generaremos dos conjuntos de entrenamiento y test. Uno para el dataset estandarizado y otro para el sin estandarizar.

2. Para las predicciones de closepred, haremos lo mismo solo que eliminando la variable updownpred.

De esta manera, además de poder evaluar las diferentes técnicas de modelado entre sí, podremos hacer un análisis para cada método sobre si utilizar variables estandarizadas mejora la predicción.

Recordemos que reservamos para test los años 2021 y 2022, los cuales, al haber reducido el tamaño del dataset, representan el 12,5 % de los datos, reduciendo así la posibilidad de tener overfitting al generar nuestros modelos.

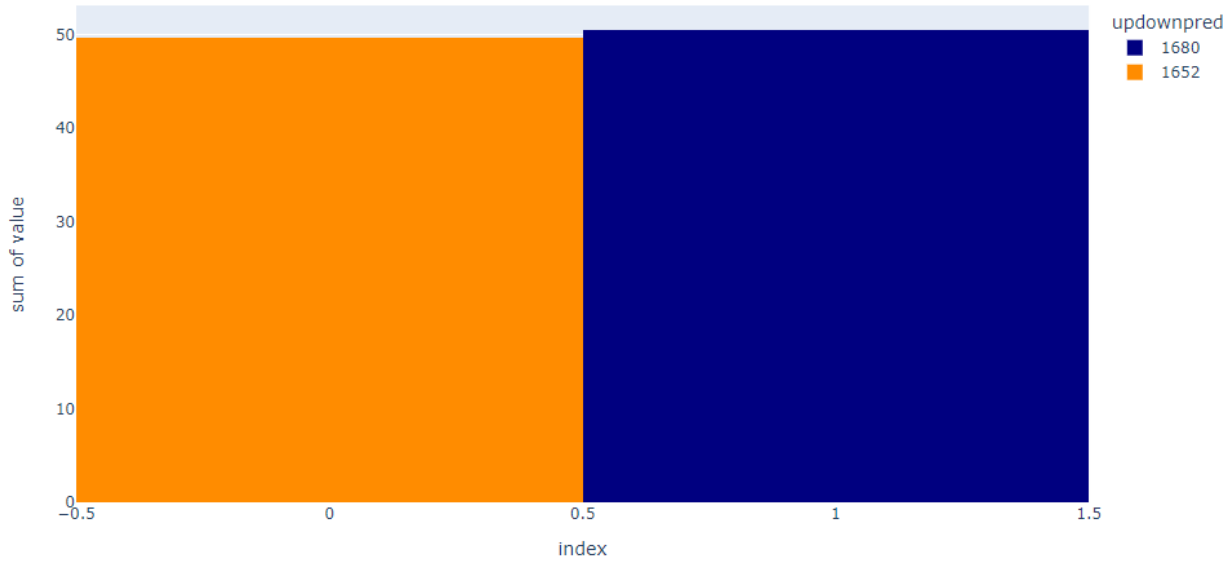


Figura 4.12: Conteo de updownpred

4.4. Modelos

Veamos un resumen de las variables finales que se utilizarán en la creación de los modelos.

Variable	Descripción
Close	Valor de cierre del EURUSD en la fecha indicada
Open	Valor de apertura del EURUSD en la fecha indicada
High	Valor máximo del EURUSD en la fecha indicada
Low	Valor mínimo del EURUSD en la fecha indicada
Var	Value at Risk. Máximo porcentaje de pérdida en una inversión
Pips	Diferencia entre el precio de apertura y el de cierre
Variation	Diferencia entre los valores máximo y mínimo alcanzados en la fecha indicada
Updown	Variable binaria que indica si e precio de cierre es superior o inferior al de apertura
MA8	Média móvil basada en los últimos 8 períodos
MA40	Média móvil basada en los últimos 40 períodos
topBB	Valor superior de las bandas de Bollinger
lowBB	Valor inferior de las bandas de Bollinger
MACD-12-26-9	Valor final del indicador MACD
MACDh-12-26-9	Média móvil exponencial de 26 períodos utilizada en el MACD
MACDs-12-26-9	Média móvil exponencial de 9 periodos utilizada en el MACD
STOCHk-14-3-3	Primera componente del oscilador estocástico
STOCHd-14-3-3	Média móvil centrada en la primera componente del indicador estocástico
plus-di	Componente del indicador ADX que mide la tendencia alcista del mercado
minu-di	Componente del indicador ADX que mide la tendencia bajista del mercado
adx	Valor final del indicador ADX
Updownpred	Variable objetivo binaria. Updown del día siguiente al indicado
Closepred	Variable objetivo continua. Close del día siguiente al indicado

Con el dataset ya preparado podemos implementar los diferentes modelos que vamos a estudiar. Como cada procedimiento ya se ha presentado matemáticamente en el capítulo 3, en esta sección haremos únicamente una breve explicación del código y de la elección de los hiperparámetros, para así centrarnos en los resultados y en la evaluación y comparación de los modelos.

4.4.1. Regresión logística

Probamos primero con la regresión logística 'Véase la sección 3'. Para implementarla, debemos añadir una columna de unos para el intercept del modelo a la matriz de predictores y generar el modelo con ese nuevo conjunto de entrenamiento. Observemos los resultados.

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.688431
         Iterations: 35
                           Logit Regression Results
==============================================================================
Dep. Variable:             updownpred   No. Observations:                 2915
Model:                          Logit   Df Residuals:                     2895
Method:                           MLE   Df Model:                           19
Date:                Fri, 02 Sep 2022   Pseudo R-squ.:                0.006469
Time:                        07:57:57   Log-Likelihood:                -2006.8
converged:                      False   LL-Null:                       -2019.8
Covariance Type:            nonrobust   LLR p-value:                    0.1265
==============================================================================
```

```
                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const             0.5853      0.575      1.017      0.309      -0.543       1.713
close            -1.2700        nan        nan        nan         nan         nan
open             -1.2692        nan        nan        nan         nan         nan
high            -19.0950        nan        nan        nan         nan         nan
low             -19.0955        nan        nan        nan         nan         nan
var               1.1423      0.394      2.896      0.004       0.369       1.915
pips             -0.0094        nan        nan        nan         nan         nan
variation        -0.0010        nan        nan        nan         nan         nan
updown           -0.1236      0.112     -1.107      0.268      -0.342       0.095
MA8               0.6497     18.549      0.035      0.972     -35.706      37.006
MA40             22.9098     15.772      1.453      0.146      -8.003      53.822
topBB            11.6727     10.675      1.093      0.274      -9.249      32.595
lowBB             5.4888     10.966      0.501      0.617     -16.004      26.982
MACD_12_26_9     75.2198        nan        nan        nan         nan         nan
MACDh_12_26_9    79.0527        nan        nan        nan         nan         nan
MACDs_12_26_9    -3.8330        nan        nan        nan         nan         nan
STOCHk_14_3_3     0.0070      0.008      0.914      0.361      -0.008       0.022
STOCHd_14_3_3    -0.0108      0.008     -1.377      0.168      -0.026       0.005
plus_di           0.0078      0.011      0.738      0.461      -0.013       0.029
minus_di         -0.0184      0.011     -1.641      0.101      -0.040       0.004
adx              -0.0039      0.004     -0.941      0.347      -0.012       0.004
==============================================================================
```

Como se puede ver, el modelo no logra converger correctamente y la mayoría de variables no son significativas (P-valor>0.05). Esto puede ser un indicador de que el modelo no va a lograr predecir muy bien las observaciones. Utilicemos las métricas definidas en la sección 3 para evaluar de manera más precisa los resultados. Recordemos que la regresión logística no devuelve exactamente 0 o 1, sino que devuelve un valor probabilístico. Debemos aproximar las predicciones primero y así podemos evaluar los resultados obtenidos.

```
              precision    recall  f1-score   support

           0       0.41      0.53      0.46      1116
           1       0.65      0.54      0.59      1799

    accuracy                           0.53      2915
   macro avg       0.53      0.53      0.52      2915
weighted avg       0.56      0.53      0.54      2915
```

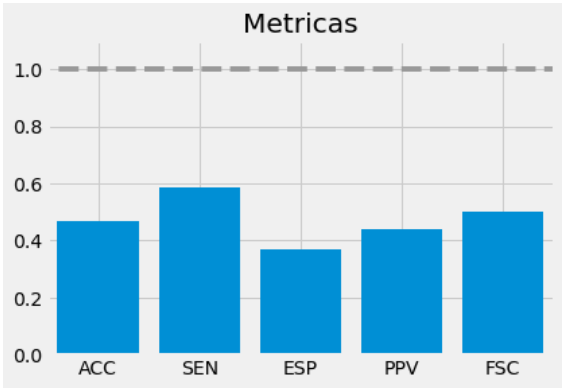
En los datos de entrenamiento acierta el 53 % de las predicciones, es un porcentaje bastante bajo pero supera al menos el umbral de referencial. Veamos su rendimiento en los datos de test.

```
              precision    recall  f1-score   support

           0       0.37      0.51      0.43       162
           1       0.59      0.44      0.50       255

    accuracy                           0.47       417
   macro avg       0.48      0.48      0.47       417
weighted avg       0.50      0.47      0.47       417
```

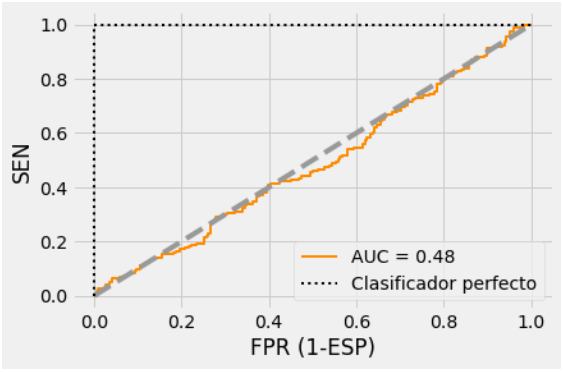
Acierta un 47 %, peor que si decidieramos elegir al azar. No tenemos un modelo válido. Observemos el resto de métricas:

```
Confusion matrix:     array([[ 83, 143],
```

```
                            [ 79, 112]], dtype=int64)
Accuracy:  0.4676258992805755
Recall (Sensitivity):  0.5863874345549738
Specificity:  0.3672566371681416
Precision:  0.4392156862745098
F1:  0.5022421524663677
```



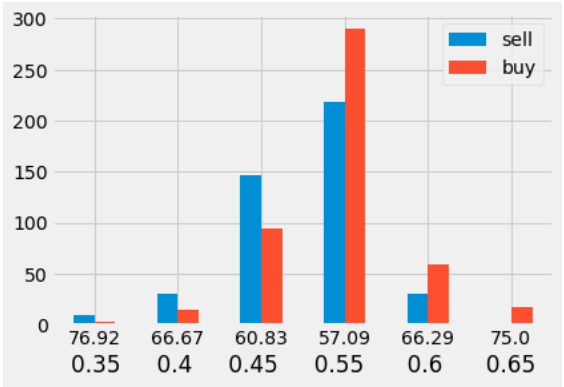
(a) Métricas regresión



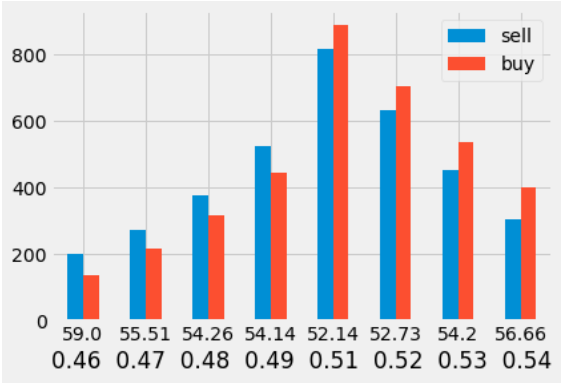
(b) Curva ROC regresión

De estas métricas se deduce que el modelo predice un poco mejor cuándo el mercado va a bajar que cuando va a subir, pero la diferencia es tan pequeña que casi no merece la pena tenerlo en cuenta. Los resultados son negativos, pero confirman lo que ya observábamos en el análisis estadístico de la sección 4.3.7 con el gráfico de correlaciones. EL tener correlaciones tan bajas afecta negativamente en el desempeño de este modelo.

Aún así, recordemos que la regresión logística devuelve un número entre el 0 y el 1. Hagámos un análisis más profundo sobre el porcentaje de aciertos que obtenemos cuando las predicciones reales se alejan un poco del 0.5 central. Confirmamos que nuestro modelo de regresión logística



(a) Gráfico de barras por probabilidad 1



(b) Gráfico de barras por probabilidad 2

predice mejor las bajadas que las subidas. Y resulta interesante ver que, aunque se tenga un número reducido de veces que la regresión logística supera el 0,6 o es inferior al 0,45, es notable cómo la cantidad de aciertos aumenta. Con esto a lo mejor se podría generar un algoritmo que invierta sólo cuando la regresión logística sobrepase esos valores. Lo estudiaremos más adelante.

Con el dataset estandarizado se generaba el mismo modelo de regresión, por lo que hemos decidido no repetir los mismos resultados dos veces.

4.4.2. Adaptative boosting

Avancemos ahora con uno de los métodos de boosting más utilizados. Recordemos del capítulo 3 que el modelo de adaptative boosting está diseñado para evitar el overfitting, reduciendo el sesgo y la varianza, por lo que esperamos obtener resultados parecidos entre el conjunto de entrenamiento y el de test.

Una de las grandes ventajas del adaptative boosting es que requiere de pocos parámetros para funcionar óptimamente (a diferencia de otros modelos de boosting como el SVM o el XGBoost). Por lo que únicamente vamos a tener que definir los siguientes:

1. Learning rate: La velocidad a la que queremos que aprenda el modelo. Cuanto más pequeño sea más despacio aprende pero puede encontrar resultados mejores. Nosotros la dejaremos en un 0.01.

2. Base estimator: El estimador base sobre el que se basa el modelo. Por defecto se utiliza los árboles de decisión. Pero tras varias pruebas el que mejor resultados dió fue la regresión logística.

3. Number of estimators: El número de estimadores base que utilizará nuestro modelo. Lo dejamos en 1000.

Con todo ello, y tras generar el modelo, podemos ver los resultados obtenidos.

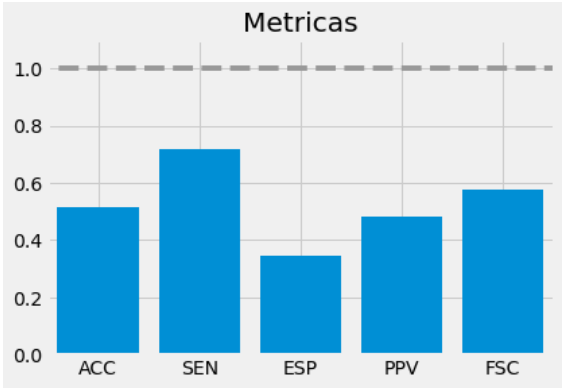
	precision	recall	f1-score	support
0	0.52	0.36	0.43	1426
1	0.53	0.69	0.60	1489
accuracy			0.53	2915
macro avg	0.53	0.52	0.51	2915
weighted avg	0.53	0.53	0.51	2915

Obtenemos un 53 % de aciertos, igual que en la regresión logística. Veamos si esta vez se mantiene con el conjunto de test.

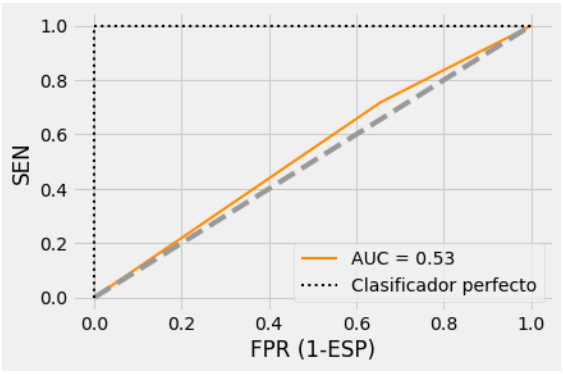
	precision	recall	f1-score	support
0	0.59	0.35	0.44	226
1	0.48	0.72	0.58	191
accuracy			0.52	417
macro avg	0.54	0.53	0.51	417
weighted avg	0.54	0.52	0.50	417

Como predijimos antes, el modelo no ha generado prácticamente nada de overfitting y obtenemos un 52 % de aciertos en el conjunto de test. Sólo supera por un 1,5 % el umbral de referencia pero mejora en 5 puntos el 46 % obtenido en la regresión logística. Veamos el resto de métricas.

```
Confusion matrix:    array([[ 78, 148],
                            [ 54, 137]], dtype=int64)
Accuracy:  0.5155875299760192
Recall (Sensitivity):  0.7172774869109948
Specificity:  0.34513274336283184
Precision:  0.4807017543859649
F1:  0.5756302521008403
```



(a) Métricas adaptative boosting



(b) Curva ROC adaptative boosting

Observemos que éste modelo es también mejor prediciendo las caídas del mercado que las subidas con una sensitivity de 0,71. Ésta vez, la curva ROC supera a la línea de aleatoriedad. Tenemos un modelo válido, aunque sus predicciones sean muy pobres.

En el análisis del dataset estandarizado se obtienen resultados peores que con el principal, así que no lo tendremos en cuenta y no lo incluiremos en el dataset de predicciones que vamos desarrollando.

date	real	log	log_conf	adap	adap_conf
2009-11-02	0	1	0.519419	1	0.501833
2009-11-03	1	1	0.614016	1	0.504944
2009-11-04	1	0	0.495720	0	0.498727
2009-11-05	0	0	0.499710	1	0.501139
2009-11-06	1	1	0.528791	1	0.501287
...	...	...	...	...	...
2022-08-03	1	1	0.506075	0	0.498097
2022-08-04	0	0	0.493650	0	0.497151
2022-08-05	1	0	0.478417	1	0.501058
2022-08-08	1	0	0.499029	0	0.499493
2022-08-09	1	0	0.496667	1	0.500604

Observemos que las predicciones reales del adaptative boosting se encuentran, esta vez, muy próximass al 0.5. Esto indica que el propio algoritmo no era muy capaz de decidirse si decir 1 o 0. Aún así, veamos un gráfico en el que se analizan un poco las predicciones que se alejan mínimamente. Se observa cierta mejora en el porcentaje de aciertos pero no es tan notable como en la regresión logística. Programar un algoritmo fijándonos únicamente en esto sería muy arriesgado.

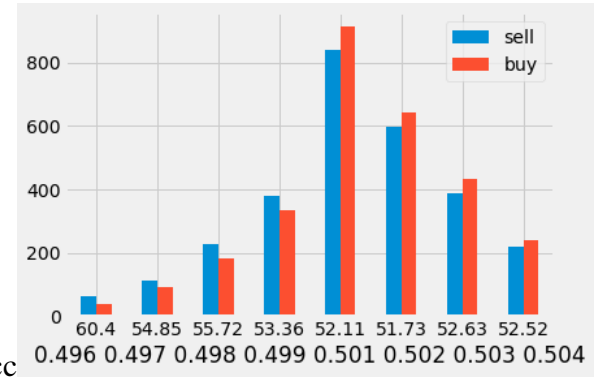
cc 

Figura 4.16: Gráfico de barras por probabilidad ADAPBOOST

4.4.3. Random Forest

Analicemos ahora, a contrapartida al modelo anterior. El Random Forest, uno de los mejores métodos de clasificación y predicción que utilizan el método de bagging. Como ya vimos en la sección 3 estos métodos tienden a provocar muchísimo overfiiting si no se controla. En nuestro caso intentamos controlar esto, pero nos dimos cuenta de que dejando que el modelo clasificara al máximo los datos de entrenamiento obteníamos mejores resultados después con el conjunto de test, que si controlábamos el overfitting reduciendo la profundidad del árbol. Gracias a eso se obtienen los siguientes resultados.

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1426
           1       1.00      1.00      1.00      1489

    accuracy                           1.00      2915
   macro avg       1.00      1.00      1.00      2915
weighted avg       1.00      1.00      1.00      2915
```

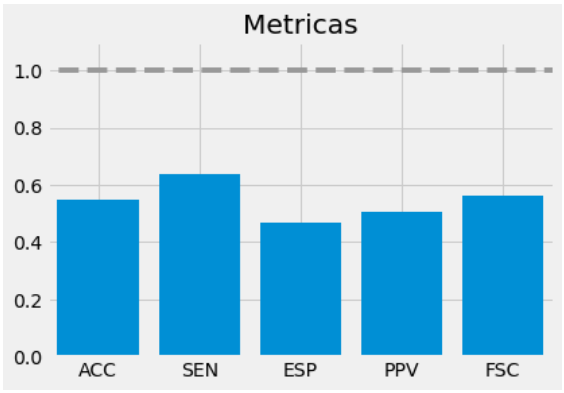
Como dijimos, obtenemos un 100 % de aciertos en el conjunto de entrenamiento. Esto no es porque el modelo sea perfecto, si no porque ha generado overfitting. Veamos su desempeño con los datos de test.

```
              precision    recall  f1-score   support

           0       0.61      0.47      0.53       226
           1       0.50      0.64      0.56       191

    accuracy                           0.55       417
   macro avg       0.55      0.55      0.55       417
weighted avg       0.56      0.55      0.54       417
```

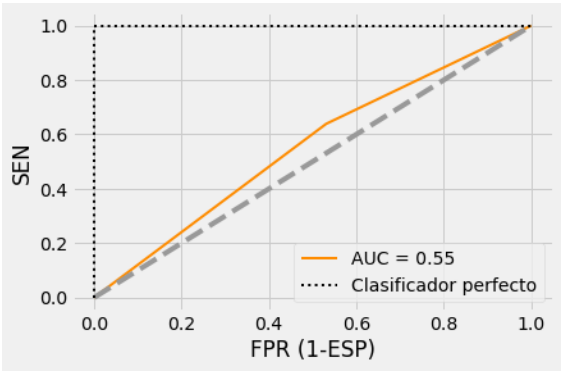
Obtenemos un 55 % de aciertos. El más alto hasta ahora. Veamos el resto de métricas.

```
Confusion matrix:     array([[106, 120],
                             [ 69, 122]], dtype=int64)
Accuracy:  0.5467625899280576
Recall (Sensitivity):  0.6387434554973822
```

```
Specificity:  0.4690265486725664
Precision:  0.5041322314049587
F1:  0.5635103926096997
```



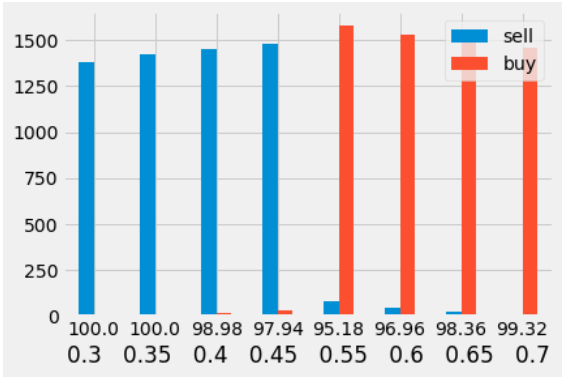
(a) Métricas random forest



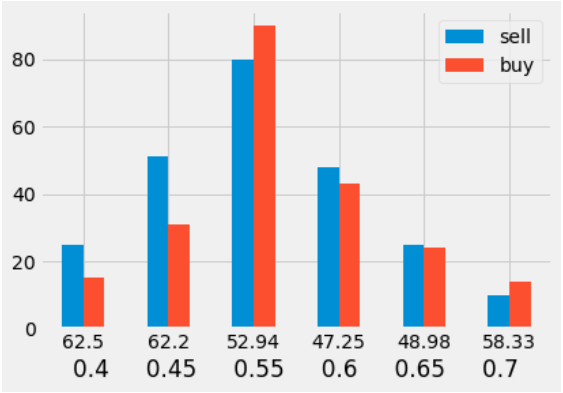
(b) Curva ROC random forest

Otra vez nuestro modelo predice muy bien las caídas del mercado con 106 aciertos sobre 69 fallos, es una notable diferencia. Además tenemos un valor de la curva ROC de 0.55, consolidando así al random forest como el mejor modelo desarrollado hasta ahora.

Otra vez, el modelo con el dataset estandarizado ha dado peores resultados por lo que no repetiremos su análisis aquí. Veamos los gráficos de barras como en los otros modelos. En la



(a) Gráfico de barras con overfitting



(b) Gráfico de barras sobre test

izquierda se puede observar el gráfico teniendo en cuenta todos los datos. Aparece contaminado por las predicciones con overfitting del conjunto de entrenamiento. Esto es algo que vamos a tener muy presente a la hora de desarrollar los algoritmos de inversión. También nos fijamos que, aunque aumente la confianza con la que el modelo nos ofrece las predicciones de subida el porcentaje de aciertos no mejora, mientras que en las de bajada sí. Esto nos confirma que el modelo es muy adecuado para predecir las caídas del mercado y un algoritmo que gire en torno a esto puede resultar muy interesante.

4.4.4. Red neuronal

Ya hemos explicado en la sección 4.3.3 el proceso de desarrollo de la red neuronal, por lo que no lo repetimos otra vez y nos centramos en los resultados. Sí que debemos mencionar que al probar los diferentes hiperparámetros, un alto número de pruebas generaban una red neuronal no válida que únicamente decía unos. Para solucionar esto hemos definido el gridsearch de tal manera que intentara obtener la mejor puntuación en 4 métricas diferentes (accuracy, precision, f1 y neg mean squared error) finalmente obteniendo la siguiente red neuronal.

```
{'activation': 'logistic',
 'alpha': 0.046416,
 'batch_size': 'auto',
 'beta_1': 0.9,
 'beta_2': 0.999,
 'early_stopping': False,
 'epsilon': 1e-08,
 'hidden_layer_sizes': (100, 150),
 'learning_rate': 'constant',
 'learning_rate_init': 0.001,
 'max_fun': 15000,
 'max_iter': 200,
 'momentum': 0.9,
 'n_iter_no_change': 10,
 'nesterovs_momentum': True,
 'power_t': 0.5,
 'random_state': 0,
 'shuffle': True,
 'solver': 'adam',
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': False,
 'warm_start': False}
```

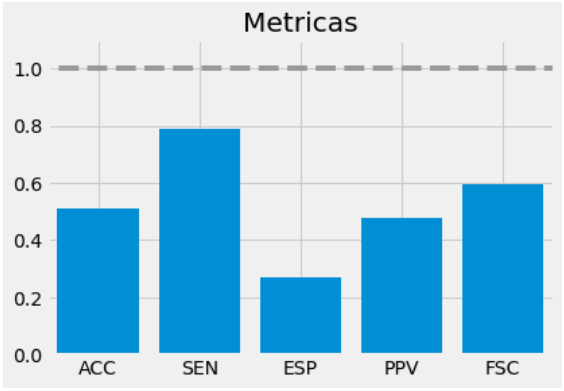
Recordemos que es importante a la hora de realizar una red neuronal que los datos estén estandarizados. Por ello hemos recurrido al dataset estandarizado para realizarla desde un principio. Veamos su rendimiento con los datos de entrenamiento y los de test.

	precision	recall	f1-score	support
0	0.53	0.33	0.40	1426
1	0.53	0.72	0.61	1489
accuracy			0.53	2915
macro avg	0.53	0.52	0.51	2915
weighted avg	0.53	0.53	0.51	2915

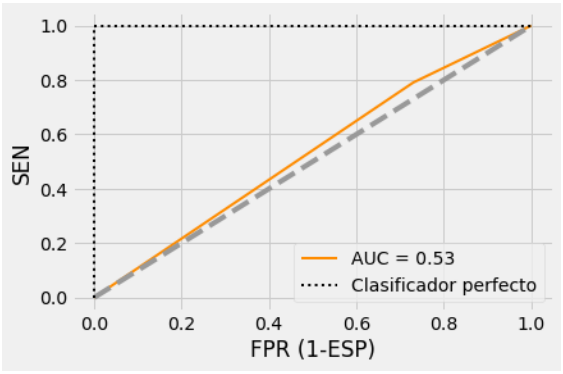
	precision	recall	f1-score	support
0	0.60	0.27	0.37	226
1	0.48	0.79	0.60	191
accuracy			0.51	417
macro avg	0.54	0.53	0.48	417
weighted avg	0.55	0.51	0.48	417

Resultados bastante pobres con un 53 % y 51 % de aciertos. Comprobemos el resto de métricas.

```
Confusion matrix:    array([[ 61, 165],
                            [ 40, 151]], dtype=int64)
Accuracy:  0.5083932853717026
Recall (Sensitivity):  0.7905759162303665
Specificity:  0.26991150442477874
Precision:  0.4778481012658228
F1:  0.5956607495069033
```

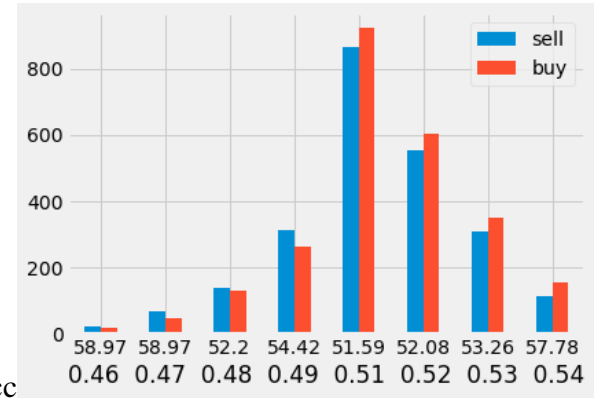


(a) Métricas red neuronal



(b) Curva ROC red neuronal

Aunque tengamos un valor de la curva ROC de 0,53, el resto de métricas indica que la red neuronal realiza las predicciones un poco peor que el adaptative boosting. Además de decir un 73 % de las veces que el mercado va a subir aunque su índice de aciertos sea del 48 %. Realicemos el ya conocido gráfico de barras que clasifica según la confianza. Se puede observar que los valores no



cc

Figura 4.20: Gráfico de barras por probabilidad red neuronal

alcanzan valores superiores a 0.55 ni inferiores a 0.45. Pero sí se mejora un poco el porcentaje de aciertos al separarse del 0,5. Aunque es tan pequeña la mejora que no merecería la pena tenerla en cuenta.

4.4.5. Observaciones

Tras el desarrollo y evaluación de los cuatro métodos. Podemos corroborar la hipótesis planteada al comienzo de este trabajo y concluir que la predicción diaria de algo tan cambiante y volátil como es el mercado de divisas es, de momento, imposible con los modelos que tenemos hoy en día.

A pesar de haber reducido la dificultad de la tarea a sólo saber si va a subir o bajar al día siguiente, el mejor resultado que hemos obtenido es de un 57 % de aciertos con el modelo de random forest, seguido de un 52 % generado por adaptative boosting, un 51 % ofrecido por la red neuronal y un 46 % de la regresión logística (que casi que es mejor hacer lo contrario a lo que nos diga para aumentar ese porcentaje).

En resumen, en línea de lo que le ocurría a otros autores, obtenemos resultados demasiado cercanos al 50 % como para que sean significativos y no se recomendaría invertir siguiendo únicamente lo que indique cualquiera de estos algoritmos.

Una posibilidad de futuro desarrollo podría aparecer tras la observación de que los 4 modelos han ofrecido unos mejores resultados a la hora de predecir los días en los que el mercado era bajista. Pero es una posibilidad pequeña y no podemos asegurar que esto no se haya dado por la tendencia bajista que tiene el euro con respecto al dólar desde 2009.

En conclusión, hay ciertos conjuntos de datos en los que las variables explicativas ofrecen información real sobre las variables a predecir. En esos casos, estos modelos ofrecen predicciones fiables con altos porcentajes de acierto. Por ello, el tratamiento estadístico de los datos es tan importante. Con él podemos dar una idea de antemando de la calidad de predicciones que podemos obtener de un conjunto de datos y ahorrarnos el trabajo de probar todos los modelos en el caso de que ese tratamiento prevea que las predicciones no van a ser suficientemente buenas.

4.5. Algoritmos de inversión

Hemos dicho en el apartado anterior que predecir los movimientos diarios del mercado de divisas es, por el momento, una tarea imposible. Pero multitud de inversores son capaces de obtener grandes cantidades de dinero invirtiendo en este mercado. Ellos no invierten cada día intentando predecir el día siguiente, si no que esperan pacientemente a que el grupo de indicadores que utilizan les muestren una señal de compra o de venta clara.

Teniendo en cuenta que los modelos desarrollados en este trabajo se basan en los indicadores más utilizados por estos inversores tal vez podamos crear un algoritmo que tenga en cuenta las predicciones obtenidas de forma que lo que se consiga sean señales de compra o venta que podemos utilizar para invertir.

Un ejemplo de un algoritmo de este tipo es el conocido como estrategia Benchmark[13]. En la cual las diferentes entradas y salidas de las inversiones están vinculadas a un índice o combinación de índices del sector o cualquier otro, como el S&P 500. Si aplicamos este algoritmo a nuestra serie temporal 'Ver apéndice A.2', obtenemos el siguiente resultado.

```
Benchmark profit by investing \$100k : 43 296.18
Benchmark Profit percentage : 43%
```

Como no se tiene un algoritmo perfecto que podamos utilizar de referencia para generar las métricas utilizadas en la sección 4.3.2. Lo que haremos para valorar la precisión de nuestros algoritmos será calcular el beneficio que se habría obtenido de seguir sus indicaciones con un capital de 100.000$ y calcular el porcentaje de beneficio asociado. Después los compararemos con los resultados del algoritmo Benchmark para valorar correctamente las predicciones obtenidas. Como el objetivo va a ser encontrar fuertes puntos de compra o de venta en la serie, consideramos 2 opciones para la creación de nuestro algoritmo:

1. Podemos suponer que cuando tres o más modelos coinciden en la predicción es probable que en ese punto haya una señal de compra o de venta, puesto que con las variables explicativas que hemos definido éstos puntos deberían de ser de los que tenemos más información. Por lo que los modelos habrían tenido más facilidad para predecirlos y que haya varios que coincidan en la predicción. De esta forma podemos desarrollar dos algoritmos diferentes: uno que invierta cuando coinciden 3 modelos (lo llamaremos el de alto riesgo) y uno que invierta cuando coinciden los 4 (éste lo denominaremos el de bajo riesgo).

2. La otra opción se basa en seguir un procedimiento parecido al anterior pero que se base en los valores de confianza que hemos obtenido en la creación de los modelos. De manera que cuando tres o más de ellos superen cierto valor, consideremos un punto fuerte de compra. Mientras que un punto de venta se obtendrá cuando queden por debajo de otro nivel de confianza.

4.5.1. Algoritmos basados en las puntuaciones de confianza

Tras varias pruebas 'Para el código completo junto con los resultados de estos algoritmos ver apéndice A.3', para los algoritmos basados en las puntuaciones de confianza, obtenemos que los valores que dan mejores resultados son:

1. Para los puntos de compra: La regresión logística y el random forest, superiores a 0.55, el adaptative boosting por encima de 0.501 y que la red neuronal sobrepase 0.51.

2. Para los puntos de venta: La regresión logística y el random forest por debajo de 0.45, el adaptative boosting inferior a 0.499 y la red neuronal que no superara 0.49.

Con estas condiciones para invertir desarrollamos los dos algoritmos ya mencionados y obtenemos los siguientes resultados:

Para el de alto riesgo (que coincidan tres) obtenemos:

```
Profit gained from the algoritmic conf 3 strategy by investing \$100k in EURUSD : 48408.28
Profit percentage of the algoritmic conf 3 strategy : 48%
```

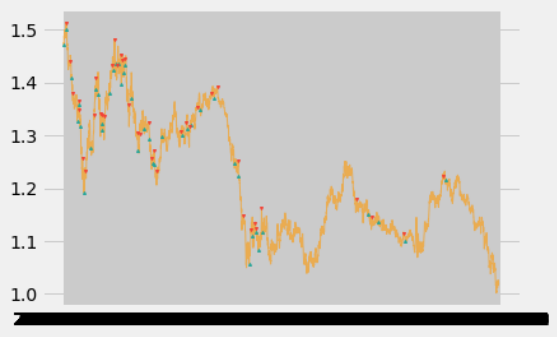
Un 5 % superior al resultado obtenido utilizando la estrategia Benchmark.

Para el de bajo riesgo (que coincidan los 4 modelos) se obtiene:

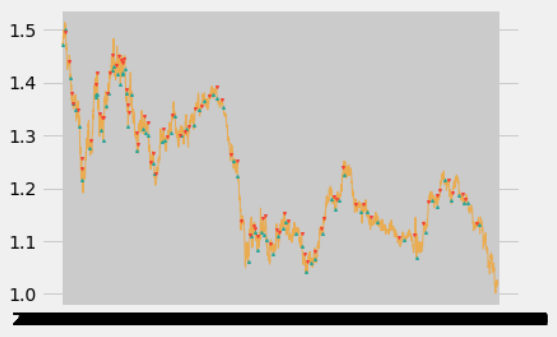
```
Profit gained from the algoritmic conf 4 strategy by investing \$100k in EURUSD : 31358.15
Profit percentage of the algoritmic conf 4 strategy : 31%
```

Obtenemos peores resultados que mediante la estrategia Benchmark. Aunque ganemos dinero siguiendo este algoritmo, no supera la prueba como para ser un algoritmo que podamos considerar útil y fiable.

Observemos cómo quedan los puntos de compra y de venta utilizando estos algoritmos en las figuras 4.21a y 4.21b. 'Se pueden observar estos gráficos ampliados junto con el código que hemos utilizado para generarlos en las figuras A.5 y A.6'. Observamos que, en general, ambas



(a) Puntos de entrada algoritmo conf3



(b) Puntos de entrada algoritmo conf4

estrategias encuentran muy buenas ocasiones de compra y de venta. Pero que cuando se equivocan las pérdidas son grandes y eso hace que nuestros resultados no sean tan buenos. Tal vez con otro algoritmo que defina las órdenes stop-loss, reduciríamos esas pérdidas y obtendríamos resultados mucho mejores. Puede ser un camino a seguir desde éste trabajo.

Vemos también, que hay pocas entradas para invertir y eso hace que el riesgo asociado a las inversiones que se hacen sea muy grande. Veremos ahora si con el otro planteamiento tenemos más entradas y solucionamos esto.

4.5.2. Algoritmos basados en los resultados directos de los modelos

Con el algoritmo creado anteriormente se obtenían unos resultados interesantes, pero observábamos el problema de que había pocos puntos de compra o venta claros. Ësto hacía que cuando se entraba en una inversión se estuviera mucho tiempo sin salir de ella y provocaba que inversiones que en un inicio eran correctas terminaran provocando pérdidas.

Utilizando los resultados de los algoritmos directamente puede resolver este problema, puesto que las condiciones para entrar en una inversión son menos restrictivas. Esto provocará más puntos de entrada y salida y reducirá el tiempo en el que estemos en una inversión.

Como hemos explicado, generaremos dos algoritmos con esta técnica 'Véase apéndice A.4.2: uno que tenga como condición de entrada que coincidan 3 modelos (el que llamaremos de alto riesgo) y otro en el que deban coincidir los 4 modelos (el de bajo riesgo).

Para el de alto riesgo obtenemos:

```
Profit gained from the algoritmic 3 strategy by investing \$100k in EURUSD : 127743.81
Profit percentage of the algoritmic 3 strategy : 127%
```

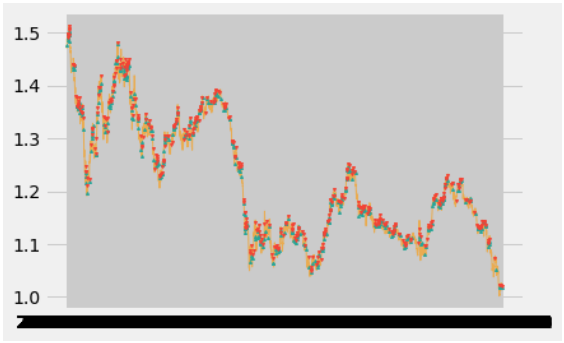
Superamos en un 79 % los resultados de la estrategia Benchmark. Esto es un resultado muy positivo, que nos habría permitido doblar la cantidad de dinero invertida en sólo 13 años. Puesto que el rendimiento obtenido es 2'5 veces mejor que el que se obtendría siguiendo la estrategia Benchmark podemos considerar que este algoritmo hace predicciones realmente buenas.

Veamos los resultados obtenidos si aplicamos el de bajo riesgo.

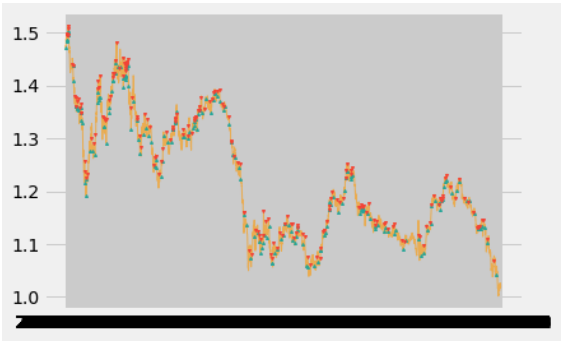
```
Profit gained from the algoritmic 4 strategy by investing \$100k in EURUSD : 157006.19
Profit percentage of the algoritmic 4 strategy : 157%
```

Triplicamos los resultados de la estrategia Benchmark. Si hubiéramos seguido las indicaciones de éste algoritmo desde finales de 2009 habríamos multiplicado nuestra cantidad inicial invertida por 2'5. Es una tasa de interés enorme que ni siquiera los mejores fondos de inversión ofrecen.

Parece ser que se han solucionado los problemas que se presentaban en los algoritmos basados en la confianza ofrecida por los modelos. En las figuras 4.22a y 4.22b se pueden observar cómo los puntos de compra o venta han aumentado con respecto a los otros algoritmos. 'Se pueden observar las imágenes ampliadas junto con el código utilizado en las figuras A.7 y A.8'.



(a) Puntos de entrada algoritmo bina3



(b) Puntos de entrada algoritmo bina4

Al ver los resultados tan buenos de estos dos algoritmos y cómo están generados los modelos que se utilizan en ellos se nos ocurre pensar que el overfitting que tenemos en el entrenamiento del modelo de Random Forest puede tener una gran influencia en las predicciones. Por ello decidimos utilizar estos algoritmos únicamente en los datos de test y compararlos con los resultados de la estrategia Benchmark en estos dos años para ver si estos porcentajes de beneficios son reales o no. 'Códigos en apéndice A.4.3'

Resultados de aplicar la estrategia Benchmark:

```
Benchmark profit by investing \$100k : 26206.87
Benchmark Profit percentage : 26%
```

Resultados de aplicar el algoritmo bina3 (alto riesgo):

```
Profit gained from the algoritmic 3 strategy by investing \$100k in EURUSD : 44181.15
Profit percentage of the algoritmic 3 strategy : 44%
```

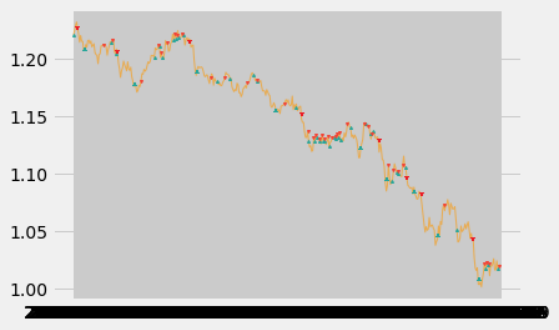
Mejoramos el rendimiento obtenido por la estrategia Benchmark en un 18 %. Si hubiéramos invertido 100.000 dólares durante los últimos dos años esto se traduciría en que hubiéramos obtendo 18.000 dólares más de beneficio.

Veamos a ver el beneficio que hubiéramos obtenido si hubiéramos seguido las indicaciones del algoritmo bina4 durante los dos últimos años.

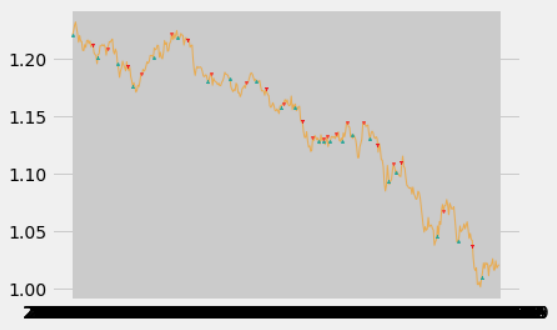
```
Profit gained from the algoritmic 4 strategy by investing \$100k in EURUSD : 64689.21
Profit percentage of the algoritmic 4 strategy : 64%
Strategy 4 profit is 38% higher than the Benchmark Profit
```

Duplicamos el resultado obtenido con la estrategia Benchmark. Esto se traduciría en conseguir 38.000 dólares más de beneficio en sólo dos años.

Se pueden observar las señales de compra y venta obtenidas por ambos algoritmos en las figuras 4.23a y 4.23b. 'Para ver estas figuras en grande junto con el código asociado ir a las figuras A.9 y A.10'.



(a) Puntos de entrada bina3 en 2021-2022



(b) Puntos de entrada bina4 en 2021-2022

Con estos resultados podemos rechazar la hipótesis antes planteada de que los algoritmos obtenían tan buenos resultados gracias al sobreajuste que se generaba en el modelo de random forest. Observemos que tanto el algoritmo bina3 como el bina4, obtienen cerca de un tercio de los beneficios que obtendrían desde 2009 en los últimos dos años, que corresponden a la partición de test y que no fue utilizada para entrenar a los modelos.

4.6. Últimos comentarios

Además del análisis que hemos realizado sobre la variable Updownpred, donde intentábamos predecir si el tipo de cambio entre el Euro y el dólar iba a aumentar o disminuir al día siguiente, hemos hecho un análisis parecido sobre la variable Closepred. Con ese análisis hemos intentado predecir lo mejor posible el precio exacto del día siguiente.

Como finalmente los resultados obtenidos de los algoritmos que hemos generado a partir de los algoritmos que hemos diseñado en ese análisis y por motivos de espacio no lo hemos podido recoger en este trabajo. Sin embargo, si se tiene interés en verlo se puede encontrar el Jupiter Notebook en la siguiente dirección: https://github.com/sermor04/TFG.git

En el Jupiter se encuentra todo el código explicado del análisis expuesto en este trabajo, el de la variable Closepred, otro que tuvimos que hacer sobre una variable que llamamos Pipspred, en la que se predice los *pips* del día siguiente (que es como predecir el precio de cierre pero evitábamos así un problema que nos surgía con la métrica variance score al intentar predecir Closepred) y los algoritmos desarrollados a partir de las predicciones obtenidas con los modelos de esta parte.

Sólo comentar que en ese análisis se comparan los modelos de regresión lineal múltiple, adaptative boosting (para caso continuo), random forest (para caso continuo) y una red neuronal. Obtiendo los mejores resultados con la red neuronal, seguida de la regresión lineal múltiple. Y los peores con el random forest que, en el dataset binario había conseguido los mejores resultados (podemos observar cómo random forest es un algoritmo que, por lo general, funciona mejor para problemas de clasificación).

También se utiliza una serie temporal GARCH(1,1) para predecir la suma de errores al cuadrado de la serie. Esto nos da una aproximación de la volatilidad que introducimos en el dataset para realizar un nuevo un análisis con los cuatro métodos mencionados antes y comparar los resultados. El código completo de SAS (el lenguaje de programación que hemos utilizado para la serie temporal) también se encuentra en el enlace https://github.com/sermor04/TFG.git.

De éste segundo análisis obtenemos unos resultados ligeramente mejores que del anterior. Con mejores de alrededor de un 1 % en las predicciones. Pero dónde es realmente interesante es a la hora de generar el algoritmo de inversión. El mejor algoritmo desarrollado con los modelos sin el GARCH(1,1) ofrece un resultado de un 115 % de beneficio. Mientras que en el mejor algoritmo desarrollado con el GARCH(1,1) se obtiene un 127 % de beneficio. Eso se traduciría, si hubiéramos invertido con una cantidad inicial de 100.000$ en 12.000$ más de beneficio.

Capítulo 5

Conclusiones

Para finalizar este estudio es conveniente repasar los resultados obtenidos y comprobar si estos responden a la pregunta inicial, así como proponer alternativas y mejoras.

Por un lado tenemos que los modelos por sí solos no realizan predicciones lo suficientemente buenas como para invertir únicamente con uno de ellos. Con esto podemos rechazar la hipótesis inicial y hacer una primera conclusión diciendo que predecir de manera fiable las fluctuaciones de las monedas mundiales con los métodos de aprendizaje automático desarrollados hasta ahora es imposible. Quén sabe si en el futuro (tal vez con ordenadores cuánticos y con millones de datos) se puede predecir. Pero de momento, hay tantas variables y factores impredecibles que pueden afectar a estos cambios, que imposibilitan la tarea. También hemos demostrado que el análisis estadístico previo al desarrollo de los algoritmos es crucial. De él se pueden obtener ya ciertas indicaciones sobre qué podemos esperar de los modelos y hemos visto que un buen tratamiento de los datos puede optimizar el rendimiento de éstos.

Por otro lado, hemos conseguido desarrollar algoritmos que se asemejan a las estrategias de inversión diseñadas por los inversores. De manera que, en vez de predecir exactamente lo que va a pasar en el mercado, el algoritmo encuentra puntos importantes de compra o de venta en los que hay una gran probabilidad de acierto. Consiguiendo así resultados muy buenos y grandes cantidades de beneficios. Resumiendo, a pesar de que los modelos por sí solos no consiguen buenas predicciones, combinando sus resultados bajo un mismo algoritmo podemos prever ciertos movimientos y aprovecharnos de ellos.

Finalmente nos preguntamos, ¿cuáles podrían ser algunas mejoras que hiciéramos en nuestro trabajo y qué líneas de investigación nos abren estos resultados? Por un lado, sería interesante investigar más modelos y analizar los resultados obtenidos. Con estos resultados, se podrían desarrollar nuevos algoritmos con diferentes condiciones de entrada y evaluar los rendimientos. ¿Qué combinación de modelos funciona mejor? ¿Qué condiciones de entrada dan mejores resultados? Y, respondiendo estas preguntas, obtener un algoritmo que ofrezca mejores resultados que el obtenido en este trabajo. Por otro lado, podría ser de interés intentar optimizar el algoritmo aquí presentado. Se nos ocurre, por ejemplo, generar un procedimiento para crear las órdenes stop-loss y take-profit, disminuyendo así el riesgo de cada inversión y maximizando el beneficio. En el libro de Isabel Nogales Navarro, Curso Avanzado Forex Profesional[22] se proponen ciertas ideas que podrían ayudar en esta tarea.

Para terminar, quedaría ver si incluyendo nuevas variables o utilizando herramientas para introducir el análisis fundamental en el trabajo, podemos crear modelos con mayor índice de acierto en las predicciones y, a su vez, algoritmos que encuentren mejor los puntos de entrada idóneos para invertir.

Bibliografía

[1] Nikhil Adithyan. Algorithmic trading with average directional index in python, 2021.

[2] Gerald Appel. *Technical Analysis: Power Tools for Active Investors.* Reprint, FT Press, 2005.

[3] Andrés Sevilla Arias. Valor en riesgo (var), 2007.

[4] Avatrader. Pips y el cálculo de beneficios y pérdidas, 2008.

[5] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] Leo Breiman and Adele Cutler. *Random decision forests*. Springer, 1995.

[7] Raul Canessa C. Análisis de sentimiento en el mercado forex, 2020.

[8] D. R. Cox. *The Regression Analysis of Binary Sequences*, volume 20, No. 2. Wiley, 1958.

[9] Marc Garrigasait. 16-sept-1992, el día que se rompió el sistema monetario europeo. devaluaciones y "miércoles negro", 2012.

[10] Heghine Grigoryan. Bandas de bollinger - bollinger bands indicator, 2022.

[11] Tibshirani Hastie and Friedman. *Elements of Statistical Learning*. Springer, 2009.

[12] Albert J. Jovell. *Análisis de regresión logística*, volume 15. Centro de Investigaciones Sociológicas, 1995.

[13] Ovidijus Jurevicius. Benchmarking, 2022.

[14] Paul Krugman. Análisis fundamental forex : Definición, 2016.

[15] Raphaël Lederman. Adaptative boosting (adaboost) supervised learning algorithms, 2019.

[16] Dora Linares. Historia del euro. pages 2–8, 2005.

[17] John J. Murphy. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance, 1999.

[18] John J. Murphy. *Análisis técnico de los mercados financieros*. Gestión 2000, 2000.

[19] John J. Murphy. Cómo funcionan las medias móviles: media móvil simple, exponencial y ponderada, 2020.

[20] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. The MIT Press, 2022.

[21] Alejandro Muttach. *LA ALQUIMIA DEL ANALISIS TECNICO: Entiende ¿dónde? ¿por qué? y ¿cuándo? hay una oportunidad en el mercado financiero*. Muttach Capital LLC, 2022.

[22] Isabel Nogales Naharro. *Curso Avanzado Forex Profesional*, volume 2. Independently published, 2017.

[23] Martin Pring. How to trend trade with guppy multiple moving average (gmma), 2018.

[24] Frank Rosenblatt. Estructura y funcionamiento del perceptrón. 1958.

[25] Kian Katanforoosh Tengyu Ma, Anand Avati and Andrew Ng. Apuntes "deep learning"del curso ml de stanford. pages 1–12, 2021.

[26] Zack West. Moving average convergence divergence (macd). pages 1–9, 2022.

[27] Freund Y. and Schapire R. *A decision-theoretic generalization of on-line learning and an application to boosting*. Springer, 1995.

Apéndice A

Código en python

A.1. Análisis estadístico

A.1.1. Datos faltantes

```
df = df.dropna()
df.isna().sum()
```

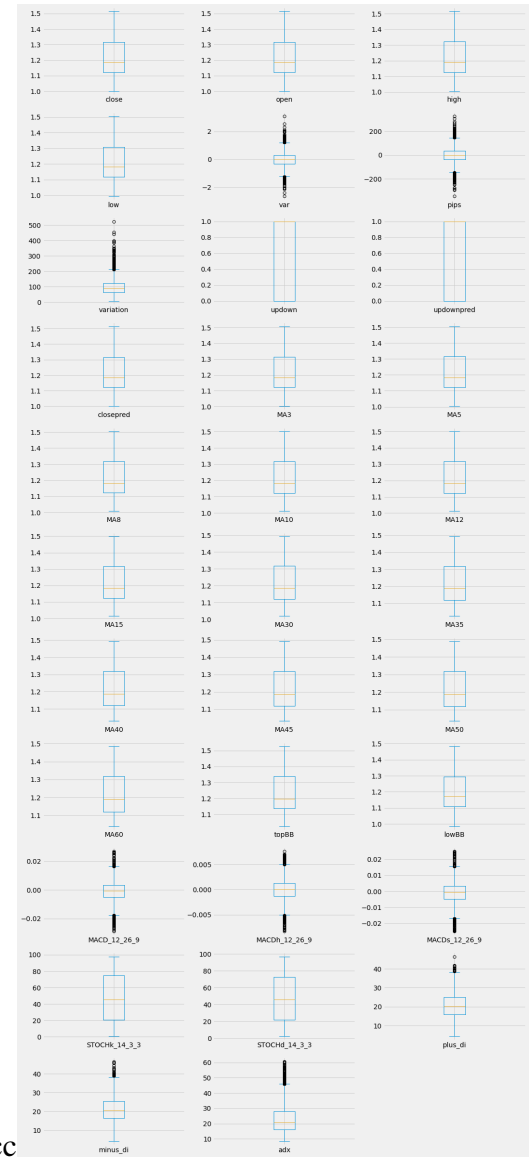
```
close          0
open           0
high           0
low            0
var            0
pips           0
variation      0
updown         0
updownpred     0
closepred      0
MA3            0
MA5            0
MA8            0
MA10           0
MA12           0
MA15           0
MA30           0
```

```
MA35            0
MA40            0
MA45            0
MA50            0
MA60            0
topBB           0
lowBB           0
MACD_12_26_9    0
MACDh_12_26_9   0
MACDs_12_26_9   0
STOCHk_14_3_3   0
STOCHd_14_3_3   0
plus_di         0
minus_di        0
adx             0
dtype: int64
```

date	2009-11-02	2009-11-03	2009-11-04	2009-11-05	2009-11-06
close	1.476900	1.471500	1.487800	1.488100	1.484900e+00
open	1.471300	1.477200	1.473100	1.487100	1.486900e+00
high	1.484600	1.481100	1.490900	1.491800	1.491500e+00
low	1.468400	1.462400	1.470100	1.480800	1.481300e+00
var	0.350000	-0.370000	1.110000	0.020000	-2.200000e-01
pips	56.000000	-57.000000	147.000000	10.000000	-2.000000e+01
variation	162.000000	187.000000	208.000000	110.000000	1.020000e+02
updown	1.000000	0.000000	1.000000	1.000000	0.000000e+00
updownpred	0.000000	1.000000	1.000000	0.000000	1.000000e+00
closepred	1.471500	1.487800	1.488100	1.484900	1.499400e+00
MA8	1.484225	1.480375	1.478813	1.478950	1.479450e+00
MA40	1.474422	1.474997	1.475790	1.476545	1.477243e+00
topBB	1.508865	1.509729	1.509239	1.508458	1.507997e+00
lowBB	1.468295	1.465511	1.465361	1.465289	1.464989e+00
MACD_12_26_9	0.004302	0.003103	0.003429	0.003670	3.560874e-03
MACDh_12_26_9	-0.003619	-0.003854	-0.002822	-0.002065	-1.739315e-03
MACDs_12_26_9	0.007920	0.006957	0.006251	0.005735	5.300189e-03
STOCHk_14_3_3	24.267648	17.670798	33.807448	45.709947	5.588459e+01
STOCHd_14_3_3	24.099886	20.591080	25.248632	32.396064	4.513399e+01
plus_di	12.635846	11.277421	14.699851	14.144255	1.342783e+01
minus_di	16.705081	17.864961	15.938552	14.872013	1.411873e+01
adx	24.066022	23.360553	23.207494	21.793318	2.038108e+01

A.1.2. Gráficos de cajas y bigotes

```
df.plot(figsize=(16,48),kind='box', subplots=True,layout=(13,3),sharex=
    False, sharey=False)
plt.show()
```



cc

Figura A.1: Boxplots 2009

A.1.3. Variables categóricas

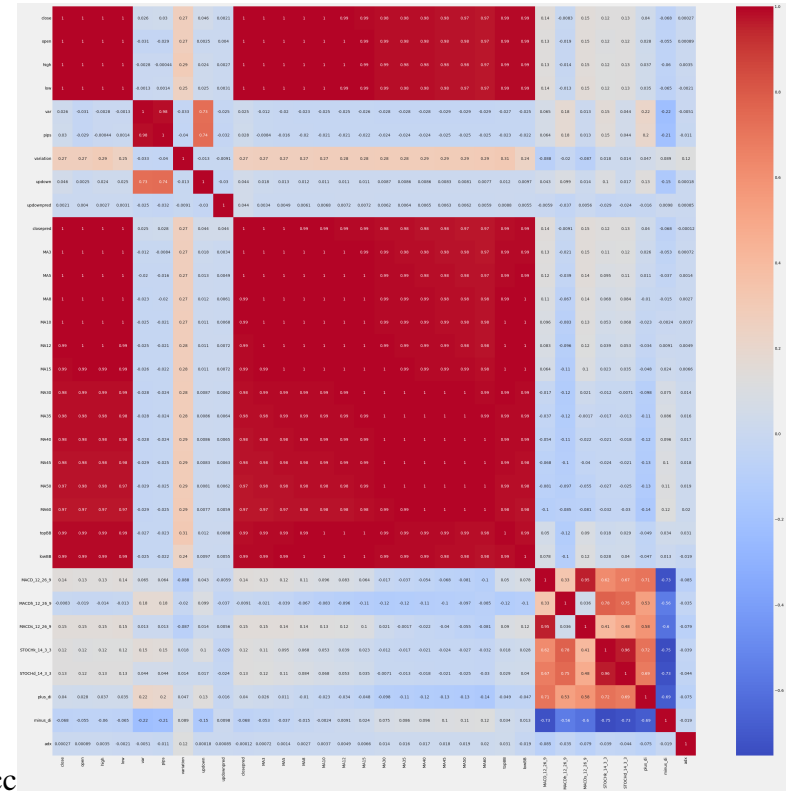
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3332 entries, 2009-11-02 to 2022-08-09
Data columns (total 32 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   close           3332 non-null   float64
 1   open            3332 non-null   float64
 2   high            3332 non-null   float64
 3   low             3332 non-null   float64
 4   var             3332 non-null   float64
 5   pips            3332 non-null   float64
 6   variation       3332 non-null   float64
 7   updown          3332 non-null   int64
 8   updownpred      3332 non-null   int64
 9   closepred       3332 non-null   float64
 10  MA3             3332 non-null   float64
 11  MA5             3332 non-null   float64
 12  MA8             3332 non-null   float64
 13  MA10            3332 non-null   float64
 14  MA12            3332 non-null   float64
 15  MA15            3332 non-null   float64
 16  MA30            3332 non-null   float64
 17  MA35            3332 non-null   float64
 18  MA40            3332 non-null   float64
 19  MA45            3332 non-null   float64
 20  MA50            3332 non-null   float64
 21  MA60            3332 non-null   float64
 22  topBB           3332 non-null   float64
 23  lowBB           3332 non-null   float64
 24  MACD_12_26_9    3332 non-null   float64
 25  MACDh_12_26_9   3332 non-null   float64
 26  MACDs_12_26_9   3332 non-null   float64
 27  STOCHk_14_3_3   3332 non-null   float64
 28  STOCHd_14_3_3   3332 non-null   float64
 29  plus_di         3332 non-null   float64
 30  minus_di        3332 non-null   float64
 31  adx             3332 non-null   float64
dtypes: float64(30), int64(2)
```

```
memory usage: 859.0+ KB
```

A.1.4. Gráficos de correlaciones

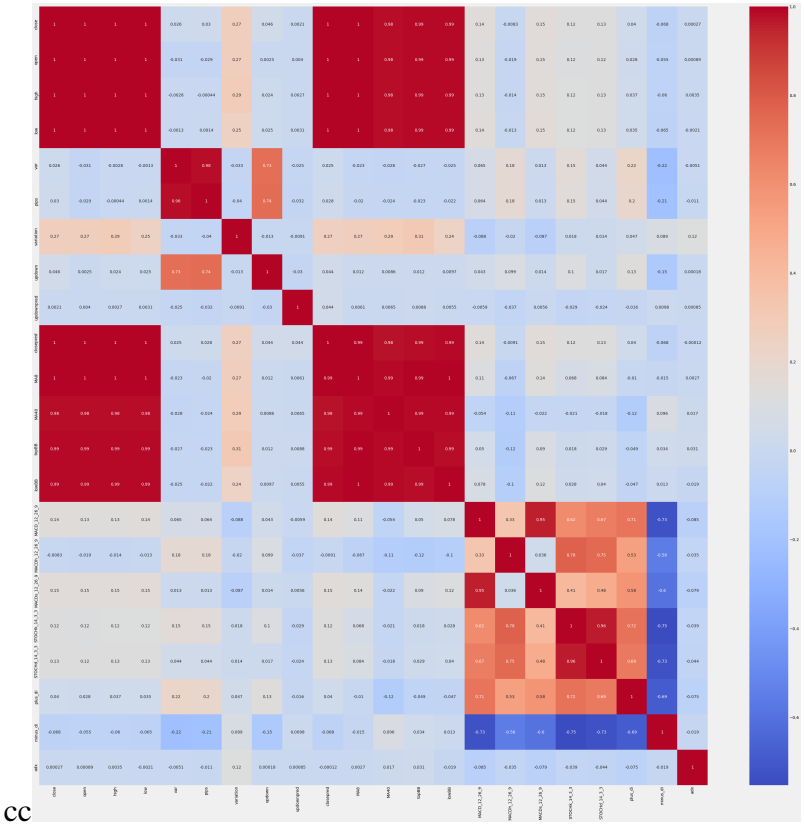
```
plt.figure(figsize=(46,46))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm')
```



cc

Figura A.2: Correlaciones 2009

```
df=df.drop(['MA3','MA5','MA10','MA12','MA15','MA30','MA35','MA45','MA50
    ','MA60'],axis=1)
plt.figure(figsize=(46,46))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm')
```



cc

Figura A.3: Correlaciones 2009 arreglada

A.1.5. Conteo y umbral de referencia

```
df_Price = df['updownpred']\
        .value_counts(normalize=True)\
        .mul(100).rename('percent').reset_index()

df_Price_conteo =df['updownpred'].value_counts().reset_index()
df_Price_pc = pd.merge(df_Price,
                                df_Price_conteo, on=['index'], how='
                                    inner')

fig = px.histogram(df_Price_pc, x="index", y=['percent'])
fig.show()
```

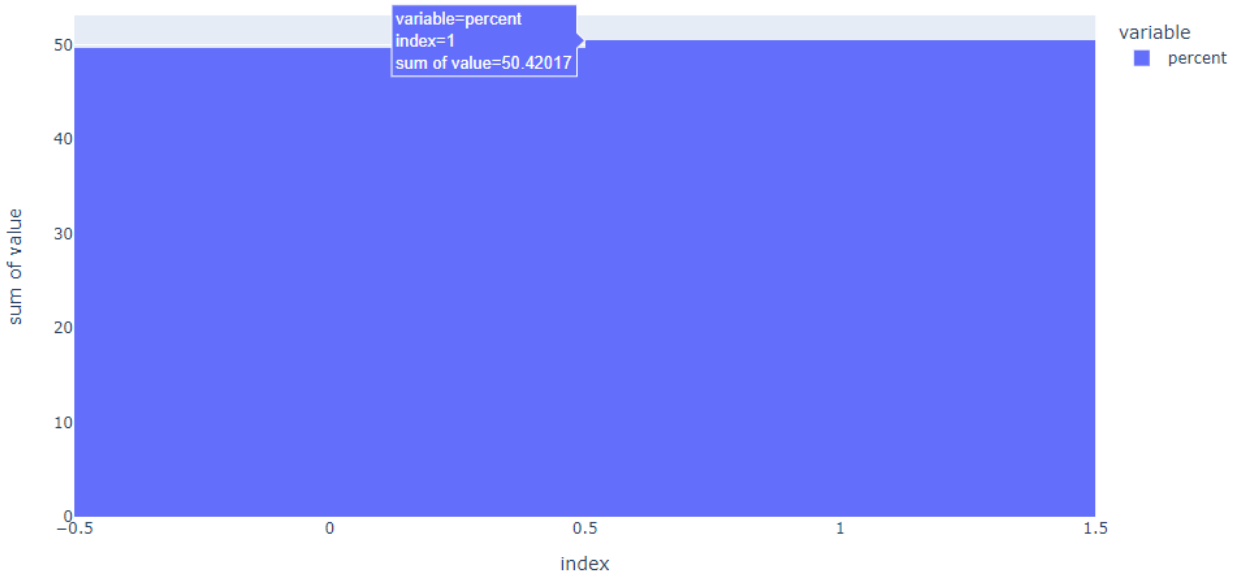


Figura A.4: Conteo de updownpred

A.2. Estrategia Benchmark

```
benchmark = pd.DataFrame(np.diff(df)).rename(columns = {0:'
    benchmark_returns'})

investment_value = 100000
number_of_stocks = math.floor(investment_value/df['close'][-1])
benchmark_investment_ret = []

for i in range(len(benchmark['benchmark_returns'])):
    returns = number_of_stocks*benchmark['benchmark_returns'][i]
    benchmark_investment_ret.append(returns)

benchmark_investment_ret_df = pd.DataFrame(benchmark_investment_ret).
    rename(columns = {0:'investment_returns'})

```

```
13
14 total_benchmark_investment_ret = round(sum(benchmark_investment_ret_df[
       'investment_returns']), 2)
15 benchmark_profit_percentage = math.floor((
       total_benchmark_investment_ret/investment_value)*100)
16 print(cl('Benchmark profit by investing $100k : {}'.format(
       total_benchmark_investment_ret), attrs = ['bold']))
17 print(cl('Benchmark Profit percentage : {}%'.format(
       benchmark_profit_percentage), attrs = ['bold']))
18 print(cl('Strategy 4 profit is {}% higher than the Benchmark Profit'.
       format(profit_percentage - benchmark_profit_percentage), attrs = ['
       bold']))
```

```
Benchmark profit by investing \$100k : 43296.18
Benchmark Profit percentage : 43%
Strategy 4 profit is 114% higher than the Benchmark Profit
```

A.3. Algoritmos basados en confianza

A.3.1. Confianza con bajo riesgo

```
1 def implement_strategyconf4(close, log, adap, rand, neural):
2     buy_price = []
3     sell_price = []
4     algor_signal = []
5     signal = 0
6
7     for i in range(len(close)):
8         if log[i] > 0.55 and rand[i] > 0.55 and adap[i] > 0.501 and
              neural[i] > 0.51:
```

```
            if signal != 1:
                buy_price.append(close[i])
                sell_price.append(np.nan)
                signal = 1
                algor_signal.append(signal)
            else:
                buy_price.append(np.nan)
                sell_price.append(np.nan)
                algor_signal.append(0)
        elif log[i] < 0.45 and rand[i] < 0.45  and adap[i] < 0.499 and
           neural[i] < 0.49:
            if signal != -1:
                buy_price.append(np.nan)
                sell_price.append(close[i])
                signal = -1
                algor_signal.append(signal)
            else:
                buy_price.append(np.nan)
                sell_price.append(np.nan)
                algor_signal.append(0)
        else:
            buy_price.append(np.nan)
            sell_price.append(np.nan)
            algor_signal.append(0)

    return buy_price, sell_price, algor_signal

buy_price, sell_price, algor_signal = implement_strategyconf4(
   updownpred['close'], updownpred['log_conf'], updownpred['adap_conf'
   ], updownpred['rand_conf'], updownpred['neural_conf'])
```

```
36
37 plt.figure()
38 plt.plot(updownpred.index, updownpred['close'], linewidth = 1, color =
      '#ff9800', alpha = 0.6, animated = True)
39 plt.plot(updownpred.index, buy_price, marker = '^', color = '#26a69a',
      markersize = 2, linewidth = 0, label = 'BUY SIGNAL')
40 plt.plot(updownpred.index, sell_price, marker = 'v', color = '#f44336',
       markersize = 2, linewidth = 0, label = 'SELL SIGNAL')
41 plt.show()
```

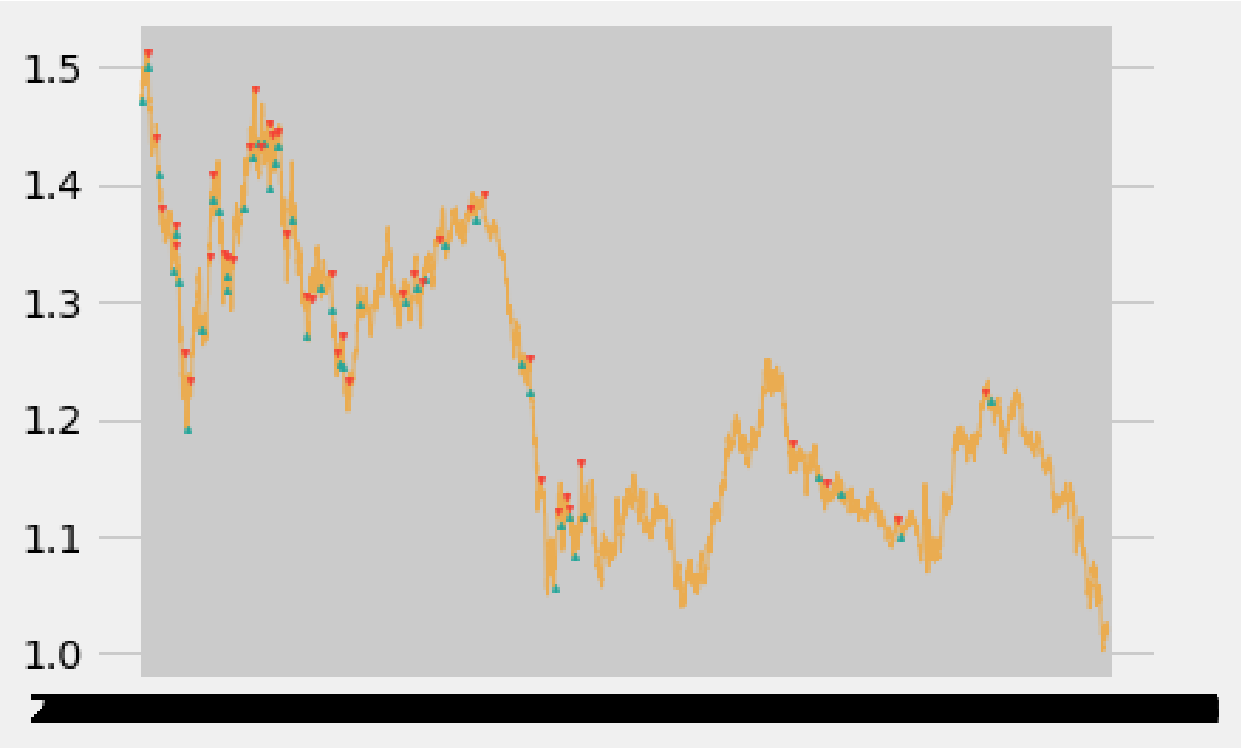


Figura A.5: Puntos de compra y venta de la estrategia conf4

```
1 position = []
2 for i in range(len(algor_signal)):
3     if algor_signal[i] > 1:
4         position.append(0)
```

```
    else:
        position.append(1)

for i in range(len(updownpred['close'])):
    if algor_signal[i] == 1:
        position[i] = 1
    elif algor_signal[i] == -1:
        position[i] = 0
    else:
        position[i] = position[i-1]

algor_signal = pd.DataFrame(algor_signal).rename(columns = {0:'
   algor_signal'}).set_index(updownpred.index)
position = pd.DataFrame(position).rename(columns = {0:'algor_position'
   }).set_index(updownpred.index)

frames = [updownpred['close'], updownpred['log_conf'], updownpred['
   adap_conf'], updownpred['rand_conf'], updownpred['neural_conf'],
   algor_signal, position]
strategy = pd.concat(frames, join = 'inner', axis = 1)

strategy
```

close	log_conf	adap_conf	rand_conf	neural_conf	algor_signal	algor_position
date						
2009-11-02	1.4769	0.519419	0.501833	0.29	0.528425	0	1
2009-11-03	1.4715	0.614016	0.504944	0.86	0.554339	1	1
2009-11-04	1.4878	0.495720	0.498727	0.77	0.511650	0	1
2009-11-05	1.4881	0.499710	0.501139	0.26	0.523186	0	1
2009-11-06	1.4849	0.528791	0.501287	0.78	0.535357	0	1
...	...	...	...	...	...	...	...
2022-08-03	1.0164	0.506075	0.498097	0.58	0.495546	0	1

2022-08-04	1.0243	0.493650	0.497151	0.65	0.476496	0	1
2022-08-05	1.0181	0.478417	0.501058	0.56	0.508200	0	1
2022-08-08	1.0195	0.499029	0.499493	0.59	0.491057	0	1
2022-08-09	1.0211	0.496667	0.500604	0.58	0.495142	0	1

```
import math
from termcolor import colored as cl
algor_ret = pd.DataFrame(np.diff(updownpred['close'])).rename(columns =
    {0:'returns'})
algor_strategy_ret = []

for i in range(len(algor_ret)):
    returns = algor_ret['returns'][i]*strategy['algor_position'][i]
    algor_strategy_ret.append(returns)

algor_strategy_ret_df = pd.DataFrame(algor_strategy_ret).rename(columns
    = {0:'algor_returns'})
investment_value = 100000
number_of_stocks = math.floor(investment_value/updownpred['close'][-1])
algor_investment_ret = []

for i in range(len(algor_strategy_ret_df['algor_returns'])):
    returns = number_of_stocks*algor_strategy_ret_df['algor_returns'][i
       ]
    algor_investment_ret.append(returns)

algor_investment_ret_df = pd.DataFrame(algor_investment_ret).rename(
   columns = {0:'investment_returns'})
total_investment_ret = round(sum(algor_investment_ret_df['
   investment_returns']), 2)
profit_percentage = math.floor((total_investment_ret/investment_value)
```

```
    *100)
print(cl('Profit gained from the algoritmic conf 4 strategy by
    investing $100k in EURUSD : {}'.format(total_investment_ret), attrs
    = ['bold']))
print(cl('Profit percentage of the algoritmic conf 4 strategy : {}%'.
    format(profit_percentage), attrs = ['bold']))
```

```
Profit gained from the algoritmic conf 4 strategy by investing \$100k in EURUSD : 31358.15
Profit percentage of the algoritmic conf 4 strategy : 31%
```

A.3.2. Confianza con alto riesgo

```
def implement_strategyconf3(close, log, adap, rand, neural):
    buy_price = []
    sell_price = []
    algor_signal = []
    signal = 0

    for i in range(len(close)):
        if (log[i] > 0.55 and rand[i] > 0.55 and adap[i] > 0.504) or (
            log[i] > 0.55 and rand[i] > 0.55 and neural[i] > 0.52) or (
            neural[i] > 0.52 and rand[i] > 0.55 and adap[i] > 0.504) or
            (log[i] > 0.55 and neural[i] > 0.52 and adap[i] > 0.504):
             if signal != 1:
                 buy_price.append(close[i])
                 sell_price.append(np.nan)
                 signal = 1
                 algor_signal.append(signal)
             else:
                 buy_price.append(np.nan)
```

```
                sell_price.append(np.nan)
                algor_signal.append(0)
        elif (log[i] < 0.45 and rand[i] < 0.45  and adap[i] < 0.498) or
            (log[i] < 0.45 and rand[i] < 0.45 and neural[i] < 0.49) or
           (neural[i] < 0.49 and rand[i] < 0.45 and adap[i] < 0.498) or
            (log[i] < 0.45 and neural[i] < 0.49 and adap[i] < 0.498):
            if signal != -1:
                buy_price.append(np.nan)
                sell_price.append(close[i])
                signal = -1
                algor_signal.append(signal)
            else:
                buy_price.append(np.nan)
                sell_price.append(np.nan)
                algor_signal.append(0)
        else:
            buy_price.append(np.nan)
            sell_price.append(np.nan)
            algor_signal.append(0)

    return buy_price, sell_price, algor_signal

buy_price, sell_price, algor_signal = implement_strategyconf3(
   updownpred['close'], updownpred['log_conf'], updownpred['adap_conf'
   ], updownpred['rand_conf'], updownpred['neural_conf'])

plt.figure()
plt.plot(updownpred.index, updownpred['close'], linewidth = 1, color =
   '#ff9800', alpha = 0.6, animated = True)
```

```
plt.plot(updownpred.index, buy_price, marker = '^', color = '#26a69a',
    markersize = 2, linewidth = 0, label = 'BUY SIGNAL')
plt.plot(updownpred.index, sell_price, marker = 'v', color = '#f44336',
     markersize = 2, linewidth = 0, label = 'SELL SIGNAL')
plt.show()
```

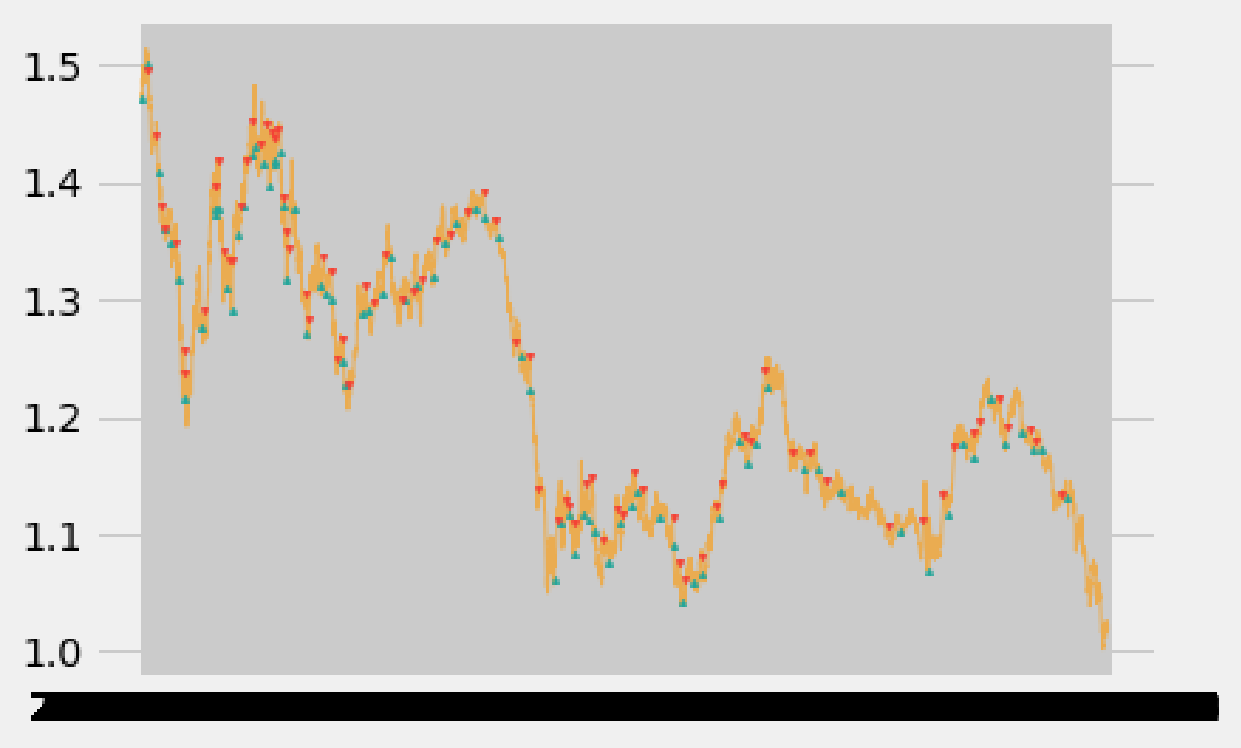


Figura A.6: Puntos de compra y venta de la estrategia conf3

```
position = []
for i in range(len(algor_signal)):
    if algor_signal[i] > 1:
        position.append(0)
    else:
        position.append(1)

for i in range(len(updownpred['close'])):
```

```
    if algor_signal[i] == 1:
        position[i] = 1
    elif algor_signal[i] == -1:
        position[i] = 0
    else:
        position[i] = position[i-1]

algor_signal = pd.DataFrame(algor_signal).rename(columns = {0:'
    algor_signal'}).set_index(updownpred.index)
position = pd.DataFrame(position).rename(columns = {0:'algor_position'
    }).set_index(updownpred.index)

frames = [updownpred['close'], updownpred['log_conf'], updownpred['
    adap_conf'], updownpred['rand_conf'], updownpred['neural_conf'],
    algor_signal, position]
strategy = pd.concat(frames, join = 'inner', axis = 1)

strategy
```

close	log_conf	adap_conf	rand_conf	neural_conf	algor_signal	algor_position
date						
2009-11-02	1.4769	0.519419	0.501833	0.29	0.528425	0	1
2009-11-03	1.4715	0.614016	0.504944	0.86	0.554339	1	1
2009-11-04	1.4878	0.495720	0.498727	0.77	0.511650	0	1
2009-11-05	1.4881	0.499710	0.501139	0.26	0.523186	0	1
2009-11-06	1.4849	0.528791	0.501287	0.78	0.535357	0	1
...	...	...	...	...	...	...	...
2022-08-03	1.0164	0.506075	0.498097	0.58	0.495546	0	1
2022-08-04	1.0243	0.493650	0.497151	0.65	0.476496	0	1
2022-08-05	1.0181	0.478417	0.501058	0.56	0.508200	0	1
2022-08-08	1.0195	0.499029	0.499493	0.59	0.491057	0	1
2022-08-09	1.0211	0.496667	0.500604	0.58	0.495142	0	1

```
import math
from termcolor import colored as cl
algor_ret = pd.DataFrame(np.diff(updownpred['close'])).rename(columns =
    {0:'returns'})
algor_strategy_ret = []

for i in range(len(algor_ret)):
    returns = algor_ret['returns'][i]*strategy['algor_position'][i]
    algor_strategy_ret.append(returns)

algor_strategy_ret_df = pd.DataFrame(algor_strategy_ret).rename(columns
    = {0:'algor_returns'})
investment_value = 100000
number_of_stocks = math.floor(investment_value/updownpred['close'][-1])
algor_investment_ret = []

for i in range(len(algor_strategy_ret_df['algor_returns'])):
    returns = number_of_stocks*algor_strategy_ret_df['algor_returns'][i
       ]
    algor_investment_ret.append(returns)

algor_investment_ret_df = pd.DataFrame(algor_investment_ret).rename(
   columns = {0:'investment_returns'})
total_investment_ret = round(sum(algor_investment_ret_df['
   investment_returns']), 2)
profit_percentage = math.floor((total_investment_ret/investment_value)
   *100)
print(cl('Profit gained from the algoritmic conf 3 strategy by
   investing $100k in EURUSD : {}'.format(total_investment_ret), attrs
   = ['bold']))
```

```
print(cl('Profit percentage of the algoritmic conf 3 strategy : {}%'.
    format(profit_percentage), attrs = ['bold']))
```

```
Profit gained from the algoritmic conf 3 strategy by investing \$100k in EURUSD : 48408.28
Profit percentage of the algoritmic conf 3 strategy : 48%
```

A.4. Algoritmos de inversión basados en los resultados

A.4.1. Binaria con bajo riesgo

```
def implement_strategy3(close, log, adap, rand, neural):
    buy_price = []
    sell_price = []
    algor_signal = []
    signal = 0

    for i in range(len(close)):
        if (log[i] == 1 and rand[i] == 1 and adap[i] == 1) or (log[i]
           == 1 and rand[i] == 1 and neural[i] == 1) or (neural[i] == 1
            and rand[i] == 1 and adap[i] == 1) or (log[i] == 1 and
           neural[i] == 1 and adap[i] == 1):
            if signal != 1:
                buy_price.append(close[i])
                sell_price.append(np.nan)
                signal = 1
                algor_signal.append(signal)
            else:
                buy_price.append(np.nan)
                sell_price.append(np.nan)
                algor_signal.append(0)
```

```
        elif (log[i] == 0 and rand[i] == 0 and adap[i] == 0) or (log[i]
            == 0 and rand[i] == 0 and neural[i] == 0) or (neural[i] ==
           0 and rand[i] == 0 and adap[i] == 0) or (log[i] == 0 and
           neural[i] == 0 and adap[i] == 0):
            if signal != -1:
                buy_price.append(np.nan)
                sell_price.append(close[i])
                signal = -1
                algor_signal.append(signal)
            else:
                buy_price.append(np.nan)
                sell_price.append(np.nan)
                algor_signal.append(0)
        else:
            buy_price.append(np.nan)
            sell_price.append(np.nan)
            algor_signal.append(0)

    return buy_price, sell_price, algor_signal

buy_price, sell_price, algor_signal = implement_strategy3(updownpred['
   close'], updownpred['log'], updownpred['adap'], updownpred['rand'],
   updownpred['neural'])

plt.figure()
plt.plot(updownpred.index, updownpred['close'], linewidth = 1, color =
   '#ff9800', alpha = 0.6, animated = True)
plt.plot(updownpred.index, buy_price, marker = '^', color = '#26a69a',
   markersize = 2, linewidth = 0, label = 'BUY SIGNAL')
```

```
plt.plot(updownpred.index, sell_price, marker = 'v', color = '#f44336',
    markersize = 2, linewidth = 0, label = 'SELL SIGNAL')
plt.show()
```

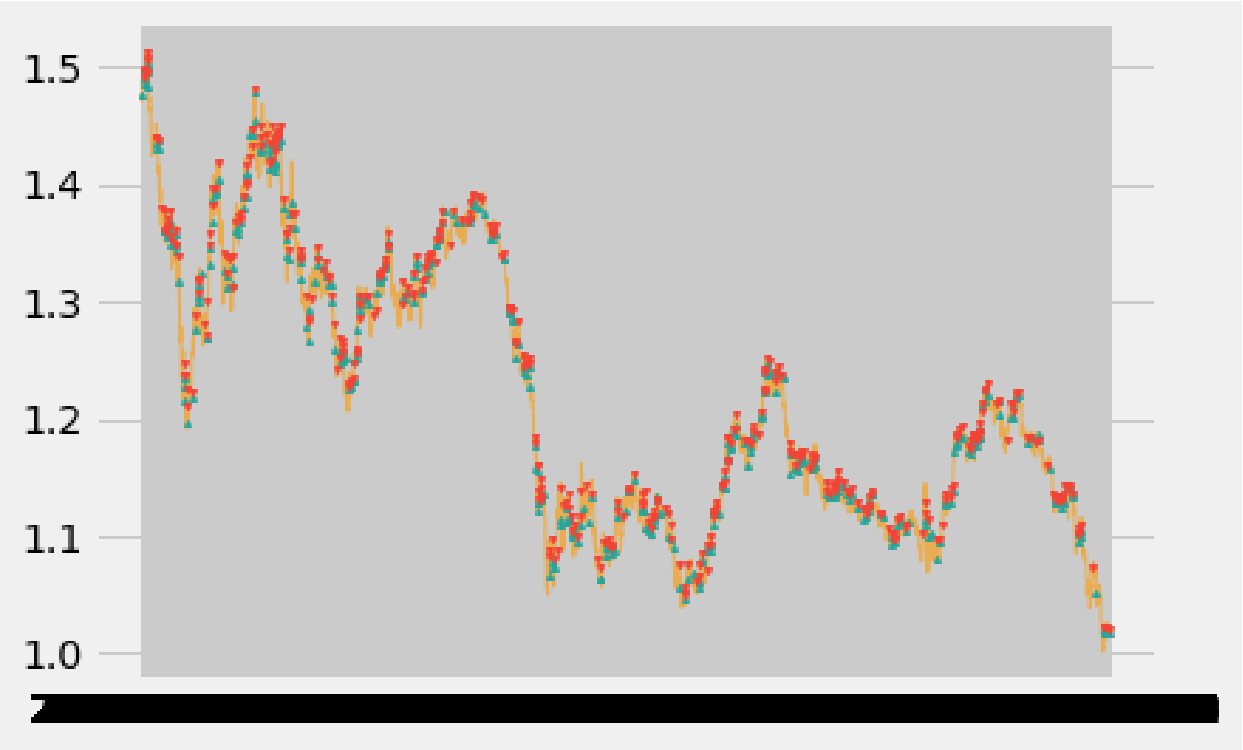


Figura A.7: Puntos de compra y venta de la estrategia bina3

```
position = []
for i in range(len(algor_signal)):
    if algor_signal[i] > 1:
        position.append(0)
    else:
        position.append(1)

for i in range(len(updownpred['close'])):
    if algor_signal[i] == 1:
        position[i] = 1
```

```
    elif algor_signal[i] == -1:
        position[i] = 0
    else:
        position[i] = position[i-1]

algor_signal = pd.DataFrame(algor_signal).rename(columns = {0:'
   algor_signal'}).set_index(updownpred.index)
position = pd.DataFrame(position).rename(columns = {0:'algor_position'
   }).set_index(updownpred.index)

frames = [updownpred['close'], updownpred['log'], updownpred['adap'],
   updownpred['rand'], updownpred['neural'], algor_signal, position]
strategy = pd.concat(frames, join = 'inner', axis = 1)

strategy
```

	close	log	adap	rand	neural	algor_signal	algor_position
date							
2009-11-02	1.4769	1	1	0	1	1	1
2009-11-03	1.4715	1	1	1	1	0	1
2009-11-04	1.4878	0	0	1	1	0	1
2009-11-05	1.4881	0	1	0	1	0	1
2009-11-06	1.4849	1	1	1	1	0	1
...	...	...	...	...	...	...	...
2022-08-03	1.0164	1	0	1	0	0	0
2022-08-04	1.0243	0	0	1	0	0	0
2022-08-05	1.0181	0	1	1	1	1	1
2022-08-08	1.0195	0	0	1	0	-1	0
2022-08-09	1.0211	0	1	1	0	0	0

```
import math
from termcolor import colored as cl
algor_ret = pd.DataFrame(np.diff(updownpred['close'])).rename(columns =
    {0:'returns'})
```

```
algor_strategy_ret = []

for i in range(len(algor_ret)):
    returns = algor_ret['returns'][i]*strategy['algor_position'][i]
    algor_strategy_ret.append(returns)

algor_strategy_ret_df = pd.DataFrame(algor_strategy_ret).rename(columns
    = {0:'algor_returns'})
investment_value = 100000
number_of_stocks = math.floor(investment_value/updownpred['close'][-1])
algor_investment_ret = []

for i in range(len(algor_strategy_ret_df['algor_returns'])):
    returns = number_of_stocks*algor_strategy_ret_df['algor_returns'][i
       ]
    algor_investment_ret.append(returns)

algor_investment_ret_df = pd.DataFrame(algor_investment_ret).rename(
   columns = {0:'investment_returns'})
total_investment_ret = round(sum(algor_investment_ret_df['
   investment_returns']), 2)
profit_percentage = math.floor((total_investment_ret/investment_value)
   *100)
print(cl('Profit gained from the algoritmic 3 strategy by investing
   $100k in EURUSD : {}'.format(total_investment_ret), attrs = ['bold'
   ]))
print(cl('Profit percentage of the algoritmic 3 strategy : {}%'.format(
   profit_percentage), attrs = ['bold']))
```

```
Profit gained from the algoritmic 3 strategy by investing \$100k in EURUSD : 127743.81
Profit percentage of the algoritmic 3 strategy : 127%
```

A.4.2. Binaria con bajo riesgo

```
def implement_strategy4(close, log, adap, rand, neural):
    buy_price = []
    sell_price = []
    algor_signal = []
    signal = 0

    for i in range(len(close)):
        if log[i] == 1 and rand[i] == 1 and adap[i] == 1 and neural[i]
           == 1:
            if signal != 1:
                buy_price.append(close[i])
                sell_price.append(np.nan)
                signal = 1
                algor_signal.append(signal)
            else:
                buy_price.append(np.nan)
                sell_price.append(np.nan)
                algor_signal.append(0)
        elif log[i] == 0 and rand[i] == 0 and adap[i] == 0 and neural[i
           ] == 0:
            if signal != -1:
                buy_price.append(np.nan)
                sell_price.append(close[i])
                signal = -1
                algor_signal.append(signal)
            else:
```

```
                buy_price.append(np.nan)
                sell_price.append(np.nan)
                algor_signal.append(0)
        else:
            buy_price.append(np.nan)
            sell_price.append(np.nan)
            algor_signal.append(0)

    return buy_price, sell_price, algor_signal

buy_price, sell_price, algor_signal = implement_strategy4(updownpred['close'], updownpred['log'], updownpred['adap'], updownpred['rand'], updownpred['neural'])

plt.figure()
plt.plot(updownpred.index, updownpred['close'], linewidth = 1, color = '#ff9800', alpha = 0.6, animated = True)
plt.plot(updownpred.index, buy_price, marker = '^', color = '#26a69a', markersize = 2, linewidth = 0, label = 'BUY SIGNAL')
plt.plot(updownpred.index, sell_price, marker = 'v', color = '#f44336', markersize = 2, linewidth = 0, label = 'SELL SIGNAL')
plt.show()
```

```
position = []
for i in range(len(algor_signal)):
    if algor_signal[i] > 1:
        position.append(0)
    else:
        position.append(1)

```

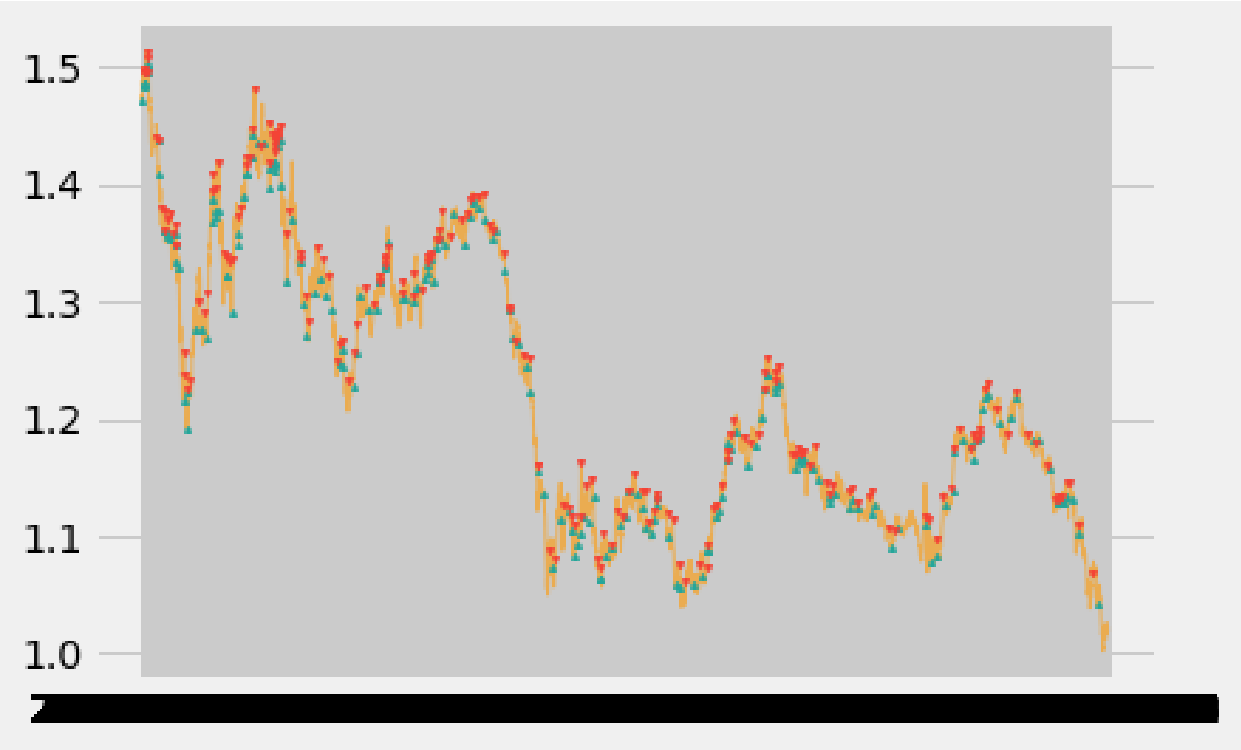


Figura A.8: Puntos de compra y venta de la estrategia bina4

```
for i in range(len(updownpred['close'])):
    if algor_signal[i] == 1:
        position[i] = 1
    elif algor_signal[i] == -1:
        position[i] = 0
    else:
        position[i] = position[i-1]
algor_signal = pd.DataFrame(algor_signal).rename(columns = {0:'
   algor_signal'}).set_index(updownpred.index)
position = pd.DataFrame(position).rename(columns = {0:'algor_position'
   }).set_index(updownpred.index)

frames = [updownpred['close'], updownpred['log'], updownpred['adap'],
   updownpred['rand'], updownpred['neural'], algor_signal, position]
```

```
strategy = pd.concat(frames, join = 'inner', axis = 1)

strategy
```

close	log	adap	rand	neural	algor_signal	algor_position	
date							
2009-11-02		1.4769	1	1	0	1	0	1
2009-11-03		1.4715	1	1	1	1	1	1
2009-11-04		1.4878	0	0	1	1	0	1
2009-11-05		1.4881	0	1	0	1	0	1
2009-11-06		1.4849	1	1	1	1	0	1
...	...	...	...	...	...	...	...	
2022-08-03		1.0164	1	0	1	0	0	1
2022-08-04		1.0243	0	0	1	0	0	1
2022-08-05		1.0181	0	1	1	1	0	1
2022-08-08		1.0195	0	0	1	0	0	1
2022-08-09		1.0211	0	1	1	0	0	1

```
import math
from termcolor import colored as cl
algor_ret = pd.DataFrame(np.diff(updownpred['close'])).rename(columns =
    {0:'returns'})
algor_strategy_ret = []

for i in range(len(algor_ret)):
    returns = algor_ret['returns'][i]*strategy['algor_position'][i]
    algor_strategy_ret.append(returns)

algor_strategy_ret_df = pd.DataFrame(algor_strategy_ret).rename(columns
    = {0:'algor_returns'})
investment_value = 100000
number_of_stocks = math.floor(investment_value/updownpred['close'][-1])
algor_investment_ret = []
```

```
14
15 for i in range(len(algor_strategy_ret_df['algor_returns'])):
16     returns = number_of_stocks*algor_strategy_ret_df['algor_returns'][i
       ]
17     algor_investment_ret.append(returns)
18
19 algor_investment_ret_df = pd.DataFrame(algor_investment_ret).rename(
   columns = {0:'investment_returns'})
20 total_investment_ret = round(sum(algor_investment_ret_df['
   investment_returns']), 2)
21 profit_percentage = math.floor((total_investment_ret/investment_value)
   *100)
22 print(cl('Profit gained from the algoritmic 4 strategy by investing
   $100k in EURUSD : {}'.format(total_investment_ret), attrs = ['bold'
   ]))
23 print(cl('Profit percentage of the algoritmic 4 strategy : {}%'.format(
   profit_percentage), attrs = ['bold']))
```

```
Profit gained from the algoritmic 4 strategy by investing \$100k in EURUSD : 157006.19
Profit percentage of the algoritmic 4 strategy : 157%
```

A.4.3. Rendimineto de los algoritmos en 2021-2022

Se define el dataset que se utilizará con sólo las observaciones de 2021 y 2022.

```
1 updownpred2 = updownpred.loc['2021-01-01:]
2 updownpred2
```

```
real    log     log_conf        adap    adap_conf       rand    rand_conf       neural
    neural_conf     close
date
```

```
2021-01-01      1       1       0.530621        1       0.502103        1       0.83    1
    0.526166        1.2212
2021-01-04      1       1       0.501390        1       0.502857        1       0.64    1
    0.511520        1.2248
2021-01-05      1       1       0.501009        1       0.500263        0       0.37    1
    0.505066        1.2294
2021-01-06      0       1       0.509385        0       0.499226        1       0.51    1
    0.500096        1.2325
2021-01-07      0       1       0.513065        1       0.501054        1       0.56    1
    0.520230        1.2270
...     ...     ...     ...     ...     ...     ...     ...     ...     ...     ...
2022-08-03      1       1       0.506075        0       0.498097        1       0.58    0
    0.495546        1.0164
2022-08-04      0       0       0.493650        0       0.497151        1       0.65    0
    0.476496        1.0243
2022-08-05      1       0       0.478417        1       0.501058        1       0.56    1
    0.508200        1.0181
2022-08-08      1       0       0.499029        0       0.499493        1       0.59    0
    0.491057        1.0195
2022-08-09      1       0       0.496667        1       0.500604        1       0.58    0
    0.495142        1.0211
```

Implementemos sobre estos años la estrategia Benchmark para ver su rendimineto.

```
df2 = df.loc['2021-01-01':]
benchmark = pd.DataFrame(np.diff(df)).rename(columns = {0:'
    benchmark_returns'})

investment_value = 100000
number_of_stocks = math.floor(investment_value/df['close'][-1])
benchmark_investment_ret = []

for i in range(len(benchmark['benchmark_returns'])):
    returns = number_of_stocks*benchmark['benchmark_returns'][i]
    benchmark_investment_ret.append(returns)

benchmark_investment_ret_df = pd.DataFrame(benchmark_investment_ret).
```

```
    rename(columns = {0:'investment_returns'})
13
14
15  total_benchmark_investment_ret = round(sum(benchmark_investment_ret_df[
        'investment_returns']), 2)
16  benchmark_profit_percentage = math.floor((
        total_benchmark_investment_ret/investment_value)*100)
17  print(cl('Benchmark profit by investing $100k : {}'.format(
        total_benchmark_investment_ret), attrs = ['bold']))
18  print(cl('Benchmark Profit percentage : {}%'.format(
        benchmark_profit_percentage), attrs = ['bold']))
```

```
Benchmark profit by investing \$100k : 26206.87
Benchmark Profit percentage : 26%
```

Ahora utilicemos los algoritmos bina3 y bina4 para encontrar las posiciones de entrada para invertir.

Comencemos con bina3:

```
1  buy_price, sell_price, algor_signal = implement_strategy3(updownpred2['
       close'], updownpred2['log'], updownpred2['adap'], updownpred2['rand'
       ], updownpred2['neural'])['adap'], updownpred['rand'], updownpred['
       neural'])
2
3  plt.figure()
4  plt.plot(updownpred2.index, updownpred2['close'], linewidth = 1, color
       = '#ff9800', alpha = 0.6, animated = True)
5  plt.plot(updownpred2.index, buy_price, marker = '^', color = '#26a69a',
        markersize = 2, linewidth = 0, label = 'BUY SIGNAL')
6  plt.plot(updownpred2.index, sell_price, marker = 'v', color = '#f44336'
       , markersize = 2, linewidth = 0, label = 'SELL SIGNAL')
```

```
plt.show()
```

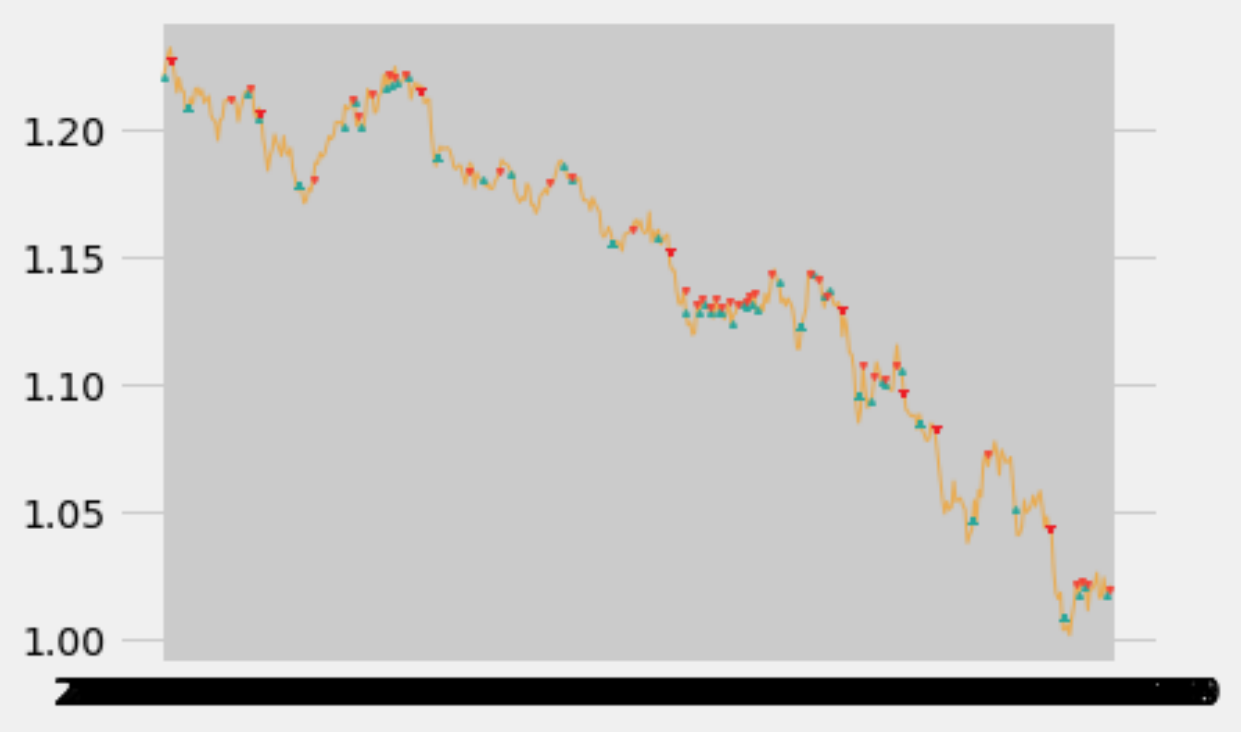


Figura A.9: Puntos de compra y venta de la estrategia bina3 años 2021-2022

```
position = []
for i in range(len(algor_signal)):
    if algor_signal[i] > 1:
        position.append(0)
    else:
        position.append(1)

for i in range(len(updownpred2['close'])):
    if algor_signal[i] == 1:
        position[i] = 1
    elif algor_signal[i] == -1:
        position[i] = 0
```

```
    else:
        position[i] = position[i-1]

algor_signal = pd.DataFrame(algor_signal).rename(columns = {0:'
   algor_signal'}).set_index(updownpred2.index)
position = pd.DataFrame(position).rename(columns = {0:'algor_position'
   }).set_index(updownpred2.index)

frames = [updownpred2['close'], updownpred2['log'], updownpred2['adap'
   ], updownpred2['rand'], updownpred2['neural'], algor_signal,
   position]
strategy = pd.concat(frames, join = 'inner', axis = 1)

strategy
```

date	close	log	adap	rand	neural	algor_signal	algor_position
2021-01-01	1.2212	1	1	1	1	1	1
2021-01-04	1.2248	1	1	1	1	0	1
2021-01-05	1.2294	1	1	0	1	0	1
2021-01-06	1.2325	1	0	1	1	0	1
2021-01-07	1.2270	1	1	1	1	0	1
...	...	...	...	...	...	...	...
2022-08-03	1.0164	1	0	1	0	0	0
2022-08-04	1.0243	0	0	1	0	0	0
2022-08-05	1.0181	0	1	1	1	1	1
2022-08-08	1.0195	0	0	1	0	-1	0
2022-08-09	1.0211	0	1	1	0	0	0

```
import math
from termcolor import colored as cl
algor_ret = pd.DataFrame(np.diff(updownpred2['close'])).rename(columns
   = {0:'returns'})
```

```
algor_strategy_ret = []

for i in range(len(algor_ret)):
    returns = algor_ret['returns'][i]*strategy['algor_position'][i]
    algor_strategy_ret.append(returns)

algor_strategy_ret_df = pd.DataFrame(algor_strategy_ret).rename(columns
    = {0:'algor_returns'})
investment_value = 100000
number_of_stocks = math.floor(investment_value/updownpred2['close'
   ][-1])
algor_investment_ret = []

for i in range(len(algor_strategy_ret_df['algor_returns'])):
    returns = number_of_stocks*algor_strategy_ret_df['algor_returns'][i
       ]
    algor_investment_ret.append(returns)

algor_investment_ret_df = pd.DataFrame(algor_investment_ret).rename(
   columns = {0:'investment_returns'})
total_investment_ret = round(sum(algor_investment_ret_df['
   investment_returns']), 2)
profit_percentage = math.floor((total_investment_ret/investment_value)
   *100)
print(cl('Profit gained from the algoritmic 3 strategy by investing
   $100k in EURUSD : {}'.format(total_investment_ret), attrs = ['bold'
   ]))
print(cl('Profit percentage of the algoritmic 3 strategy : {}%'.format(
   profit_percentage), attrs = ['bold']))
```

```
Profit gained from the algoritmic 3 strategy by investing \$100k in EURUSD : 44181.15
Profit percentage of the algoritmic 3 strategy : 44%
```

Vamos ahora con bina4:

```
buy_price, sell_price, algor_signal = implement_strategy4(updownpred2['
    close'], updownpred2['log'], updownpred2['adap'], updownpred2['rand'
    ], updownpred2['neural'])

plt.figure()
plt.plot(updownpred2.index, updownpred2['close'], linewidth = 1, color
    = '#ff9800', alpha = 0.6, animated = True)
plt.plot(updownpred2.index, buy_price, marker = '^', color = '#26a69a',
     markersize = 2, linewidth = 0, label = 'BUY SIGNAL')
plt.plot(updownpred2.index, sell_price, marker = 'v', color = '#f44336'
    , markersize = 2, linewidth = 0, label = 'SELL SIGNAL')
plt.show()
```

```
position = []
for i in range(len(algor_signal)):
    if algor_signal[i] > 1:
        position.append(0)
    else:
        position.append(1)

for i in range(len(updownpred2['close'])):
    if algor_signal[i] == 1:
        position[i] = 1
    elif algor_signal[i] == -1:
        position[i] = 0
    else:
```

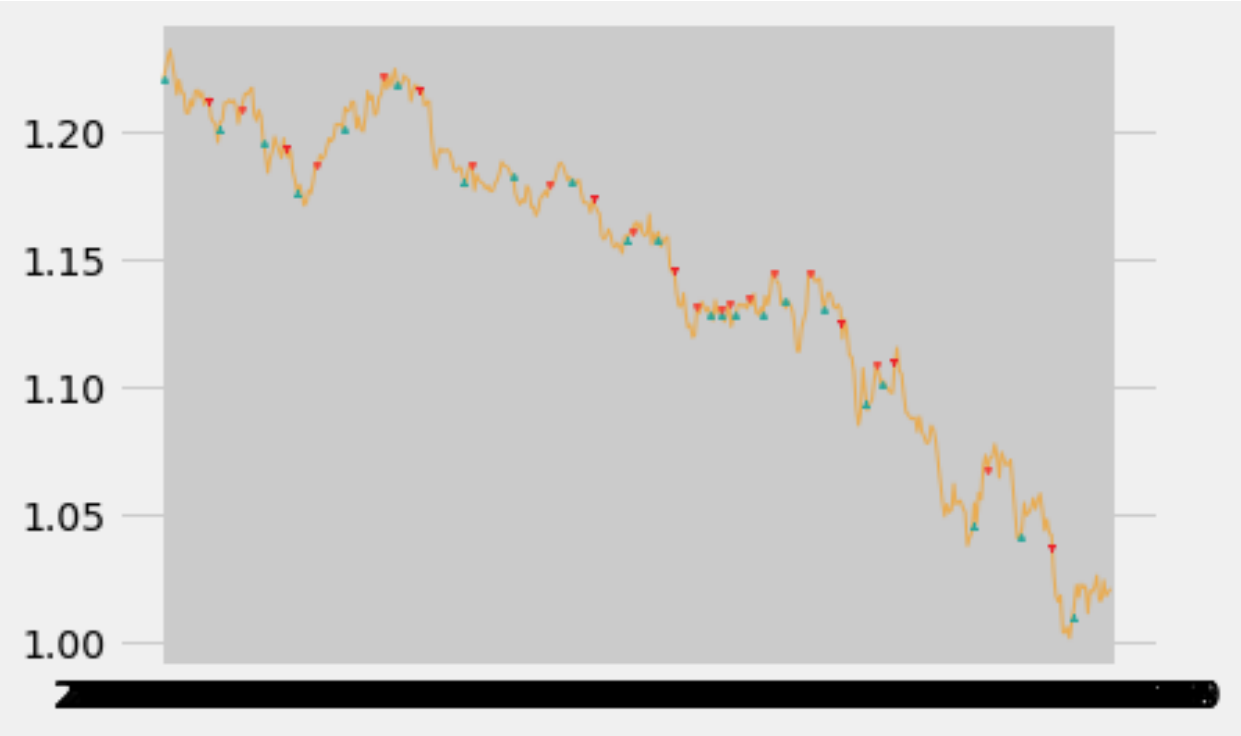


Figura A.10: Puntos de compra y venta de la estrategia bina4

```
14          position[i] = position[i-1]
15  algor_signal = pd.DataFrame(algor_signal).rename(columns = {0:'
        algor_signal'}).set_index(updownpred2.index)
16  position = pd.DataFrame(position).rename(columns = {0:'algor_position'
        }).set_index(updownpred2.index)
17
18  frames = [updownpred2['close'], updownpred2['log'], updownpred2['adap'
        ], updownpred2['rand'], updownpred2['neural'], algor_signal,
        position]
19  strategy = pd.concat(frames, join = 'inner', axis = 1)
20
21  strategy
```

	close	log	adap	rand	neural	algor_signal	algor_position

date							
2021-01-01	1.2212	1	1	1	1	1	1
2021-01-04	1.2248	1	1	1	1	0	1
2021-01-05	1.2294	1	1	0	1	0	1
2021-01-06	1.2325	1	0	1	1	0	1
2021-01-07	1.2270	1	1	1	1	0	1
...	...	...	...	...	...	...	...
2022-08-03	1.0164	1	0	1	0	0	1
2022-08-04	1.0243	0	0	1	0	0	1
2022-08-05	1.0181	0	1	1	1	0	1
2022-08-08	1.0195	0	0	1	0	0	1
2022-08-09	1.0211	0	1	1	0	0	1

```
import math
from termcolor import colored as cl
algor_ret = pd.DataFrame(np.diff(updownpred2['close'])).rename(columns
   = {0:'returns'})
algor_strategy_ret = []

for i in range(len(algor_ret)):
    returns = algor_ret['returns'][i]*strategy['algor_position'][i]
    algor_strategy_ret.append(returns)

algor_strategy_ret_df = pd.DataFrame(algor_strategy_ret).rename(columns
    = {0:'algor_returns'})
investment_value = 100000
number_of_stocks = math.floor(investment_value/updownpred2['close'
   ][-1])
algor_investment_ret = []

for i in range(len(algor_strategy_ret_df['algor_returns'])):
    returns = number_of_stocks*algor_strategy_ret_df['algor_returns'][i
       ]
    algor_investment_ret.append(returns)
```

```
18
19 algor_investment_ret_df = pd.DataFrame(algor_investment_ret).rename(
       columns = {0:'investment_returns'})
20 total_investment_ret = round(sum(algor_investment_ret_df['
       investment_returns']), 2)
21 profit_percentage = math.floor((total_investment_ret/investment_value)
       *100)
22 print(cl('Profit gained from the algoritmic 4 strategy by investing
       $100k in EURUSD : {}'.format(total_investment_ret), attrs = ['bold'
       ]))
23 print(cl('Profit percentage of the algoritmic 4 strategy : {}%'.format(
       profit_percentage), attrs = ['bold']))
24 print(cl('Strategy 4 profit is {}% higher than the Benchmark Profit'.
       format(profit_percentage - benchmark_profit_percentage), attrs = ['
       bold']))
```

```
Profit gained from the algoritmic 4 strategy by investing \$100k in EURUSD : 64689.21
Profit percentage of the algoritmic 4 strategy : 64%
Strategy 4 profit is 38% higher than the Benchmark Profit
```