uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Informática

Curso 2022-2023

Título

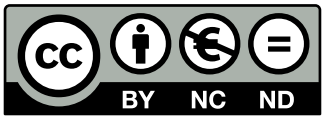
Aproximación a tendencias en Bitcoin mediante algoritmos predictivos

Carlos Mozo Nieto

Tutor

Lisardo Prieto González

Leganés, Septiembre 2023



Agradecimientos

Tras la realización de este trabajo, y por lo tanto el final de mi etapa académica, al menos por el momento. Me gustaría dedicar unas pequeñas y simples letras para aquellas personas, que sin embargo se sentirán aludidas por ellas. Gracias a familia y amigos por servirme de apoyo y de guía tanto en lo académico como en lo vital. Aunque suene cliché, y justificarlo lo sea también, de veras, gracias por acompañarme en los peores y mejores momentos. Espero que continuéis a mi lado.

Por último, mi más sincero agradecimiento a mi tutor Lisardo, que me sorprendió desde el primer día que le conocí y por el que siento completa admiración. Porque cuándo no tenía por qué, y tenía otras obligaciones más importantes, confió en mi y aceptó ser mi tutor.

Gracias a todos, Carlos Mozo.

Resumen

En los últimos años la adopción de las criptomonedas por el público general es más que una evidencia. Mientras en sus inicios tan solo un escaso número de entusiastas de la criptografía hacían uso de este sistema de intercambio de valor digital, en la actualidad, numerosas son las personas que se adentran en este mundo con la sola idea de conseguir dinero fácil.

Es por ello que esta tesis pretende servir de ayuda para todas aquellas personas que decidan adentrarse en este mundo. Esto lo pretende conseguir informando sobre qué es realmente aquello en lo que invierten su dinero y sus riesgos, mientras también se crearán diferentes modelos para tratar de reducir el riesgo de su inversión.

A lo largo del desarrollo del trabajo, se hará un estudio considerando tres algoritmos de *Machine Learning* (ML): Random Forest, XGBoost, y redes neuronales *long short-term memory* (LSTM). Además, considera la previa obtención y procesamiento de los datos, para finalizar realizando diferentes modelos de clasificación sobre la tendencia de cierre del próximo día sobre el par USD-BTC, siendo el modelo *Random Forest* el que mejor predicciones consigue, y alcanzando una tasa de acierto similar al de otros trabajos de investigación existentes, pero aún siendo subsceptible de mejora (por ejemplo considerando más fuentes y características para los datos de entrada).

Finalmente, se presenta un análisis de los resultados obtenidos así como los posibles trabajos posteriores a la realización de este proyecto. Además, también se muestra un desglose de los costes asociados al desarrollo del mismo.

Palabras clave:

Machine Learning (ML); Bitcoin (BTC); predicción de tendencias; Random Forest; XGBoost; LSTM; Redes de neuronas

Abstract

In recent years, the adoption of cryptocurrencies by the general public is more than evident. While in its beginnings, only a few cryptography enthusiasts used this digital value exchange system, today, many people enter this world to make easy money.

That is why this thesis aims to help all those who decide to enter this world. It aims to achieve this by informing about which money is invested in and its associated risks, while different models will also be created to try to reduce the risk of the investment.

Throughout the development of the work, a study will be carried out considering three *Machine Learning* (ML) algorithms: Random Forest, XGBoost, and *long short-term memory* (LSTM) neural networks. In addition, it considers the previous obtaining and processing of the data to end by carrying out different classification models on the closing trend of the next day on the USD-BTC pair, with the *Random Forest* model being the one that achieves the best predictions, reaching a success rate similar to that of other existing research works, but still susceptible to improvement (for example, considering more sources and characteristics for the input data).

Finally, an analysis of the obtained results is presented with the consequent possible lines of work after the completion of this project. In addition, a breakdown of the costs associated with its development is also shown.

Keywords:

Machine Learning (ML); Bitcoin (BTC); trends prediction; Random Forest; XGBoost; LSTM; Neural Networks

Índice general

1. Introducción **1**

1.1. Motivación 1

1.2. Objetivos 2

1.3. Marco Regulador 2

2. Estado del arte **3**

2.1. BTC 3

2.2. Machine Learning 4

2.2.1. Aprendizaje supervisado 5

2.2.2. Aprendizaje no supervisado 6

2.2.3. Hipótesis y función coste 6

2.2.4. Gradiente descendente 7

2.2.5. Overfitting 9

2.2.6. Regularización 9

2.2.7. Feature Scaling 10

2.3. Algoritmos de Machine Learning 11

2.3.1. Árboles de derivación 11

2.3.2. RandomForest 12

2.3.3. XGBoosting 13

2.3.4. Redes Neuronales 14

2.4. Procesadores del lenguaje natural 15

3. Estudio matemático de los algorítmos utilizados **17**

3.1. Random Forest 17

3.2. Gradient Boosting 18

3.2.1. Gradient Tree Boosting 20

3.3. Redes de Neuronas 20

3.4. Trabajos previos 22

4. Diseño e Implementación del sistema **23**

4.1. Elecciones de diseño 23

4.2. Obtención y Estructura de los datos 23

4.3. Preprocesamiento de Datos 26

4.3.1. Análisis de Sentimientos 27

4.4. Implementación 27

4.4.1. Funciones 29

4.5. Ajuste de los modelos 30

4.5.1. Ajuste de RandomForest 30

4.5.2. Ajuste de XGBoost 32

4.5.3. LSTM 33

5. Discusión **37**

5.1. Conclusiones 37

5.2. Trabajo Futuro 38

5.3. Originalidad y valor añadido 38

6. Desarrollo del proyecto **39**

6.1. Planificación 39

6.2. Presupuesto 39

6.3. Impacto socioeconómico 41

Acrónimos **43**

Glosario **45**

7. Anexo - Repositorio de código en GitHub **51**

Índice de figuras

2.1. Desarrollo ML 5
2.2. Clustering 6
2.3. Gradiente descendiente de una regresión lineal 8
2.4. Tasas de aprendizaje 8
2.5. Gradiente descendente sin y con feature scaling 11
2.6. Método RandomForest 12
2.7. Evolución de algoritmos basados en Árboles de derivación 13
2.8. Método XGBoost 13
2.9. Red de Neuronas Artificiales 15
2.10. Aplicaciones de NLP 16

3.1. Red Neuronal básica 21

4.1. Estructura de Datos BTC 24
4.2. Estructura revisiones Wikipedia 24
4.3. Estructura de la opinión y los comentarios 25
4.4. Data set 25
4.5. Balance de los datos 26
4.6. Matriz de Confusión RandomForest 28
4.7. Matriz de Confusión XGBoost 28
4.8. Matriz de Confusión LSTM 29
4.9. Matriz de Confusión mejor modelo RandomForest 32

4.10. Matriz de Confusión LSTM tunned 34
4.11. Matriz de Confusión mejor LSTM con umbral 34
4.12. Matriz de Confusión mejor LSTM rollback 35

Índice de tablas

4.1. Ajuste Random Forest 31
4.2. Ajuste XGBoost 33

6.1. Presupuesto recursos humanos 40
6.2. Coste de recursos utilizados 41

1

Introducción

En este capítulo nos proponemos crear el marco sobre el cual realizaremos el desarrollo de este TFG, haciendo incapié en la motivación y los objetivos del mismo.

1.1 Motivación

El auge de las criptomonedas es una realidad. En los últimos años ha pasado de ser un mundo completo desconocido, a ser patrocinado por estrellas mundiales, Cristiano Ronaldo o Messi, clubes de fútbol en todo el planeta, incluso llegando a aparecer incluso en la competición reina del automovilismo, la F1. Este boom por las criptomonedas, ha provocado que cada vez gente más inexperta se acerque a este mundo sin tener un conocimiento real sobre su propósito o sus riesgos, con la única idea en mente de conseguir dinero fácil.

Ser capaz de predecir cualquier tipo de mercado es complicado, pero si hablamos del mercado “cripto” lo es aún más. Este mercado es uno de los más volátiles del mundo [1] haciendo que si, los beneficios puedan hacerte rico en cuestión de minutos, pero también arruinarte en segundos.

1.2 Objetivos

Los objetivos de la realización de esta tesis están enfocados en la instrucción sobre BTC y su mercado, e intento de predicción del mismo con el objetivo conjunto de minimizar el riesgo al que se pueda exponer la población al adentrarse en el mismo. De forma desglosada los objetivos principales son:

- Dar a conocer la historia, funcionamiento y propuesta de BTC, para informar a aquellos que quieran participar en el proyecto. Reduciendo el desconocimiento sobre el tema en cuestión para así minimizar el riesgo de las personas que decidan adentrarse en este mundo.

- Concienciar del riesgo de la inversión en criptomonedas, así como intentar crear un modelo capaz de minimizar el riesgo de invertir en BTC, para ello utilizaremos técnicas de inteligencia artificial. Al ser un mercado extremadamente difícil de predecir utilizaremos información externa a los valores propios del mercado, para ajustar mejor nuestras predicciones.

1.3 Marco Regulador

Actualmente las criptomonedas, carecen de un marco regulatorio común [2]. Internacionalmente hay países como El Salvador en los que se acepta Bitcoin como moneda de curso legal, sin embargo en la mayoría de países su uso esta permitido pero carecen de legislación y organismos de control. [3]. En Europa, el propio Banco Central Europeo (BCE), en 2015 [4], a pesar de advertir de algunos riesgos del uso de criptomonedas, derivados de la imposibilidad de conectar personas reales con pseudónimos o de la volatilidad del mismo, desde hacía años carecía de un marco regulador común. El último año se aprobó la Ley MiCA, en inglés Markets in Crypto Assets, que entrará en vigor el 30 de diciembre de 2024, que apunta a aquellos exchange que operen dentro de la Unión Europea, haciendo obligatorio el KYC (del inglés Know Your Customer), lo cual implicará el fin del anonimato de los usuarios dentro de los exchanges.

2

Estado del arte

2.1 BTC

Los precursores de Bitcoin se remontan a la década de 1990, donde surgieron propuestas como ecash, hashcash y Bit Gold que sentaron las bases para una moneda digital descentralizada. Sin embargo, no fue hasta 2008 cuando se publicó el white paper [5] que presentó formalmente al mundo la idea detrás de Bitcoin, un innovador sistema peer-to-peer para transferir valor digital sin intermediarios. Su supuesto creador fue Satoshi Nakamoto, del cuál a día de hoy se sigue sin conocer su verdadera identidad, aunque algunos apuntan a que se trata de un seudónimo bajo el que operaron varios indivíduos.

- Lanzamiento: Bitcoin fue lanzado propiamente en 2009 [6] cuando Nakamoto minó el bloque génesis y realizó la primera transacción con Hal Finney. Inicialmente, Bitcoin tuvo un uso limitado, restringido prácticamente a entusiastas de la criptografía.
- Crecimiento inicial de Bitcoin: Para 2013, cuatro años después de su lanzamiento, el uso de Bitcoin se había expandido significativamente, atrayendo creciente interés tanto de usuarios como de comerciantes. En este periodo, surgió y cayó el primer exchange Mt.Gox (2010-2014), se acptó como pago en Silk Road una página de compra de artículos ilegales, posteriormente cerrada por el FBI, lo que ayudó a la apreciación de su valor pese a que también supuso su posterior asociación con actividades ilegales [7]. Durante esta etapa el precio tuvo grandes fluctuaciones pasando de menos de

un centavo de dólar, hasta más de 1200 para desplomarse hasta los 300 dólares después del hackeo de Mt.Gox.

- La burbuja de Bitcoin: A finales de 2017 el precio de Bitcoin llegó a máximos históricos, generando una euforia especulativa, que finalmente hizo que en 2018 el precio colapsase provocando grandes pérdidas [8]. Tras estos hechos diferentes autoridades alrededor del mundo comenzaron a promover la regulación de Bitcoin, a la vez que inversionistas institucionales mostraban creciente interés.

- Bitcoin durante la pandemia: En 2020, factores como los estímulos económicos, el auge del comercio digital y mayor adopción institucional impulsaron a Bitcoin a nuevos récords. Algunos países como El Salvador le dieron estatus legal [9], pero otros como China lo prohibieron su uso. Además aparecieron nuevas tecnologías con la base de Bitcoin y otras cryptomonedas como fueron DeFi y NFTs.

- Perspectivas de Bitcoin: El futuro de Bitcoin sigue polarizando opiniones. Mientras países planean adoptarlo, otros buscan prohibirlo. Su volatilidad también genera debate sobre su valor real. Sin embargo, para sus defensores Bitcoin es la solución para el sistema bancario y la inflación de la que la mayoría de individuos carecen de control.

2.2 Machine Learning

El análisis que se desarrollará a través de esta sección es una combinación de diferentes fuentes, incluido contenido del curso académico de las universidades politécnicas UC3M (Leganés, Madrid) y Politécnica de Varsovia.

Según Tom M. Mitchell Aprendizaje Automático o por su acrónimo ML es el proceso en el que una ”máquina aprende con respecto a una tarea T en particular, una métrica de rendimiento P y un tipo de experiencia E, si la confiabilidad del sistema mejora su rendimiento P en la tarea T, siguiendo la experiencia E”[10]. La definición de T, P y E dependerá de la tarea de aprendizaje. Es decir, ML es el proceso en el que una máquina aprende a partir de unos datos proporcionados, estén estos etiquetados o no, a través de una implementación y proporciona, información o posibles predicciones basados en dichos datos.

Es importante destacar las diferencias entre ML y la propia informática (CS, del inglés Computer Science). Mientras CS implica la programación manual y la obtención de datos se produce mediante la ejecución de un código previamente escrito. En ML el resultado varía en función de la interpretación de los datos proporcionados. El programa deberá ser capaz de ajustarse a los datos para producir

los resultados más precisos.

En el mundo animal todavía se desconoce a ciencia cierta como funcionan los procesos de aprendizaje, más aún si hablamos de humanos, como así concluye Tom M. Mitchell. Sin embargo, la sinergia humano-máquina ha sido sin lugar a dudas uno de los mayores avances para la humanidad, promoviendo la revolución que más abundantes y más rápidos progresos ha conseguido.

En la actualidad, el ML es utilizado en multitud de campos. Desde los algoritmos de recomendación más adictivos como TikTok o Netflix, a técnicas de deepfake, hasta el reciente boom de los chatbots.

Los procesos de aprendizaje de ML se dividen en aprendizaje supervisado y no supervisado.

2.2.1 Aprendizaje supervisado

El aprendizaje supervisado consiste en el entrenamiento de un algoritmo capaz de relacionar un conjunto de variables de entrada X y una variable de salida y, dependiente de la entrada. Posteriormente se utilizará este modelo para la predicción de salidas de datos de los que se desconoce su resultado. Según [11], "el aprendizaje supervisado es la metodología más importante en el aprendizaje automático y también tiene una importancia central en el procesamiento de datos multimedia". Dependiendo del resultado proporcionado hay dos tipos de aprendizaje supervisado:

- Clasificación: El algoritmo predice un valor discreto para la salida. Por ejemplo, si un activo aumentará o no su valor, 1 y 0 en cada caso.

- Regresión: El algoritmo predice un valor continuo para la salida. Por ejemplo, el precio de un activo a lo largo del tiempo.

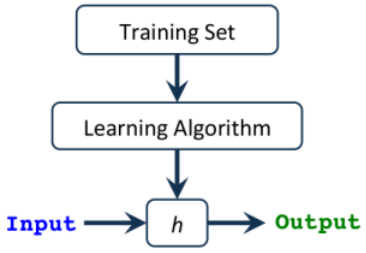


Figura 2.1: Desarrollo ML

2.2.2 Aprendizaje no supervisado

En el aprendizaje no supervisado las entradas se proporcionan sin ningún resultado objetivo. El objetivo será encontrar patrones para construir una representación de los datos de la entrada, capaz de predecir datos futuros. Su aplicación es útil en problemas en los que hay poca información y se desconoce la forma de abordarlos. Los ejemplos clásicos de aprendizaje no supervisado son[12]:

- Clustering: el algoritmo trata de lograr el agrupamiento de conjuntos de objetos no etiquetados, para lograr construir subconjuntos de datos o Clusters.

- Reducción de dimensionalidad: como su nombre indica el algoritmo se encarga de convertir un conjunto de datos de dimensiones elevadas en un conjunto de datos de dimensiones menores, asegurando que la información que proporcionada con el conjunto final es similar al inicial.

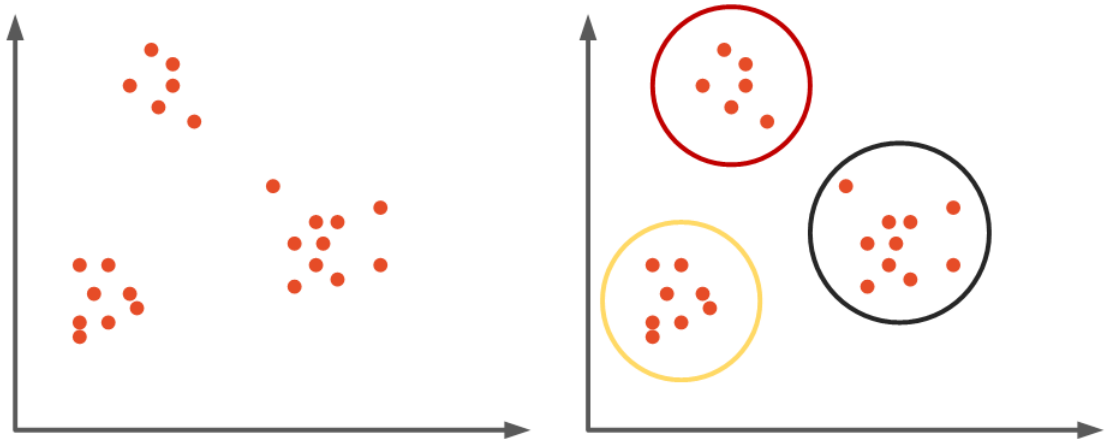


Figura 2.2: Clustering

2.2.3 Hipótesis y función coste

La notación usada a lo largo del TFG será la siguiente. EL número de datos del conjunto de entrenamiento es m, los valores de las características x, que formarán la matriz de características X, y la ecuación de salida se denotará con la letra y. La ecuación de la hipótesis será expresada con h o $h\theta$, donde θ será la variable cuyo valor buscará ser optimizado.

Para el aprendizaje supervisado los valores de la hipótesis deben corresponder a los valores de la variable de salida. La optimización supondrá minimizar el error entre estos valores.

Para medir la distancia entre hipótesis y salida hay diferentes medidas pe-

ro la más común es la función coste $J(\theta)$. Por lo tanto, como hemos explicado anteriormente, el proceso de aprendizaje consistirá en reducir esta función.

Para ayudar a su comprensión, utilizaremos como ejemplo, usando la notación nombrada, la representación más simple de hipótesis y función coste, correspondientes a las de la regresión lineal univariable:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 \tag{2.1}$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{2.2}$$

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1) \tag{2.3}$$

Usaremos x0 como sesgo, para generalizar la expresión:

$$h_\theta(x) = g(\theta^T x) \tag{2.4}$$

Dónde g() es la función de activación.

2.2.4 Gradiente descendente

El gradiente descendente es un algoritmo ampliamente usado para hallar mínimos locales o globales de funciones y es fundamental en el entrenamiento de modelos de *machine learning*. Dada una función de coste $J(\theta)$, queremos encontrar el θ que minimice J. Para esto:

1. Inicializamos θ con un valor aleatorio.
2. Calculamos la derivada (gradiente) de J respecto a θ. Esto indica la pendiente de J en ese punto.
3. Actualizamos θ en dirección opuesta al gradiente, para descender en J. Vamos en dirección negativa del gradiente.
4. Repetimos calculando el nuevo gradiente y actualizando θ, iterativamente.
5. El algoritmo converge cuando el gradiente es casi 0, indicando que se llegó a un mínimo.

Con cada actualización disminuye el coste. Así, el gradiente descendiente "explora" la curva de J para hallar el mínimo, como descendiendo por una pendiente.

Diversos autores [13] lo comparan como un descenso a ciegas colina abajo. Vamos dando pasos en dirección de mayor pendiente negativa hasta no poder descender más.

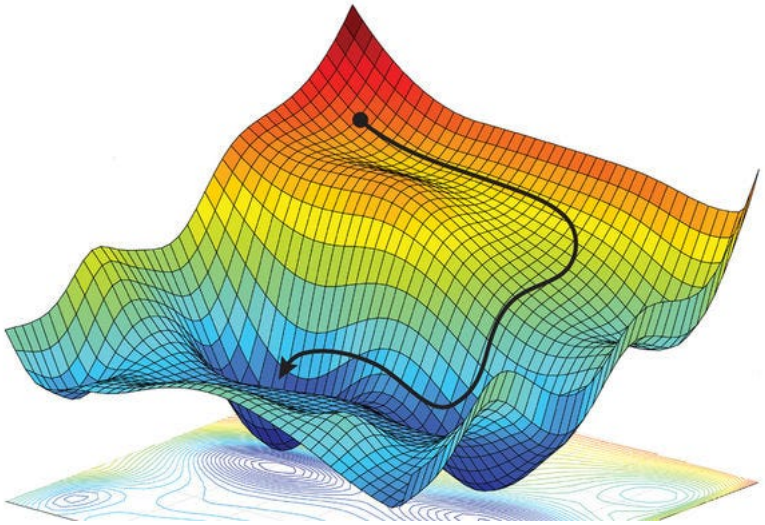


Figura 2.3: Gradiente descendiente de una regresión lineal

La fórmula general del gradiente descendiente es la siguiente:

$$\theta = \theta - \alpha \nabla J(\theta) \tag{2.5}$$

O de forma más desarrollada:

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \qquad \text{para } j = 0, 1, 2, ..., n \tag{2.6}$$

Donde α es la tasa de aprendizaje. Al igual que una mala elección de un valor θ inicial, la elección de una tasa de aprendizaje incorrecta puede ocasionar iteraciones infinitas o la no convergencia en un mínimo.

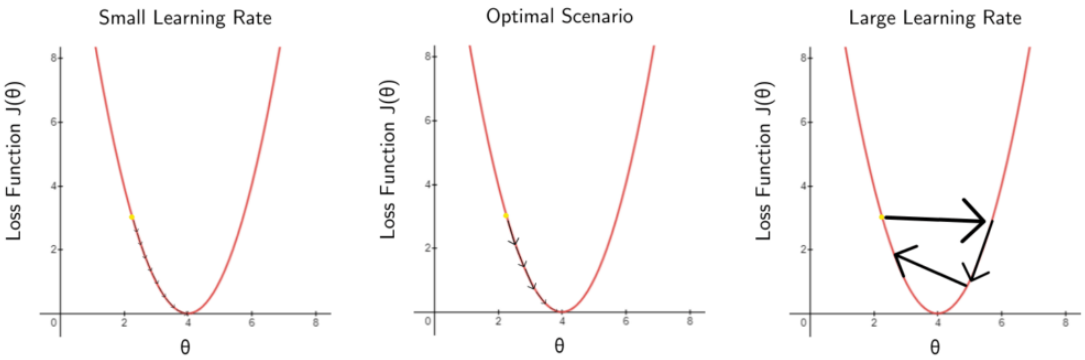


Figura 2.4: Tasas de aprendizaje

2.2.5 Overfitting

El overfitting o sobreajuste es un problema frecuente en modelos de machine learning que ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento. Al capturar tanto el patrón como el ruido pierde la capacidad de generalización [14].

Generalmente se produce cuando el modelo tiene muchos parámetros en relación al número de datos, o cuando se entrena por demasiadas iteraciones.

Entre las estrategias más comunes para evitar el overfitting se encuentran técnicas como:

- Simplificar el modelo, reduciendo el número de parámetros.
- Regularización, agregando penalizaciones en la función de costo.
- Conseguir más datos de entrenamiento.
- Interrumpir el entrenamiento temprano, antes de que ocurra el sobreajuste.
- Validación cruzada y monitoreo del error en conjunto de validación.

2.2.6 Regularización

La regularización es una técnica utilizada en machine learning para mejorar la capacidad de generalización de los modelos y evitar overfitting [15]. Consiste en agregar una penalización a la función de costo durante el entrenamiento, el parámetro de regularización, λ.

Al asignar un valor λ, debemos de tener cuidado ya que, al controlar la penalización, valores elevados podrían hacer que el modelo recayese en underfitting, es decir, dejase de hacer predicciones ajustadas.

Los conceptos clave son:

- La regularización penaliza la complejidad, forzando al modelo a ser más simple y menos propenso a sobreajustarse.
- Las técnicas más usadas son L1, L2 y dropout. L2 consiste en la distribución de los pesos.
- Ayuda a mejorar la generalización al suavizar el modelo, balanceando sesgo y varianza.
- Es útil cuando se tiene muchos parámetros respecto al número de muestras.

Al aplicar regularización L2, la función de coste será:

$$J(\theta) = \frac{1}{2m}\left[\sum i = 1^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right] \tag{2.7}$$

2.2.7 Feature Scaling

El *feature scaling* o escalado de variables es una técnica utilizada como parte de la preprocesamiento de datos en *machine learning*. Consiste en normalizar las variables numéricas de los datos dentro de un rango específico, generalmente entre 0 y 1 o con media 0 y varianza 1. Esto presenta algunas ventajas:

- Evita que variables con rangos mayores dominen a aquellas con rangos menores.
- Los algoritmos convergen más rápido cuando las variables tienen la misma escala.
- Reduce el riesgo de *ill-conditioning* en los datos.

Algunas técnicas populares de *feature scaling* son:

- Normalización: escala los datos entre un rango [a,b].

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} * (b - a) + a \tag{2.8}$$

- Estandarización: centra los datos en media 0 y escala a varianza 1.

$$x' = \frac{x - \mu}{\sigma} \tag{2.9}$$

  Donde μ y σ son la media y desviación estándar de x, respectivamente.

- Escala Min-Max: reescala entre los valores mínimo y máximo.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} * (b - a) + a \tag{2.10}$$

En resumen, el *feature scaling* o escalado de variables busca dejar las variables en el mismo rango para mejorar el desempeño y estabilidad de la mayoría de modelos de *machine learning* [16]. Cabe destacar que incluso en modelos en los que no es necesario, por ejemplo en arboles de derivación, puede mejorar la velocidad de entrenamiento.

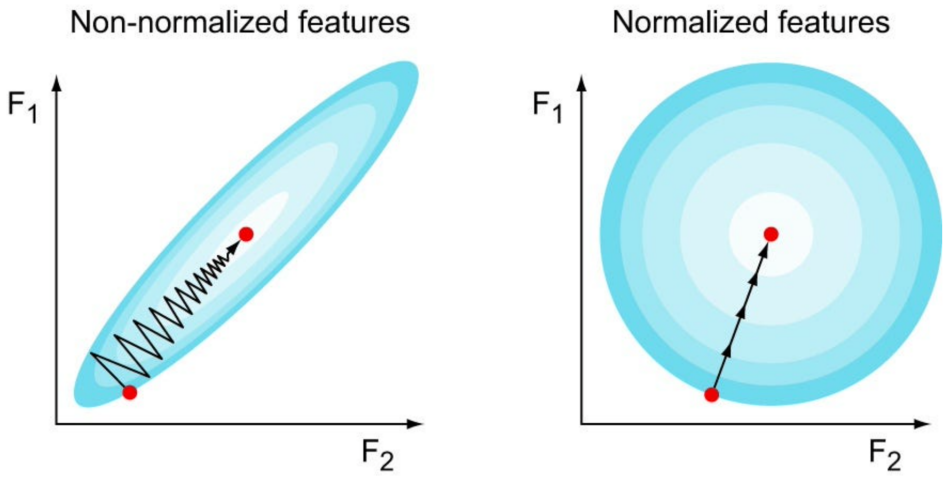


Figura 2.5: Gradiente descendente sin y con feature scaling

2.3 Algoritmos de Machine Learning

En esta sección se introducirán los algorítmos elegidos para realizar esta tesis. Estos son RandomForest, XGBoost y red neuronal recurrente LSTM. Posteriormente se explicará el motivo de la elección del algoritmo utilizado.

2.3.1 Árboles de derivación

Dado que RandomForest es un algoritmo de ensamble que consta de múltiples árboles de decisión, parece conveniente explicar primero qué es un árbol de decisión.

Los árboles de decisión (decision trees) son modelos de predicción supervisada[17] que parten de un conjunto de datos de entrenamiento X para modelar una variable objetivo y. Funcionan dividiendo recursivamente el espacio de características X en regiones distintas con el fin de agrupar observaciones similares y maximizar la homogeneidad de y dentro de cada grupo formado.

Formalmente, el algoritmo parte de un nodo raíz que contiene todas las observaciones. Evalúa cada característica x_j como posible partición, eligiendo el corte que maximice una métrica de impureza como entropía:

$$H(T) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

Donde $H(T)$ es la entropía del nodo T y p_i es la proporción de observaciones de

clase i en T.

Este proceso se repite en cada nodo hijo de forma recursiva hasta alcanzar nodos puros o cumplir un criterio de parada. Los nodos finales u hojas representan una región del espacio X asociada a un valor de y. Así, el árbol particiona el espacio predictivo según condiciones if-then aplicadas a las variables originales o transformadas.

Los árboles de decisión son versátiles, fáciles de entender e implementar. Capturan relaciones no lineales y son robustos incluso con valores faltantes o atípicos en los datos. Sin embargo, son propensos a sobreajuste, por lo que se recurre a métodos de poda o ensambles (RandomForest).

2.3.2 RandomForest

Como hemos mencionado anteriormente Random Forest es un algoritmo de ensamble que consta de múltiples árboles de decisión [18]. Funciona construyendo una multitud de árboles de decisión en el entrenamiento y luego promediando sus predicciones.

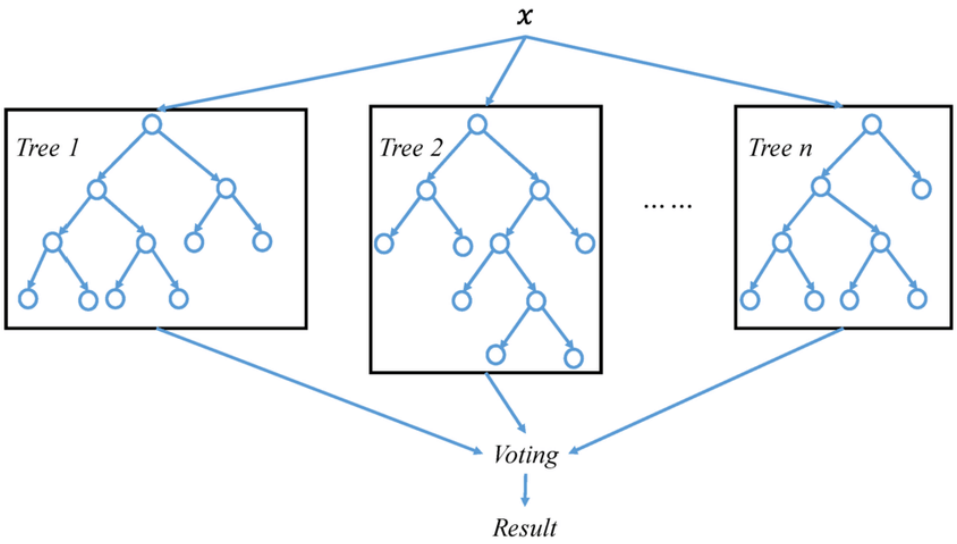


Figura 2.6: Método RandomForest

Las principales ventajas de Random Forest son reducir sobreajuste al promediar múltiples modelos y poder capturar relaciones no lineales entre variables. Es un algoritmo muy potente y versátil tanto para tareas de clasificación como regresión[18].

2.3.3 XGBoosting

XGBoost es una técnica de ensamble similar a un Random Forest, que utiliza árboles de decisión. Según Espinosa [19], el estado del arte para algoritmos basados en arboles de decisión:

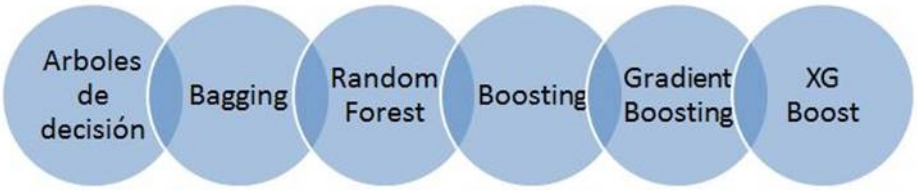


Figura 2.7: Evolución de algoritmos basados en Árboles de derivación

La diferencia radica en que mientras Random Forest genera los diferentes árboles indipendientes, XGBoost comienza con un árbol y agrega un nuevo aprendiz para minimizar el error. Es decir, se supone que cada nuevo aprendiz corrijirá los errores de sus predecesores. En esencia XGBoost, genera cada nuevo aprendiz después de que se completa el árbol anterior y se calcula el error del mismo, para que sirva de aprendizaje al nuevo árbol. Este proceso se puede apreciar en la siguiente figura.

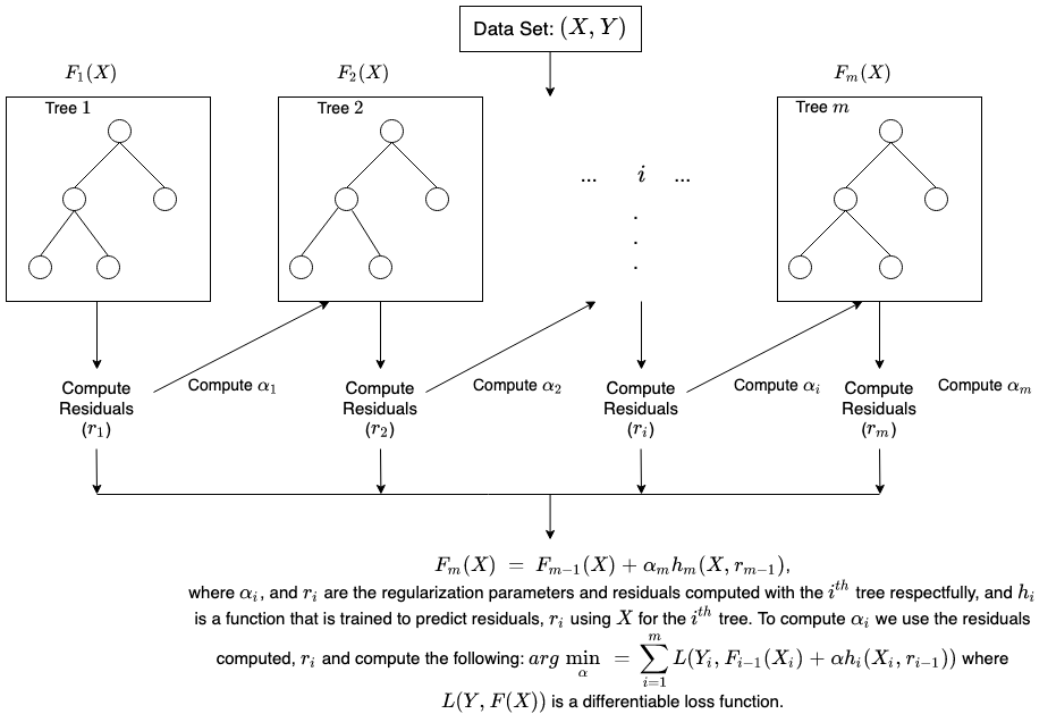


Figura 2.8: Método XGBoost

Algunas características principales de XGBoost:

- Utiliza el algoritmo de Gradient Boosting (GBM) para combinar árboles de decisión secuencialmente.
- Minimiza una función objetivo regularizada para controlar overfitting.
- Optimiza el proceso de crecimiento de los árboles, haciéndolo más rápido y escalable.
- Admite paralelización para entrenar en múltiples CPUs o núcleos de CPU.
- Incorpora regularización L1 y L2 para manejar la complejidad y evitar overfitting.
- Tiene muy buen desempeño tanto en problemas de clasificación como regresión.
- Es configurable y permite muchos ajustes para optimizar el modelo.

En resumen, XGBoost es una implementación optimizada y de alto rendimiento del método boosting usando árboles de decisión. Es ampliamente utilizado en clasificación dada su precisión predictiva por lo que se utilizará como alternativa para mejorar el modelo base.

2.3.4 Redes Neuronales

Las redes neuronales son modelos de aprendizaje profundo (*deep learning*) [20] inspirados en la estructura neuronal del cerebro humano. Están compuestas por unidades simples llamadas neuronas artificiales. Estas neuronas se organizan en capas y están interconectadas mediante “pesos” (valores numéricos). El funcionamiento de una red neuronal es el siguiente:

- Cada neurona recibe entradas (*input*), realiza un cálculo en base a sus pesos sinápticos y genera una salida (*output*).
- Las salidas se propagan a las neuronas de la siguiente capa, donde el proceso se repite.
- Mediante un algoritmo de entrenamiento, se ajustan iterativamente los pesos para lograr la salida deseada a partir de las entradas dadas.
- Las redes neuronales “aprenden” patrones en los datos modelando las relaciones entre las variables de entrada y salida.

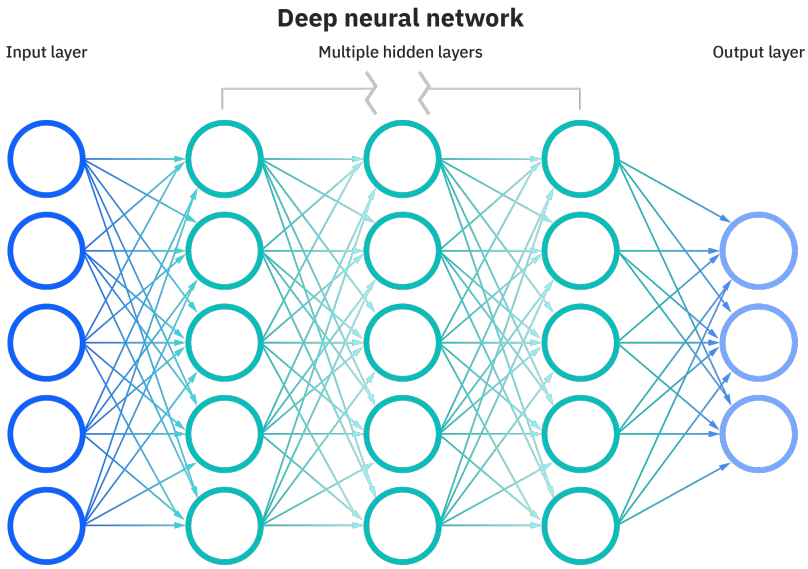


Figura 2.9: Red de Neuronas Artificiales

Las principales ventajas son su capacidad para aproximar funciones complejas, detectar patrones, realizar predicciones y extraer características de grandes conjuntos de datos. Se utilizan en numerosas tareas como clasificación de imágenes, reconocimiento de voz, traducción automática, entre otras.

2.4 Procesadores del lenguaje natural

Como explica Liddy E.D.[21] la definición de NLP o Procesadores del lenguaje natural es: “El procesamiento del lenguaje natural es una gama de técnicas computacionales teóricamente motivadas para analizar y representar textos que ocurren naturalmente en uno o más niveles de análisis lingüístico con el fin de lograr un procesamiento del lenguaje similar al humano para una variedad de tareas o aplicaciones”. Las principales aplicaciones del NLP son:

- Traducción automática entre idiomas
- Análisis de sentimientos y clasificación de textos
- Extracción y recuperación de información
- Recuperación de información
- Resumen y generación automática de textos

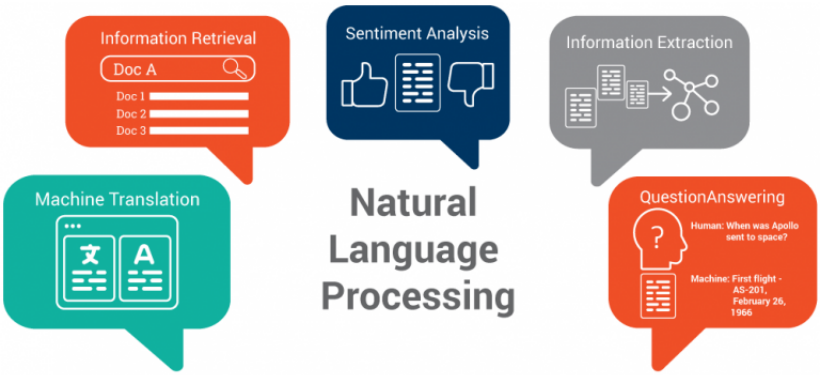


Figura 2.10: Aplicaciones de NLP

Para lograr estos objetivos, el NLP se sirve de modelos estadísticos y de deep learning entrenados con grandes conjuntos de datos. Algunas de las técnicas utilizadas son redes neuronales recurrentes, LSTM, Transformers...

Como explica Gobinda [22], los criterios para su entrenamiento se basan en técnicas como: el análisis léxico y morfológico del texto, la semántica, significados de palabras individuales o frases. El estudio y desarrollo de NLP ha experimentado una revolución en los últimos años, que ha supuesto la aparición de grandes modelos de inteligencia artificial como ChatGPT o BERT.

En concreto para la realización de la tésis es importante el subespacio de NLP que componen los NLP dedicados a sentiment analysis. El análisis de sentimiento o sentiment analysis es una rama del procesamiento de lenguaje natural que se enfoca en determinar la actitud, opinión o emoción expresada en un texto. Se busca clasificar la polaridad de un texto como positiva, negativa o neutral, utilizando diferentes técnicas de ML anteriormente mencionadas.

Es una tecnología ampliamente utilizada hoy en día. Es aplicada con fines políticos, con su uso en redes sociales para determinar la opinión del publico general sobre conflictos, elecciones... o como ayuda en análisis de mercado y de marca.

3

Estudio matemático de los algorítmos utilizados

3.1 Random Forest

Como hemos explicado anteriormente, es un algoritmo de ensamble formado por árboles de derivación, donde cada árbol depende de una colección de variables aleatorias. Formalmente, para un vector aleatorio p-dimensional $\mathbf{X} = (X_1, ..., X_p)^T$, que representa la entrada y una variable, dependiente de esta, $\mathbf{Y}$ de respuesta, suponemos una distribución conjunta $P_{XY}(X, Y)$. El objetivo del algoritmo es encontrar una función $f(X)$ capaz de predecir $\mathbf{Y}$ minimizando la pérdida.

$$E_{XY}[L(Y, f(X))] \tag{3.1}$$

Donde E_{XY} es el operador de esperanza con respecto a la distribución conjunta de X e Y. Intuitivamente, $L(Y, f(X))$ mide cuán proximas son $f(X)$ e Y; penalizando valores distantes. Las elecciones de L dependen de si se utilizará para regresión en cuyo caso la función de pérdida más tipica es la media cuadrática:

$$L(Y, f(X)) = (Y - f(X))^2 \tag{3.2}$$

Mientras que para clasificación se suele utilizar zero-one:

$$L(Y, f(X)) = I(Y \neq f(X)) = \begin{cases} 0 & \text{si } Y = f(X) \\ 1 & \text{si } Y \neq f(X) \end{cases} \tag{3.3}$$

Al minimizar la pérdida cudrática se obtiene la esperanza condicional, mejor conocida como *función de regresión*:

$$f(x) = E(Y|X = x) \tag{3.4}$$

Mientras que en el caso de la clasificación obtenemos la *regla de Bayes*:

$$f(x) = \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(Y = y|X = x) \tag{3.5}$$

El Random Forest construye f en términos de una colección de aprendices base" $h_1(x), ..., h_J(x)$, estos se combinan para dar el predictor conjunto" $f(x)$. En regresión, los aprendices base se promedian:

$$f(x) = \frac{1}{J} \sum_{j=1}^{J} h_j(x) \tag{3.6}$$

Mientras que en clasificación, $f(x)$ es la clase predicha más frecuentemente (se produce una cvotación"):

$$f(x) = \underset{y \in \mathbb{Y}}{\operatorname{argmax}} \sum_{j=1}^{J} I(y = h_j(x)) \tag{3.7}$$

En Random Forest el j-ésimo aprendiz base es un árbol denotado $h_j(\mathbf{X}, \Theta_j)$, donde Θ_j es un conjunto de variables aleatorias, independientes para $j = 1, ..., J$. La definición de Random Forest es muy general, sin embargo, "casi siempre se implementan de la manera descrita por Breiman"[23], [24]

3.2 Gradient Boosting

Al igual que Random Forest y la mayoría de problemas de *aprendizaje supervisado* hay una variable de salida y y un vector de variables de entrada x, relacionadas entre sí con alguna distribución probabilística. De nuevo, el objetivo es encontrar una función $\hat{F}(x)$ que aproxime la salida en función de los valores de las entradas, minimizando el error cuadrático medio $\frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$.

Esto se formaliza introduciendo la *función de pérdida* $L(y, F(x))$ y minimizándola en función de la esperanza:

$$\hat{F} = \underset{F}{\operatorname{argmin}}, \mathbb{E}_{x,y}[L(y, F(x))] \tag{3.8}$$

El método de gradient boosting asume que y tiene valores reales. Busca una aproximación $\hat{F}(x)$ en forma de una suma ponderada de M funciones $h_m(x)$ de la clase $\mathcal{H}$, las funciones aprendices base o débiles:

$$\hat{F}(x) = \sum_{m=1}^{M} \gamma_m h_m(x) + \text{const} \tag{3.9}$$

Para minimizar la función el XGBoost comienza con un modelo, que consta de una función constante $F_0(x)$, y lo expande incrementalmente con una técnica greedy. Esta técnica algorítmica consiste en la búsqueda de la solución óptima paso a paso. En cada paso se realiza la elección local óptima, sin considerar opciones futuras[25].

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{3.10}$$

Esto quiere decir que γ es el valor predicho para el cúal la función pérdida es mínima.

$$F_m(x) = F_{m-1}(x) + \left(\arg\min_{h_m \in \mathcal{H}} \left[\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i)) \right] \right)(x) \quad \text{Para } m \geq 1 \tag{3.11}$$

Desafortunadamente, elegir la mejor función en cada paso para una función de pérdida L cualquiera, generalmente es un problema de optimización computacionalmente inviable. Por lo que, se restringe el enfoque a una versión simplificada del problema.

La idea es aplicar steepest descendent al problema de minimización (descenso de gradiente funcional). La intención es encontrar un mínimo local de la función de pérdida iterando sobre $F_{m-1}(x)$, la dirección de descenso máximo local será el gradiente negativo.

Moviendo una pequeña cantidad γ tal que la aproximación lineal siga siendo válida, es decir, $\gamma < 0$ y $L(y_i, F_m(x_i)) \leq L(y_i, F_{m-1}(x_i))$

$$F_m(x) = F_{m-1}(x) - \gamma \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)) \tag{3.12}$$

Podemos optimizar el modelo aún más encontrando el valor mínimo de γ, tal que la función pérdida tenga un mínimo.

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_m(x_i)) \tag{3.13}$$

En el caso de tratarse de un caso concreto, podría actualizarse la ecuación como:

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)) \tag{3.14}$$

Donde γ_m sería la longitud del paso. Sin embargo, en el caso discreto, elegimos la función candidata h más cercana al gradiente de la función pérdida L para la cual el coeficiente γ, puede calcularse con la ayuda de la búsqueda lineal en las ecuaciones anteriores.

3.2.1 Gradient Tree Boosting

La selección de árboles de derivación como aprendices base es la aplicación principal de Gradient Boosting. Dada la explicación anterior, el pseudo código del algoritmo quedaría:

1. Inicializar el modelo con una constante:

$$\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta)$$

2. Para $m = 1$ de M :

   Calcular el gradiente, pendiente de la función de pérdida:

$$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}_{(m-1)}(x)}$$

   Y hessiano, curvatura de la misma:

$$\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}_{(m-1)}(x)}$$

   Ajustar un aprendiz base usando $\{x_i, -\frac{\hat{g}m(x_i)}{\hat{h}m(x_i)}\}_{i=1}^{N}$ resolviendo:

$$\hat{\phi}_m = \arg\min_{\phi \in \mathbf{\Phi}} \sum_{i=1}^{N} \frac{1}{2}\hat{h}_m(x_i)\left[\phi(x_i) - \frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right]^2$$

$$\hat{f}_m(x) = \alpha\hat{\phi}_m(x)$$

   Actualizar el modelo:

$$\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x)$$

Salida $\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x)$

Donde:

$\hat{f}_m(x)$ representa la contribución del modelo base en la iteración m al modelo final.

$\hat{\phi}_m$ representa el modelo base.

3.3 Redes de Neuronas

Las redes neuronales son algoritmos de aprendizaje muy complejos que se inspiran en la interconectividad del cerebro. Es por ello que, las unidades básicas del algoritmo reciben el nombre de neuronas, al igual que sus homólogas biológicas.

Las redes neuronales están formadas por una capa de entrada, con un número variable de capas ocultas que desembocan en una capa de salida. Algunas aclaraciones pertinentes sobre la notación utilizada durante la explicación del funcionamiento del algoritmo son:

$a_i^{(j)}$: Valor de activación de la unidad i en la capa j.

$\Theta^{(j)}$: Matriz de pesos desde la capa j a la capa $j+1$.

$z^{(j)}$: vector de activación, salida de la capa j

$$z^{(j)} = \Theta^{(j-1)} a^{(j-1)} \tag{3.15}$$

$g(z)$: Es la función sigmoide sobre el vector de activación. Es la función de activación más común en redes de neuronales:

$$g(z) = \frac{1}{1 + e^{-\Theta a^{(j-1)}}} \tag{3.16}$$

Las dimensiones de la matriz $\Theta^{(j)}$ estarán determinadas por el número de unidades en cada capa. Si una red tiene s_j unidades en la capa j y s_{j+1} unidades en la capa $j+1$, las dimensiones de la matriz serán $s_{j+1} \times (s_j + 1)$. Por convenio se suele agregar una unidad de sesgo en cada capa.

Teniendo en cuenta esto las dimensiones de las matrices de pesos $\Theta^{(j)}$ para la siguiente imagen serían:

- $\Theta^{(1)}$: matriz de pesos entre la capa de entrada y la capa oculta. Como la capa de entrada tiene 3 neuronas y la oculta 4,$\Theta^{(j)} \in \mathbb{R}^{4x4}$ (4x3 + 1 bias).
- $\Theta^{(2)}$: matriz de pesos entre la capa oculta y la capa de salida. Con misma explicación $\Theta^{(j)} \in \mathbb{R}^{1x5}$.

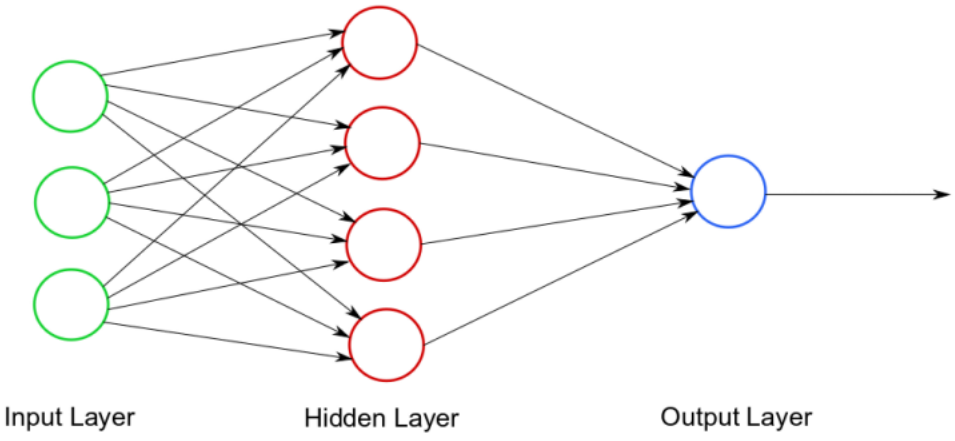


Figura 3.1: Red Neuronal básica

La función activación de las neuronas vendrá dada por:

$$a_i^{(j)} = g\left(\sum_{k=0}^{s_{j-1}} \Theta_{ik}^{(j-1)} a_k^{(j-1)}\right) \tag{3.17}$$

Por lo que quedaría:

$$\begin{aligned} a_1^{(2)} &= g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3) \\ a_2^{(2)} &= g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3) \\ a_3^{(2)} &= g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3) \\ a_4^{(2)} &= g(\Theta_{40}^{(1)} x_0 + \Theta_{41}^{(1)} x_1 + \Theta_{42}^{(1)} x_2 + \Theta_{43}^{(1)} x_3) \\ h_\Theta(x) = a^{(3)} &= g(\Theta_{10}^{(2)} a_1^{(2)} + \Theta_{11}^{(2)} a_2^{(2)} + \Theta_{12}^{(2)} a_3^{(2)} + \Theta_{13}^{(2)} a_4^{(2)} + \Theta_{14}^{(2)}) \end{aligned} \tag{3.18}$$

Cada neurona de una capa i se interconectan con todas las neuronas de la capa inmediatamente anterior $i-1$. Las neuronas de capa i consideran entradas a lo que consideraban salidas las neuronas de la capa $i-1$. Este proceso se llama forward propagation.

3.4 Trabajos previos

Hasta la investigación realizada durante esta tesis, el único estudio similar al presentado aquí, es el realizado por Ikhlaas Gurrib y Firuz Kamalov [26], bajo la universidad Canadiense de Dubai. En dicho artículo, los autores exponían también, que consideraban que el suyo era el primer estudio con un enfoque similar sobre este tema ya que desconocían de otros estudios similares.

El objetivo de Gurrib[26] era parecido en cuanto a que, trataba de crear un método capaz de predecir la dirección de BTC usando técnicas de ML y combinado con análisis de sentimientos. Su motivación, era tambien la creciente atención, que ha recibido el mercado crypto en los últimos años, sin embargo, el algoritmo de ML seleccionado en dicho caso fue LDA (Linear Discriminant Analysis).

Según explican los propios autores su enfoque consistía en el entrenamiento de un clasificador basado en LDA que utiliza la información actual del precio de BTC y los titulares de las noticias sobre BTC. Así, pretendían pronosticar la dirección del precio de BTC al día siguiente.

Entre los hallazgos y resultados del estudio cabe destacar la precisión de pronóstico más óptima obtenida 0,585. Esta precisión fue conseguida con el modelo tras la inclusión del análisis de sentimiento sobre los titulares. Defendían además la selección de tanto los valores como el sentimiento eran valiosos para realizar la predicción ya que la precisión era ”superior a una suposición aleatoria”.

4

Diseño e Implementación del sistema

4.1 Elecciones de diseño

Teniendo en cuenta el objetivo a alcanzar,los modelos a implementar debían ser clasificadores binarios para la subida 1, o bajada 0, del precio del BTC. Esto se debe además a que la implementación de un diseño de regresión, aunque pudiese acercarse de forma excelente al precio diario, este podría ser superior o inferior al real, llevando a la perdida de capital. Es por ello que consideramos más interesante la idea de implementar un clasificador, capaz de indicarnos la dirección del mercado.

4.2 Obtención y Estructura de los datos

Para trabajo de fin de grado, los datos relacionados con el precio de BTC fueron descargados mediante el uso de la librería $yfinance$. Esta librería nos permite acceder al precio de un cierto símbolo en YahooFianance, en este caso ”BTC-USD”. Tras realizar la descarga del data set, obtendremos los datos diarios de BTC desde los primeros registros de Yahoo, que corresponden a 2014. La estructura será la siguiente:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2014-09-17 00:00:00+00:00	465.864014	468.174011	452.421997	457.334015	21056800	0.0	0.0
2014-09-18 00:00:00+00:00	456.859985	456.859985	413.104004	424.440002	34483200	0.0	0.0
2014-09-19 00:00:00+00:00	424.102997	427.834991	384.532013	394.795990	37919700	0.0	0.0
2014-09-20 00:00:00+00:00	394.673004	423.295990	389.882996	408.903992	36863600	0.0	0.0
2014-09-21 00:00:00+00:00	408.084991	412.425995	393.181000	398.821014	26580100	0.0	0.0
...	...	...	...	...	...	...	...

Figura 4.1: Estructura de Datos BTC

Dispondremos de más de 3200 días a nuestra disposición, por lo que, en principio el volumen de datos no supondrá un problema para entrenar el modelo. Como se puede apreciar, hay 2 columnas con valores 0, correspondientes a los dividendos y el desdoblamiento de acciones. Estos datos los proporciona *yfinance* ya que generalmente los símbolos representan valores de acciones, sin embargo, son innecesarios y por ello se eliminarán dichas columnas.

Por otro lado, también se utilizarán datos correspondientes al las modificaciones sobre la página de BTC de Wikipedia. Estos datos parecen útiles para ajustar más precisamente el modelo. La hipótesis parte de que como sabemos, Wikipedia es la página de consultas más conocida y utilizada por la mayoría de usuarios. Así mismo, cuántas más modificaciones se produjesen en la misma, podría implicar un mayor interés en el activo o eventos significativos que implicasen la necesidad de ediciones y por lo tanto supusieran cambios en la tendencia.

Para descargar los datos de las revisiones de Wikipedia haremos uso de la librería *mwclient* media wiki client. Una vez completa la operación las revisiones tendrán una estructura como la visible en la figura.

```
OrderedDict([('revid', 275832581),
             ('parentid', 0),
             ('user', 'Pratyeka'),
             ('timestamp',
              time.struct_time(tm_year=2009, tm_mon=3, tm_mday=8,
yday=67, tm_isdst=-1)),
             ('comment', 'creation (stub)')])
```

Figura 4.2: Estructura revisiones Wikipedia

Además, al descubrir que las ediciones de la página tenían comentarios, se decidió incluir la opinión recogida en ellos, tras realizar análisis de sentimientos sobre los mismos. Esta decisión partía de la hipótesis de que tener cierta idea

sobre la opinión de un activo, podría ayudar a determinar la subida o bajada del mismo. Esto se debe a que, por regla general, la opinión sobre un mercado suele implicar si este está alcista o bajista.

Tras procesar los datos como explicaremos en la sección 4.3. Los datos se mostrarían como en la siguiente figura:

	edit_count	sentiment	neg_sentiment
2009-03-08	4	-0.550525	0.75
2009-03-09	0	0.000000	0.00
2009-03-10	0	0.000000	0.00
2009-03-11	0	0.000000	0.00
2009-03-12	0	0.000000	0.00
...	...	...	...

Figura 4.3: Estructura de la opinión y los comentarios

Posteriormente, uniríamos ambos dataset, teniendo en cuenta que, los datos de comentarios de Wikipedia comienzan en 2009. Por lo tanto, reduciremos las filas del mismo para coincidir con el precio de BTC, por lo que el Data set quedará como en la figura siguiente.

	open	high	low	close	volume	edit_count	sentiment	neg_sentiment
2014-09-17	465.864014	468.174011	452.421997	457.334015	21056800	3.600000	-0.210959	0.513422
2014-09-18	456.859985	456.859985	413.104004	424.440002	34483200	3.266667	-0.263395	0.541993
2014-09-19	424.102997	427.834991	384.532013	394.795990	37919700	3.466667	-0.228847	0.508660
2014-09-20	394.673004	423.295990	389.882996	408.903992	36863600	3.400000	-0.208723	0.497549
2014-09-21	408.084991	412.425995	393.181000	398.821014	26580100	3.333333	-0.195046	0.486438
...	...	...	...	...	...	...	...	...

Figura 4.4: Data set

Finalmente, al tratarse de un sistema de aprendizaje supervisado debemos proporcionarle los resultados esperados. Como nuestra intención es que el modelo sea capaz de predecir la dirección de BTC el proximo día, debemos añadir una nueva tag ”target”, que represente el valor esperado y_i, para el día i. Esta valor, será el valor binario de comparar el precio de cierre de BTC hoy con el precio de cierre de mañana.

Afortunadamente los datos presentaban una estructura bastante balanceada, es decir, como se puede ver en la figura, tenían aproximadamente el mismo número de días en los que el precio subía que en los que el precio bajaba. Esto favorecerá al modelo, haciendo que evite sesgos y permitiendo simplificar el modelo al reducir las preocupaciones sobre el procesamiento de ese desbalance.

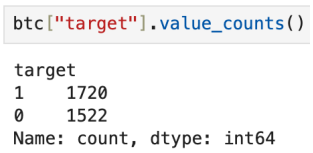


Figura 4.5: Balance de los datos

4.3 Preprocesamiento de Datos

Como ya hemos mencionado anteriormente, los datos sin procesar que obtuvimos tenían ciertos problemas en su estructura. Es por esto que, para entrenar el modelo fue preciso modificarlos. Las modificaciones realizadas fueron las siguientes:

- Sobre los datos de BTC:

  - Eliminación columnas innecesarias (Dividends y Stock Splits)

  - Crear la columna ”target”, qué será la variable objetivo

- Sobre las ediciones de la página de BTC de Wikipedia:

  - Recogida de la opinión de los comentarios usando análisis de sentimientos.

  - Transformación de la fecha de las ediciones en formato String.

  - Crear un diccionario ordenado en función del tiempo, que recoja el número de comentarios en un día, la media de la opinión de dichos comentarios y el porcentaje de comentarios negativos

  - Transformar el diccionario en un DataFrame y convertir los valores de opinión en la media de los últimos 15 días

Finalmente se ajustaron ambos data sets para que representasen los mismos días.

4.3.1 Análisis de Sentimientos

Como hemos mencionado anteriormente se decidió incluir una opinión sobre el BTC a través de los comentarios realizados en las modificaciones de la entrada BTC en Wikipedia. Para ello se realizó análisis de sentimientos utilizando un modelo preentrenado de la librería transformers. Al utilizar este modelo sobre los comentarios, este nos proporcionará un valor entre 0 y 1 representando la fuerza de ese sentimiento, acompañado de un signo positivo o negativo en función del tipo de opinión.

4.4 Implementación

El entorno utilizado para realizar el estudio ha sido python lab. La elección de Python como lenguaje para el desarrollo de la tesis se debe a la popularidad de este para realizar desarrollos de ciencia de datos y ML. Es por ello que cuenta con numerosas librerías especificas y OpenSource como: TensorFlow, Keras, PyTorch, scikit-learn, pandas, NumPy, etc. Estas, proveen funciones optimizadas y de alto nivel.

Para la implementación de los modelos basados en árboles, RandomForest y XGBoost, utilizaremos la librería sklearn. Es la librería estándar para machine learning en Python y provee eficientes implementaciones de una amplia variedad de algoritmos. Con esta librería podemos modelar fácilmente y a la vez que ajustar los hiperparámetros de forma sencilla. Para el prototipo de la red neuronal LSTM sin embargo utilizaremos keras API que se ejecuta sobre TensorFlow.

Los primeros 3 modelos, fueron modelos básicos de Random Forest, XGBoost y RNN, Red de Neuronas Recurrentes LSTM, Long Short-Term Memory, los hiperparámetros de estos modelos fueron completamente aleatorios. Los resultados de los modelos fueron los siguientes, presentados como sus matrices de confusión y su precisión:

Random Forest:

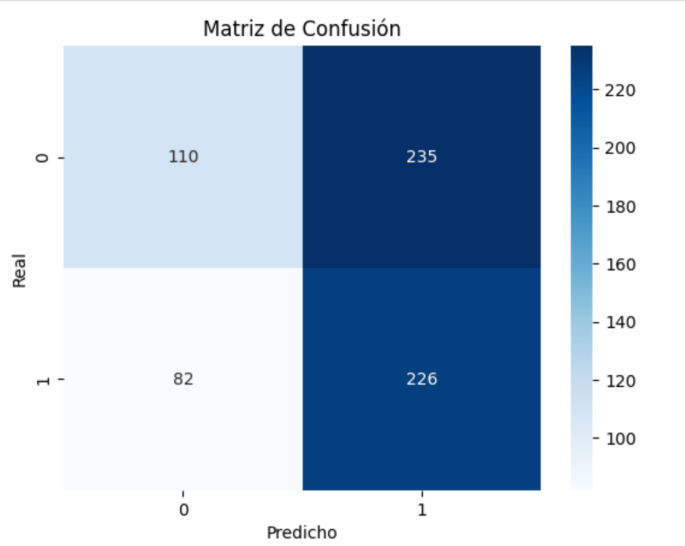


Figura 4.6: Matriz de Confusión RandomForest

XGBoost:

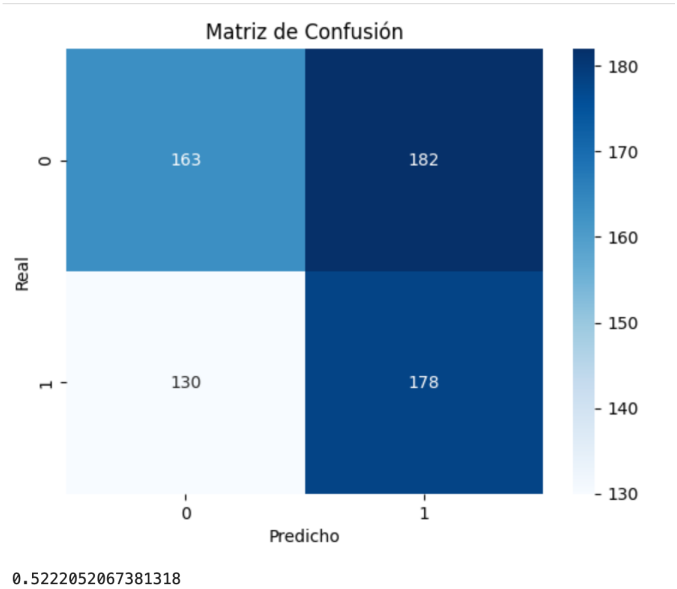


Figura 4.7: Matriz de Confusión XGBoost

LSTM:

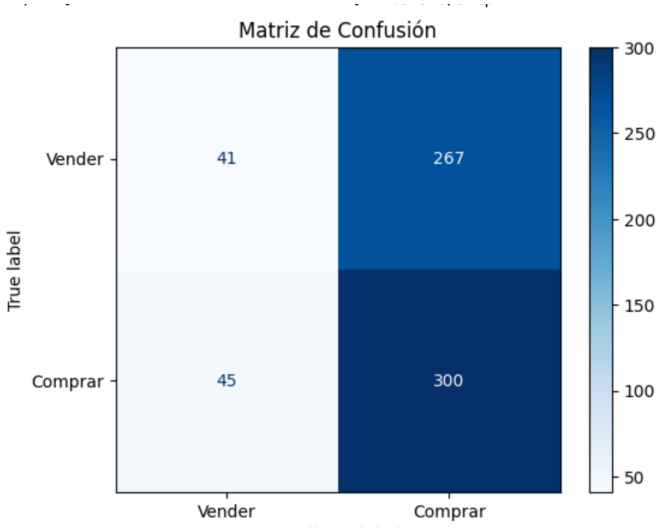


Figura 4.8: Matriz de Confusión LSTM

Sobre el modelo LSTM debemos hacer una anotación y es que parece presentar una tendencia a sobreajustarse en favor de la compra. Posteriormente, al ajustar las capas, veremos como se desarrolla esta aparente bias a favor de la compra.

4.4.1 Funciones

Las funciones que se decidieron implementar parten de diferentes hipótesis que podrían mejorar la precisión del modelo y son la siguientes:

- **backtest**: es una función creada para poder evaluar los modelos, no solo con los últimos datos, sino hacerlo progresivamente. Esta función permite entrenar con franjas de 3 años (start=1095) y hacer predicciones con los siguientes 6 meses (step=150). Posteriormente concatena las predicciones.

  La validación cruzada (cross-validation) es una técnica para evaluar modelos de machine learning en la que se parte el conjunto de datos en subgrupos, se entrena el modelo con una parte y se prueba con el subgrupo restante. Esto se repite intercambiando los conjuntos de entrenamiento y prueba.

  Sin embargo, al tratarse de una serie temporal no tendría sentido hacer validación cruzada convencional. Es por ello que se aplica la validación cruzada walk-forward, que es una forma especial de cross-validation para series temporales. En este caso, la división en conjuntos de entrenamiento y prueba se realiza respetando el orden temporal de los datos, no aleatoriamente,

evitando incongruencias de intentar predecir datos de 2015 con datos de 2020.

- **compute_rolling**: es una función creada con la idea de proporcionar al modelo tendencias temporales, que podrían ayudar a predecir la variable objetivo. Para ello se crean las nuevas variables usando las medias móviles de los intervalos de tiempo determinados (2 días, 1 semana, 2 meses y 1 año):

  - *ratio*: ratio sobre el precio de cierre en el intervalo dado.

  - *edit*: promedio de edits.

  - *trend*: tendencia objetivo con medias móviles.

  La hipótesis de la creación de esta función parte de que, por regla general, los mercados de valores siguen tendencias temporales. Parecía razonable por lo tanto proporcionarle al modelo dicha información para tratar de mejorar su precisión.

Sin embargo, para probar si estas hipótesis eran correctas, dado que no teníamos certeza de que estas funciones favorecieran a los modelos, se decidió comprobar si eran útiles ajustando los hiper-parámetros con y sin ellas.

4.5 Ajuste de los modelos

4.5.1 Ajuste de RandomForest

Al ser el modelo más simple el tiempo en relizar pruebas fue menor dado a que su coste computacional era mucho más simple. Esto permitió realizar muchas más pruebas en comparación con los otros modelos. Los parámetros que se consideraron para ajustar fueron:

- **n_estimators**: El número de árboles generados. Se eligieron los siguientes valores [25, 50, 100, 150, 200].

- **max_depth**: profundidad máxima de cada árbol. Los valores seleccionados fueron [10, 15, 20, None].

- **max_features**: número máximo de de características para dividir un nodo. Sus valores fueron [’sqrt’, ’log2’, None]. El numero de features se decidía haciendo la raíz, el logaritmo en base 2 y none nada, sobre el número de carácterísticas.

- **min_sample_split**: mínimo número de muestras para separar un nodo. Sus valores fueron [2, 3, 5, 10, 50].

La selección de estos hyperparámetros, al igual que sus rangos no se realizó de manera aleatoria, sino que fueron valores dentro de los rangos recomendados según la bibliografía consultada[27].

Los resultados del testeo de todas estas combinaciones de hiperparámetros, con distintos tamaños de entrenamiento y aplicación de las funciones, reportaron los datos que se pueden encontrar en el apartado 7. Los mejores modelos para cada caso fueron los siguientes:

Mejor modelo	Entrenamiento:Tests	Hiperparámetros				Test Accuracy	Precision
		n_estimators	max_depth	min_sample	max_features		
Random Forest	80:20	100	None	10	log2	0.524	0.531
	75:25	25	None	5	log2	0.538	0.557
	70:30	50	10	5	sqrt	0.540	0.558
Random Forest RollBack	80:20	25	15	2	None	0.529	0.539
	75:25	25	None	5	sqrt	0.529	0.543
	70:30	150	15	10	log2	0.538	0.530
Random Forest BackTest	80:20	25	20	50	log2	0.511	0.521
	75:25	25	20	50	log2	0.511	0.521
	70:30	25	20	50	log2	0.511	0.521
Random Forest Backtest y RollBack	80:20	50	10	5	None	0.512	0.522

Table 4.1. Ajuste Random Forest

Dado el sobreajuste que presentaron estas pruebas se intento proporcionar el conjunto de test y train sin orden temporal, que aunque pudiera no parecer la mejor solución podría ayudar al mejor reconocimiento de patrones y reducir el sobreajuste. Con esta técnica se consiguió el siguiente modelo que es el modelo con mejor capacidad de predicción, 0.585 accuracy sobre test.

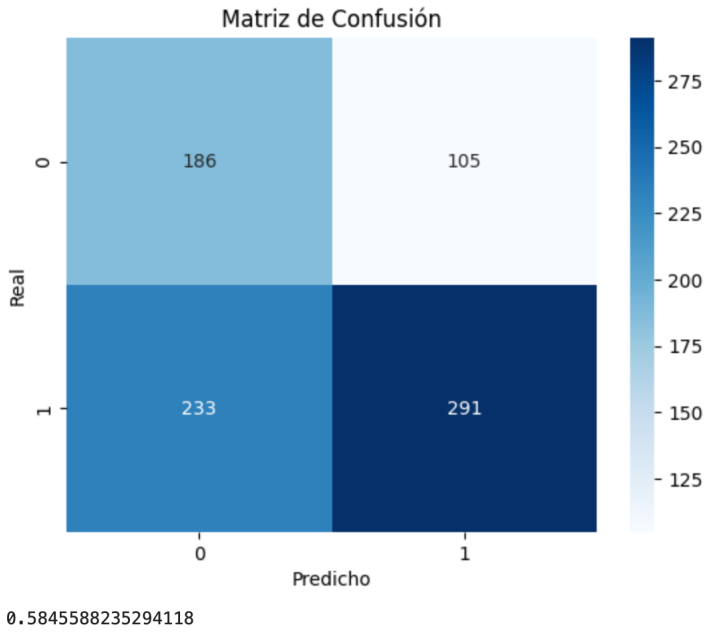


Figura 4.9: Matriz de Confusión mejor modelo Random-Forest

4.5.2 Ajuste de XGBoost

Al igual que para Random Forest, se realizó una selección de los hiper-parámetros más significativos, dentro de los rangos recomendados por Amazon web service[28]. Los hiperparámetros considerados fueron:

- **n_estimators**: vuelve a ser número de árboles generados. Se eligieron los siguientes valores [25, 50, 100, 200].
- **max_depth**: profundidad máxima de cada árbol. Los valores seleccionados fueron [10, 15, 20, None].
- **learning_rate**: Tasa de aprendizaje para actualizar pesos en cada boosting round. Los elegidos fueron [0.01, 0.1, 0.2, 0.3]
- **min_child_weight**: Mínimo número de muestras requeridas en un nodo hoja. Sus valores fueron [1, 2, 3, 4].
- **gamma**: controla poda de hojas, reduce overfitting. Los valores fueron [0, 1, 2, 3].

Al igual que para Random Forest, todas las pruebas se pueden consultar en los anexos. Los mejores modelos fueron:

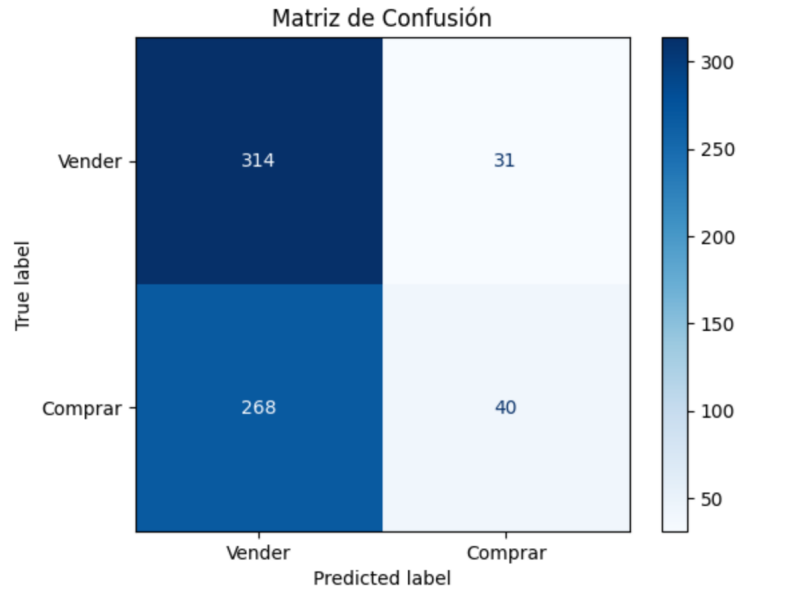
Mejor modelo	Entrenamiento:Tests	Hyperprámetros					Test Accuracy	Precision
		n_estimators	max_depth	learning_rate	min_child_ weight	gamma		
XGBoost	80:20	200	15	0.3	3	0	0.524	0.672
	75:25	100	20	0.3	2	2	0.518	0.614
	70:30	200	10	0.3	2	0	0.536	0.679
XGBoost RollBack	80:20	200	15	0.3	3	0	0.524	0.711
	75:25	25	20	0.3	2	2	0.518	0.772
	70:30	200	10	0.3	2	0	0.536	0.8125
XGboost BackTest	80:20	50	10	0.3	4	1	0.516	0.529
	75:25	50	10	0.3	4	1	0.516	0.529
	70:30	50	10	0.3	4	1	0.516	0.529
XGBoost Desordenado	80:20	100	20	0.1	1	1	0.559	0.692
	75:25	150	20.0	0.01	2	0.3	0.567	0.731
	70:30	50	15	0.2	2	1	0.541	0.623

Table 4.2. Ajuste XGBoost

Como podemos apreciar al igual que pasaba con RandomForest, al realizar la división de los datos ordenados, producía un mayor sobreajuste, empeorando el modelo.

4.5.3 LSTM

Como pudimos ver en el primer modelo, de LSTM, primeramente este modelo parecía tener un sesgo a favor de uno de los dos conjuntos a clasificar. Aún así, para comprobar si era cierto, se decidió realizar un ajuste del número de capas mediante random search y tamaños de prueba [32,64,128], para comprobar si se había producido por casualidad o los modelos tenían la tendencia a ajustarse en favor a sesgos, producidos por el pequeño desbalance de los datos. Tras realizar el ajuste de capas, a pesar de lograr un accuracy decente la precisión era bastante mala, pudimos comprobar que había un sesgo por parte del modelo.




```
test_acc
```

```
0.5421133041381836
```

Figura 4.10: Matriz de Confusión LSTM tunned

Al observar que efectivamente el modelo tenía un sesgo, se decidió aplicar un umbral al modelo. El valor del umbral fue seleccionado para compensar el desbalance de los datos positivos y negativos, por lo que su valor fue similar al desbalance entre los mismos. Los resultados de este modelo fueron mucho más precisos como podemos ver en la siguiente figura:

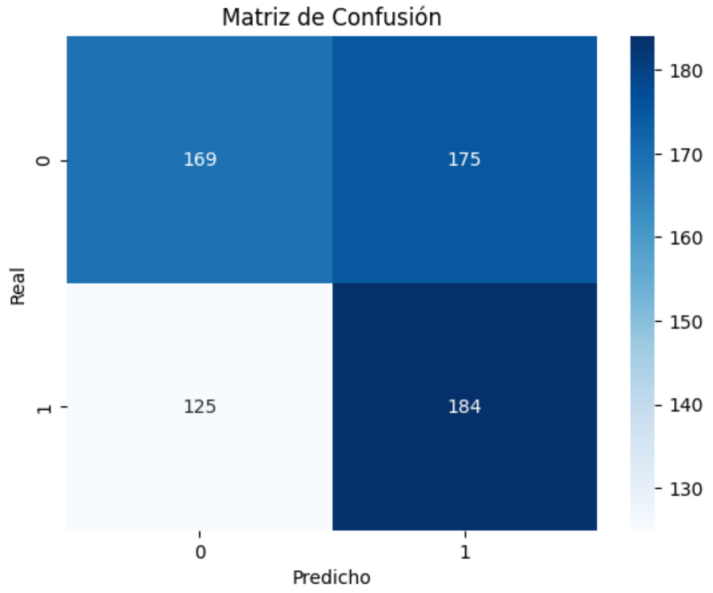


Figura 4.11: Matriz de Confusión mejor LSTM con umbral

Dado el alto coste computacional de este modelo, se debió reducir el número

de pruebas y es por esto que no se realizaron tantas pruebas como en anteriores modelos. Sin embargo, se decidió realizar una última prueba con el umbral y rollback de los datos, cuyo mejor resultado fue el siguiente:

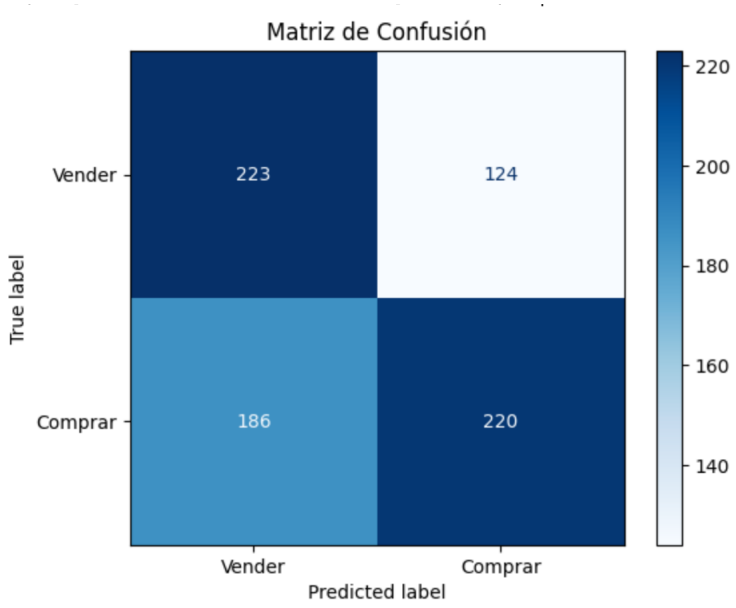


Figura 4.12: Matriz de Confusión mejor LSTM rollback

4 Diseño e Implementación del sistema

5

Discusión

En este capítulo explicaremos las conclusiones a las que hemos llegado tras el desarrollo de la tesis, así como posibles mejoras correspondientes a trabajos futuros para mejorar los modelos.

5.1 Conclusiones

Con respecto a los resultados obtenidos tras el desarrollo de los diferentes modelos, podemos concluir que tal y como afirmábamos en la introducción el mercado de las criptomonedas en un sector de alto riesgo y de difícil predicción, dados sus bruscos cambios de tendencia, volatilidad e inestabilidad.

Tras el estudio de los diferentes algoritmos, podemos afirmar que efectivamente, las características del mercado previamente mencionadas dificultan en gran medida la capacidad de encontrar patrones y por lo tanto empeora el desempeño de los modelos. Podemos concluir también que, a pesar de que los resultados de esta tésis son susceptibles a mejoras, alcanzar con el mejor modelo una precisión idéntica a la obtenida por Ikhlaas Gurrib y Firuz Kamalov[26] en un paper publicado, podría considerarse un éxito.

Al igual que afirmaban estos autores, aunque pudieran no parecer niveles de predicción extremadamente altos, son niveles bastante aceptables dadas condiciones y volatilidad del mercado a predecir. Además puesto que el desarrollo es el correspondiente a un trabajo de fin de grado, alcanzar los niveles de un artículo publicado parece más que satisfactorio.

5.2 Trabajo Futuro

Para el desarrollo de esta tesis, se tuvo en cuenta la posibilidad de añadir las búsquedas sobre BTC, proporcionadas por Google trends, desafortunadamente, finalmente no se pudieron incluir. El motivo fue que, a conocimiento del autor de esta tesis, acceder registro de búsquedas diarias sobre BTC resultó imposible. Los datos tan solo aparecen como un porcentaje de interés, y al evaluar los años se ofrece la media mensual, por lo que sería inútil para el modelo creado.

Sin embargo, creemos que teniendo la posibilidad de acceder a dichos datos en formato diario, ayudaría en gran medida a la creación de un modelo mucho más preciso. Además, dados los buenos resultados que proporcionó añadir el análisis de sentimientos, acompañados por la numerosa bibliografía apoyando la implementación de análisis de sentimientos para predecir mercados bursátiles, parece concluyente que añadir por ejemplo, si fuese posible análisis de sentimientos sobre los principales artículos publicados sobre BTC en cada día o en una franja temporal no mayor a una semana.

Entre los trabajos futuros también se debería considerar la tendencia al sobreajuste de los modelos basados en árboles, que presentaron una gran brecha entrenamiento-prueba. En cuanto a los modelos, quizá sería conveniente el estudio de otros modelos que pudiesen tener un mejor desempeño que los seleccionados para la realización de la tesis, como por ejemplo SVM como plantean Ikhlaas Gurrib y Firuz Kamalov [26].

Sin embargo, independientemente de posibles mejoras, podemos concluir que el proyecto ha sido fructífero puesto que ambos objetivos planteados en la sección 1.2 fueron completados de manera satisfactoria.

5.3 Originalidad y valor añadido

Como publicaban Ikhlaas Gurrib y Firuz Kamalov [26], el suyo parecía ser el único estudio sobre técnicas de predicción de BTC con técnicas de ML y análisis de sentimientos. Tras el estudio realizado por el alumno en este trabajo de fin de grado, esta afirmación parece seguir siendo real. Por tanto, podemos afirmar que, la idea detrás del desarrollo de esta tesis es original y dada esa originalidad le confiere al contenido de esta tesis un cierto valor añadido.

6

Desarrollo del proyecto

6.1 Planificación

La planificación de este trabajo se puede dividir en tres etapas: fase de estudio, fase de análisis y diseño y fase de creación del documento.

- Fase de estudio: Al inicio del proyecto el conocimiento sobre este tema de estudio era limitado. Por lo tanto, fue esencial un estudio profundo del campo para entender las perspectivas teóricas y posibilidades de aplicaciones de ML.
- Fase de análisis y diseño: En esta fase se esperaba desarrollar los modelos y entrenamiento de los mismos. Esto implicó la selección de los datos y preprocesamiento de los mismos, implementación de los diferentes modelos, desarrollo de resultados, mejora del código, etc.
- Fase de creación del documento: La fase final consistió en plasmar el trabajo realizado en un documento. Esto implicó la visualización de resultados y la redacción de la tesis de grado.

6.2 Presupuesto

Para el presupuesto estimado se han considerado los costes humanos y de recursos asociados, incluidos los costes de hardware y software.

Para los costos humanos asociados, se considerarán tan solo los salarios del tutor y estudiante de este TFG. El coste estudiantil será el representado por el salario medio anual de un ingeniero recién egresado, que según jobted [29], es de 20.450€ brutos (antes de impuestos). Según el mismo sitio web, el salario medio para un ingeniero informático senior, con ”10-20 años de experiencia gana un promedio de 59.500€”.

Por lo tanto, el coste total del proyecto asociado a recursos humanos se presenta en la siguiente tabla: Para analizar los costes asociados de los recursos

	Horas	Salario por hora	Coste sobre el proyecto
Ingeniero Junior	400	10,65€	4260€
Ingeniero Senior	20	30,99€	619,80€
			Total: 4879,80€

Table 6.1. Presupuesto recursos humanos

utilizados se evalúa tanto hardware como software, costes energéticos y de documentación.

- El hardware, tan solo se tendrá en cuenta el ordenador utilizado, obviando costes referentes a material de oficina (folios, boligrafos...).
- Referente al software, dado que todos los elementos de software son open-source, o la universidad nos proporciona licencia para su uso, no existe coste alguno.
- En cuanto al coste energético, según los datos proporcionados por la Organización de Consumidores y Usuarios, OCU [30] referentes al teletrabajo el consumo eléctrico de una jornada laboral sería:

  - Consumo del portátil: 1,6 kW en una jornada de 8 horas al día, lo que supone 50 cts/día.

  - Climatización: Estiman 1kWh por hora, unos 2,4 euros por día çon precios de la electricidad y el gas actuales”.

  Teniendo en cuenta estos datos implicaría unos 0,3625cts/h, que solo se aplicarán a las horas de trabajo del alumno.

- Referente a la documentación, cabe destacar la necesidad de la compra del artículo[26] sobre el estudio realizado por Ikhlaas Gurrib y Firuz Kamalov.

Recursos	Coste	Tiempo de uso	Depreciación por uso
Hardware	1.298,85€	≃ 3 meses	≃130€
Software	0	≃ 3 meses	-
Energético	145€	400h	-
Documentación	32,00€	-	-

Table 6.2. Coste de recursos utilizados

El coste final de los recursos asciende a los 307€ que se añadirán a los de recursos humanos para dar un precio final del proyecto de 5186,80€

6.3 Impacto socioeconómico

El impacto socio-económico esperado con este proyecto es muy similar a la motivación y objetivo del mismos, explicados en el capítulo 1.

Dado el creciente aumento del mercado cripto, y su adopción, también parece crecer el desconocimiento de sus usuarios. Son más y más las personas que se introducen en el ecosistema, promoviendo la adopción masiva. Sin embargo, la mayoría lo hace desde el desconocimiento con la idea de conseguir dinero fácil, poniendo en peligro sus ahorros. Es por ello que el estudio aquí presentado espera servir para proteger a aquellos que quieran introducirse en este sector, trayendo concienciando sobre qué es realmente BTC, los riesgos del mercado y proporcionando una pequeña ayuda para aquellos que estén dispuestos a invertir en él.

6 Desarrollo del proyecto

Acrónimos

BCE Banco Central Europeo. 2, *Glosario:* Banco Central Europeo

KYC Know Your Customer. 2, *Glosario:* Know Your Customer

LDA Linear Discriminant Analysis. 22, *Glosario:* Linear Discriminant Analysis

LSTM Long Short-Term Memory. 11, 16, 27–29, 33–35, IX, X, *Glosario:* Long Short-Term Memory

MiCA Markets in Crypto Assets. 2, *Glosario:* Markets in Crypto Assets

ML Machine Learning. 4, 5, 16, 22, 27, 38, IX, *Glosario:* Aprendizaje Automático

NLP Natural Language Processing. 15, 16, IX, *Glosario:* Procesadores del lenguaje natural

OCU Organización de Consumidores y Usuarios. 40, *Glosario:* Organización de Consumidores y Usuarios

Glosario

Aprendizaje Automático Desarrollo de sistemas informáticos que son capaces de aprender y adaptarse sin seguir instrucciones explícitas, mediante el uso de algoritmos y modelos estadísticos para analizar y sacar inferencias a partir de patrones en los datos.. 4

Banco Central Europeo El Banco Central Europeo es el banco central de los Estados de la Unión Europea que tienen el euro como moneda. Dirige el Eurosistema y conforma también, junto con los bancos centrales nacionales de los demás Estados miembros ajenos a la eurozona, el Sistema Europeo de Bancos Centrales. 2

Know Your Customer Es el proceso de una empresa que identifica y verifica la identidad de sus clientes. El término también se utiliza para referirse a las regulaciones bancarias y anti-lavado de dinero que rigen estas actividades. 2

Linear Discriminant Analysis Es un algoritmo de aprendizaje supervisado que se utiliza para tareas de clasificación en ML. 22

Long Short-Term Memory Es un tipo de red neuronal recurrente (RNN) diseñada para resolver el problema del gradiente evanescente mediante la introducción de una célula de memoria que puede almacenar información durante periodos de tiempo más largos. 27

Markets in Crypto Assets Hace referencia a la normativa europea que regula la emisión y prestación de servicios relacionados con criptoactivos y 'stablecoins'. 2

Organización de Consumidores y Usuarios es una organización sin ánimo de lucro y una asociación privada española, totalmente independiente, creada en 1975 con el objetivo de defender los derechos de los consumidores. 40

Procesadores del lenguaje natural rama de la inteligencia artificial o IA encargada de dar a los ordenadores la capacidad de comprender textos y palabras habladas de la misma manera que los seres humanos. 15

Bibliografía

[1] Z. Chen, C. Li y W. Sun, «Bitcoin price prediction using machine learning: An approach to sample dimension engineering», *Journal of Computational and Applied Mathematics*, vol. 365, pág. 112 395, 2020.

[2] J. P. Cadena, X. E. Herrera, S. C. Llaguno y B. P. Alcivar, «Criptomonedas: Funcionamiento, oportunidades y amenazas: Cryptocurrency: Operation, opportunities and threats», *Res Non Verba Revista Científica*, vol. 11, n.º 2, págs. 174-193, 2021.

[3] E. S. Wong Chiriboga, G. Navarrete Mendieta y X. Espinoza Herrera, «¿ Pueden ser reguladas las criptomonedas? Caso Bitcoin y Libra», 2021.

[4] B. C. Europeo, *Informe Anual. Eurosistema, 112.* 2015. dirección: https://www.ecb.europa.eu/pub/pdf/annrep/ar2015es.pdf.

[5] S. Nakamoto, «Bitcoin: A peer-to-peer electronic cash system», *Decentralized business review*, 2008.

[6] F. P. Dan Ashmore, *Bitcoin Price History 2009 to 2022*, 2023. dirección: https://www.forbes.com/advisor/in/investing/cryptocurrency/bitcoin-price-history-chart/.

[7] N. Popper, *Bitcoin Has Lost Steam. But Criminals Still Love It.* 2023. dirección: https://www.nytimes.com/2020/01/28/technology/bitcoin-black-market.html.

[8] R. M. Carmen Aguilar Garcia y A. J. Martin, *Behind the collapse: The real cost of Bitcoin's fall from grace*, 2018. dirección: https://news.sky.com/story/behind-the-collapse-the-real-cost-of-bitcoins-fall-from-grace-11585936.

[9] J. Tidy, *Fear and excitement in El Salvador as Bitcoin becomes legal tender*, 2021. dirección: https://www.bbc.com/news/technology-58473260.

[10] R. S. Michalski, J. G. Carbonell y T. M. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[11] P. Cunningham, M. Cord y S. J. Delany, «Supervised learning», en *Machine learning techniques for multimedia: case studies on organization and retrieval*, Springer, 2008, págs. 21-49.

[12] Z. Ghahramani, «Unsupervised learning», en *Summer school on machine learning*, Springer, 2003, págs. 72-112.

[13] S. Ruder, «An overview of gradient descent optimization algorithms», *arXiv preprint arXiv:1609.04747*, 2016.

[14] X. Ying, «An overview of overfitting and its solutions», en *Journal of physics: Conference series*, IOP Publishing, vol. 1168, 2019, pág. 022 022.

[15] F. Girosi, M. Jones y T. Poggio, «Regularization theory and neural networks architectures», *Neural computation*, vol. 7, n.º 2, págs. 219-269, 1995.

[16] X. Wan, «Influence of feature scaling on convergence of gradient iterative algorithm», en *Journal of physics: Conference series*, IOP Publishing, vol. 1213, 2019, pág. 032 021.

[17] S. B. Kotsiantis, «Decision trees: a recent overview», *Artificial Intelligence Review*, vol. 39, págs. 261-283, 2013.

[18] G. Biau y E. Scornet, «A random forest guided tour», *Test*, vol. 25, págs. 197-227, 2016.

[19] J. J. Espinosa-Zúñiga, «Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito», *Ingeniería, investigación y tecnología*, vol. 21, n.º 3, 2020.

[20] J. Schmidhuber, «Deep learning in neural networks: An overview», *Neural networks*, vol. 61, págs. 85-117, 2015.

[21] E. D. Liddy, «Natural language processing», 2001.

[22] K. Chowdhary y K. Chowdhary, «Natural language processing», *Fundamentals of artificial intelligence*, págs. 603-649, 2020.

[23] L. Breiman, «Random forests», *Machine learning*, vol. 45, págs. 5-32, 2001.

[24] A. Cutler, D. R. Cutler y J. R. Stevens, «Random forests», *Ensemble machine learning: Methods and applications*, págs. 157-175, 2012.

[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest y C. Stein, *Introduction to algorithms*. MIT press, 2022.

[26] I. Gurrib y F. Kamalov, «Predicting bitcoin price movements using sentiment analysis: a machine learning approach», *Studies in Economics and Finance*, vol. 39, n.º 3, págs. 347-364, 2022.

[27] *Range of Values for Hyperparameter Fine-Tuning in Random Forest Classification*, 2021. dirección: https://stats.stackexchange.com/questions/558060/range-of-values-for-hyperparameter-fine-tuning-in-random-forest-classification.

[28] A. Aws, *Ajustar un modelo XGBoost*, 2023. dirección: https://docs.aws.amazon.com/es_es/sagemaker/latest/dg/xgboost-tuning.html.

[29] Jobted, *Sueldo del Ingeniero Informático en España*, 2023. dirección: https://www.jobted.es/salario/ingeniero-inform%C3%A1tico.

[30] O. de Consumidores y Usuarios, *¿Cuánto se ahorra con el teletrabajo?*, 2022. dirección: https://www.ocu.org/coches/gasolina-y-carburantes/noticias/cuanto-ahorra-teletrabajo.

BIBLIOGRAFÍA

7

Anexo - Repositorio de código en GitHub

Se puede acceder al código y las pruebas mediante el siguiente repositorio de GitHub:

https://github.com/CarlosMozo/Aproximacion-a-tendencias-en-bitcoin-mediante-algoritmos-predictivos

7 Anexo - Repositorio de código en GitHub