UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Desarrollo de diferentes modelos predictivos basados en Inteligencia Artificial para su uso en escenarios de inteligencia visual y series temporales financieras.

Francisco Linaje Moreno

Rafael Gadea

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2017-18

Valencia, 31 de agosto de 2018

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG.NO. ES-CV- 000030

Resumen

Desarrollo de diferentes modelos clasificatorios en entornos de inteligencia visual mediante redes neuronales convolucionales y modelos regresivos/clasificadores para predecir tendencias en series temporales financieras.

Resum

Desenrotllament de diferents models classificatoris en entorns d'intel·ligència visual per mitjà de xarxes neuronals convolucionales i models regresiuss/classificatoris per a predir tendencies en sèries temporals financieres.

Abstract

Development of different classification models in visual intelligence environments using convolutional neural networks and regressive models / classifiers to predict trends in financial time series.

Índice

Capítulo 1. Introducción, objetivos y justificación.

1.1 Introducción.

La inteligencia artificial se ha convertido en uno de los campos con mayor potencial de crecimiento en las próximas décadas, a pesar de que en la última década haya sufrido buena parte de su total desarrollo hasta la fecha, su historia no es tan reciente.
El primer año que se fundamentó como disciplina fue en 1956 y desde entonces ha tenido un ritmo de crecimiento exponencial, calculándose que para 2056 [1] podríamos desarrollar una IA de un nivel similar al humano.
El interés sobre este campo ya se remonta mucho más atrás, incluso antes de la existencia de la electricidad, desde siempre el ser humano ha soñado con la creación de máquinas que pudieran pensar, una referencia temporal significativa la podemos ya encontrar en la antigua Grecia; Galatea una estatua que tomo vida.
También podemos encontrar otros ejemplos un siglo antes de que la primera máquina programable fuera concebida; Lovelace en 1842 creo un algoritmo destinado a ejecutarse en una máquina, esta ingeniera y matemática fue la primera persona en realmente vislumbrar el potencial de las maquinas computacionales.
Sin embargo, no fue hasta mucho más tarde, mediados del siglo XX cuando el verdadero crecimiento comenzó a efectuarse gracias al desarrollo de la tecnología.

¿Pero qué es exactamente una Inteligencia Artificial?

Se conoce por Inteligencia Artificial o "agente inteligente" toda aquella máquina que de alguna forma detecta el medio desde donde está siendo aplicada y es capaz de inferir una serie de acciones que ayuden a conseguir desarrollar las tareas o objetivos marcados.
Dentro de esta amplia definición podemos encontrar diferentes tipos de agentes inteligentes que son aplicados de manera específica a determinados entornos.

Podemos distinguir principalmente dos grandes grupos:

-Machine Learning: Un algoritmo de aprendizaje automático, es un algoritmo que es capaz de aprender por si mismo de la experiencia aportada en los datos introducidos y desarrollar una serie de previsiones de cara a una tarea específica.
En este tipo de tecnología, el ser humano aun tiene un papel principal a la hora de desarrollarlo. El principal problema al que se enfrentan los desarrolladores de ML es describir de una forma útil la tarea a resolver como reconocer audio, caminar, etc.

-Deep Learning: El aprendizaje profundo en maquina o comúnmente llamado "redes neuronales" se basa principalmente en un conjunto de capas de neuronas que van generalizando el patrón existente en los datos con el fin de desarrollar predicciones con un error objetivo.
Dentro de este grupo podemos encontrar una gran variedad de redes que más adelante explicare como son las redes neuronales convolucionales, redes feedforward, redes neuronales recurrentes, etc.

El futuro para la inteligencia artificial es más que esperanzador, ya hay estudios muy exhaustivos sobre el impacto de la IA en materia económica, como el de Accenture sobre el impacto de IA en diversos países, donde se detalla que para 2035 el PIB de los principales países de occidente podría crecer a una tasara del 30% al 50% por encima de su nivel base, si absorben completamente su uso en todos sus sectores económicos.

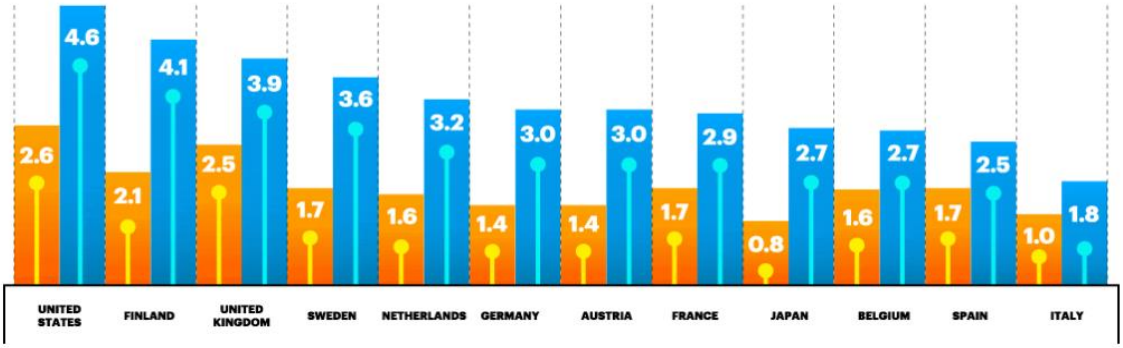

Figura 1: Crecimiento anual en 2035 del PIB comparando el crecimiento económico base del país(naranja) vs crecimiento del PIB con una economía con plena absorción de la IA. *[2]*

En su informe también explican como el nivel de productividad crecerá gracias a la absorción de esta nueva tecnología entre un 10% y un 40%. Desglosando este crecimiento económico por sectores podemos observar claramente su impacto:

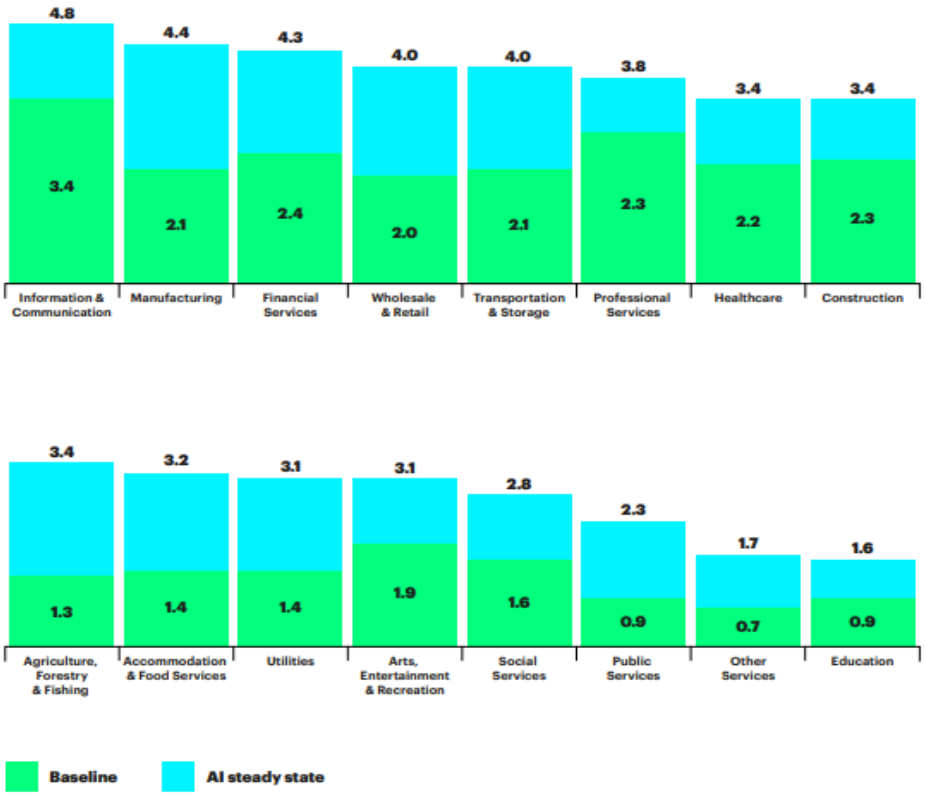

Figura 2: Crecimiento por sectores desglosado. *[2]*

Así como sus beneficios empresariales en varios puntos porcentuales según el sector económico al que pertenezcan sus empresas.

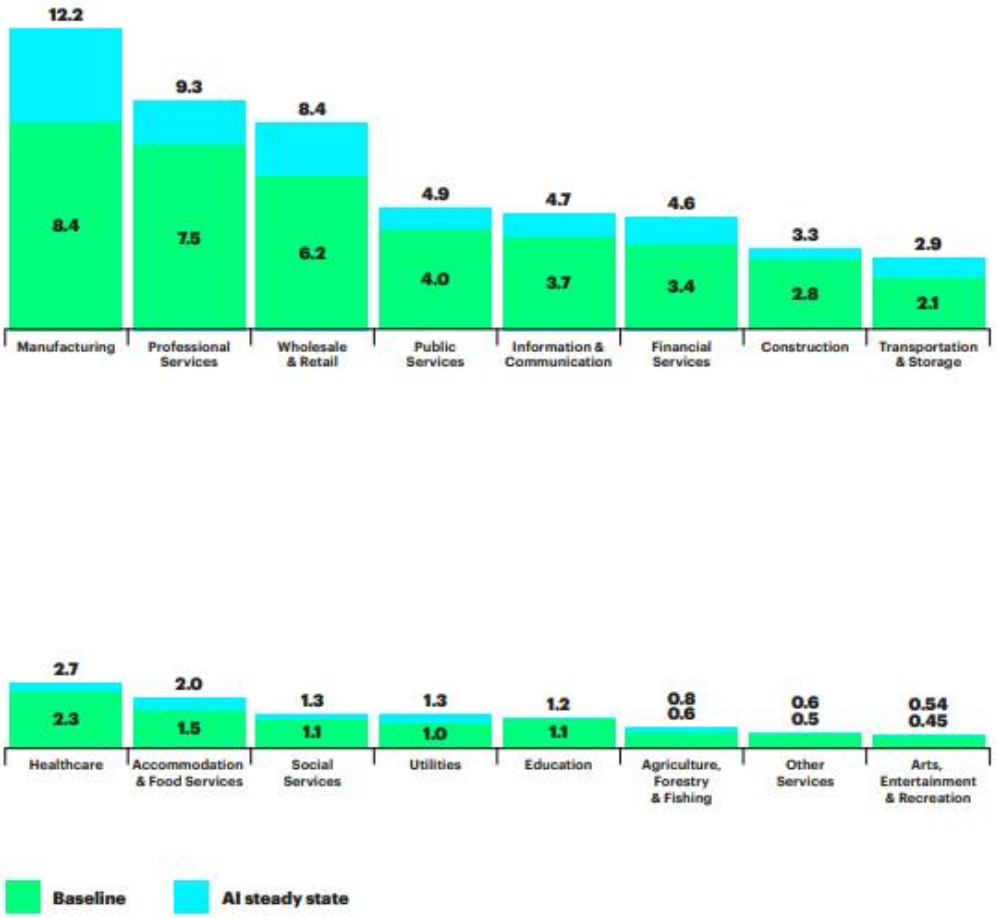

Figura 3: Beneficios empresariales desglosados por sector. *[2]*

El panorama que nos dibuja la IA, es más que alentador, demuestra un claro potencial a largo plazo para mejorar nuestra productividad en tareas específicas, pero sus beneficios no se acotan solo al plano económico, en el campo de la investigación científica podría suponer la cura o encontrar nuevas vías de detección y tratamiento en diversas enfermedades médicas, como en la detección de diversos tipos de cancer, diversas cardiopatías, etc. Estos avances no solo se dan en enfermedades físicas, sino también en el campo de la salud mental donde se han conseguido grandes avances en el tratamiento de enfermedades mentales como la depresión, psicosis, etc. En el caso concreto del diagnóstico de pacientes con psicosis el software desarrollado por IBM [4] conseguía hasta un 80% de precisión y no solo eso, establecían una ventana de precisión de hasta 2 años, en los que un individuo sin antecedentes podría bajo ciertos factores volverse psicótico. Esto supone un gran avance porque en materia de psique humana no se disponen en muchos casos de pruebas científicas que permitan diagnosticar con precisión, sino que el diagnostico se realiza mediante un diagnostico diferenciado muy limitado a veces.

1.2 Objetivos y justificación

Este proyecto surgió como línea de investigación para la implementación de tecnología basada inteligencia artificial para la startup NextNight, dirigida por el propio autor de este TFG.

NextNight, es una startup que propone una serie de servicios multiplataforma web y móvil para la digitalización del sector del ocio nocturno.

NextNight SN se basa en una aplicación móvil que tiene como ambición convertirse en una red social para sus usuarios donde puedan tener todas las actividades y promociones desarrolladas dentro de este sector en una única aplicación y así consultar y planear de forma rápida y sencilla sus actividades.

Por otro lado, esta startup implementa otra aplicación, llamada NextNight BC, desarrollada para que los clientes o locales de ocio nocturno puedan subir a esta red social sus actividades. Esta aplicación permitirá acceder a un dashboard web donde poder visualizar diferentes métricas y alertas obtenidas bajo una arquitectura Big Data.

A la aplicación se pueden subir tanto videos, como imágenes, texto, etc, además recolecta la información de las diferentes interacciones que realizan los usuarios dentro de NextNight app.

Es en este contexto, donde se plantean los siguientes objetivos:

1. Introducción y desarrollo de algoritmos de detección de objetos y clasificación de imágenes, para el procesamiento de los diferentes anuncios subidos y la clasificación de estos.

2. Desarrollo de modelos predictivos orientados a serie temporales sobre procesos estocásticos, para el análisis de posibles patrones que optimicen el éxito de los locales a la hora de llegar a sus usuarios, como predicción de ventas, eventos potenciales, etc.

Debido a que la startup no dispone actualmente del dataset necesario para realizar estos modelos, se emplearan otros diferentes a modo de ejemplo, tanto para el desarrollo de modelos orientados a la inteligencia visual, como para el análisis de series temporales.
Proponiéndose como datasets de ejemplo cats vs dogs" y Cifar10 para inteligencia visual y el índice DJIA para el desarrollo de modelos predictivos en procesos estocásticos, debido a su complejidad y a su posible correlación con otros mercados como ventas de bebidas, nivel de actividad de los usuarios, etc.

Capítulo 2. Machine Learning

Cuando se habla de **Machine Learning** (ML), uno se puede acoger a la definición dada por Arthur Samuel en 1959, donde expresa lo siguiente: "hablamos de una técnica que permite dar a los ordenadores la habilidad de aprender sin ser explícitamente programados" y asimismo, otra definición data desde una perspectiva más cercana a la ingeniería realizada por Tom Mitchell en 1997 "se trata de un programa de ordenador capaz de aprender de una experiencia E respecto a una tarea T y que es capaz de desarrollar una actuación A que mejora con E".

Una vez desarrollado este agente inteligente, no solo nos servirá en la tarea específica en la que ha sido entrenada, sino que además analizando los resultados quizás se puedan ver correlaciones entre diferentes variables que no se creían posibles.

En resumen, podemos observar como ML es realmente útil para:

- Problemas donde es necesario programar múltiples normas con reglas if-else.
- Problemas donde los métodos tradicionales no permiten obtener una solución.
- Medios fluctuantes, donde los algoritmos ML puede adaptarse continuamente y mantener unos resultados en una tarea específica.
- Obtener un análisis o ideas sobre problemas complejos que requieren cantidades enormes de datos.

2.1 Tipos de sistemas

Supervisados vs No Supervisados, Semi supervised and Reinforcement Learning: Este apartado será desarrollado más en profundidad ya que es la forma más utilizada para clasificar en grueso modo un sistema ML.

Cuando se entrena un sistema mediante una técnica ***supervisada***, se dota a la máquina del valor inicial y de su solución, con el fin de que este sistema aprenda.

Algunos algoritmos utilizados en esta línea son:

a) k -Nearest Neighbors
b) Linear Regression
c) Logistic Regresion
d) Support Vector Machines (SVMs)
e) Decision Trees and Random Forest
f) Neural Networks

En los sistemas ***No Supervisados*** el sistema trata de aprender por sí mismo sin la solución final, se utiliza principalmente a la hora de clasificar elementos que aparentemente no se conoce la correlación entre ellos o los supuestos grupos a los que pertenezcan.

Algunos de los algoritmos empleados son:

a) Clustering
b) Visualization and dimensionality reduction
c) Association rule learning

En el caso de "dimensionality reduction" la maquina permite reducir y simplificar los datos sin perder la información esencial, lo hace gracias a la unión de dos variables muy correladas en una sola, esta técnica también suele ser llamada "feature extraction". Gracias a esta técnica es posible dejar preparada tu "training data" de forma que al ser utilizada para entrenar a otro sistema ML supervisado ocupe menos memoria en disco y sea procesada mucho más rápido.

Por último, la técnica "Association rule learning", permite detectar altas correlaciones entre variables, es una de las técnicas más utilizadas en ventas, ya que permite establecer patrones a priori no conocidos.

Por otro lado, se pueden encontrar en esta línea sistemas mixtos, llamados "***Semisupervised learning***", en estos sistemas algunos inputs vienen ya etiquetados y otros no, esto implica que el sistema aprende por sí mismo a etiquetar y clasificar esas variables, pero utilizando como guía los inputs ya clasificados, un ejemplo de este sistema podría ser el etiquetado de fotos en diferentes redes sociales, donde al etiquetar en algunas fotos a ciertas personas el sistema aprende a reconocerlas y al subir otras fotos con ellas presente, el sistema puede automáticamente reconocer y etiquetar las.

Por último, podemos encontrar una técnica muy diferente al resto y novedosa llamada "***Reinforcement Learning"***, el sistema de aprendizaje también llamado "agente" analiza el entorno y el contexto realizando en función de este, una serie de acciones en base a una política que ella misma desarrolla y perfecciona. En función de su desempeño obtiene una serie de recompensas o penaltis, tal y como se describe en la siguiente imagen.

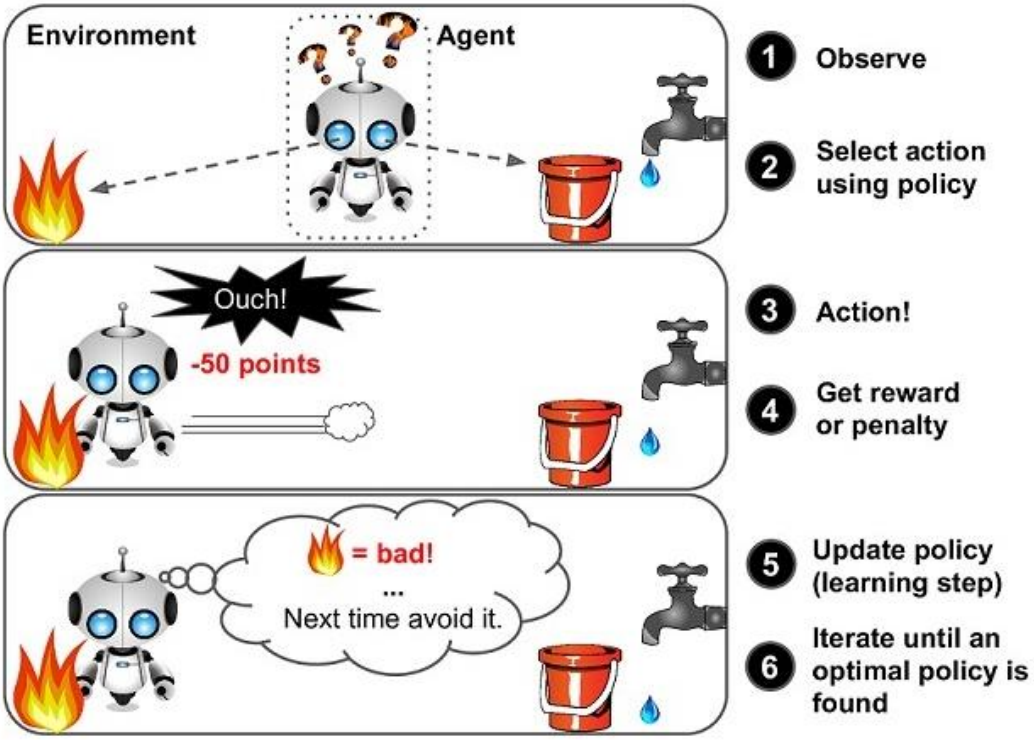

Figura 4: What is Reinforcement learning. *[4]*

Un ejemplo famoso de una IA implementando con este sistema, es Alpha Go que en mayo de 2017 venció al campeón mundial Ke Jie en el juego "Go". Sus creadores explicaron que esta IA analizo millones de partidas desarrollando y perfeccionando una política de acciones para ganar.

- **Online vs batch learning:** Este tipo de clasificación se basa en distinguir entre dos sistemas de aprendizaje, uno ***offline*** y otro ***online***, en el primero el sistema es entrenado utilizando todos los datos disponibles y una vez terminada su fase de aprendizaje es empleado directamente en la producción, de esta forma en caso de obtener nuevos datos y buscar reentrenarlo se necesitara volver a entrenarlo desde cero y por lo tanto sacar una nueva versión, tratándose por tanto de

una de las formas más utilizadas debido a su simplicidad. Sin embargo, está directamente limitado por el coste computacional y el tamaño de los datos que dispongamos, de forma que en determinados casos donde ambos sean notablemente grandes, el método offline será inviable por sus costes. En su contra parte, en "*online learning*", los datos de entrenamiento son agrupados en pequeños grupos de forma que se entrena el sistema de forma secuencial, de esta forma no es necesario cargar y procesar todos los datos al principio.

La ventaja principal de este sistema es que puede ser entrenada con nueva información sin necesidad de crear una nueva versión y reentrenarla, gracias a que se implementan algoritmos "out of core" que permiten extraer datos de sistemas de almacenamiento externo, convirtiéndose así en uno de los sistemas más utilizados en mercados financieros donde el sistema tiene que readaptarse continuamente a las nuevas condiciones. Sin embargo, esta situación hace que en caso de que los datos no sean los apropiados se pueda dañar el modelo y estropear el trabajo anteriormente hecho.

- Instance based vs model based: La agrupación de los algoritmos en uno de estos dos grupos se realiza en base a generalización que se obtiene de ellos, en el primero se compara los nuevos inputs con los inputs ya vistos en el entrenamiento para ver si hay similitudes y producir una solución, ha sido llamado de esta forma precisamente ya que construye las hipótesis directamente del entrenamiento, un ejemplo de este tipo de entrenamiento es k-nearest neightbor algorithm o kernel machines.

Por otro lado, nos encontramos con los algoritmos "*model based*", son el sistema más empleado en machine learning para obtener predicciones, se basa principalmente en el análisis de los datos, observando que variables pueden tener mayor correlación e impacto en la predicción y conforme a este análisis se desarrolla el modelo .Por ejemplo, en problemas de regresión si comparando dos variables, vemos una clara tendencia lineal, utilizaremos un algoritmo de regresión lineal con una función de coste para que el modelo reduzca el error conforme es entrenado, por ultimo si el error entra dentro de nuestros objetivo podremos utilizar el modelo desarrollado y entrenado para realizar predicciones, en caso de que este modelo no sea el apropiado deberemos reanalizar los datos y buscar nuevas variables que hayamos pasado por alto.

En el siguiente ejemplo vemos un posible modelo que analiza la felicidad en función del producto interior bruto con una regresión lineal.

$$life_{satisfaction} = \emptyset_1 + \emptyset_2 \cdot GDP \quad (1)$$

2.2 Training Data

Posiblemente uno de los pasos más importantes a la hora de desarrollar un modelo de inteligencia artificial, sea crear un training data adecuado, puesto que para que este cumpla con los objetivos propuestos se tendrá que sortear diferentes problemas que pueden hacer que el modelo no funcione correctamente.

Uno de los principales problemas a lo que hay que enfrentarse es no disponer de datos suficientes para entrenar nuestro modelo, este problema tiene difícilmente solución, sin embargo, en casos muy concretos es posible generar artificialmente datos propios haciendo duplicados con una ligera variación. Por otro lado, otro problema importante es que los datos que disponibles no sean representativos para el modelo que se desea crear, en consecuencia, las predicciones podrían perder precisión, para ello, se debe decidir previamente que tipo de variables serán empleadas que puedan estar inferidas en los datos elegidos. Otro problema similar ocurre cuando nuestros datos tienen una calidad pobre, debido a que hayan podido estar

sometidos a medidas erróneas, ruido, etc. Por ende, será esencial en la medida de lo posible tratarlos y filtrarlos como sea posible, hasta alcanzar una calidad óptima.

Finalmente, cuando se dispongan ya de los datos necesarios para entrenar el modelo, pueden surgir dos complicaciones principales durante el entrenamiento:

a) **Overfitting**: Es un problema muy común que surge cuando el modelo es entrenado de forma redundante y es incapaz finalmente de generalizar correctamente. No necesariamente puede ser debido a que los datos sean excesivamente repetitivos o el training data sea demasiado largo, sino incluso al tipo de arquitectura que se haya creado, una red neuronal con excesivas neuronas puede llegar a causar también este efecto. Una forma muy efectiva para evitar que suceda este problema, es aplicar técnicas de regularización como Lasso, Ridege, etc que permitan mantener nuestro modelo lo más simple posible.

b) **UnderFitting**: Tal y como se puede intuir, se trata del problema opuesto, ocurre cuando el modelo es demasiado simple y no dispone del tamaño suficiente para inferir el patrón ya sea porque la arquitectura de la red, modelo o tamaño del training data es insuficiente.

Para evaluar si está ocurriendo alguno de estos problemas en el modelo no es necesario esperar al test final, sino que a medida que se entrena, se puede utilizar una pequeña porción de los datos que no hayan sido empleados en el entrenamiento del actual modelo y así examinar si se está produciendo realmente un aprendizaje o mejora de la precisión. En adición, otra técnica algo más compleja pero muy aceptada para examinar la calidad del modelo y sus predicciones es la técnica de "**cross-validation**", básicamente funciona dividiendo los datos en diferentes fragmentos y se entrena y testea cada vez con combinaciones diferentes de estos, con el fin de ver si la precisión aumenta según se entrena.

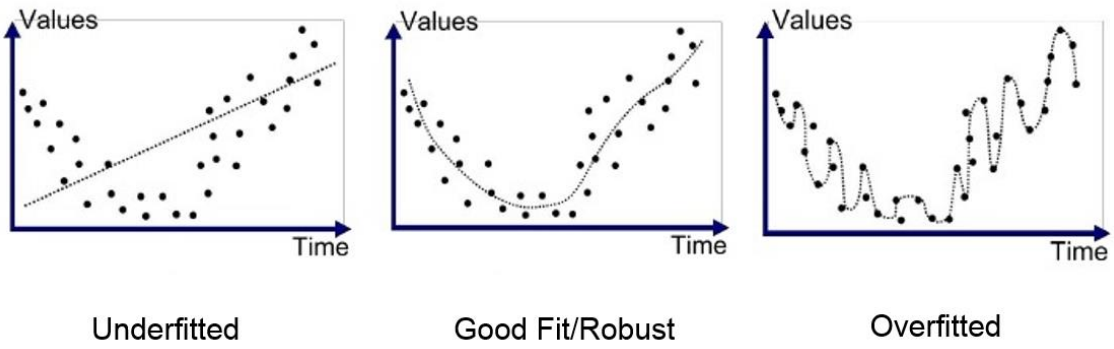

Figura 5: OverFitting vs UnderFitting *[5]*

2.3 Machine Learning Models

En este apartado se explicarán los diferentes algoritmos empleados en el aprendizaje automático y que posiblemente serán empleados en este proyecto.

Regresión Lineal:

Este algoritmo es uno de los más simples que se pueden encontrar, permite realizar predicciones simplemente computando el peso de algunos de sus inputs más una constante llamada bias, dependiendo del número de variables se pueden clasificar en dos tipos básicos; simple si solo presenta una variable dependiente y múltiple si presenta 2 o más.

$$\hat{y} = \emptyset_0 + \emptyset_1 x_1 + \emptyset_2 x_2 + \cdots + \emptyset_n x_n + \varepsilon \text{ (2)}$$

Donde $\hat{y}$ es el valor predicho, n es el numero de variables, x_i el input, $\emptyset_i$ es el peso incluyendo el bias $\emptyset_0$,por último una variable aleatoria que representa el error.

Para encontrar el valor $\emptyset$ que permite calcular tanto el bias como la pendiente, se utiliza la ecuación Normal, que no es más que la derivada del coste de la función.

Como se puede observar se incluye un 2 a la función MSE, para que a la hora de derivar el exponente sea cancelado.

$$J(\emptyset) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i) \text{ (3)}$$

$$\frac{dJ}{d\emptyset} = \widehat{\emptyset} \text{ (4)}$$

$$\widehat{\emptyset} = (X^T \cdot X)^{-1} \cdot X^T y \text{ (5)}$$

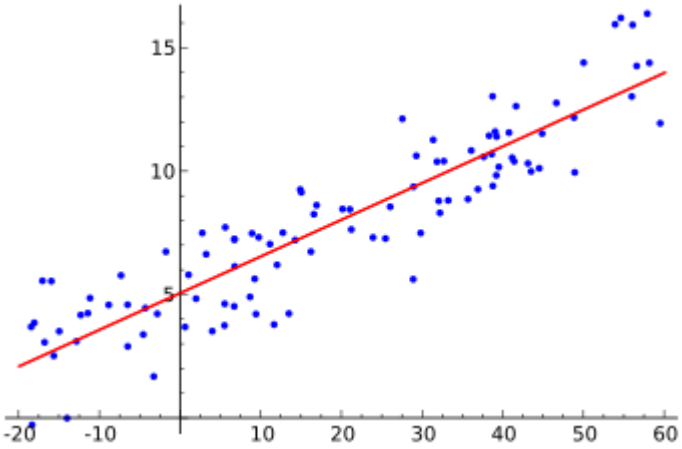

Figura 6: Modelo regresivo lineal

Gradiente Descendente y otras variantes:

Posiblemente uno de los mejores algoritmos a la hora de encontrar soluciones de forma óptima a una gran amplitud de problemas. El algoritmo reduce el error de forma iterativa en base a un "learning rate" hasta alcanzar el error mínimo objetivo, más tarde en el punto 3.2 de este trabajo se explicará cómo calcularlo. Por otro lado, el algoritmo en este caso actúa sobre todos los datos, es decir cuando se habla de Batch se refiere a la totalidad del training data, sin embargo, si se divide el conjunto de datos en fragmentos se hablará por lo tanto de mini-batch. Por último, para medir el error se utiliza normalmente una función de coste como el error cuadrático medio (MSE) al tratarse al fin y al cabo de un problema de regresión.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \text{ (6)}$$

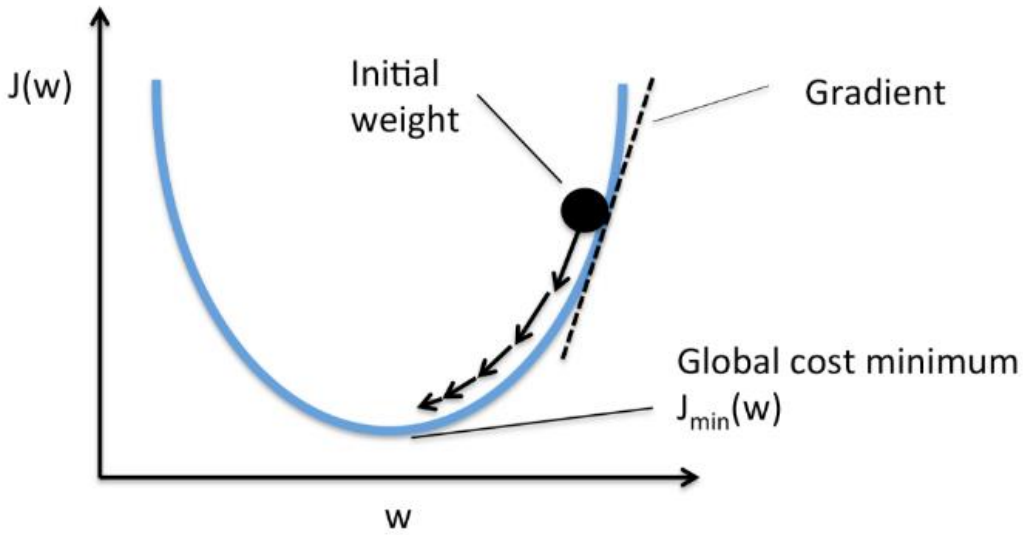

Figura 7: Gradiente descendente

Sobre la configuración de la red, no hay una fórmula que permita obtener los máximos resultados, siempre que se respete la lógica anteriormente expuesta se basa en ir realizando un testeo rápido de un par de muestras y ver como evoluciona el error, seleccionando la mejor configuración posible.

Otro algoritmo derivado del gradiente descendente que también es posible implementarlo en redes neuronales, es el sistema de entrenamiento "Adam" o Short Adaptative Moment Estimation, una variante del RMSProp. Se basa en que los pesos en ved de ser adaptados en base a un "learning rate" en concreto como en el gradiente descendiente, en Adam, el valor del learning rate, se adapta utilizando la media exponencial de los momentos del gradiente.

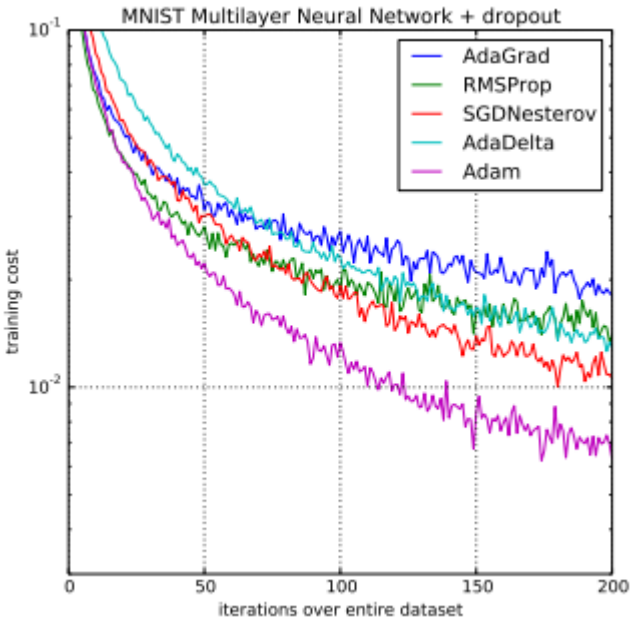

Figura 8: Comparación de diferentes algoritmos de aprendizaje [6]

En la figura 8, se puede observar como el algoritmo de entrenamiento funciona mucho mejor que las otras variantes del gradiente descendiente.

Gradiente Descendente Estocástico (SGD):

El principal problema al que nos enfrentamos con el gradiente descendente es la necesidad de aplicarlo en cada una de las iteraciones, implicando por ende un coste computacional muy alto. Con el fin de solventar este problema, se creó este nuevo algoritmo (SGD) que permite seleccionar al azar el número de muestras que empleara para el entrenamiento del modelo, una forma similar de implementarlo es organizar al azar las muestras del training set.
Por otro lado, el "learning rate" no tiene un valor fijo, sino que a medida que se va iterando este se va reduciendo para converger y no ir rebotando por la curva de mínimos.

Gradiente Descendente con Mini-Batch:

Este nuevo algoritmo permite colocar un mini-batch o grupo de muestras permitiendo calcular el error por pequeños grupos reduciendo significativamente el tiempo de cómputo y memoria necesaria para procesar todo el dataset.

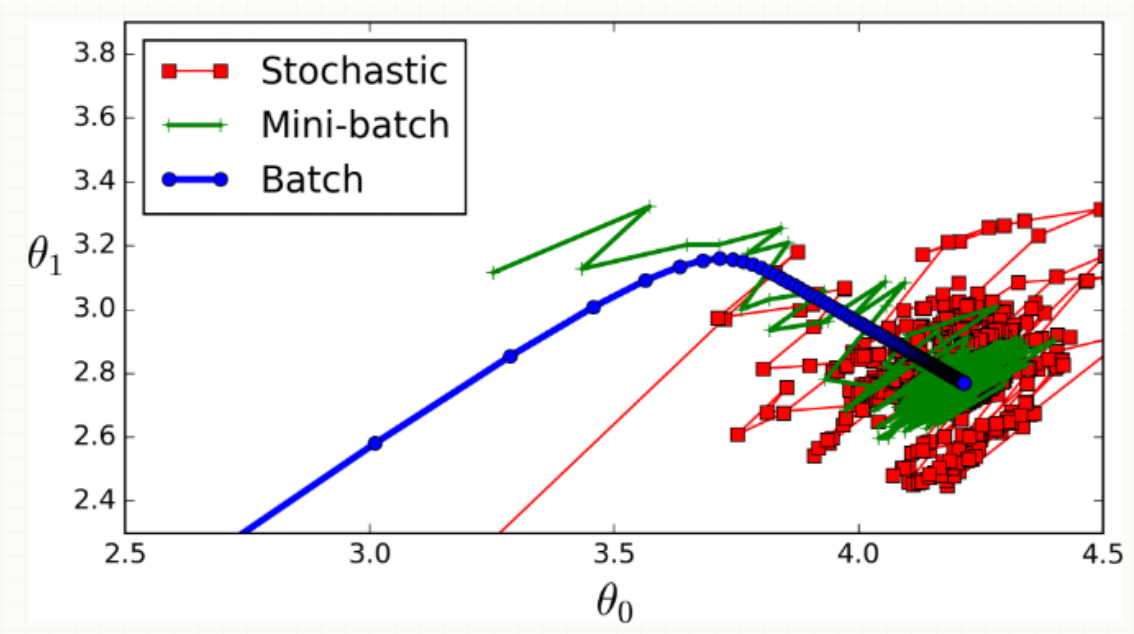

Figura 9: Comparación de SGD, mini-batch y un único batch *[7]*

Regresión Polinomial o no lineal:

Cuando en el modelo actúan datos no lineales, el sistema de regresión lineal no será capaz de funcionar correctamente, por ello se desarrollará un modelo basado en variables no lineales que permitan captar esta complejidad. Sin embargo, hay que actuar con mucha cautela, porque ciertos sistemas aparentemente no lineales pueden ser transformados matemáticamente y operar como lineales, un ejemplo:

$$y = ae^{bx} \quad (7)$$

$$\ln(y) = \ln(a) + bx \quad (8)$$

La fórmula general empleada para un modelo no lineal de cualquier grado es:

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_0 x^n \quad (9)$$

Logistic Regression o Sigmoid Function:

Algunos algoritmos pueden ser utilizados como clasificadores, la regresión logística permite a través de unos inputs pasados por una serie de pesos realizar una clasificación binaria, si el resultado es mayor que 0.5 se considera identificado "1", sino se consideraría rechazado "0".

$$\hat{p} = \sigma(\theta^T x) \quad (10)$$

En concreto, σ, este símbolo llamado logic denota la función logística siguiente:

$$\sigma = \frac{1}{1+\exp(-t)} \quad (11)$$

$$\hat{y} = \begin{cases} 0 \; if \; \hat{p} < 0.5 \\ 1 \; if \; \hat{p} \geq 0.5 \end{cases} \quad (12)$$

Softmax Regression:

Se trata de una generalización de la regresión logística, permite realizar clasificaciones sobre k clases.

$$S_k(x) = (\theta^K)^T x \quad (13)$$

$$\widehat{p_k} = \sigma\big(S_k(x)\big) = \frac{\exp(S_k(x))}{\sum_{j=1}^{k} \exp(S_j(x))} \quad (14)$$

$$\hat{y} = argmax(S_k(x)) \quad (15)$$

Categorical Cross entropy:

Se usa con frecuencia para medir con que precisión una lista de clases estimadas cuadran con sus respectivas clases reales.

$$J(\emptyset) = -\frac{1}{m} \sum_{1=1}^{m} \sum_{k=1}^{k} {y_k}^i \; \log({\hat{p}_k}^i) \quad (16)$$

Support Vector Machines SVM:

Posiblemente uno de los modelos ML más empleados, capaz de realizar clasificaciones o regresiones lineales o no lineales e incluso detección de patrones anómalos al resto.

1- Classification: Permite establecer mediante unos vectores la separación de los datos que siguen un determinado patrón detectado. Es importante señalar que al tratarse de vectores una correcta normalización o escalado de los datos permitirá separar con mayor precisión las diferentes clases.

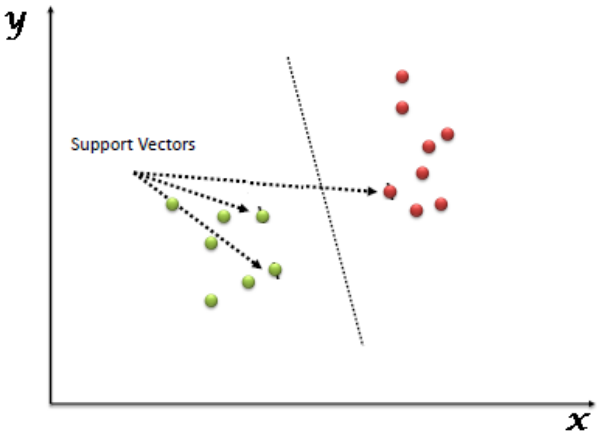

Figura 10: SVM vectors *[8]*

a) Lineal o Soft Margin Classification: aunque puede ser empleado en algunos casos concretos no lineales, la realidad es que solamente es útil en clasificaciones lineales y con ciertas limitaciones. Se basa en reducir o aumentar el margen entre los vectores de soporte y el vector principal, de esta forma ajustando el valor C podemos obtener un margen mayor y al contrario aumentando C podemos reducirlo. Normalmente si tenemos un modelo que está sufriendo overfitting se tendera a reducir C para ver si es capaz de generalizar mejor y lo mismo a la inversa.

b) NonLinear SVM Classification: De forma similar a lo explicado anteriormente en modelos no lineales polinomiales, normalmente podremos clasificar estos patrones no lineales empleando una transformación polinómica seguido de un escalado y un SVC lineal, pero a veces no es tan sencillo y será necesario emplear otras técnicas.

   a) Polinomial Kernel: Si nuestro dataset es lo suficientemente complejo será necesario incluir un modelo polinomial de mayor grado haciendo que este modelo sea mucho más lento a la hora de entrenarlo. Para solventar este problema podremos utilizar una aproximación, llamada "Kernel Trick", permite transformar el espacio dado en otro de mayores dimensiones. Para poder comprenderlo de forma sencilla, se ha propuesto la siguiente función:

$$x_1^2 x_2^2 + y_1^2 y_1^2 + 2x_1 y_1 * x_2 y_2 \quad (17)$$

   Obteniendo dos puntos en su espacio al separar las variables S(P1,P2):

$$S(P1) = \left(x_1^2, y_1^2, \sqrt{2} x_1 y_1\right) \quad (18)$$

$$Kernel\ Trick = K(x, y) = (x^2, y^2, \sqrt{2} xy) \quad (19)$$

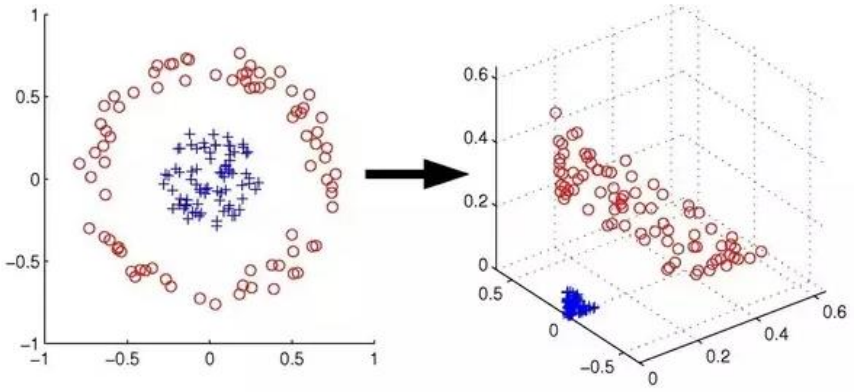

Figura 11: Transforming the data can make it linearly separable *[9]*

2.4 Minera de datos:

Se llama minería de datos al proceso general de la extracción de un patrón en un volumen de datos disponible mediante una serie de pasos:

- Selección del conjunto de datos: selección de las variables independientes y dependientes del proceso.
- Análisis de las propiedades de los datos: análisis de dispersión, valores atípicos, histogramas, etc.
- Transformación del conjunto de datos de entrada: se aplicarán las transformaciones necesarias para que los datos se puedan emplear de la mejor forma posible con la técnica seleccionada.
- Selección del modelo: se diseña y construye el modelo regresivo o clasificatorio.
- Extracción e interpretación de lo datos y evaluación: Una vez obtenidos los resultados con el modelo, se evalúan e interpretan para ver si el modelo cumple con los objetivos marcados, si no se cumplen las expectativas es necesario realizar un cambio de paradigma.

Este proceso conlleva cerca del 80% del tiempo empleado por el científico de datos a la hora de construir el modelo, tal y como se explica en *Dasu and Johnson, 2003*[17].
Para transformar el conjunto de datos y adaptarlos a las necesidades de las funciones de activación se emplean principalmente los dos siguientes métodos:

2.4.1 Estandarización o variable centrada reducida/ unidad tipificada.

Se basa en centrar y reducir las variables empleadas en el modelo de forma que se puedan realizar comparaciones entre ellas, se consigue restando su media a cada uno de los valores que toma la variable y reduciéndola, dividiendo por su desviación típica.

$$z = \frac{x - \bar{a}}{s} \quad (20)$$

2.4.2 Normalización.

Se basa en colocar todos los datos de una variable dentro del rango óptimo para ser utilizado dentro del modelo, por ejemplo, en el caso de algunas funciones de activación como la logística, se necesita que estos valores estén entre 0 y 1. Por consiguiente la formula óptima para normalizar dentro del rango es la siguiente:

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (21)$$

La principal desventaja de la normalización vs estandarización es que cuando se aplica a un rango de datos perdemos información sobre su distribución, sobre todo debido a los valores atípicos, causando que al usar el gradiente descendiente a este le cueste más converger. Sera necesario estudiarlos y ver si realmente deben de ser eliminados del dataset.

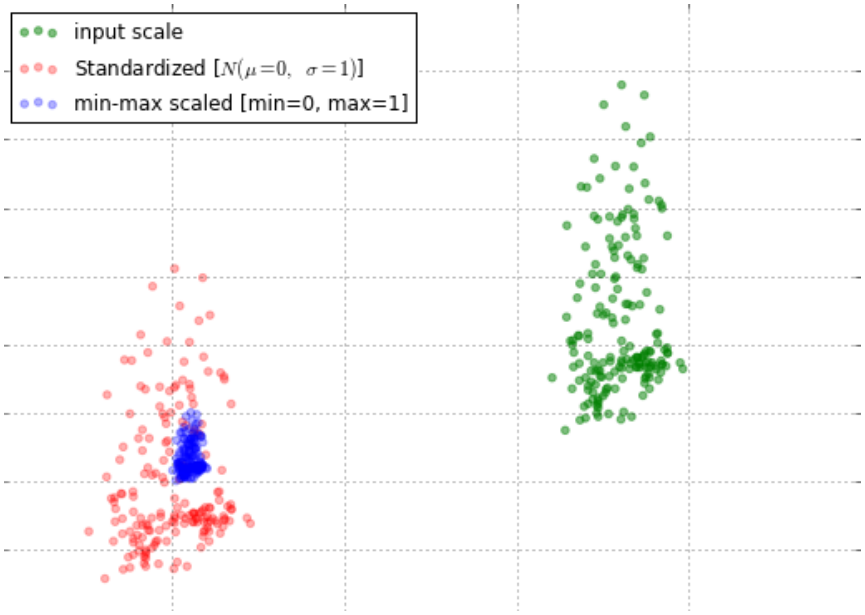

Figura 12: Normalization vs Standarization

2.5 Evaluación del modelo

Para realizar la evaluación de un modelo se deberá distinguir generalmente entre dos grandes tipos: los modelos regresivos y los modelos clasificadores. Las métricas utilizadas en estos para analizar la calidad de sus predicciones pueden variar siendo algo más simple en los clasificadores, pero generalmente algunas serán comunes y compartidas.

Comunes:

a) Cross-Validation: En esta técnica se guarda una pequeña porción de los datos para evaluar si el modelo está realmente reduciendo su error mientras es entrenado, el modelo no debe usar estos inputs en su entrenamiento nunca, sino el resultado estaría siendo adulterado.

Regresión:

a) Error Medio Absolute (MAE): Mide el error medio en toda la muestra de predicciones.

$$MAE = \frac{\sum_1^n |y_i - \hat{y}_i|}{n} \quad (22)$$

b) Error Cuadrático Medio (RMSE): Mide la variación del estimador, es la raíz cuadrada de la varianza conocida también por desviación estándar, en definitiva, mide el grado de dispersión de los datos respecto al valor promedio.

$$MSE = \frac{\sum_1^n (\hat{y}_i - y_i)^2}{n} \quad (23)$$

c) Error Cuadrático Medio Logarítmico (RMSLE): Empleado para ver la diferencia en forma de ratio, penaliza en mayor medida las ratios grandes producidos por diferencias pequeñas.

$$RMSLE = \log\frac{\hat{y}_i + 1}{y_i + 1} \quad (24)$$

d) Mean Absolute Percentage Error (MAPE): Permite analizar el % de error total, como se puede observar en la formula, si hay algún valor real normalizado a 0, el MAPE daría infinito.

$$MAPE = \frac{100\%}{n}\sum_{1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \quad (25)$$

e) Time Series Cross-Validation: Se divide todo el dataset en un conjunto de observaciones, el conjunto de entrenamiento se basa en observaciones que ocurrieron siempre antes del conjunto utilizado para test, estas se van desplazando en orden cronológico hacia el futuro, tal y como se explica muy bien en la siguiente imagen.

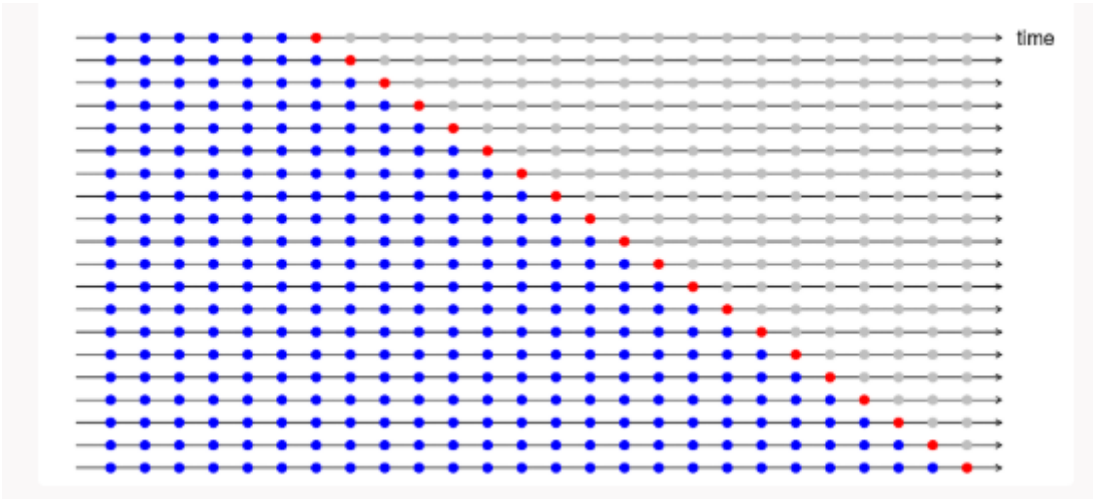

Figure 13: Cross Validation for time series. *[10]*

La precisión global del modelo se computa en base a la media obtenida del conjunto de test realizados.

Clasificadores:

a) Matriz de Confusión: Posiblemente el método más utilizado a la hora de analizar los resultados obtenidos. Se trata de una matriz donde cada columna representa el número de clases predichas y su conteo y las filas representan las clases reales. Podemos observar como el eje "y" mide la precisión y el eje "x" mide la sensibilidad.

		Clase Predicha	
Clases Real		Clase Referencia	Clase No Referencia
	Clase Referencia	TP	FP
	Clase No Referencia	FN	TN

Donde un TP o Verdadero positivo significa correctamente identificado, FP o falso positivo es incorrectamente identificado, TN o verdadero negativo es correctamente rechazado y por último falso negativo es incorrectamente rechazado.

b) Precisión: Es la forma más concreta de analizar la capacidad de predicción del modelo, a veces también se llama especifidad o True Negative Rate (TNR).

$$precision = \frac{TP}{TP + FP} \quad (26)$$

c) Recall: También llamada sensibilidad o True Positive Rate (TPR), mide el número de predicciones que han sido correctamente detectadas

$$recall = \frac{TP}{TP + FN} \quad (27)$$

d) F1 Score: Se trata de una métrica donde se puede comparar directamente la precisión y sensibilidad del modelo, está construido en base a realizar una media harmónica o también llamada "subcontrary mean", de esta forma solo si ambas métricas son altas se podra obtener un F1 alto.

$$F_1 = 2 \cdot \frac{precision\ x\ recall}{precision + recall} \quad (28)$$

Sin embargo como se puede intuir, este indicador favorece a modelos con un ratio de precisión y sensibilidad similares, sin embargo en la vida real habrá situaciones donde necesites favorecer la precisión en contra posición a la sensibilidad, un ejemplo podría ser, el desarrollo de un clasificador para una enfermedad coronaria donde se daria prioridad a tener una sensibilidad alta aunque quizás se tenga una precisión baja y al revés por ejemplo en la detección de contenido apto para niños.

e) FPR or False Positive Rate:

$$FPR = \frac{FP}{FP + TN} \quad (29)$$

f) Accuracy or ACC:

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \quad (30)$$

g) Curva ROC: Se trata de una representación gráfica de la sensibilidad frente a la tasa de falsos positivos. Por lo tanto, podemos observar en la gráfica que a mayor ratio de TPR y menor ratio de FPR nuestro modelo será predictivamente mejor, encontrando el modelo perfecto en la coordenada (0,1).

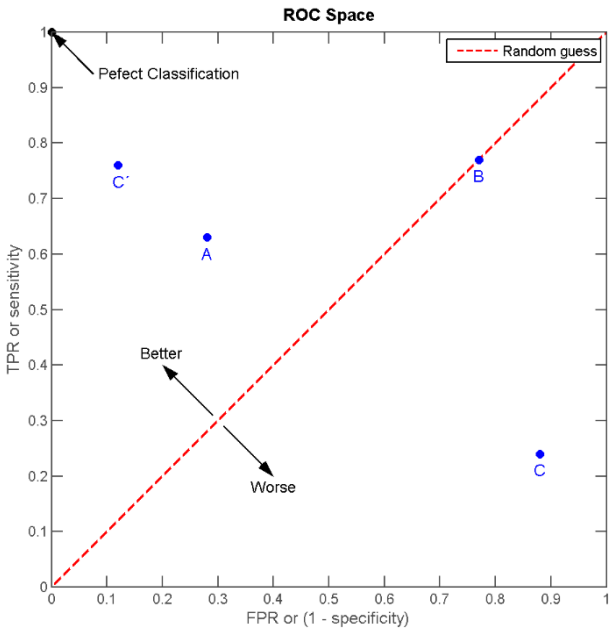

Figure 14: Curva ROC *[11]*

Ratios de riesgo para evaluar portafolios:

a) Sharpe: El ratio Sharpe se basa en el retorno medio obtenido del total de diferencias del retorno del portafolio contra un retorno libre de riesgo como puede ser el bono del tesoro estadounidense T-BILL a 180 días, posteriormente dividido por la desviación típica del portafolio. Cuanto mayor sea el ratio, más atractivo resulta la inversión.

$$Sharpe\ Ratio = \frac{r_p - r_f}{\sigma_p} \quad (31)$$

Capítulo 3. Redes Neuronales

Las redes neuronales toman su inspiración en el comportamiento del cerebro humano, a pesar de que estas no lo representan como tal, ya que aún no se conoce su funcionamiento por completo, sin embargo, sí que utiliza la idea de un conjunto de neuronas conectadas que aplican un determinado algoritmo del cual se puede extraer a partir de unos inputs una serie de outputs o conclusiones.

Estas neuronas son organizadas por capas y a su vez están conectadas con unos pesos que simbolizan las sinapsis de un cerebro real, de la forma que los pesos ajustan la fuerza de cada conexión.

La investigación en el campo de las redes neuronales se inició en torno a 1940 con la hipótesis del doctor Hebb sobre el mecanismo neuronal de plasticidad conocido como "Hebbian learning". Farley y clark trataron de simularla en una maquina en el año 1956.

A esta gran fecha le siguieron otras también muy importante como la creación del perceptrón en 1958 un tipo básico de red neuronal, unos años después, en 1959 se descubrió dos tipos de células del cortex visual, las células simples y complejas, que supusieron la base para lo que fue más tarde las redes convolucionales.

Como podemos observar gran parte de la teoría en la que se sustentan las redes neuronales ya había sido creada, pero había un problema principal, la tecnología que había hasta el momento era insuficiente para producir toda la capacidad necesaria de cómputo para poder entrenar redes neuronales complejas, por lo tanto la idea de redes neuronales fue dejada a un lado y se centraron los esfuerzos en desarrollar modelos basados en machine learning, o algoritmos desarrollados por expertos siguiendo operaciones condicionales if-else.

3.1 Tipos de redes neuronales

En este punto, se explicará que tipos de redes neuronales hay y cuáles son sus principales funciones:

a) Perceptrón: Se trata del tipo de red neuronal más básica de todas, que permite realizar una generalización lineal de un patrón. El algoritmo se encuentra limitado a la expresamente a problemas lineales, como puertas lógicas or, and, etc, por lo tanto, sería incapaz de resolver un problema no lineal como la puerta lógica x-or.
   Por otro lado, no utiliza algoritmos que empleen técnicas de derivación como la regla delta para actualizar los pesos en base al error estimado, esto hace que el aprendizaje sea mucho más largo.

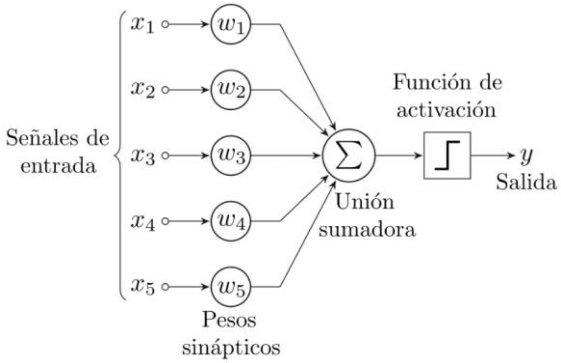

Figura 15: Perceptrón simple.

a) Adaptative Linear Element (Adaline): Se trata de una red neuronal mono capa, similar al perceptrón utilizado de la misma forma en problemas lineales, la gran diferencia radica en que en las redes Adaline, sí emplean a la hora de actualizar los pesos de las conexiones neuronales, el algoritmo de descenso del gradiente que permite mediante la diferencia de error entre el valor estimado y esperado, reducir iterativamente esta diferencia.

b) Red Neuronal Feedforward (FNN): Se trata de la red neuronal más utilizada posiblemente de todas, debido a su efectividad y simplicidad. A diferencia de las redes neuronales recurrentes, en las neuronas de la FNN no se establece ningún ciclo recursivo, sino que flujo va continuamente hacia delante. Dentro de este tipo de redes se pueden encontrar el perceptrón y su forma no lineal, el perceptrón multicapa MPL.

c) Red Neuronal Recurrente: (RNN): En este tipo de red neuronal las neuronas guardan un estado interno (memoria) que les permite procesar los inputs utilizando discriminatoriamente el pasado. De esta forma, son increíblemente útiles para realizar reconocimiento de voz o reconocimiento de handwriting.

d) Red Neuronal Convolucional: Este tipo de red basada en el comportamiento de las neuronas del córtex visual, permite realizar tareas de detección de objetos o clasificación de imágenes, son posiblemente las que actualmente están ofreciendo mejores resultados en este cometido.

e) Autoencoder: Se trata de un tipo de red artificial con aprendizaje no supervisado. El objetivo de este tipo de redes es reducir la dimensionalidad y representarla de esta nueva forma ya reducida produciendo un nuevo tipo de input data llamada “codings”.
Pese a que no se van a emplear en este trabajo, sí que merece la pena nombrar sus usos potenciales más allá de reducir la dimensionalidad, han permitido detectar nuevas variables en los datos y pre-entrenar la red, además permiten también generar nuevos datos muy similares a los originales, a partir de esta nueva representación.

3.2 Arquitectura de una Red Neuronal FNN

Este tipo de red neuronal es una de las más extendidas en uso, pero no será empleada en el proyecto, simplemente servirá como introducción a futuras topologías de otros tipos de redes más complejas como las recurrentes (LSTM) o convolucionales.

Las FNN multicapa están compuestas por diferentes capas que permiten inferir patrones no lineales frente a sus versiones simples monocapa, de esta forma se tiene:

1-Input Layer: En este nivel se alimenta a la red neuronal con los inputs, es muy importante que dependiendo la función de activación de la neurona estén los datos correctamente tratados y normalizados.

2- Hidden Layer: Compuesta por una o más capas escondidas, con diferentes neuronas desde donde se computarán los pesos por el valor de salida obtenido en la capa anterior y se filtrarán por la función de activación seleccionada para estas neuronas.

3- Output Layer: Se trata de la última capa, donde se reciben todas las salidas de cada una de las neuronas de la última Hidden Layer, filtrando una última vez se obtiene la salida.

4- Error: Una vez obtenido un valor estimado (valor de salida) y utilizando el valor esperado podremos aplicar el algoritmo "backpropagation" que se explicara más adelante, que permite el reajuste de los pesos y por ende aprendizaje de la red.
Este algoritmo será aplicado en cada iteración, obteniendo un aprendizaje iterativo.

5- Funciones de Activación: Permite definir la salida en función de una entrada dada o un conjunto de entradas, se llama precisamente activación porque dependiendo de esta entrada decide activar o no a la neurona viéndose representado este "estado" en su valor de salida.
Entre las principales funciones de activación se pueden encontrar funciones ya explicadas anteriormente como la función logística y softmax, pero también es posible encontrar otra función muy utilizada, como la tangente hiperbólica:

5.1-Tangente hiperbólica: Permite computar valores negativos y obtener así salidas entre -1 y 1.

$$g_{tanh}(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \quad (32)$$

3.3 Backpropagation

Se trata del principal algoritmo de aprendizaje empleado que sirve para entrenar a la red neuronal, aunque hoy en dia ya no es necesario programar la lógica que hay detrás para implementarlo gracias a la gran cantidad de APIs ya creadas como Keras o Tensorflow, si es esencial entender bien la matemática que justifica el proceso de aprendizaje o inferencia de conocimiento de la red, ya que en muchos otros casos es utilizada.

Los conocimientos previos necesarios para poder entenderla son: conocer el gradiente de un vector y saber derivar correctamente con la regla de la cadena.

-Gradiente de un vector: Se define como el campo vectorial cuyas coordenadas son las derivadas parciales del campo escalar.

$$\nabla \mathrm{f} = \left(\frac{\partial f(r)}{\partial x1}, \ldots, \frac{\partial f(r)}{\partial xn}\right) \quad (33)$$

Para una correcta explicación se ha decidido explicar paso a paso 1 ciclo de entrenamiento o 1 epoch.

En este ejemplo [6] se dispone de una red neuronal básica compuesta por una primera capa con 2 inputs, seguido de una capa oculta o también llamada Hidden Layer de 2 neuronas y por último una capa de salida con otras dos neuronas. Todas las neuronas de cada capa están conectadas con la anterior, de forma que se crea una malla de conexiones donde cada neurona recibe las salidas de la anterior multiplicada por los pesos de las conexiones o sinapsis.

Paso1: Inicialización.

Se inicializan las conexiones con un número aleatorio y valor de bias como 1.

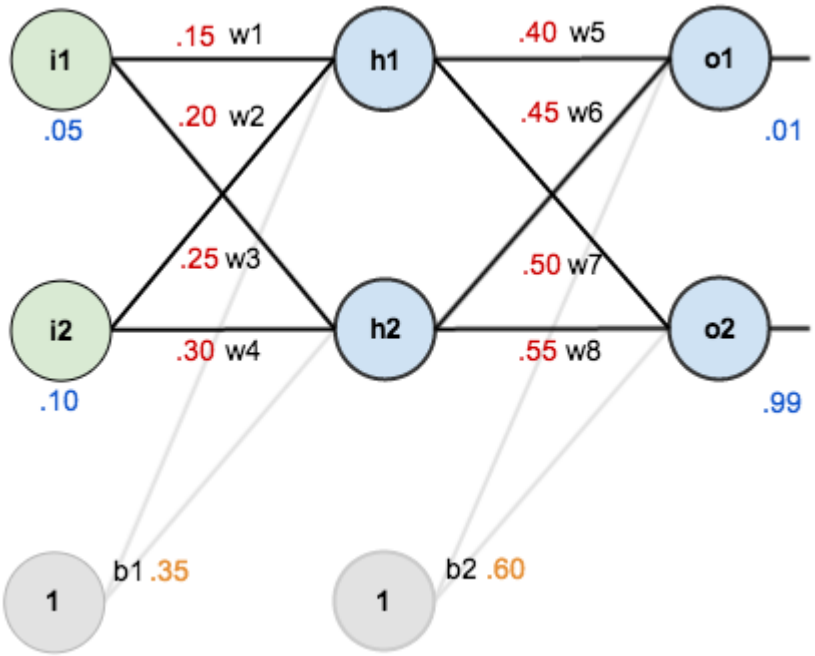

Figura 16: Red neuronal multi capa ForwardPass. *[12]*

Paso 2: Forward Pass

Se calcula tanto el valor a la entrada de la neurona como a la salida después de pasar por la función de activación.

$$neth1 = w1 * i1 + w2 * i2 + b1 * 1 \quad (34)$$

$$outh1 = \frac{1}{1 + e^{-neth1}} \quad (35)$$

Y asi sucesivamente con cada neurona de la capa oculta que conecta con los inputs, el siguiente paso es idéntico, se trata de calcular el valor de entrada y salida pero esta vez de la Output Layer, para ello se aplicaran las siguientes formulas:

$$netO1 = w5 * outh1 + w6 * outh2 + b2 * 1 \quad (36)$$

$$outo1 = \frac{1}{1 + e^{-neto1}} \quad (37)$$

Paso 3: Calculando el error total.

El error permitirá saber cómo de acertadas son las previsiones y a la vez será empleado en el gradiente descendiente para poder reducir así el error de las siguientes iteraciones. Se calcula el error total con la siguiente formula:

$$E_{total} = \frac{1}{2}\sum(target - output)^2 \quad (38)$$

Paso 3: Backward Pass.

En esta parte se ejecuta el algoritmo “Backpropagation” que permitirá ajustar los pesos en función del error de cada iteración.

Tal y como se aprecia en la gráfica este algoritmo utiliza el gradiente descendiente que es necesario para reajustar los pesos de los enlaces de las neuronas con el fin de reducir el error total.

Este método clasificado como uno de los métodos utilizados en la diferenciación automática (AD) que permite evaluar la derivada de la función implementada por un programa, por lo tanto, aplicando la regla de la cadena repetidamente a las diferentes operaciones realizadas podemos ir consiguiendo mayor precisión.

Regla de la cadena:

$$\frac{dg}{dx} = \frac{dg}{df}\frac{df}{dx} \quad (39)$$

Gradiente:

$$\nabla f = \frac{df}{dx}i + \frac{df}{dy}j + \frac{df}{dz}k \quad (40)$$

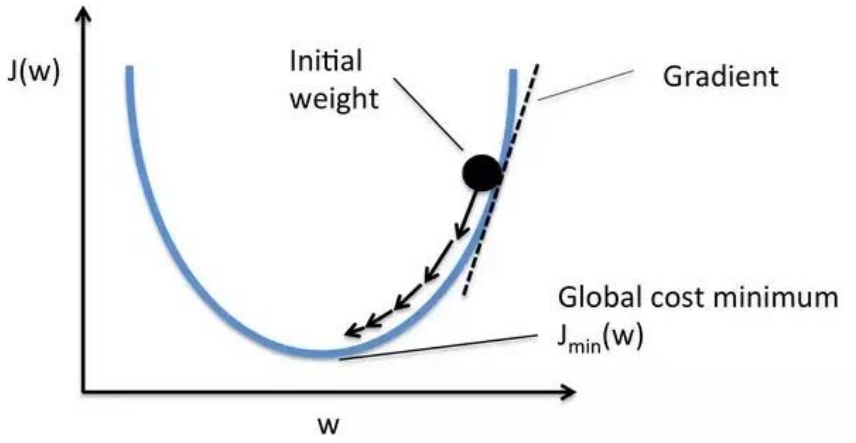

Figura 17: Gradiente Descendente. [12]

Utilizando el ejemplo propuesto anteriormente vamos a aplicar el algoritmo por cada camino de cada neurona de cada capa:

$$\frac{dE_{total}}{dw6} = \frac{dE_{total}}{dout_{o1}} \cdot \frac{dout_{o1}}{dnet_{o1}} \cdot \frac{dnet_{o1}}{dw6} \quad (41)$$

Derivando cada elemento en función de la variable del denominador podemos obtener el error total derivado en función del peso del camino, luego solo quedaría actualizarlo según el learning rate utilizado (LR).

$$w6 = w6 - LR \cdot \frac{dE_{total}}{dw6} \quad (42)$$

En las siguientes capas es ligeramente diferente, ya que el error estaría acumulado por las salidas de todas las neuronas de la output layer, tal y como podemos observar en la siguiente imagen.

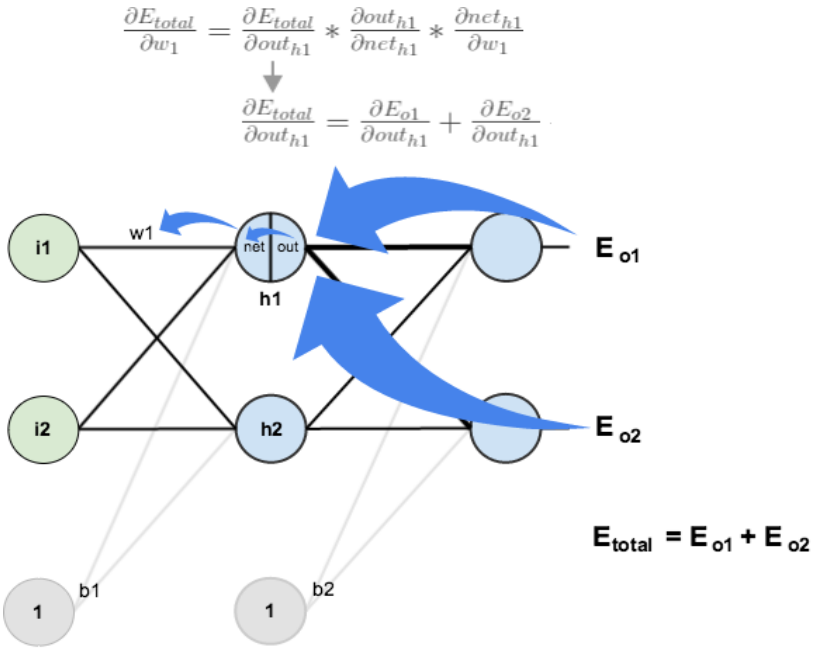

Figura 18: Red neuronal multi capa ForwardPass *[12]*

Como se puede observar no es un algoritmo excesivamente complejo, pero es tremendamente efectivo. A medida que se realizan iteraciones se reduce el error. El problema de este algoritmo y de otros similares está en encontrar realmente el mínimo absoluto, algo bastante difícil y complejo ya que para ello se deberá sortear los mínimos relativos que se irán obteniendo, ¿Entonces cómo se puede alcanzar un mínimo absoluto? Se emplean diferentes técnicas, que seran explicadas en el último punto de este trabajo, pero la más efectiva es dotar a cada neurona una probabilidad "p" de ser desechada "drop-out", de esta forma es más difícil que el error alcance un mínimo relativo y no sea capaz de seguir convergiendo, ya que la arquitectura sufrirá continuamente pequeñas variaciones que permitirá su convergencia, de esta forma el error se irá reduciendo y sorteando los diferentes mínimos relativos.

Capítulo 4. Redes neuronales convoluciones e Inteligencia Visual

En las CNN se aplica una filosofía similar a las FNN, este tipo de redes encuentran su inspiración en el cerebro, concretamente en el cortex visual, donde este tipo de neuronas se activan si el patrón de entrada responde al patrón almacenado en ellas, pueden albergar patrones tanto simples como complejos; líneas, bordes, etc. Gracias al experimento realizado por Hubel y Wiesel en 1962 *[15]* pudo comprobarse este proceso de activación de neuronas a determinados estímulos. Estas capas se van organizando en complejidad, es decir van construyendo e interpretando el estímulo capa por capa.

Convolutional Layer

Debido a la elevada complejidad y al gasto de recursos de tener que computar cada pixel de una imagen con su neurona correspondiente, se diseñaron las capas convoluciones, el proceso se basa en tomar una imagen con una resolución determinada (height, width, number of color channels) y procesar cada pixel con el peso del filtro correspondiente sin necesidad de conectar los pixeles a cada neurona de la capa oculta como podría ser en una NN ForwardPass.
Se delimitan pequeñas regiones de inputs, llamadas "receptive fields" que se conectaran a un *filtro* o *kernel*, este filtro estará formado por una matriz de valores o pesos que detectaran una determinado feature, este filtro ha de tener la misma profundidad que la entrada, normalmente el número de canales de colores codificado, a su vez el filtro recorrerá toda la imagen mediante un determinado paso de columnas conocido como "stride length".
Pongamos un ejemplo, imaginemos una imagen de 32x32x3 y queremos aplicar 6 filtros de 5x5x3, en total realizaremos 28x28 posiciones únicas (32-5+1) para cada uno de los 6 filtros. El resultado de la convolución entre el filtro y los diferentes receptive fields surgidos del barrido de la imagen se denomina "feature map".

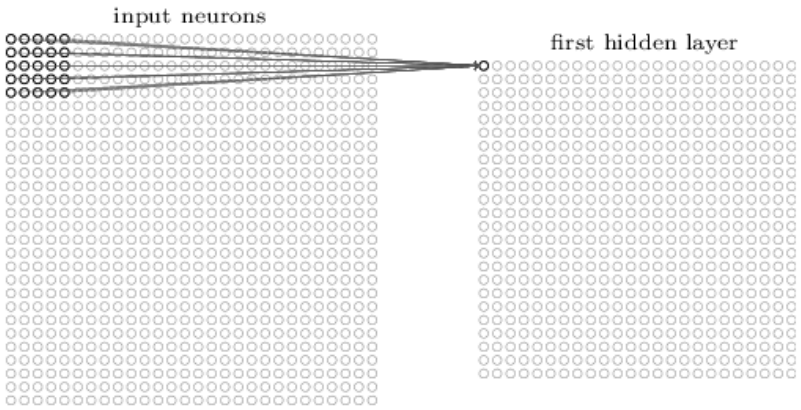

Figura 19: Introducing Convolutional Neural Network *[13]*

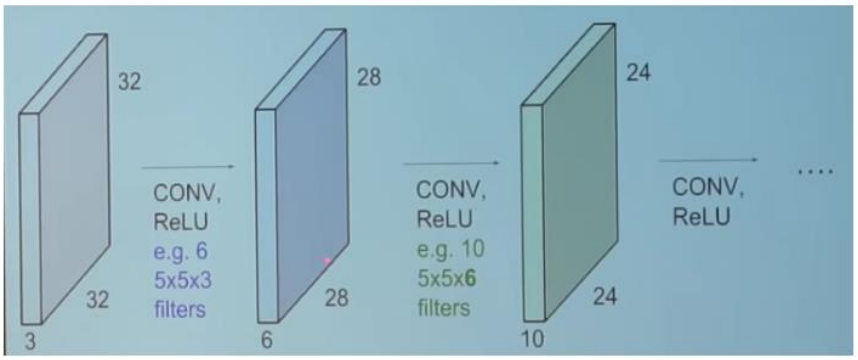

Figura 20: Introducing Convolutional Neural Network *[27]*

Encontramos por lo tanto en las redes convolucionales un proceso por el cual se multiplica la matriz de pesos del filtro por la matriz de pixeles de la receptive field correspondiente, representando así el valor final de salida de la neurona para ese filtro en ese receptive field, tal y como se explica en la *figura 20*.

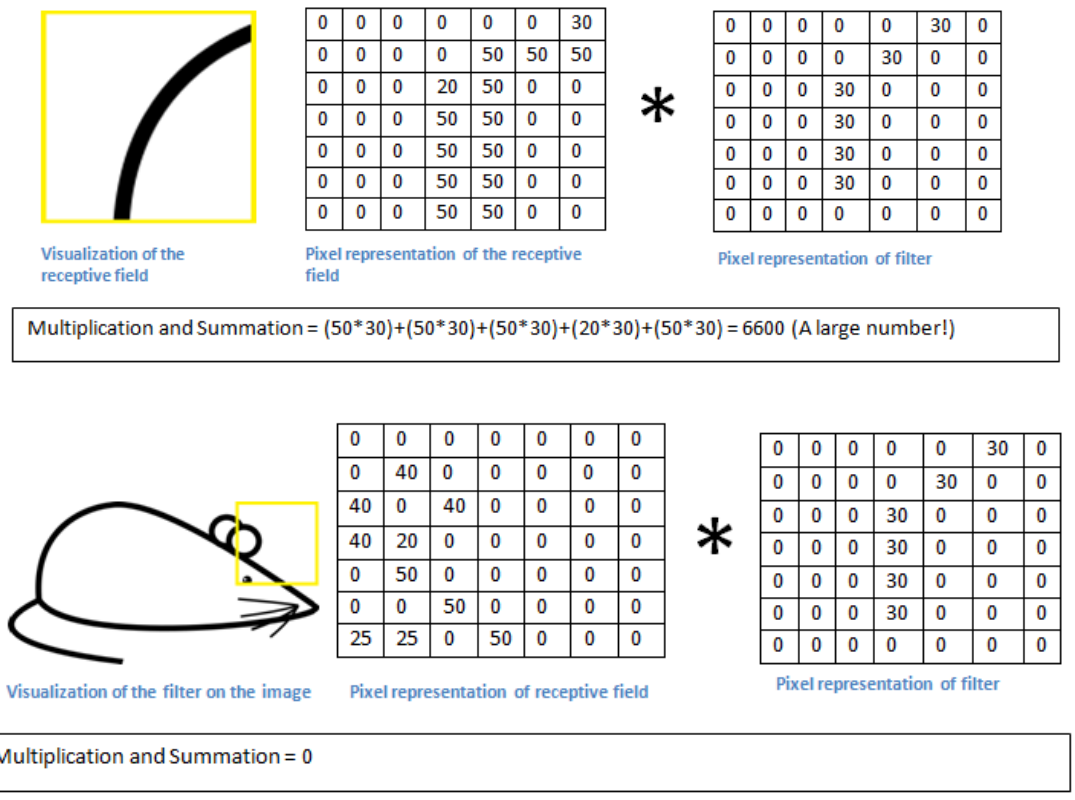

Figura 21: Operación de convolución *[14]*

La salida Yj de una neurona j es una matriz que se calcula por medio de la combinación lineal de las salidas Yi de las neuronas de la capa anterior operadas con el núcleo de convolución Kij correspondiente a cada conexión, al resultado se le suma un bias (Bj) y se pasa por la función de activación g no-lineal, como la función ReLu. Esta salida ha de ser sumada junto a las salidas calculadas de la misma forma para el resto de canales, en este caso al ser RGB, son 3 canales;

rojo,azul y verde, las dimensiones de la matriz de salida será igual al número de receptive fields operados.

$$Yj = g(bj + \sum Kij \otimes Yi \quad (43)$$

Para cada particular "feature map" estará compuesto de un numero de neuronas igual al número de posiciones únicas obtenido en el barrido, se llaman neuronas ya que computan los pesos del filtro con una funcion de activación , además éstas, podrán implementar un proceso llamado "shared weights", donde todas las neuronas de este feature map comparten los mismos pesos, de esta forma no será necesaria que cada neurona aprenda para sí misma a detectar su feature, reduciendo así el número de parámetros de la red y los requisitos finales de computo. Es importante señalar que los pesos entre distintos feature maps pueden ser distintos.

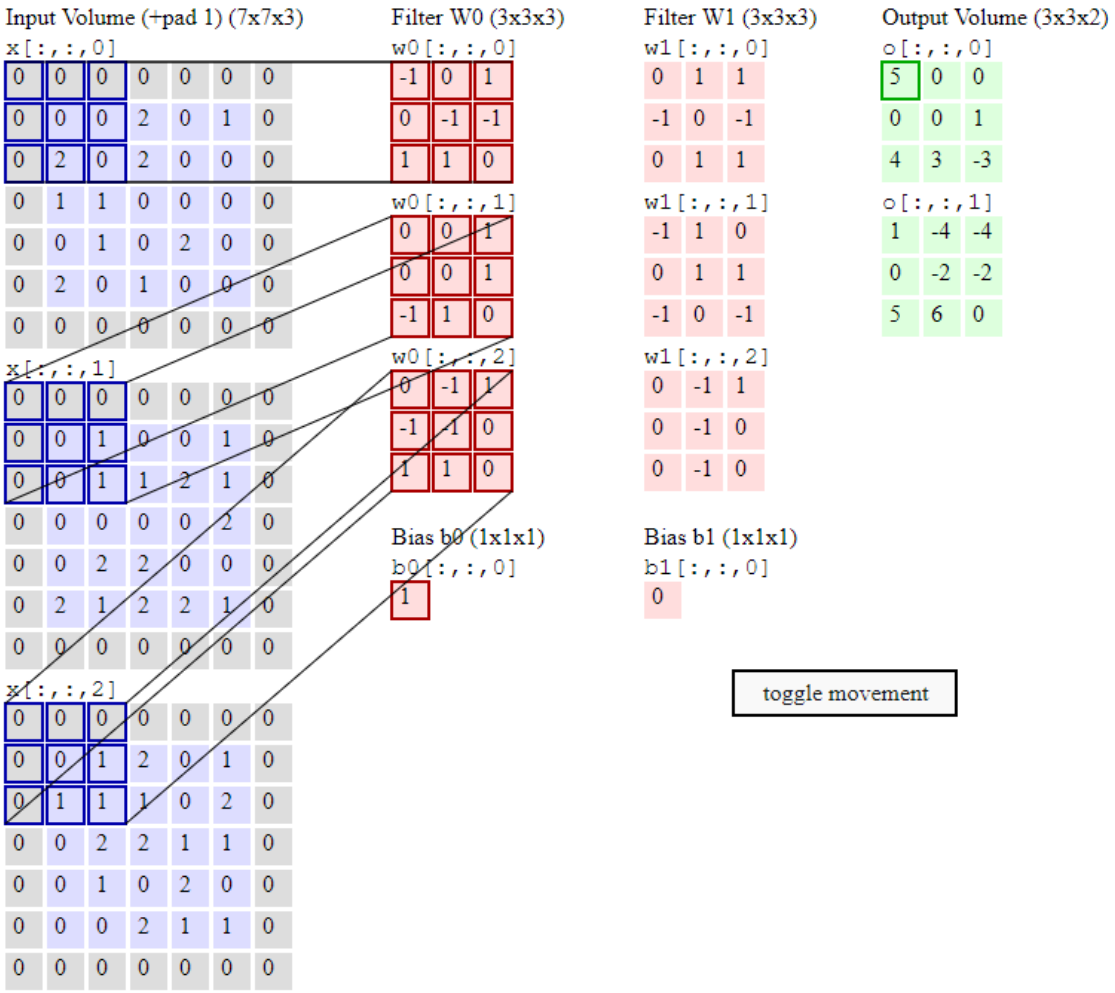

Figura 22: Convolution operation with shared weights between neurons of the same feature map [26]

Otro parámetro muy a tener en cuenta en las CNN, es el **padding,** con ello buscamos mantener capa a capa el mismo volumen de inputs, veamos un caso concreto; en un input de 25x25 con un filtro de 5x5 pasaría a ser de 21x21 pero si aplicamos un zero padding de 2, tendríamos otra vez una entrada para las siguientes capas de 25x25.

Pooling Layer

En esta capa se busca producir una reducción del muestreo para reducir así la resolución y por ende el exceso de información reduciendo la carga computacional de la red y la memoria utilizada, mediante una función de agregado como por ejemplo una media o como en el siguiente caso de la *Figura 23*, un máximo.

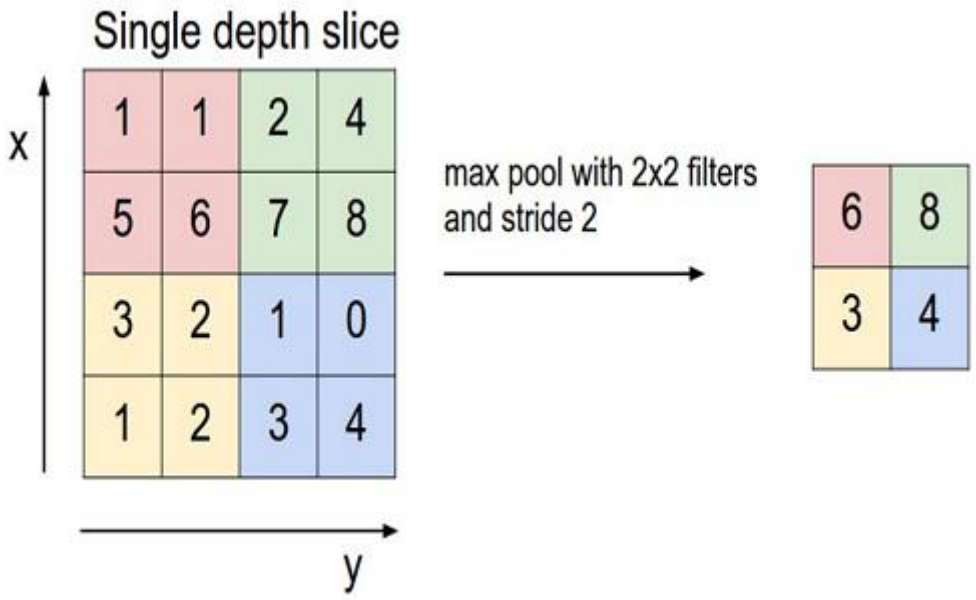

Figura 23: Pooling Layer *[15]*

La gran ventaja que tiene utilizar "max-pooling", es que la red puede preguntarse si una característica está en alguna región de la imagen, por ende, no es necesario saber la localización exacta sino más bien como está relacionada con las otras características, por lo tanto permite reducir así el número de parámetros necesarios en las próximas capas ya que hay menos características agrupadas.

Neuronas de Clasificación "Fully Connected Layer":

Después de pasar por las diferentes capas de extracción de características, los datos finales han de clasificarse hacia una categoría u otra según los objetivos. Las neuronas de estas capas son idénticas a otras redes como las FNN, pueden utilizar diferentes algoritmos de aprendizaje backpropagation, Adam, RMSProp, etc.

$$yj = g(bj + \sum wij \cdot yi \quad (44)$$

Por último, la última capa es un conjunto de neuronas simples que sirven para hacer la clasificación final de las características inferidas en el proceso mediante una función *sigmoid* en el caso de que sea una clasificación binaria o *softmax* en el caso de la existencia de múltiples categorías.

Tenemos que tener muy en cuenta que este tipo de redes demandan una gran capacidad de cómputo y de memoria RAM, por lo que es necesario utilizarlas con equipos que permitan el uso de GPUs.

Una forma muy útil de reducir este enorme consumo de recursos es reduciendo el *mini-batch size* o por otro lado aumentando los *stride* o pasando de float32 a float16.

Algoritmo de aprendizaje

El algoritmo normalmente empleado para el aprendizaje es el mismo que en otras FNN, se trata del "backpropagation" empleando el gradiente descendiente, siendo utilizado tanto en las capas convolucionales como en las fully connected.

La estructura final, podría ser algo similar a la *Figura 24* donde se puede observar la estructura general de una CNN, con sus múltiples capas convolucionales, pooling y una última capa de clasificación.

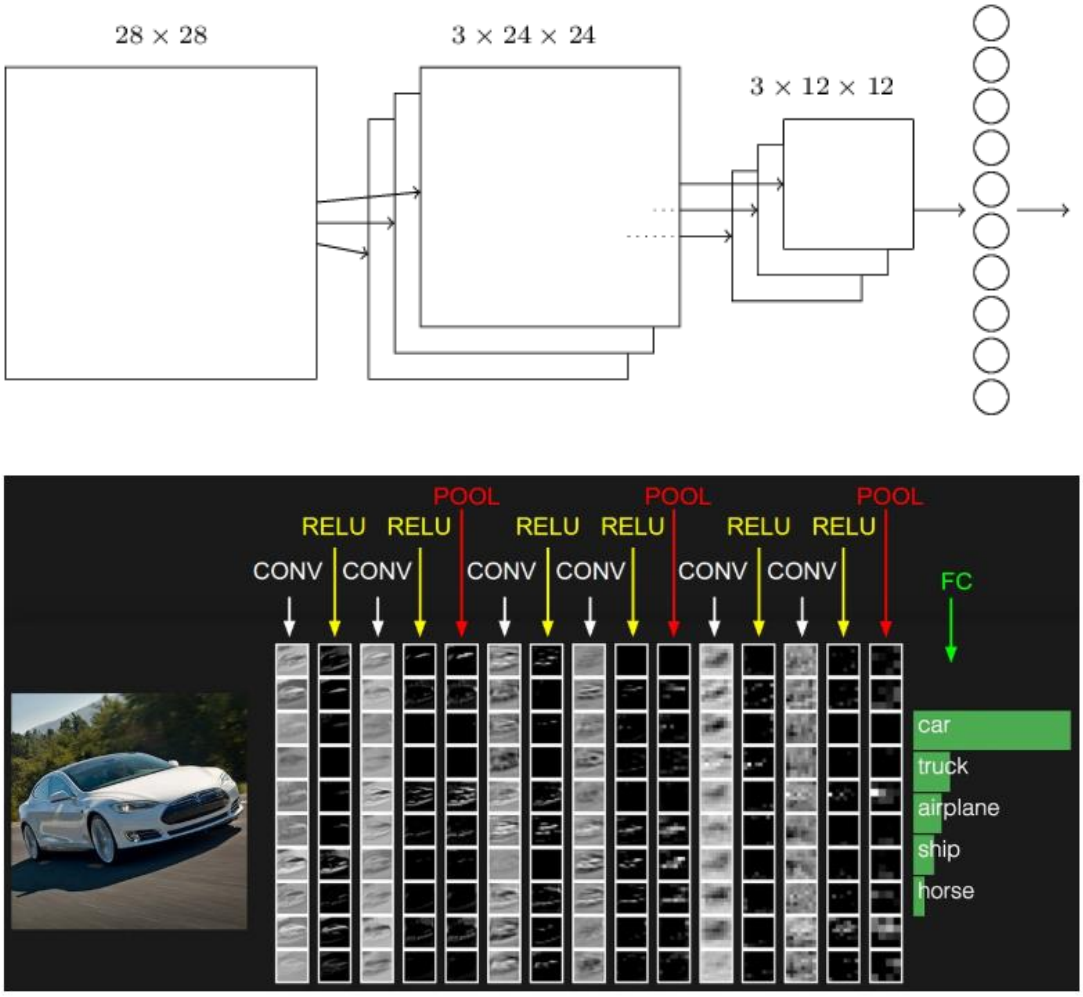

Figura 24: Convolutional Neural Network Layers *[15]*

4.2 TensorFlow y Keras

Python dispone de multiples librerías como podrían ser Keras o TensorFlow, para el desarrollo de redes neuronales o algoritmos de machine learning. En concreto, en este apartado se utilizarán ambas para la construcción de la CNN propuestay observaremos el desempeño de ambas APIs.

4.2.1 TensorFlow y TFLearn

El módulo de capas de TensorFlow nos ofrece una API desde donde la podemos construir de forma rápida y sencilla cualquier tipo de red. Sin embargo, han diseñado una nueva High-API que funciona por encima de Tensorflow, aún más sencilla y será la que utilizaremos, llamada TFLearn desde donde poder crear la red mediante unos métodos más simplificados.

Los métodos empleados fueron los siguientes:

- conv_2d() : Construye una capa convolucional de 2 dimanesiones, se ha de especificar el número de filtros, su tamaño (ej 5x5) , el padding y su función de activación.

-max_pool_2d(): contruye una pooling layer de 2-dimensiones, toma como parámetros el tamaño de filtro y el stride.

-fully_connected(): se le indica la función de activación y el número de neuronas que incorporara

-dropout(): se indica la probabilidad de que una neurona sea desechada, se usa para evitar los minimos locales y mejorar la optimización.

-regression: donde se indicará el algoritmos de aprendizaje, learning rate, función de perdidas, etc del modelo.

```
tf.reset_default_graph()
convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.5)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')
```

Figura 25: modelo de red neuronal para clasificacion “cats vs dogs”

En la creación de este modelo se empleó una función de perdidas llamada “categorical-cross entropy” y una función de activación softmax, esta elección no resulta errónea puesto que softmax es la generalización para la clasificación de múltiples categorías de la funcion sigmoid, pero es necesario indicar que al tratarse de una clasificación binaria, es mucho más optimo

emplear una función de activación sigmoid y una función de perdidas “binary_crossentropy” si se van a emplear datasets más complejos.

4.2.2 Keras

De forma similar a Tensorflow, fue creada Keras, se trata de una de las librerías más utilizadas y con más documentación disponible para la creación de algoritmos ML o NN.
Esta librería se empleara para construir la CNN propuesta para la clasificación de múltiples categorías con el dataset Cifar10, así mismo esta librería se empleara en el apartado 5 apartado, para la creación de modelos regresivos o clasificadores para series temporales mediante LSTM, SVM, etc.

Los métodos que se emplearan y futuro modelo son los siguientes:

-sequential(): se indica a Keras que se va a crear un secuenciado de capas

-Conv2D(): de forma similar a TFLearn, indicamos el número de neuronas de la capa, su input shape y su función de activación.

- MaxPoolingLayer(): indicamos el tamaño del pool o de ventana, que permitirá la reducción de información, tal y como hemos explicado anteriormente.

-Dropout(): probabilidad de que una neurona de esta capa sea desechada.

-Dense(): capa ForwardPass.

```
# Create the model
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=x_train.shape[1:]))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=LR, decay=1e-6),
              metrics=['accuracy'])
return model
```

Figura 26: modelo de red neuronal para clasificación del dataset *Cifar10[17]*

El dataset Cifar10 dispone de hasta 10 categorías diferentes de objetos para ser clasificados, por ello es totalmente necesario emplear a su salida un numero de neuronas igual al total de categorías y como función de activación final softmax, junto a una función de perdidas “categorical_crossentropy”.

4.3 Desarrollo de una CNN para la clasificación de imágenes con TensorFlow y Keras

Uno de los puntos más importantes de este trabajo final de grado, era desarrollar una red neuronal convolucional para el reconocimiento de imágenes y su posterior clasificación.
A nivel técnico ha sido muy instructivo, ya que me ha permitido tomar contacto y aprender a programar en Python, uno de los lenguajes más usados en Data Science.

Para el proyecto decidí desarrollar un modelo de red neuronal y probarla en 2 escenarios diferentes, uno simple donde se clasificasen dos categorías muy diferentes y otro más complejo donde se enseñase a la red a clasificar múltiples categorías. El objetivo es analizar el potencial de las CNN para extraer en un futuro datos de los propios anuncios de NextNight y procesarlos después con otro tipo de redes.

4.3.2 Escenario "Cats vs Dogs"

El primer escenario se basaba en desarrollar un modelo que permita distinguir entre perros y gatos a partir de una base de datos creada por la universidad de Toronto, donde se recogían cerca de 25000 imágenes de ambos animales, al solo haber dos categorías y una variedad importante de inputs, la capacidad para inferir generalización en el modelo y por lo tanto lograr una precisión optima era mayor.

El modelo que se ha propuesto es el de la imagen anterior, 5 redes convoluciones seguidas cada una por una red Pooling para reducir las dimensiones, una red ForwardPass de 1024 neuronas con funciones de activación "relu", seguida de una output layer de dos neuronas con función de activación "softmax" para clasificar el resultados según probabilidad de ser gato o perro, al ser solo dos categorías podría haberse empleado la función logística.

Si el resultado numéricamente mayor se encuentra en la primera neurona significa que la red neuronal considera que la imagen es un gato y si se encuentra en la segunda neurona considera que es un perro.

Al no existir una fórmula que permita obtener la configuración optima de la red, el número de capas convolucionales, pooling layers, etc es necesario realizarla a través de una aproximación puramente experimental.

Para ello jugaremos con las variables que nos permiten configurarla:

1-Numero de "epochs", es decir número de veces que la red es entrenada con los datos disponibles. Hay que tener en cuenta que, aunque a priori parezca una buena idea un numero de epochs excesivamente alto puede acarrearnos overfiting en el modelo.

2-Numero de redes convolucionales y polling layers, además de su distribución y orden.

3-Numero de redes ForwardPass , numero de neuronas de cada una y su función de activación.

4-Dropouts

5-Learning Rate

En cuanto a los datos que se usarán como inputs, disponemos de 24000 imágenes para entrenar, de ellas las ultimas 600 serán usadas para realizar la parte de validación en cada epoch, permitiendo ver si realmente la red está aprendiendo correctamente, cabe recalcar que los datos empleados en la validación o en el test jamás pueden ser usados en el entrenamiento porque si no la red ya habría tenido contacto con ellos y los resultados estarían adulterados.

El hardware empleado para realizar este escenario es el siguiente:

GPU: nvidia GTX745.

CPU: Intel(R) Core(TM) i7-4790S CPU @ 3.20GHz, 3201 Mhz, 4 procesadores principales, 8 procesadores lógicos.

SO: Windows 10.

RAM: 12GB.

Los resultados de las múltiples pruebas fueron los siguientes:

Indice	**Epoch**	**Nº de capas convolucionales y polling layers**	**Nº de redes Forward Pass**	**Funcion de activacion**	**Dropout**	**Learning Rate**	**Precision con el test de validacion**
1	1	5 de 5x5 (32,64,128,64,32) 5 kernel stride en polling	2 (1x1024 1x2)	relu	0.5	1E-3	0.73
2	1	5 de 5x5 (32,64,128,64,32) 5 kernel stride en polling	2 (1x1024 1x2)	relu	0.5	1E-2	0,48
3	3	5 de 5x5 (32,64,128,64,32) 5 kernel stride en polling	2 (1x1024 1x2)	relu	0.5	1E-6	0,578
4	3	5 de 5x5 (32,64,128,64,32) 5 kernel stride en polling	2 (1x1024 1x2)	relu	0.8	1E-6	0.585
5	3	5 de 5x5 (32,64,128,64,32) 5 kernel stride en polling	2 (1x1024 1x2)	relu	0.8	1E-3	0.76
6	5	5 de 5x5 (32,64,128,64,32)	2 (1x1024 1x2)	relu	0.8	1E-3	0.771

		5 kernel stride en polling					
7	5	5 de 5x5 (32,64,128,64, 32) 5 kernel stride en polling	2 (1x1024 1x2)	relu	0.9	1E-3	0.775
8	10	5 de 5x5 (32,64,128,64, 32) 5 kernel stride en polling	2 (1x1024 1x2)	relu	0.9	1E-3	0.785
9	10	5 de 5x5 (32,64,128,64, 32) 5 kernel stride en polling	1x512,1x256,1 x2	relu	0.9	1E-3	0.736
10	10	5 de 5x5 (32,64,128,64, 32) 5 kernel stride en polling	1x2048,1x2	relu	0.9	1E-3	0.795(10) 0.805(6)
11	6	5 de 5x5 (32,64,128,64, 32) 5 kernel stride en polling	1x3072,1x2	relu	0.9	1E-3	0.773
12	6	3 de 7x7,2 de 5x5 (32,64,128,64, 32) 5 kernel stride en polling	1x2048,1x2	relu	0.9	1E-3	0.71
13	6	5 de 5x5 (32,64,128,64, 32) 3 kernel stride en polling	1x2048,1x2	relu	0.9	1E-3	0.49 (1)
14	6	5 de 5x5 (32,64,128,64, 32)	1x2048,1x2	relu	0.9	1E-3	0.793

		8 kernel stride en polling					
15	6	8 de 5x5 (32,64,128,64, 32) 8 kernel stride en polling	1x2048,1x2	relu	0.9	1E-3	0.748
16	8	5 de 5x5 (32,64,128,64, 32) 8 kernel stride en polling	1x2048,1x2	relu	0.8	1E-3	0.7917
17	**8**	5 de 5x5 (32,64,128,64, 32) 8 kernel stride en polling	1x2048,1x2	relu	0.6	1E-3	0.805
18	10	5 de 5x5 (32,64,128,64, 32) 8 kernel stride en polling	1x2048,1x2	relu	0.4	1E-3	No converge bien
19	10	5 de 5x5 (32,64,128,64, 32) 13 kernel stride en polling	1x2048,1x2	relu	0.6	1E-3	0.79
20	10	5 de 5x5 (32,64,128,64, 32) 8 kernel stride en polling	1x2048,1x2	relu	0.6	1E-3	0.7917
21	20	5 de 5x5 (32,64,128,64, 32) 8 kernel stride en polling	1x2048,1x2	relu	0.6	1E-3	0.77

Tabla 1: Secuenciado de test realizados sobre el escenario 1.

```
| Adam | epoch: 008 | loss: 0.39336 - acc: 0.8137 -- iter: 24128/24400
Training Step: 3052  | total loss: 0.39240 | time: 23.828s
| Adam | epoch: 008 | loss: 0.39240 - acc: 0.8167 -- iter: 24192/24400
Training Step: 3053  | total loss: 0.38567 | time: 23.891s
| Adam | epoch: 008 | loss: 0.38567 - acc: 0.8194 -- iter: 24256/24400
Training Step: 3054  | total loss: 0.39594 | time: 23.953s
| Adam | epoch: 008 | loss: 0.39594 - acc: 0.8140 -- iter: 24320/24400
Training Step: 3055  | total loss: 0.39763 | time: 24.016s
| Adam | epoch: 008 | loss: 0.39763 - acc: 0.8092 -- iter: 24384/24400
Training Step: 3056  | total loss: 0.40548 | time: 25.079s
| Adam | epoch: 008 | loss: 0.40548 - acc: 0.8048 | val_loss: 0.46383 - val_acc: 0.8050 -- iter: 24400/24400
```

Figura 27: Resultado final del entrenamiento sobre la mejor arquitectura seleccionada (índice 17 tabla 1).

Figura 28: Ejemplo visual del datatest realizado.

Como podemos observar hemos conseguido una precisión del 80%, con mayor tiempo de estudio será posible establecer una configuración mucho más efectiva que eleve nuestra precisión. Ahora pasaremos a analizar más exhaustivamente un escenario de clasificación mucho más complejo.

4.3.1 Escenario cifar10

En este escenario vamos a analizar el dataset *Cifar10* [17] mediante una red convolucional desarrollada con Keras capaz de distinguir entre 10 clases diferentes de objetos.

Hemos cambiado de lenguaje o Hight API, debido a algunos problemas de TFLearn para clasificar grandes dataset con múltiples categorías con tarjetas gráficas de la serie 700, donde se observa que el error dado por la función de aprendizaje no convergía.

El modelo de arquitectura de red que se ha creado es el siguiente y esta propuesto en el GitHub de Keras por defecto, ya fue explicado e introducido en la *Figura 26*.

Ha llevado cerca de 35 minutos entrenar el modelo, siendo necesario entender que debido a que es una red convolucional con un dataset bastante considerable, cerca de 50000 imágenes, realizara un consumo considerable de memoria RAM y GPU , al tener que computar los inputs por las múltiples capas creadas, por lo tanto debemos evitar tener programas activados mientras la entrenamos que puedan consumir alguno de estos dos requisitos, sino podrían surgir errores de falta de recursos mientras se lleva a cabo el entrenamiento.

La configuración de la red decidida para el entrenamiento, validación y test son los siguientes:

Algoritmo de aprendizaje = Adam,

Función de perdidas = categorical crossentropy,

Training set = 50000 imágenes,

Validation y Test set = 10000 imágenes.

Learning Rate	Epochs	Precisión
0.0001	30	0.705
0.0001	45	0.721
0.001	40	Pico de precisión en epoch 16 con 0.8

Tabla 2: Secuenciado de test para optimizacion de los parametros del modelo.

Como podemos observar en la propia grafica de entrenamiento,las perdidas han dejado de converger apartir del epoch 16, quedandose ya muy atrás respecto a las perdidas en el aprendizaje del training set. Tecnicamente no ha llegado a ver overfitting ya que la precision en la validacion no ha sufrido una reduccion, pero si estamos en claro riesgo de sufrirlo a mayor numero de iteracciones puesto que la ganancia de precision real en el modelo parece haber llegado a su limite con esta configuracion.

Con mayor tiempo de estudio, se podra proponer modelos mucho más eficaces.

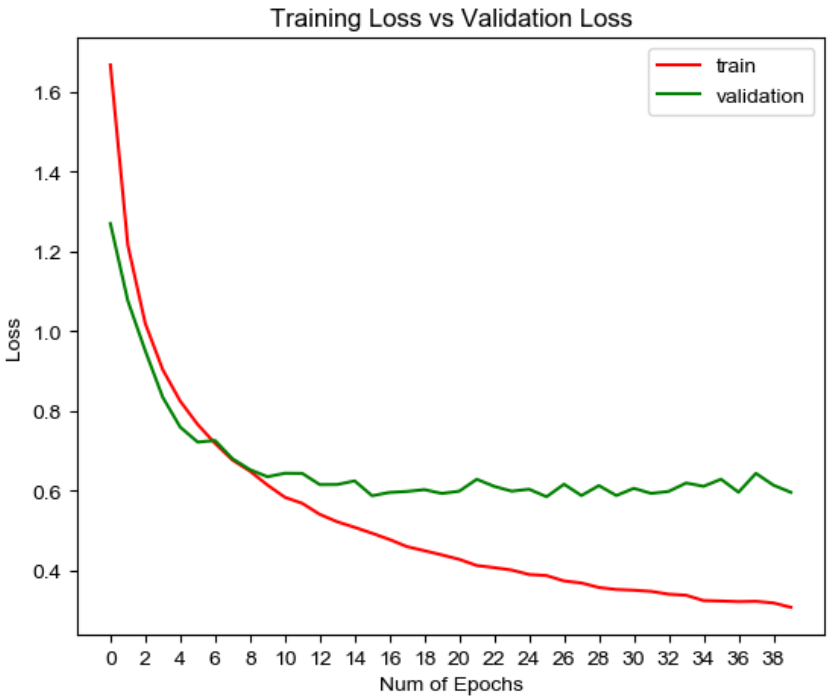

Figura 29

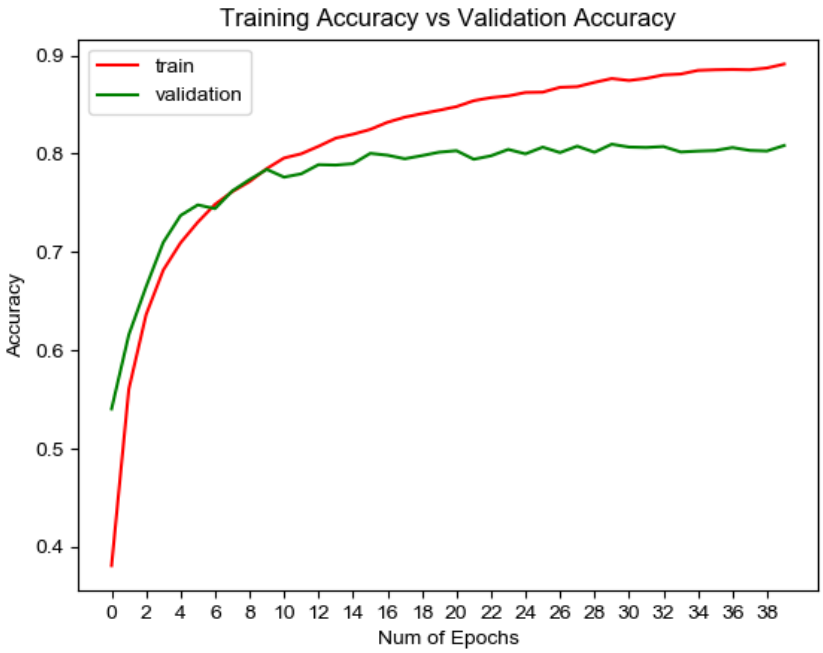

Figura 30

De esta manera, procedemos a cambiar el numero óptimo de iteraciones (16) y entrenaremos el modelo nuevo buscando resultados similares a los anteriores.

Por último, lo evaluaremos utilizando las métricas propuestas en el punto 2.4 de este trabajo.

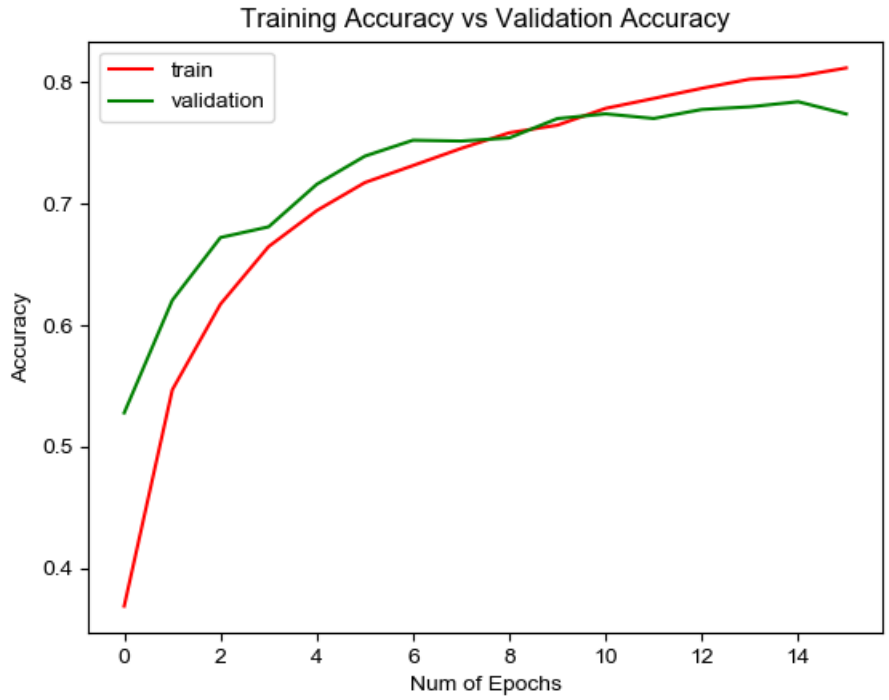

Figura 31

```
Loss: 0.669
Accuracy: 77.40%
```

Como podemos observar en la siguiente imagen, tenemos la matriz de confusión de nuestro modelo clasificatorio sobre el dataset cifar10. En el eje x tenemos las clases reales del test y en el eje y las predicciones para cada una de ellas.

etiquetas = ['airplane','automobile','bird','cat','deer','dog','frog','horse','ship','truck']

```
[[788   5  32  11   8   4   9  12  90  41]
 [ 14 793   3   4   2   4   5   2  34 139]
 [ 41   1 695  24  62  74  49  26  18  10]
 [ 18   4  64 431  54 287  55  47  16  24]
 [ 10   0  56  31 729  44  30  81  15   4]
 [ 10   1  40  47  22 801  12  55   3   9]
 [  3   2  42  20  28  32 843  18   8   4]
 [ 11   3  23  11  24  76   1 835   4  12]
 [ 29   8   7   3   6   3   5   5 906  28]
 [  8  15   7   2   2   5   6   8  28 919]]
```

Figura 32: Matriz de confusión del test.

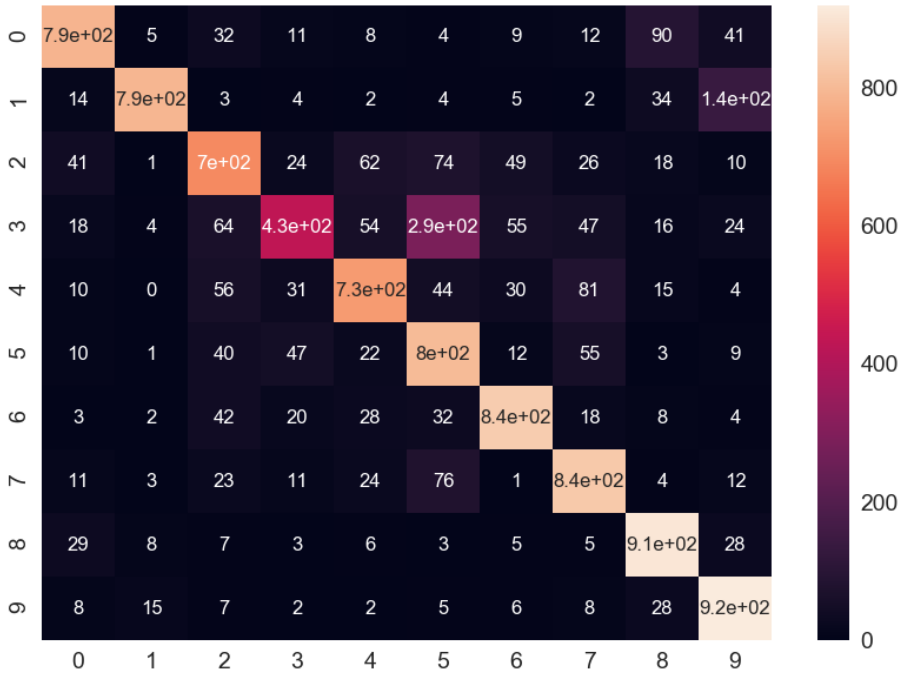

Figura 33: Matriz de confusión con python del test.

Gracias al método “sequential_model_to_ascii_printout()” de Keras podemos ver la arquitectura de la red convolucional con sus respectivos pesos .

```
   OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

       Input   #####      3   32   32
      Conv2D    \|/  -------------------       896     0.1%
        relu   #####     32   30   30
      Conv2D    \|/  -------------------     18496     2.4%
        relu   #####     64   28   28
MaxPooling2D   Y max -------------------         0     0.0%
               #####     64   14   14
     Dropout    | ||  -------------------         0     0.0%
               #####     64   14   14
      Conv2D    \|/  -------------------     73856     9.5%
        relu   #####    128   12   12
MaxPooling2D   Y max -------------------         0     0.0%
               #####    128    6    6
      Conv2D    \|/  -------------------    147584    19.0%
        relu   #####    128    4    4
MaxPooling2D   Y max -------------------         0     0.0%
               #####    128    2    2
     Dropout    | ||  -------------------         0     0.0%
               #####    128    2    2
     Flatten   |||||  -------------------         0     0.0%
               #####         512
       Dense   XXXXX -------------------    525312    67.7%
        relu   #####        1024
     Dropout    | ||  -------------------         0     0.0%
               #####        1024
       Dense   XXXXX -------------------     10250     1.3%
     softmax   #####          10
```

Figura 34: Esquema de peso del modelo CNN.

Por último, añado una imagen con un fragmento del testset, para poder visualizar la precisión del modelo.

Figura 35: Muestra del testset.

Adjunto en la bibliografía un enlace a un blog donde se propone un modelo de red capaz de alcanzar un 90% de precisión *[25]*, sin embargo, se trata de una red bastante densa y profunda que demandara una GPU acorde a su nivel.

Capítulo 5. Desarrollo de un algoritmo predictivo para el índice Dow Jones

En este capítulo, se introducirá el análisis y desarrollo de modelos predictivos sobre series temporales, con el fin de ver la viabilidad del uso de inteligencia artificial para detectar tendencias o resultados futuros, debido a la falta de los datos necesarios, se decide emplear un marco diferente para la experimentación, como pueden ser las series temporales financieras, pero extrapolables a series menos complejas o al menos con un componente aleatorio menor como pueda ser el consumo de cerveza, interés en determinados tipos de eventos, etc. Por consiguiente, se explicará que es un mercado financiero, que factores le afectan y como seleccionar variables para desarrollar modelos predictivos.

5.1 Introducción.

Desde hace décadas se han ido desarrollando conforme el paso del tiempo, sistemas que pudieran emplearse para poder operar el mercado de forma fructuosa y a ser posible incluso batirlo.

Los diferentes fondos de inversión han empleado técnicas muy diferentes según los productos financieros a los que estuvieran orientados, perfiles de riesgo de sus clientes, contexto, etc. Todos ellos no solo buscan ser rentables, es decir tener una rentabilidad positiva, sino que además a ser posible batir al mercado generando un Alpha por parte de su gestor. Cuando hablamos de Alpha o Beta en el mundo financiero nos estamos refiriendo a dos tipos de resultados.

Beta: Se trata de como varían las operaciones respecto al índice de referencia

Alpha: Es la rentabilidad extra conseguida por el gestor o inversor.

Un ejemplo concreto sería, si un fondo tiene una Beta de 0,6 y una Alpha de 0,2, si el índice de referencia crece un 15% significaría que el fondo ha obtenido un 9% de rentabilidad por el Beta más un 3% por el Alpha.
Como se puede observar, el desempeño del gestor es clave para los buenos resultados de una inversión, sin embargo todos los humanos estamos expuestos y sujetos a determinados sesgos cognitivos que de alguna forma pueden afectar a nuestras decisiones y por ende al resultado final.

Debido a esto y a que realmente muy pocas personas son capaces de controlar la gran presión a la que pueden ser sometidos por parte del mercado en sus momentos más depresivos, se ha intentado que la tarea de gestión del portafolio sea llevada a cabo de la forma más racional y para ello se han usado algoritmos ejecutados por máquinas que permitirían eliminar al menos en parte los sesgos emocionales.

Para este objetivo, no solo se han creado algoritmos, sino también productos financieros con muy buenos resultados, como es el caso de los llamados ETFs o fondos pasivos. Pero ¿Qué es un fondo pasivo? Se trata de un producto financiero donde por el precio de cotización de este, se dispone una parte proporcional de las acciones que están contenidas en ese fondo. Son denominados “pasivos” debido a que no disponen de un gestor en activo, sino que en todo momento están compuestos de forma ponderada por todas las compañías del índice al que pretenden replicar, por lo tanto, juzgamos su comportamiento en el caso de que sean capaces de generar una beta muy cercana a 1, es decir que repliquen el comportamiento del índice.

Los ETF ,se han convertido en uno de los productos financieros con mayor éxito de la historia y además con muy buenos resultados, ya que aunque a priori asimilamos la bolsa a riesgo, sobre todo en las últimas décadas debido a las dos grandes burbujas que asolaron el mundo en el año 2000 “burbuja .com” y 2008 “burbuja subprime”, los mercados han generado una rentabilidad positiva, como por ejemplo el índice SP500, índice de referencia estadounidense, donde desde

1928 ha crecido un 10% anual, superando a la inflación media que ha sido del 3% siendo esta el enemigo real de los ahorradores.

El gran reto que se abordara en este TFG, es crear un algoritmo que opere sobre el Down Jones y además que no solo sea rentable, sino que obtenga una Beta similar al ETF que replica al índice en cuestión y quizás hasta un Alpha, con el fin de demostrar que los algoritmos basados en inteligencia artificial son aplicables de forma exitosa a una gran diversidad de escenarios.

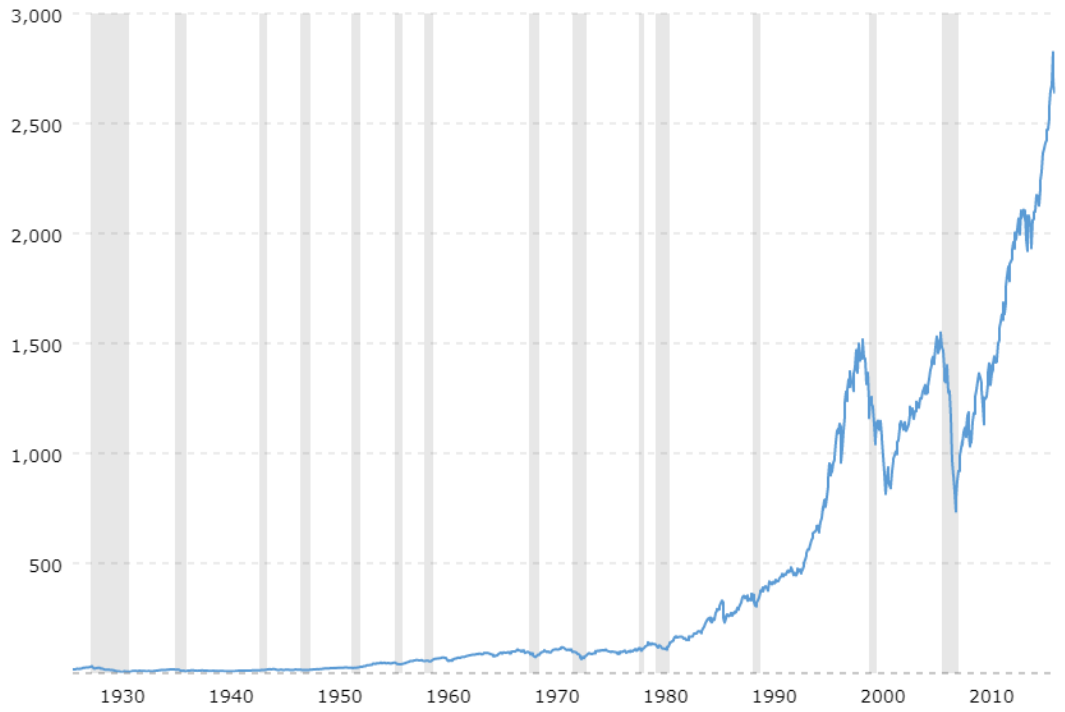

Figura 36: Indice Dow Jones escala lineal.

En el grafico anterior, figura 36, se puede observar la evolución del índice SP500 anteriormente introducido, sin la inflación ajustada, por otro lado, las franjas grises corresponden con las recesiones. El índice recoge la capitalización de las 500 mayores empresas de los índices NYSE y NASDAQ.

He de señalar que el análisis de la gráfica ajustada linealmente con periodos tan amplios de tiempo no sirve para observar realmente si un índice ha tenido un crecimiento exponencial, para ello es necesario ajustarlo logarítmicamente, en la siguiente imagen, figura 37, podemos ver de una forma más nítida los periodos de crecimiento exponencial de las cotizaciones y las diversas burbujas o fases de euforia del mercado.

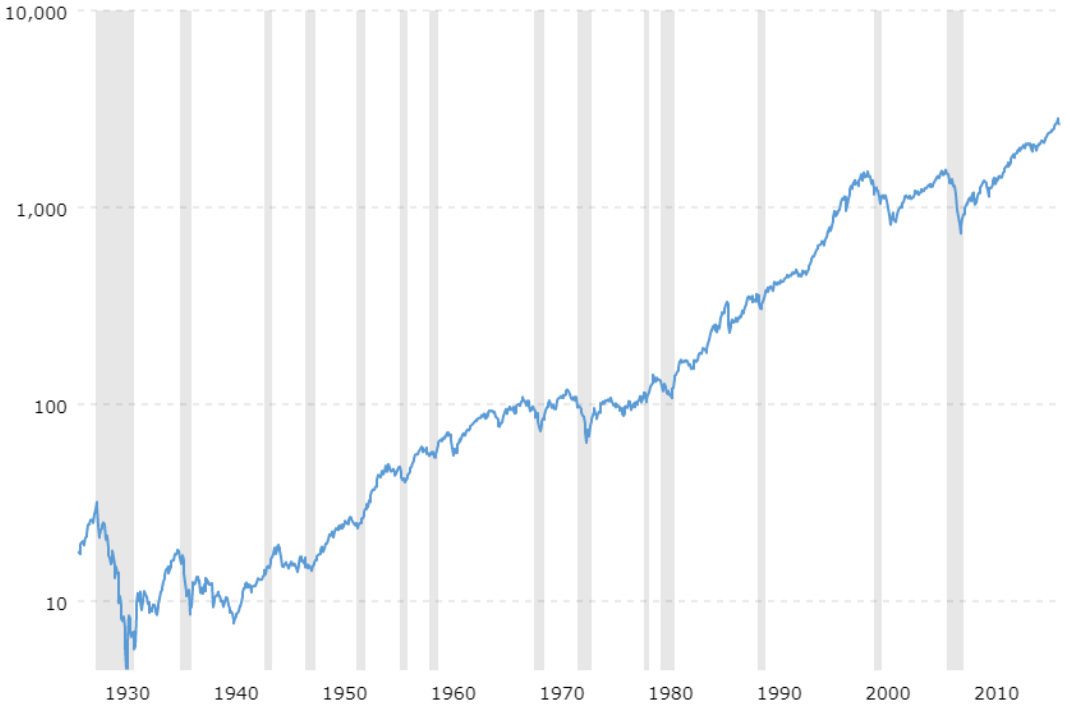

Figura 37: Indice Down Jones escala logarítmica.

Los gestores e inversores privados han ido desarrollando y aplicando determinadas técnicas con enfoques y puntos de vista muy diferentes para poder conseguir buenos resultados, además generando una gran bibliografía sobre estas técnicas y sobre el mundo financiero en general.

Las principales técnicas empleadas para operar en estos mercados son en grandes rasgos las siguientes:

Análisis macroeconómico: Entendemos por macroeconomía a la teoría que se encarga de elaborar y estudiar los diferentes indicadores económicos que pretenden cuantificar el mercado sobre el que operan en términos de variables agregadas como productividad, bienes y servicios, empleo, etc en contra posición a la microeconomía que se encargaría de estudiar a los agentes individuales consumidores, empresas, trabajadores, etc.

Análisis técnico: Se trata del análisis del mercado desde el punto de vista del precio para tratar de averiguar con graficas patrones o tendencias que hagan posible predecir futuros valores de la cotización.

Inversión en valor: Consiste principalmente en comprar acciones cuando estas se encuentran infravaloradas por el mercado o su precio está por debajo de su valor intrínseco o fundamental, para posteriori venderlas o mantenerlas en cartera a largo plazo (buy & hold).

La valoración del precio se hace conforme a una serie de indicadores como el ratio de endeudamiento, ratio de liquidez, flujo de caja, beneficios respecto a su índice de referencia, etc.

Análisis cuantitativo: Es la técnica más novedosa y con mayor potencial de crecimiento de todas las áreas anteriormente descritas, a su vez es a nivel técnico la más compleja.
Se trata de cuantificar el mercado mediante matemáticas financieras derivadas de la estadística. Por lo tanto, se emplea el cálculo centrado en derivadas parciales, econometría, estadística y probabilidad.

Este enfoque es el que se empleara a pequeña escala para el desarrollo de los diferentes algoritmos. Para ello se utilizarán indicadores estadísticos empleados en el análisis de series

temporales en econometría como Arima, junto con otros indicadores técnicos, redes neuronales, etc. En futuros trabajos se podría ampliar la selección hacia indicadores macroeconómicos y contables en otras escalas temporales.

5.2 Índice Dow Jones y contexto macroeconómico

La teoría de "ciclos económicos" es la más extendida a la hora de explicar el origen de las recesiones y expansiones económicas, en ella se explica cómo ésta sigue fases de expansión inmediatamente precedidas por fases de contracción que sirven para regular los desajustes producidos en la anterior fase del ciclo.
Ray Dalio director del fondo de inversión Bridgewater Associates valorado en 150 billones de dolares, explica de forma muy precisa en "How the Economic Machine Works" como utilizar esta teoría a la hora de comprender los mercados financieros y como usarla para tomar nuestras decisiones de inversión.

En su paper explica de forma detallada, como la materia prima de los mercados son las transacciones, una transacción consiste en un agente individual comprando a otro agente un bien o servicio a través de dinero o crédito y en definitiva un mercado financiero es el conjunto de todos las transacciones que se realizan diariamente.

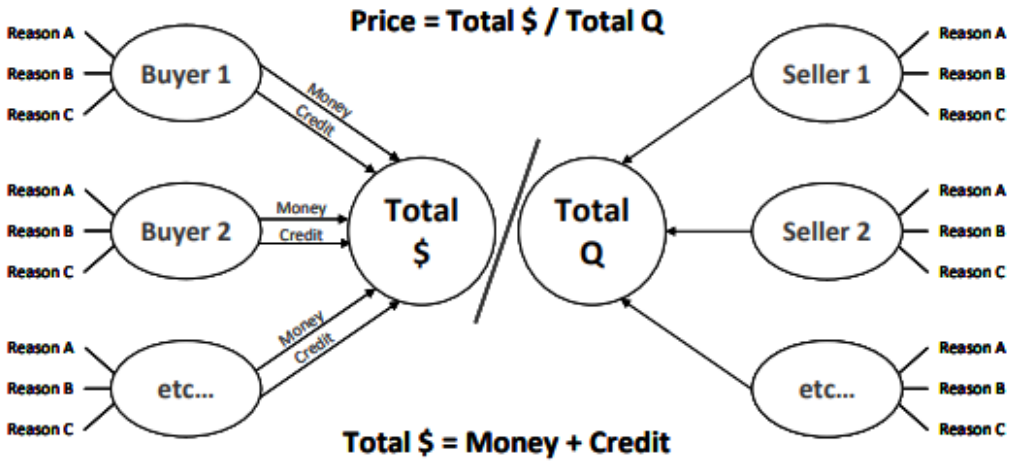

Figura 38: Transactions based approach *[19]*

Tal y como explica Ray Dalio, todos los cambios producidos en la actividad económica y en los precios de los mercados financieros son consecuencia de la oferta (Q) y demanda ($), sin embargo la variabilidad de la cantidad total que se tiene de estos afecta de forma diferente, en este caso los cambios producidos por la demanda son más importantes, debido a que modificar la oferta de dinero y crédito es muy fácil a través de la masa monetaria, tipos de interés, etc, debido a estos cambios es por lo que tenemos un ciclos de económicos.

Por esta razón, cuando hay una contracción de capital, se produce una contracción de la economía y surge debido a que ya no hay suficiente dinero y/o crédito para gastar en bienes y servicios. Estas contracciones se dividen en dos tipos:

-Recesiones: contracciones dentro de un ciclo de deuda a corto plazo.

-Depresiones: contracción económica dentro de un ciclo de desapalancamiento.

Short-term debt cycle: Comúnmente llamado "business cycle" o ciclo económico, se compone de un aumento de la demanda respecto a la oferta lo que lleva a un incremento de los precios y una expansión económica. Sin embargo, el aumento de la demanda (creada principalmente por el crédito) conlleva un aumento de la inflación. Normalmente la recesión ocurre cuando el crédito privado se reduce derivado del cese de una política expansiva de los Bancos Centrales a través de las tasas de interés para controlar la inflación.

Cuando la inflación se encuentra controlada, se vuelven a bajar los tipos de interés para volver a estimular la demanda a través del crédito. He de señalar que las tasas de interés determinan cuánto cuesta el servicio de crédito, por ende, unos tipos de interés bajos harán más accesible el acceso al crédito.

Long-term debt cycle: Una depresión surge de un incremento de las deudas por encima de los ingresos debido a que los costes del crédito crecen de una forma excesiva a pesar de que las tasas de interés se encuentran ya reducidas al máximo para paliar la carga de deuda.

De esta forma surge el proceso de desapalancamiento, que no es más que un proceso donde la carga de deuda se va reduciendo.

Las herramientas empleadas para reducir este desapalancamiento son:

- reducción de la deuda

- austeridad

- redistribución de riqueza

- monetización de la deuda

Una depresión es la fase de contracción económica dentro de un ciclo de desapalancamiento, ocurre cuando a pesar de reducir los costes del crédito la deuda no puede reducirse.

Normalmente la medida estrella es la monetización de la deuda, por la cual el banco central compra activos financieros a través de la impresión de dinero para compensar la disminución del crédito del sector privado, elevando la masa monetaria que afecta directamente a los ahorros no sujetos a mercados.

Las tres grandes fuerzas que dirige la actividad económica son:

1- La línea de crecimiento de la productividad: Es la tendencia principal y se considera alcista, la define la productividad con un crecimiento medio anual del 2%. La productividad esta ligada al conocimiento, mayor conocimiento más productividad. ¿Sin embargo, porque cae la productividad a pesar de que el conocimiento sigue siendo el mismo que antes? Ray Dalio explica que estas variaciones se deben a los ciclos de deuda a corto y largo plazo que hemos explicado antes.
2- Long Term Debt Cycle
3- Short-Term Debt Cycle.

De acuerdo con Ray Dalio, los mercados se encontrarían actualmente saliendo de una depresión creada por los altos niveles de deuda durante las décadas previas a 2008, y a su vez en una fase actual avanzada del ciclo económico "empresarial" donde se esperaría a corto-medio plazo una pequeña recesión cíclica como ajuste del propio sistema. Esto implicaría que, aunque la tendencia primaria de fondo en los mercados es alcista, el riesgo de fondo también se está incrementando gradualmente según avanzamos en él tiempo.

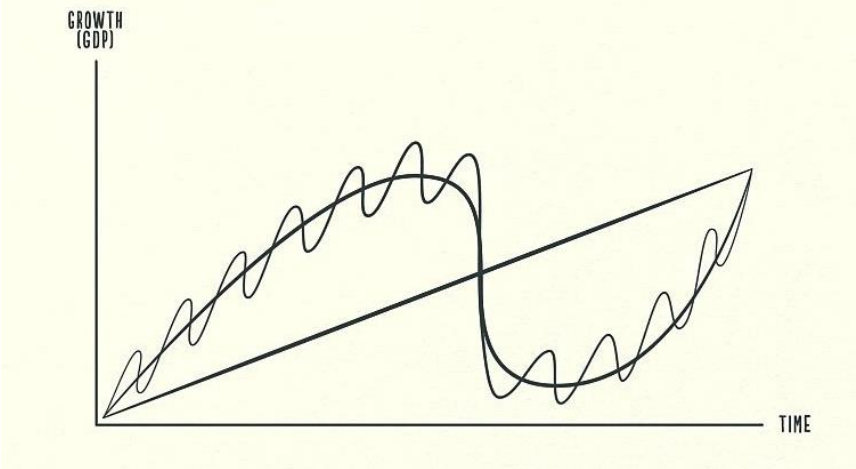

Figura 39: Economic long/short cycles *[19]*

5.3 Análisis de las variables y métricas a emplear en el modelo.

Muchos analistas tratan de crear modelos que puedan predecir el precio del día siguiente, debido al componente altamente aleatorio de los mercados en el corto plazo esta tarea resulta en muchos casos insatisfecha, por ello la mejor estrategia es desarrollar modelos que busquen inferir la función de distribución que está siguiendo el propio mercado en ese intervalo, es decir, la tendencia. Sin embargo, no se debe dejar a un lado como varia la propia distribución a lo largo del tiempo, puesto que en muchos casos nos pueden indicar información muy útil sobre que puede estar ocurriendo. Por lo tanto, podemos distinguir dos tipos de modelos, basados en la variancia del precio y por otro lado tendenciales.

Utilizaremos modelos autorregresivos de la familia ARIMA y a su vez modelos no lineales obtenidos con ML y RNN.

Los datos que se emplearan estarán en escala nominal, debido a que no estarán ajustados a la inflación, también estarán representados sin escala logarítmica debido a que está demostrado, como se indica en la *figura 40* , que no hay grandes diferencias en la autocorrelación con retornos compuestos , pero donde realmente sí se aprecian diferencias significativas es a la hora de tratarlos en valores absolutos.

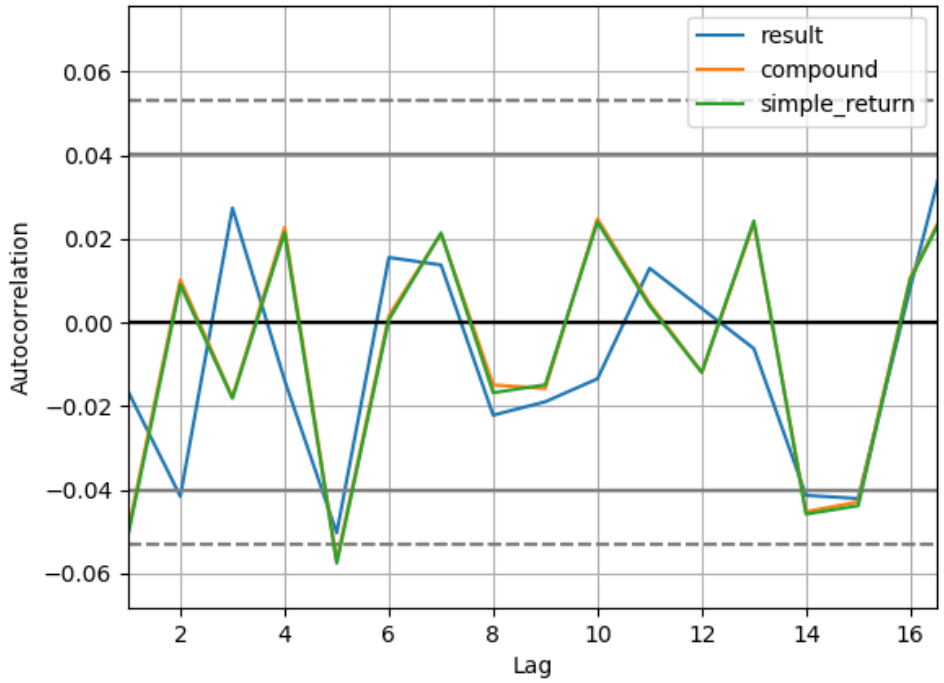

Figura 40: Autocorrelación de diferentes tipos de retornos.

$$x_t(compound) = \log(z_t + d_t) - \log(z_{t-1}) \quad (10)$$

Donde d es el dividendo si lo hay y z_t es el precio en un instante de tiempo.

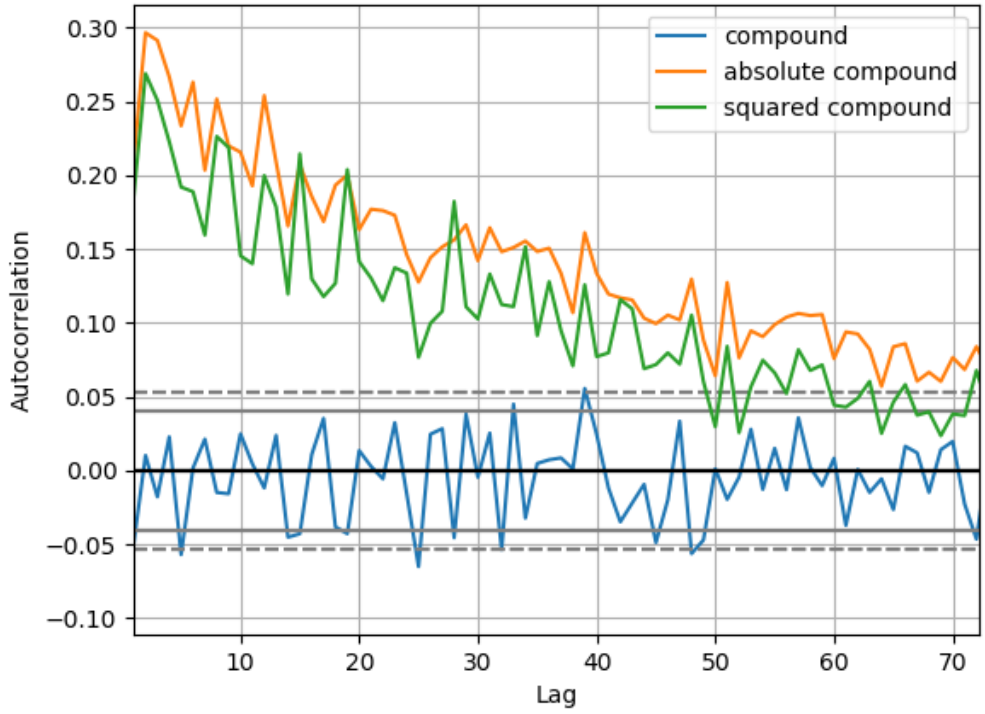

Figura 41: Autocorrelación en retornos compuestos vs retornos compuestos en valores absolutos o cuadráticos

Una posible interpretación de los resultados, es que los retornos absolutos grandes son más probables de ser seguidos por otros retornos grandes que por retornos absolutos pequeños, una posible explicación son los cambios en las varianzas Fama(1965) o como expreso Mandelbrot (1963) *"large changes tend to be followed by large changes, of either sign, and small changes tend to be followed by small changes."*.

Tal y como ya se ha explicado en el punto anterior a nivel macroeconómico, los mercados financieros pueden contener una tendencia dada y esta puede ser explicada a modo de ciclos económicos, pero también puede ser expresada a nivel estadístico.

Por lo tanto, para desarrollar los modelos posteriores, se van a asumir las siguientes **hipótesis**:

1- No es estrictamente ruido blanco SWN: Al existir alta autocorrelación entre los retornos diarios t y t-1 prueba de forma clara que no se trata de un proceso SWN a pesar del gran componente aleatorio a corto plazo, además de que observaremos como la varianza no permanece constante en el tiempo.
2- Es no-lineal: Tal y como veremos más adelante en las gráficas de correlaciones, ninguna se comporta de forma lineal y es lógico pensar esto debido al gran número de variables que pueden afectar al precio.
3- Sus funciones de distribución se caracterizan por tener colas largas y kurtosis negativa que permite el crecimiento a largo plazo generalmente de los activos.
4- Las series temporales financieras vienen expresas en la mayoría de los casos por un proceso estocástico X_t, es decir los datos varían con el tiempo, siendo a veces estacionarios o no, estando sus variables correladas o no en función de este.
5- Generalmente no son estacionarios: la media teórica de la serie temporal y su varianza varían con el tiempo, debido entre otras causas a la inflación que eleva los precios progresivamente.

Cuando se construye un sistema compuesto por diferentes variables es necesario no centrarse solo en analizar la varianza individual de cada una de ellas, sino analizar las covarianzas que puedan existir. La varianza en definitiva indicará como el precio varía con el tiempo, sin embargo, a la hora de analizar la serie se deberá ver más allá de la media y la varianza, ya que estas por si mismas no aportan gran información, en el siguiente ejemplo vemos dos distribuciones con idéntica media y desviación típica, pero con distribuciones muy diferentes.

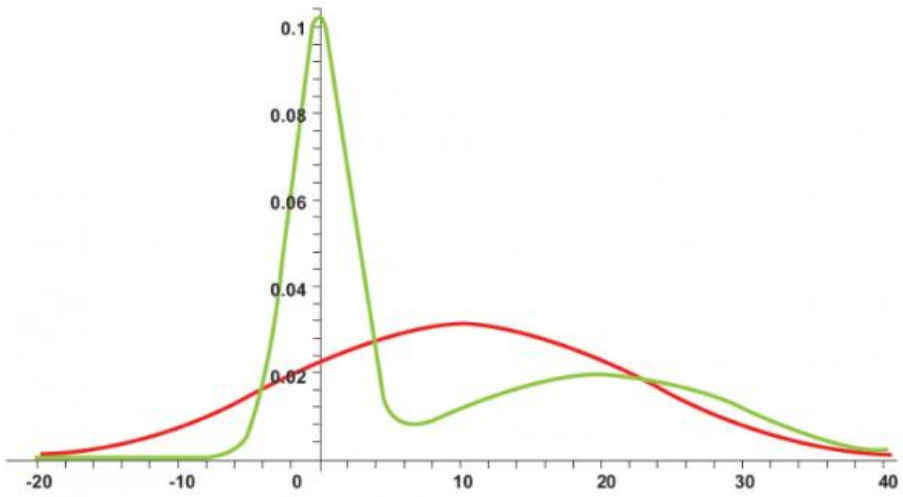

Figura 42 Distribuciones diferentes, pero con idéntica media y varianza *[20]*

De la misma forma, se necesitará utilizar también otros dos fenómenos que podrán indicar si la serie temporal tiene potencial para ser rentable:

- Skewness: mide los niveles de asimetría de una distribución entorno a su media, valores positivos de skeness indican que la distribución es asimétrica en sus colas hacia valores positivos y viceversa.

- Kurtosis: mide el nivel de aplanamiento de la curva de una distribución normal, teniendo un valor cercano a 3, valores superiores a este indican colas menos gruesas y curva apuntada, valores inferiores a 3 indican colas más gruesas y curva menos apuntada

En conclusión, se deberá buscar mercados con skewness negativa ya que permitirá obtener retornos positivos con mayor frecuencia, aunque estos estén cercanos a su media a pesar de una ligera probabilidad a sufrir pérdidas grandes, por otro lado, buscaremos valores de kurtosis cercana o superior a 3 para evitar la alta volatilidad.

Como se puede observar, se trata de analizar cuanto riesgo se está dispuesto a asumir. Estrategias que contemplen mayor riesgo implicara operar sobre distribuciones con mayor probabilidad de sufrir pérdidas menores en compensación de sufrir grandes ganancias (skewness positiva - Stocks). En conclusión, no existe una regla única puesto que al fin y al cabo es cuestión de probabilidades, pero será de utilidad a la hora de medir el riesgo de la estrategia.

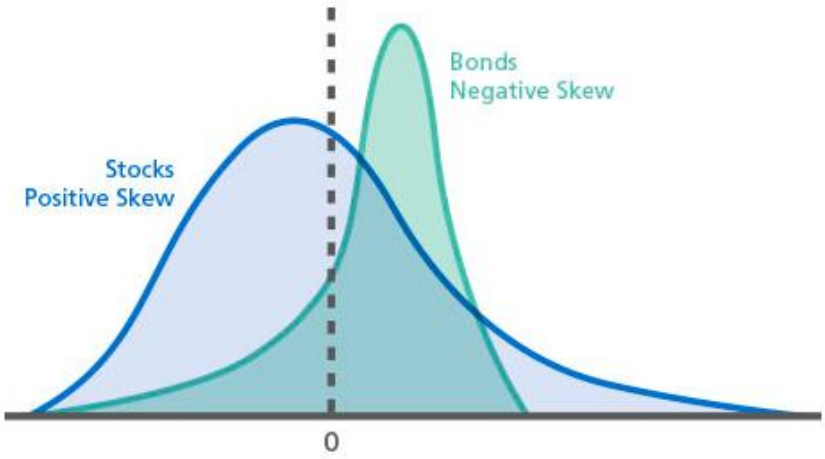

Figura 43: Riesgo de inversión según la distribución de los retornos del producto.

En nuestro caso de estudio, el índice DJIA presenta una ligera skewness negativa y un valor de curtosis grande como se puede apreciar al ser una distribución puntiaguda *figura 44*.

Deberemos esperar por lo tanto retornos no muy alejados de su media con una ligera probabilidad de sufrir retornos negativos altos, obtener un Alpha o rendimiento por encima del mercado en estas condiciones será complicado.

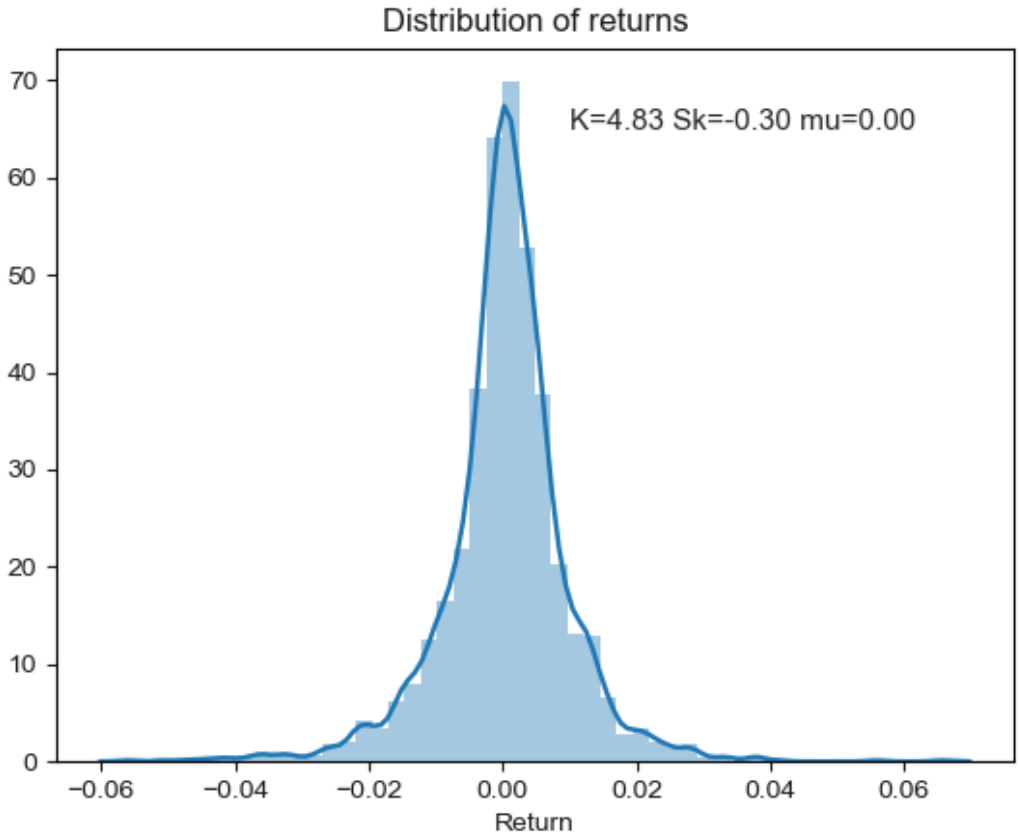

Figura 44: Distribución de los retornos del índice DJI.

5.4 Indicadores Tecnicos.

a) MACD (Moving Average Convergence Divergence): se trata de un indicador técnico que permite observar tendencias y microtendencias en un activo financiero debido al uso de diferencias entre medias móviles exponenciales.

$$MACD = EMA(x) - EMA(y) \quad (45)$$

$$Señal = EMA(9, MACD) \quad (46)$$

$$Histograma = MACD - Señal \quad (47)$$

Donde x e y son los periodos que deseemos utilizar.

b) RSI (Relative Strength Index): se trata también de un indicador del tipo oscilatorio que permite analizar la fuerza del precio a través del análisis de sucesivos precios de cierre, de esta forma se puede observar si el activo se encuentra sobrecomprado cuando supera

valores de 90% o sobrevendido cuando se encuentra por debajo del 30%. Si el precio del día es al alza se utiliza el primer caso, si es a la baja el segundo.

$$\begin{cases} \begin{cases} U = cierre_{hoy} - cierre_{ayer} \\ D = 0 \end{cases} \\ \begin{cases} U = 0 \\ D = cierre_{ayer} - cierre_{hoy} \end{cases} \end{cases} \quad (48)$$

$$RS = \frac{EMA[N]\ de\ U}{EMA[N]\ de\ D} \quad (49)$$

$$RSI = 100 - 100 \times \frac{1}{1 + RS} \quad (50)$$

c) MA (Medias Móviles) y EMA (Medias Móviles Exponenciales) : En el primer caso se trata de una media móvil, donde según se añaden más observaciones a la serie el promedio se ve desplazado, por otro lado, en el caso de las medias exponenciales además se le otorga mayor peso a las observaciones más recientes *figura 45*.

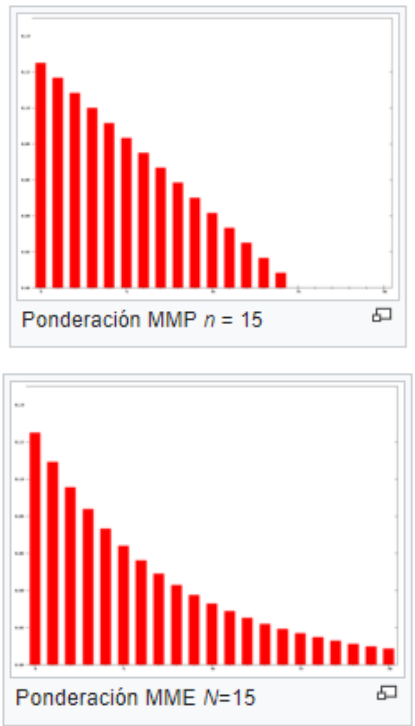

Figura 45: Diferencia entre MA y EMA

En el siguiente grafico podemos ver los diferentes indicadores explicados anteriormente actuando sobre el índice DJIA en diario.

Figura 46: Indice Dow Jones con indicadores MACD, EMA, MA y RSI.

5.4.1 Estrategias Especulativas utilizando análisis Técnico

En este punto se va a introducir algunas estrategias muy básicas pero efectivas para operar y analizaremos la precisión y resultados de éstas sobre nuestro dataset, la idea es ver realmente su efectividad y si podría ser interesante emplearlos en futuros modelos.

a) Cruce de medias móviles: Se trata de una técnica muy básica donde se definen dos medias móviles (una lenta como por ejemplo 30 periodos y otra rápida como por ejemplo 10), cuando la media rápida corta al alza a la lenta se supone una tendencia alcista, cuando corta a la baja supone una tendencia bajista.

b) Corte 0 del MACD: Cuando el histograma llega a 0 después de un mínimo se presupone una tendencia alcista, si llega después de un máximo se presupone una tendencia bajista.

   b.1) 0MACD w BULL: Si el MACD tiene un valor superior a 0 pero además en el día anterior creció un X % se compra.
   b.2) 0HIST w BULL: Igual que el caso anterior solo que con el histograma.

c) RSI: Si el RSI <=x se compra, si el RSI es > y se vende, donde x e y suelen tener valores de 35,85

La siguiente prueba, se ha realizado a lo largo de todo el dataset disponible (01-01-2008 / 15-05-2018).

Técnica	Precisión	Retorno al final del test
CR(10,30)	54.73%	5560€
0HIST	56.21%	8650€
0HIST w UP_lastday 10%	57.80%	4000€
0 HIST w UP_lastday 20%	57.86%	3343€
RSI(35,85)	52.89%	5030€
0MACD	55.54%	8250€
0 MACD w UP_lastday 10%	57.04%	4357€
0 MACD w UP_lastday 20%	57.28%	2745€
CR(7,EMA 15)	54.86%	8200€
CR(10,EMA 30)	54.77%	10530€
CR(EMA 10,EMA 30)	54.47%	6580€
CR(10,EMA 30) & 0HIST w UP_lastday 10%	60.44%	3110€
HIST UP >0% (tendencia alcista)	55.72%	8958€
HIST UP >0.5% & RSI <=70 (tendencia alcista suavizada)	55.83%	5943€

Tabla 3: Resultados sobre test de diferentes estrategias técnicas (solo compra)

Como se puede observar a la vista de los resultados en la tabla 2, es posible realizar modelos especulativos rentables en el periodo testeado mediante reglas if-else a partir de las señales o datos obtenidos con los diversos indicadores técnicos. Sin embargo, aunque los resultados en todos los casos sean positivos, solo 3 modelos, marcados en la tabla 2 con fondo azul, han obtenido ganancias con un nivel de varianza reducida que permitan su viabilidad a largo plazo.

Estos 3 modelos HIST UP >0% o tendencia alcista, HIST UP >0.5% & RSI <=60 o tendencia alcista suavizada y por último el cruce de medias CR(10,EMA 30) han demostrado como se ve en las gráficas una ganancias más constante y menos volátil. Por otro lado, se han efectuado combinaciones entre ellas sin resultados positivos.

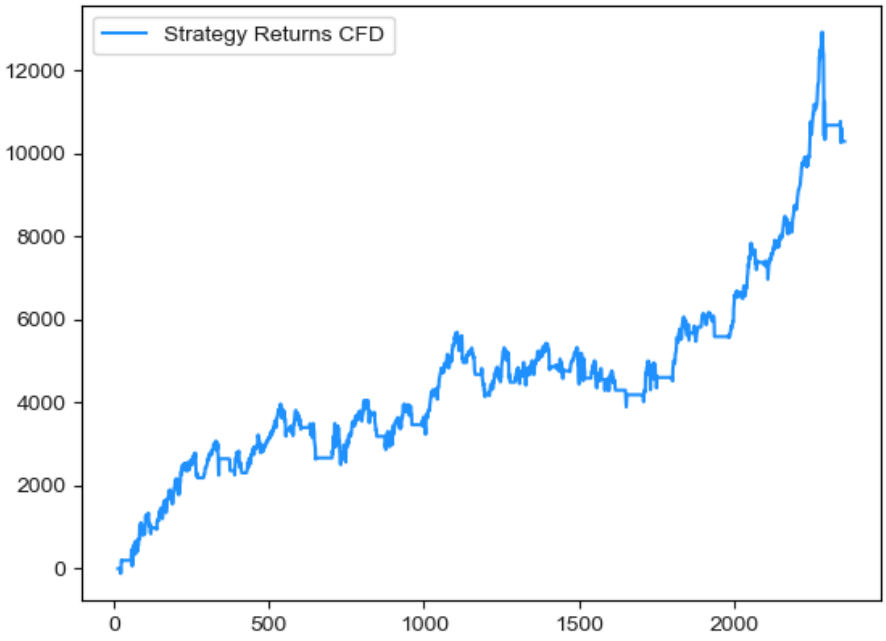

Figura 47: CR(10,EMA 30)

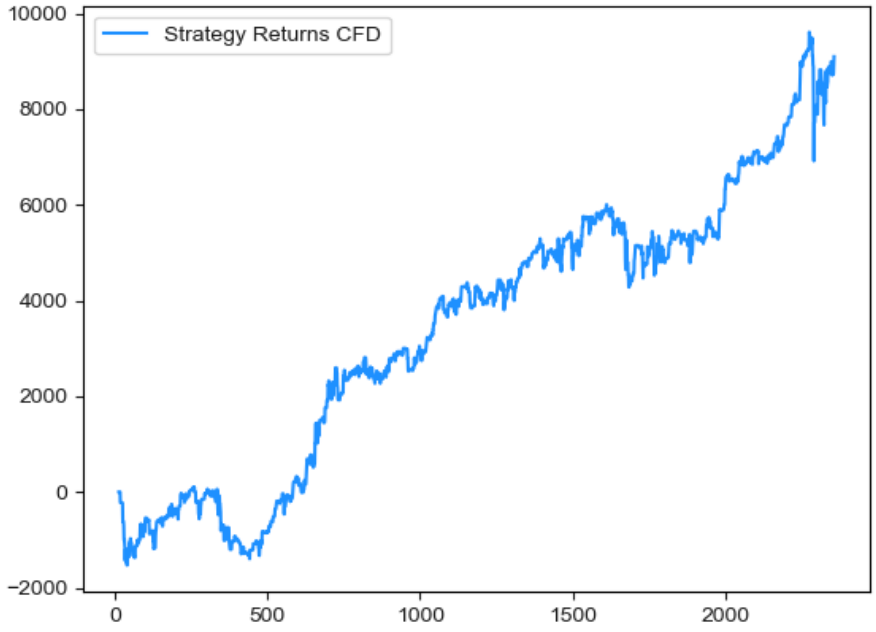

Figura 48: HIST UP >0% (tendencia alcista)

En el siguiente modelo se ha utilizado el RSI como suavizado para evitar zonas sobrecompradas, reduciendo así las pérdidas y el riesgo.

Figura 49: HIST UP >0.5% (tendencia alcista) & RSI <=60

5.5 Econometría financiera: modelos ARIMA

ARIMA(p,d,q) donde p es el orden del polinomio autorregresivo estacionario, d es el orden de integración de la serie, es decir el número de diferencias que hay que tomar a la serie para que sea estacionaria y q es el orden del polinomio de medias móviles invertible.

El orden de integración del modelo, se basa en el número de diferencias que hay que tomar del proceso para conseguir la estacionariedad en media, o dicho de otra manera el número de raíces unitarias de la serie.

$$\varphi(L)(1-L)^d Yt = \delta + \vartheta(L)at \quad (51)$$

La aplicación de la metodología Box-Jenkins se basa en determinar el modelo Arima(p,d,q) que ha podido generarla, es por lo tanto un proceso de inferencia.

La construcción de modelos ARIMA de lleva a cabo iterando cuatro pasos:

a) Identificación: El objetivo es determinar mediante la información existente los órdenes p,d,q que parecen apropiados para reproducir las características de la serie y si se incluye o no la constante δ.

   Se realiza en dos fases:

1. Análisis de la estacionariedad:

   a) Estacionariedad en varianza: transformaciones estabilizadoras de varianza, sino es estable se usa las transformaciones Box-Cox (funciones del tipo raíz cuadrada, inversa, logarítmica, etc).

   b) Estacionariedad en media: número de diferencias d que hay que tomar para que la serie sea estacionaria en media. Se ha de comprender que una serie es estacionaria en media si la serie temporal fluctua alrededor de una media constante, si sacamos su función de autocorrelación veríamos como su proceso cae exponencialmente.

   En caso de que no sea estacionaria se podrán tomar d diferencias de orden 1:

$$Z_t = Y_t(1-L)^d \quad (52)$$

2. Elección de ordenes p y q. Una vez obtenida la serie estacionaria, el objetivo es determinar el proceso estacionario ARMA(p,q) que lo haya generado.

Por lo tanto, para poder realizar este paso usaremos los gráficos de la serie original y sus transformaciones para observar su estacionariedad y autocorrelacion para ver si decrece rápidamente, una gráfica donde se pueda observar como los datos dependen del tiempo, se podrá argumentar que en principio no es estacionaria la serie.

	FAC	**FACP**
$MA(q)$	Se anula para $j > q$	Decrecimiento rápido No se anula
$AR(p)$	Decrecimiento rápido No se anula	Se anula para $j > p$
$ARMA(p,q)$	Decrecimiento rápido No se anula	Decrecimiento rápido No se anula

Figura 50: tabla de decisiones de parámetros para ARIMA *[21]*

b) Estimación: Usando los datos se realiza la inferencia sobre los parámetros condicionando que el modelo sea apropiado.

c) Validación: Se realizan contrastes de diagnóstico para comprobar si el modelo se ajusta a los datos y si no es así, ver las discrepancias para mejorarlo.

d) Predicción: Obtener pronósticos a partir del modelo.

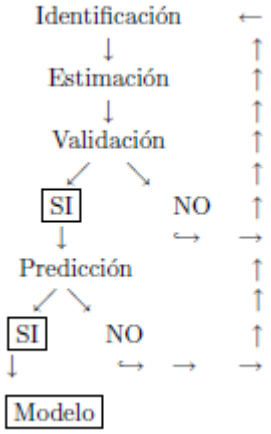

Figura 51: Cronograma de decisiones para desarrollo global del modelo *[21]*

Es importante señalar que el modelo ha de ser construido siguiendo el principio de parametrización escueta o parsimonia, donde se trata de construir un modelo que pueda representar la serie con el mínimo de parámetros posibles y únicamente ampliarlo en caso de que sea necesario.

$$Y_t = \underbrace{\phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p}}_{\text{parte autorregresiva}} + a_t \underbrace{- \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ldots - \theta_q a_{t-q}}_{\text{parte medias móviles}}$$

Figura 52: Formula general ARIMA *[21]*

5.5.1. Desarrollo de un modelo ARIMA para el índice DJIA

Como ya hemos explicado anteriormente y utilizando las hipótesis desarrolladas en el punto 5.3, crearemos un modelo ARIMA para tratar de predecir el precio y tendencia del día siguiente. Sin embargo, trataremos de seguir y comprobar si estas se cumplen.

1. Análisis de estacionariedad:

   Para realizar el test emplearemos el test de Augmented Dickey-Fuller, donde un valor p resultado del test ADF inferior a un 5% implicaría que hay una componente estacional.

   Al realizar el análisis de Dickey-Fuller directamente sobre el precio, la hipótesis de no estacionalidad se cumple, pero por ello emplearemos los retornos calculados, ya que estos no varían en exceso en el tiempo, además no será necesario calcularlos con la transformación logarítmica, como ya se explicó anteriormente.

   Realizando el test de Dickey-Fuller sobre los retornos, obtenemos que el valor de p < 5%, por lo tanto, los retornos si contienen una componente estacional, si además nos fijamos en el valor dado ADF -10.39, cumple que es menor que el 1%, por lo tanto cumple la hipótesis de estacionariedad con un 99% de probabilidad. Por tanto, no será necesario añadir transformaciones o diferencias para conseguir estacionalidad, estamos ante un modelo ARMA(p,0,q).

```
ADF Statistic: -10.394275
p-value: 0.000000
Critical Values:
        5%: -2.863
        1%: -3.433
        10%: -2.567
```

Figura 53: Test de Dickey-Fuller

2. Test de autocorrelación parcial (PACF) y completa (ACF):

El test de autocorrelación ACF, se basa en la correlación lineal existente entre los valores de la serie que se encuentran a "k lags".

La diferencia entre ambas es que en PACF, mide el grado de asociación lineal existente entre las variables entre "k-lags" una vez ajustado el efecto lineal de todas las variables intermedias.
En un proceso estocástico estacionario típico en mercados financieros la ACF deberá decrecer rápidamente hacia 0 señalando el orden p del modelo AR, de la misma forma el PACF señalara el orden q del modelo MA.

Mediante las gráficas podremos analizar el orden autorregresivo (AR) y el orden de media móvil (MA), tal y como se indica en la *figura 54 y 55*.
Podemos observar una serie donde PACF y ACF decrecen rápidamente hasta 0 sin anularse, ello implica junto al conocimiento que ya se dispone de la existencia de una componente estacional intrínseca a la serie, que se trata de un modelo AR(p)+MA(q), analizando las gráficas observamos que el número de lags a tener en cuenta donde existe correlación distinta de 0 es en el lag 1 tanto para AR como para MA, tal y como se puede observar también en la *figura 41*.

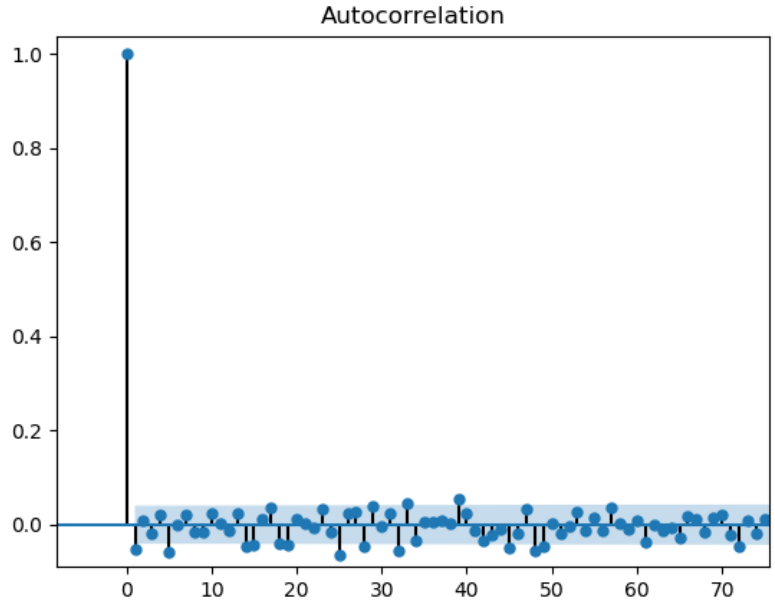

Figura 54: ACF de los retornos no logarítmicos

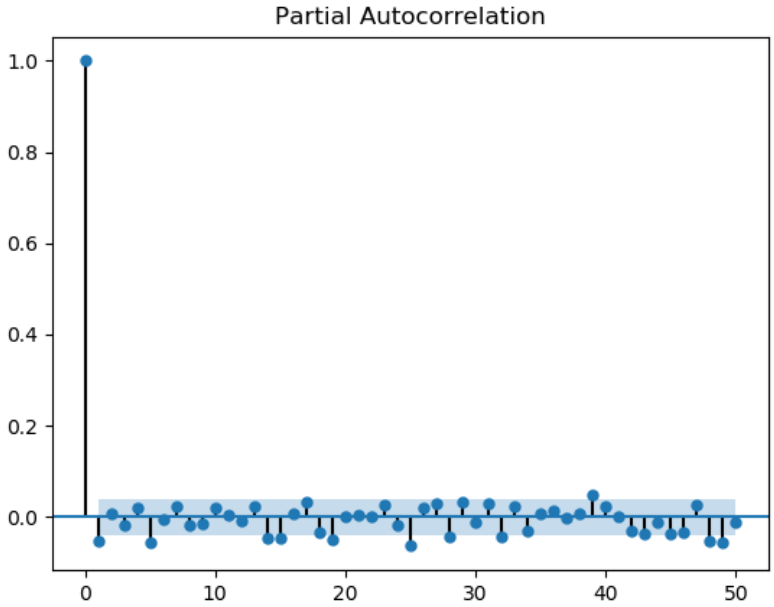

Figura 55: PACF de los retornos no logarítmicos

3. Propuesta de modelo y análisis:
   Tras realizar el análisis de las graficas ACF y PACF y decidir los órdenes de ARMA (1,0,1) , programamos el modelo importando ARIMA de la librería de Python statsmodels e introduciendo sus órdenes (1,0,1).

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 2001
Model:                     ARMA(1, 1)   Log Likelihood                6391.867
Method:                       css-mle   S.D. of innovations              0.010
Date:                Mon, 30 Jul 2018   AIC                         -12775.734
Time:                        19:21:43   BIC                         -12753.328
Sample:                             0   HQIC                        -12767.507

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0005      0.000      2.148      0.032    4.03e-05       0.001
ar.L1.y       -0.7327      0.123     -5.971      0.000      -0.973      -0.492
ma.L1.y        0.6805      0.132      5.163      0.000       0.422       0.939
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -1.3648           +0.0000j            1.3648            0.5000
MA.1           -1.4695           +0.0000j            1.4695            0.5000
-----------------------------------------------------------------------------
```

Figura 56: Resultados del modelo ARMA (1,0,1).

4. Resultados:

Tras el estudio de la serie temporal y proponer un modelo ARMA(1,0,1) fue testeado alcanzando una precisión en la predicción de la tendencia del retorno del día siguiente de un

55.5%. El resultado no es malo, pero bastante mediocre teniendo en cuenta que estamos ante una serie estacional por si mismo que no necesitaba de transformaciones.

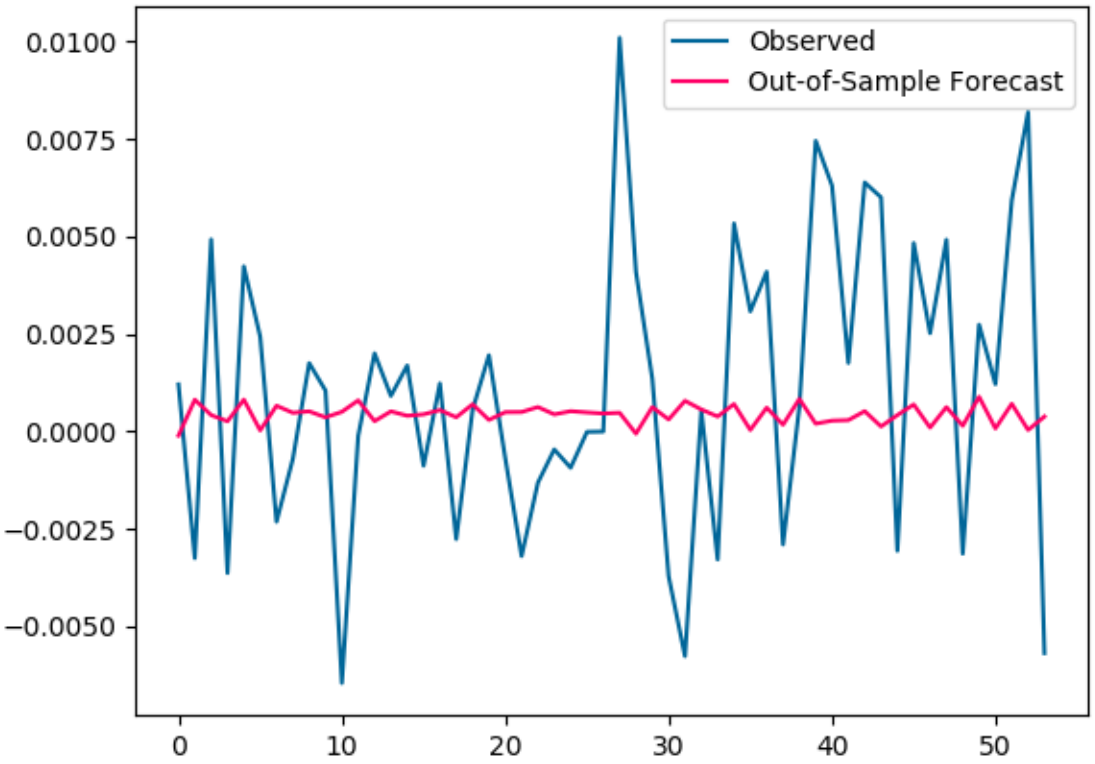

Figura 57: Resultados del test para el modelo ARMA(1,0,1).

Aunque aparentemente el modelo no cuadra bien con el test, es necesario no cometer el error de añadir ordenes extra solo por ver si cuadra mejor el patrón. Si se ha realizado correctamente el análisis teórico de la serie, el modelo obtenido será el mejor posible para los datos disponibles.

5.6 Machine Learning

En este apartado se tratara de desarrollar un algoritmo ML de compra que permita utilizar los datos y técnicas empleadas en el algoritmo desarrollado con reglas if-else. Podremos observar si la IA es capaz de generar un reconocimiento más preciso de este patrón y establecer sus propias reglas más allá de comprar cuando la diferencia porcentual del histograma del MACD es superior al día anterior.

Utilizaremos un clasificador SVM-RBF con los datos previamente estandarizados bajo una misma media y varianza para evitar que alguna variable este sobredimensionada y altere el funcionamiento del algoritmo.

Las variables empleadas definitivas posteriormente al análisis y selección para maximizar los resultados serán precio del día, precio más alto y bajo del día anterior, señal MACD, diferencia % histograma con el día anterior, RSI y el valor del histograma. Se estudiará si la IA puede optimizar las decisiones o reglas IF-ELSE.

5.6.1 Histograma UP

Se empleará la técnica introducida anteriormente, si el histograma crece respecto a su día anterior se activa un flag, en nuestro algoritmo IF-ELSE si está el flag activado será una señal de compra, la IA sin embargo no recibirá esta orden solo se le indicara que se encuentra activada, veremos así si es capaz de desarrollar sus propias reglas de inversión y si estas son mejores que las nuestras.

Algoritmo	Precisión Optimized	Sharpe Optimized	Total Returns (all variables)	Optimized IA
IA	55.33%	1.035	12060€	15676€
IF-ELSE	55.98%	0.754	8850€	8850€

Tabla 4: Resultados del clasificador long SVM-RBF con MACD

Se utilizará el sistema de cross time validation con 10 observaciones para analizar la precisión global.

Nº	Precision IA	IA Returns	Sharpe IA	Precision IF-ELSE	IF-ELSE Returns	Sharpe IF-ELSE
1	58.40%	1373€	0.97	51.08%	-966€	-0.92
2	57.00%	2002€	1.55	58.88%	1944€	2.23
3	50.00%	-252€	-0.12	52.17%	1714€	1.19
4	53.12%	1464€	1.38	63.04%	1584€	1.78
5	54.74%	1367€	0.94	50.50%	-148€	-0.24
6	58.49%	2035€	1.24	59.57%	1432€	1.30
7	48.09%	-1029€	-0.40	51.08%	-551€	-0.31
8	54.58%	1621€	0.71	51.64%	330€	0.15
9	56.77%	3641€	3.10	58.62%	1603€	1.63
10	62.08%	3030€	0.97	63.26%	1870€	0.72

Tabla 5: Resultados desagregados de los 10 test realizados con el clasificador long SVM-RBF con MACD

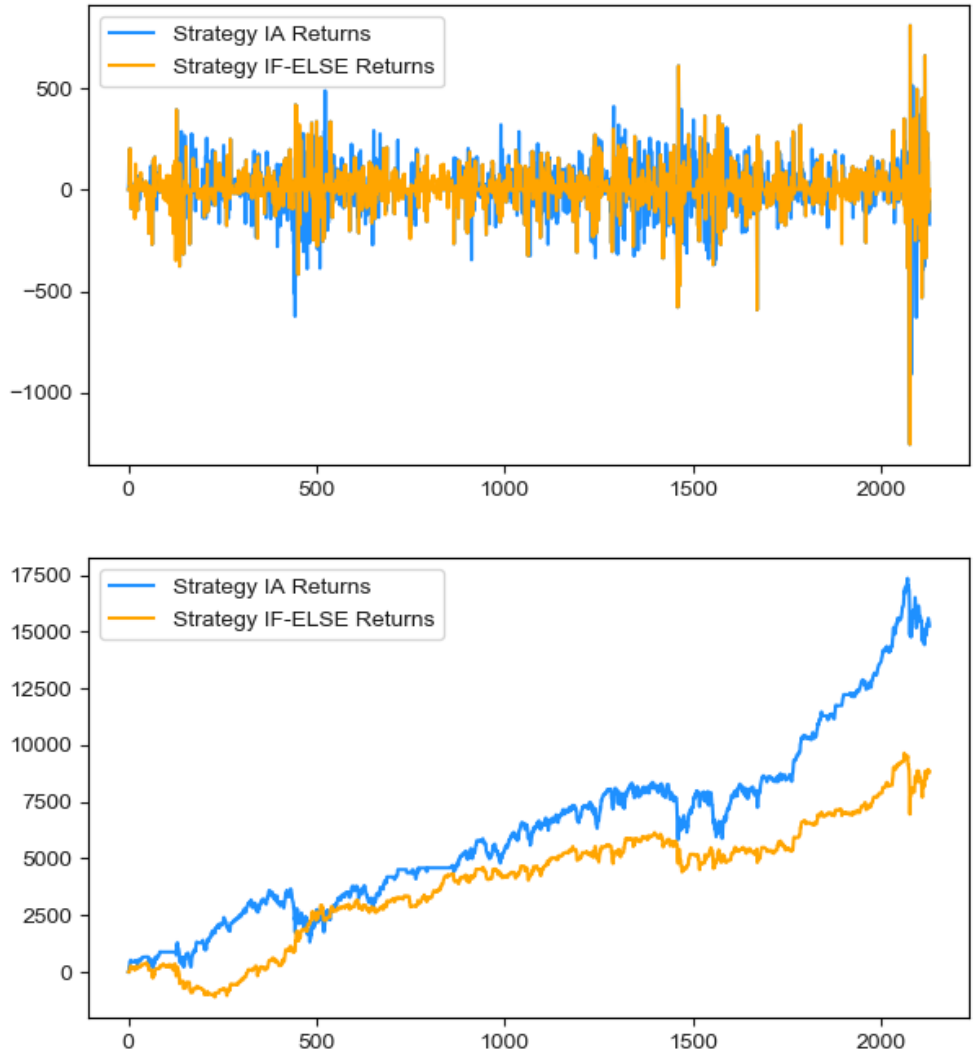

Figura 58: Retornos (imagen superior) y retornos acumulados (imagen inferior)

Como podemos observar duplica nuestro rendimiento, aunque si bien es cierto se debe sobre todo al gran rally alcista que produjo la victoria de Donald Trump. Por otro lado, en el riesgo en la distribución podemos observar mayor curtosis en los retornos de la IA, reduciendo así la volatilidad del sistema, el riesgo extremo representado por las colas, es similar en ambos modelos.

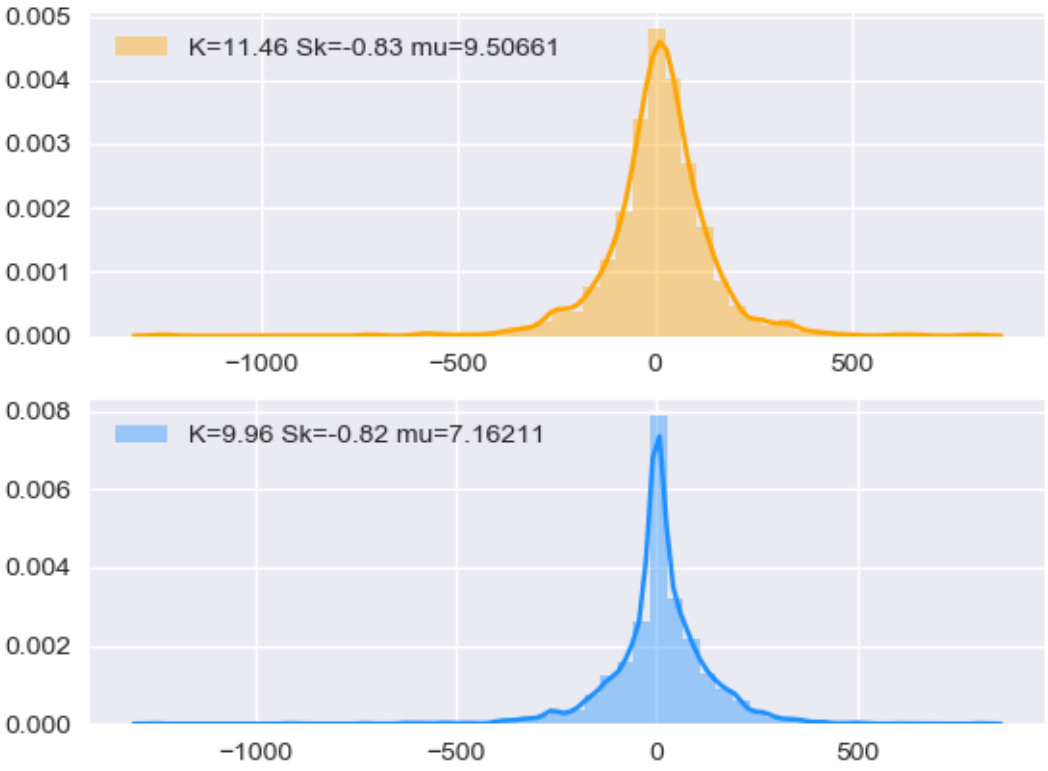

Figura 59: Función de distribución de los retornos obtenidos con el modelo IA y con IF-ELSE

El algoritmo ML que hemos desarrollado demuestra resultados superiores a nuestro modelo basado en reglas IF-ELSE, será necesario realizar en un futuro testeos en otros ciclos alcistas para observar si se obtienen resultados similares.

Para un mejor análisis en la siguiente grafica se puede ver el desempeño de los dos modelos en cada uno de los 10 test realizados en el cross time validation. En 7 de los 10 test el modelo ML demuestra unos resultados superiores, solo en 3 ocasiones el modelo IF-ELSE consiguió ligeramente superar a la IA. Por último, añado la función de autocorrelación del histograma siendo vital para mejorar nuestros resultados en la fase de optimización.

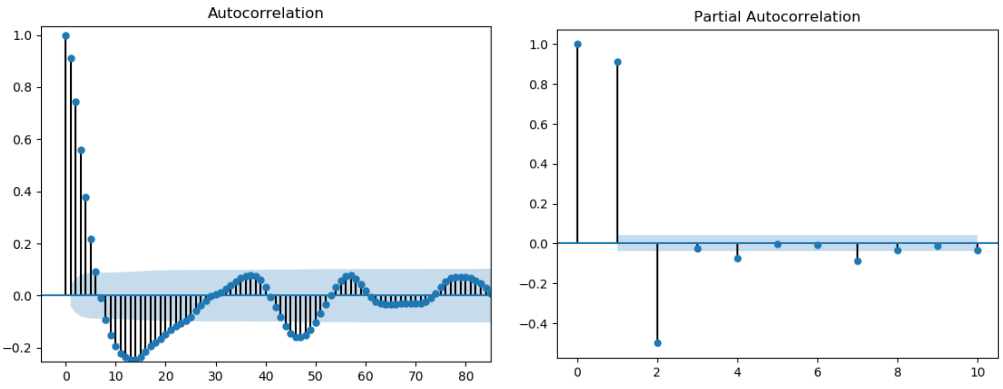

Figura 60: Funciones de autocorrelación del histograma MACD.

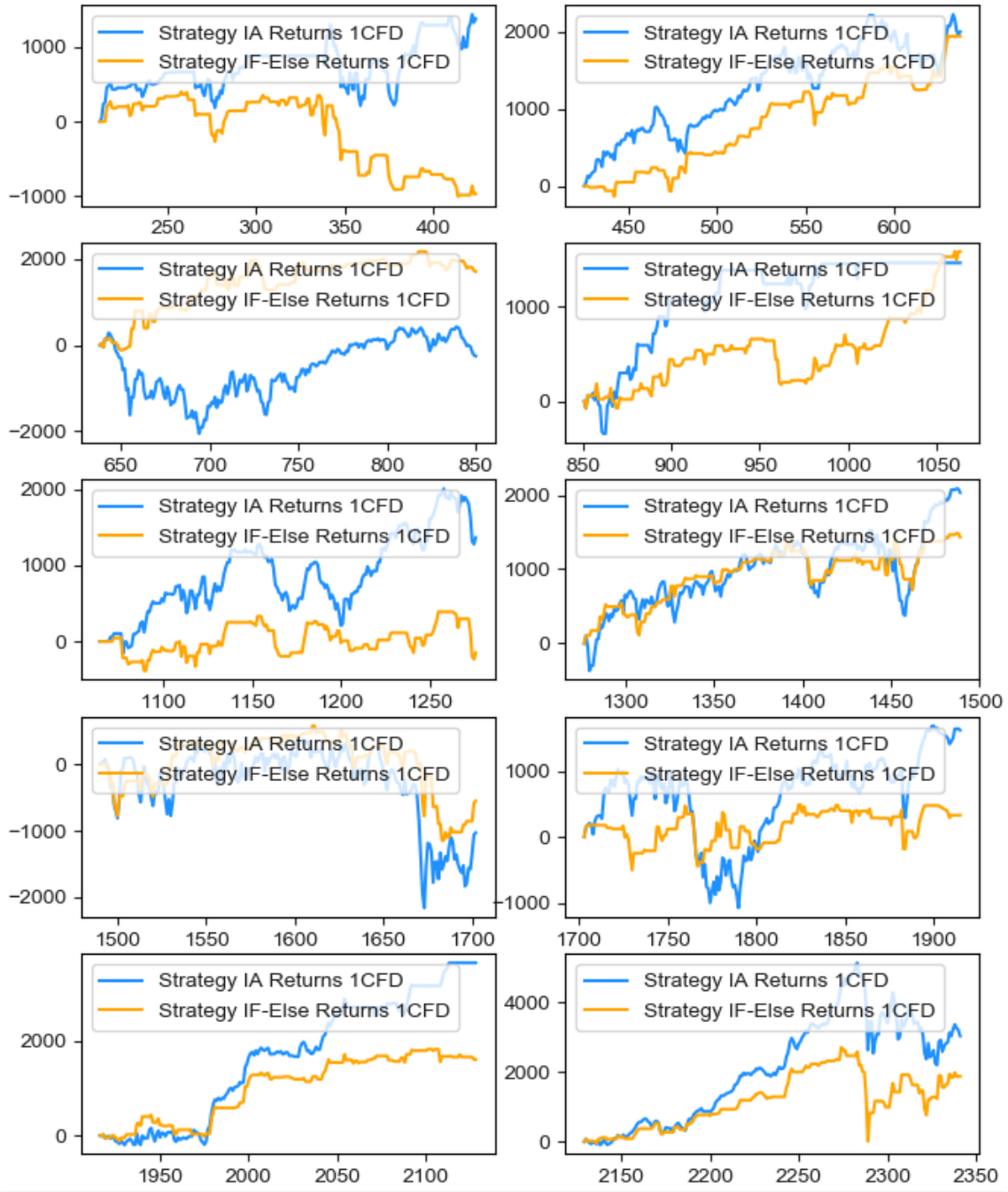

Figura 61: Resultados acumulados desagregados para los 10 test realizados.

Siguiendo con nuestro análisis pasaremos ahora analizar cómo se desenvuelve la IA bajo nuevos datos de otros indicadores técnicos como el cruce de medias móviles.

5.6.2 Algoritmo ML CM(10, EMA 30)

De forma similar al caso anterior, vamos a estudiar si la IA es capaz de crear un conjunto de reglas propias más eficientes que comprar si la MA(10) >EMA(30). En primer lugar, se probara el modelo en igualdad de condiciones, es decir solo con los datos de precio y si el flag de cruce esta activado o no, posteriormente veremos se observara si los resultados mejoran al añadir variables anteriores como el MCD o histograma y sus diferencias porcentuales.

Tal y como podemos observar en los retornos no optimizados la IA vuelve a mostrarse claramente superior con unos retornos un 52% superiores con una precisión ligeramente superior del 55.76%, un 1.56% más.

Algoritmo	Precisión Optimized	Sharpe Optimized	Returns (Not Optimized)	Optimized IA
IA	54.86%	0.874	12700€	14091€
IF-ELSE	54.24%	0.571	8329€	8329€

Tabla 6: Resultados del clasificador long SVM-RBF con cruce de medias.

Tras añadir el resto de las variables aplicadas al modelo anterior el retorno ha aumentado un 11% hasta los 14091€, los resultados para cada uno de los 10 test realizados son:

Nº	Precision IA	IA Returns	Sharpe IA	Precision IF-ELSE	IF-ELSE Returns	Sharpe IF-ELSE
1	52.89%	368€	0.20	59.55%	475€	0.35
2	56.42%	1391€	1.17	56.14%	956€	0.73
3	50.73%	365€	0.14	47.51%	-23€	-0.03
4	50.54%	1523€	1.20	53.16%	1557€	1.23
5	55.28%	1648€	1.08	51.47%	-178€	-0.21
6	59.06%	2112€	1.32	57.23%	659€	0.46
7	48.30%	-1086€	-0.43	43.01%	-1066€	-0.73
8	57.55%	2299€	1.07	52.94%	1891€	1.34
9	55.12%	1677€	1.71	56.37%	1570€	1.38
10	62.67%	3741€	1.23	65.03%	2847€	1.18

Tabla 7: Resultados desagregados de los 10 test realizados con el clasificador long SVM-RBF con cruce de medias.

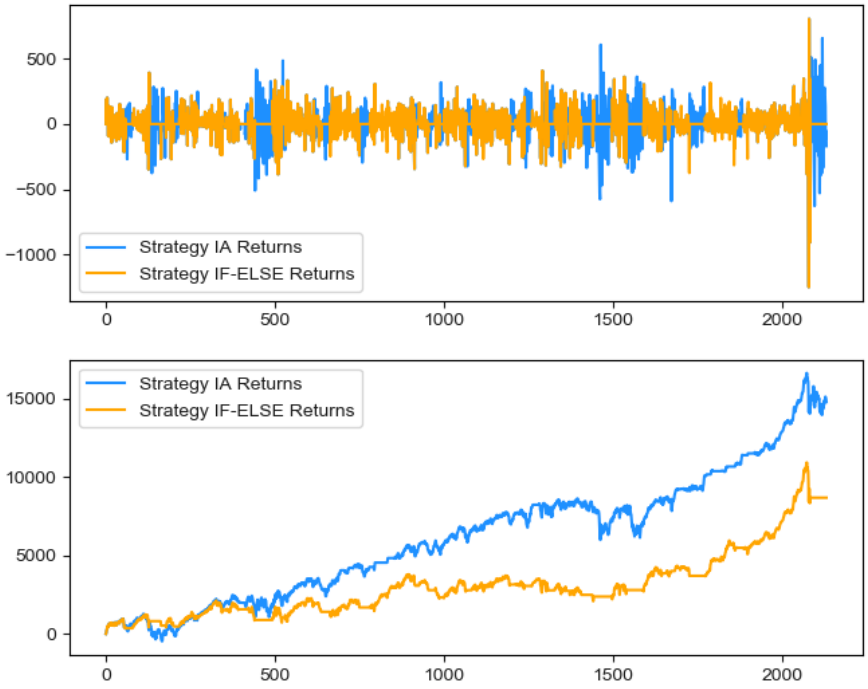

Figura 62: Retornos (imagen superior) y retornos acumulados (imagen inferior).

Podemos observar en la gráfica como la curtosis presente en la distribución de los retornos conseguidos por la IA es muy similar al caso anterior. Sin embargo, seguimos sin poder disminuir las colas de nuestra distribución aun empleando el indicador RSI que permite analizar sobre extensiones del mercado.

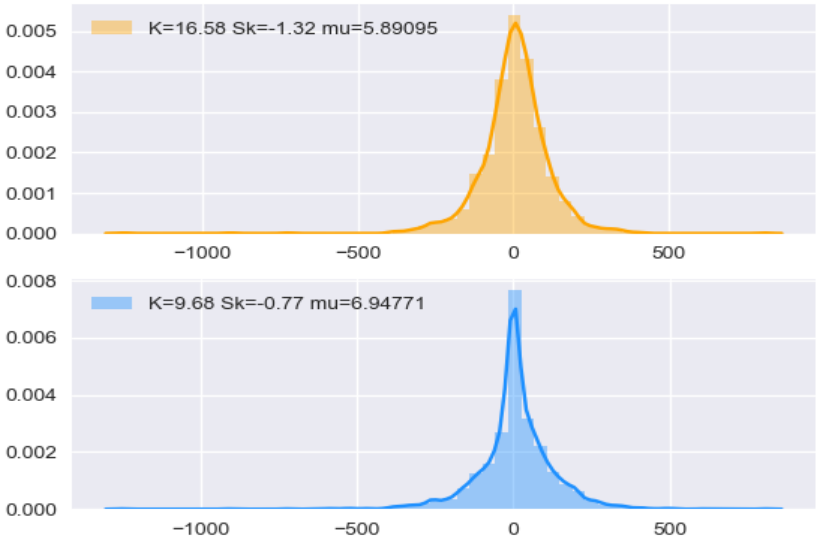

Figura 63: Función de distribución de los retornos obtenidos con el modelo IA y con IF-ELSE.

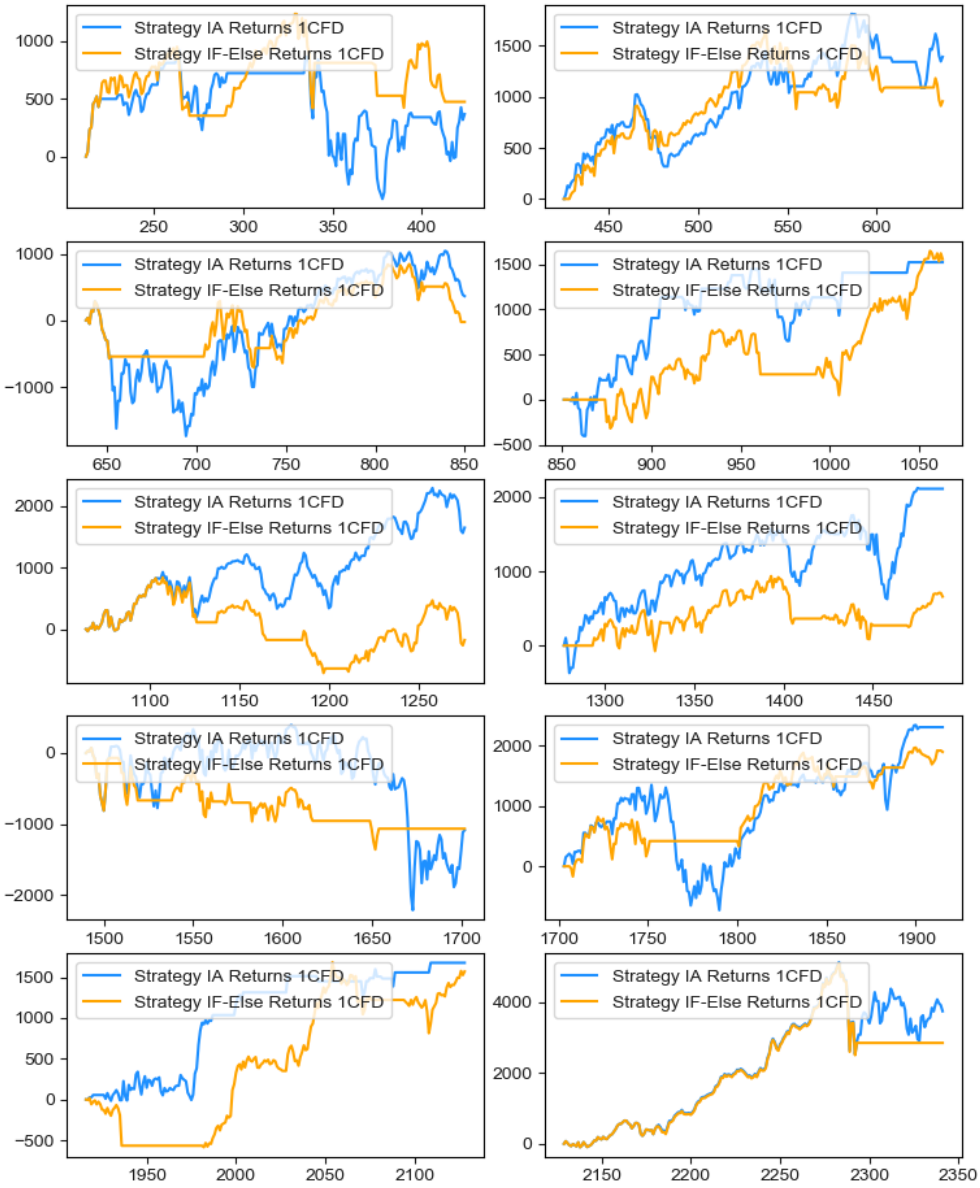

Figura 64: Resultados acumulados desagregados para los 10 test realizados.

5.6.3 Histograma UP > 5% up & RSI < 70%

De la misma forma que los dos casos anteriores analizaremos si la IA es capaz de mejorar los resultados disponiendo de las mismas variables de base que el caso anterior, que se considera ya optimizado. Tal y como se indica en el punto anterior, el suavizado incrementaba la precisión a pesar de que reducía las ganancias potenciales, pero permitía linealizar estas ganancias y hacerlo mucho más seguro a largo plazo.

Algoritmo	Precisión Optimized	Sharpe Optimized	Optimized IA
IA	54.63%	0.860	13778€
IF-ELSE	56.58%	0.578	5945€

Tabla 7: Resultados del clasificador long SVM-RBF mixto.

Como se puede observar los retornos de la IA son muy superiores gracias a obtener un patrón inherente en el resto de los datos que la regla IF-ELSE no opera.

Nº	Precision IA	IA Returns	Sharpe IA	Precision IF-ELSE	IF-ELSE Returns	Sharpe IF-ELSE
1	52.23%	305€	0.15	54.06%	-414€	-0.45
2	56.66%	1619€	1.39	60.00%	1522€	1.78
3	50.72%	397€	0.16	55.00%	2165€	1.66
4	50.92%	1543€	1.16	59.71%	665€	0.75
5	55.22%	1696€	1.12	56.83%	-62€	-0.17
6	59.25%	1880€	1.15	60.60%	1236€	1.26
7	48.30%	-1086€	-0.43	51.14%	-451€	1.26
8	58.38%	2484€	1.16	57.83%	569€	-0.26
9	52.05%	1457€	1.55	57.15%	941€	1.07
10	62.56%	3463€	1.14	53.52%	-265€	-0.19

Tabla 9: Resultados desagregados de los 10 test realizados con el clasificador long SVM-RBF mixto.

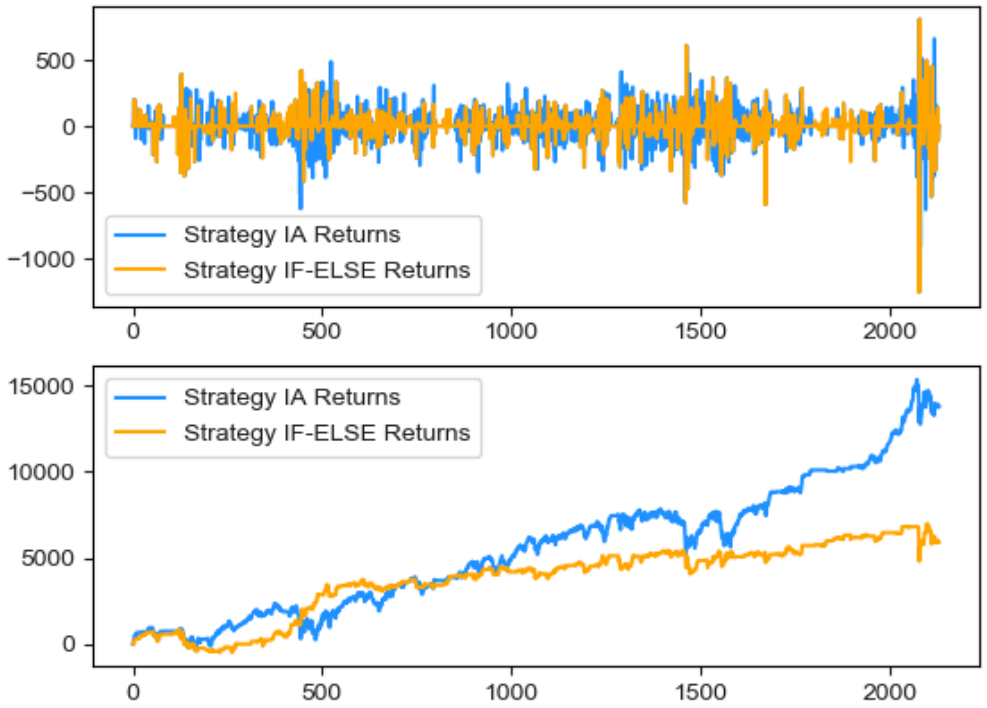

Figura 65: Retornos (imagen superior) y retornos acumulados (imagen inferior).

El riesgo de pérdidas y ganancias máximas es muy similar en ambas distribuciones quizás cabría destacar que el algoritmo IF-ELSE demuestra menos volatilidad al tener mayor curtosis.

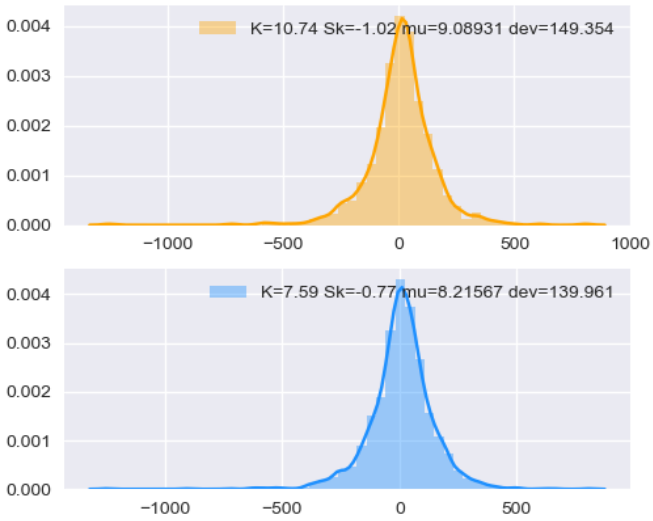

Figura 66: Función de distribución de los retornos obtenidos con el modelo IA y con IF-ELSE.

En un análisis más exhaustivo podemos observar como la IA bate al algoritmo manual en 8 de 10 test realizados mediante cross time validation.

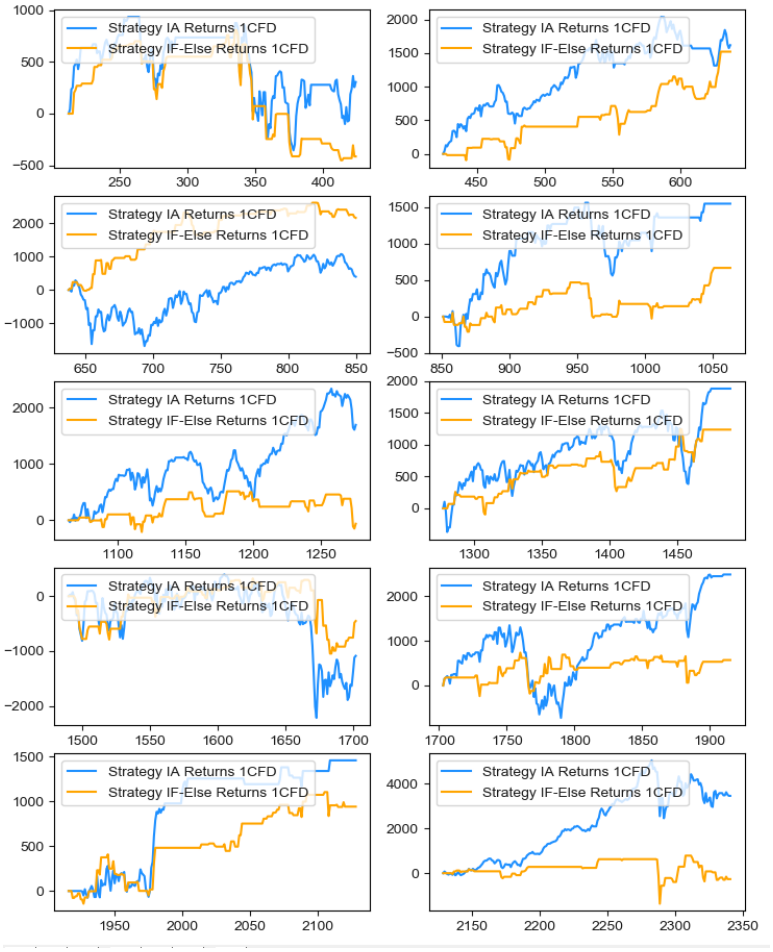

Figura 67: Resultados acumulados desagregados para los 10 test realizados.

5.6.4 Conclusiones

Se ha testeado y comparado cada una de nuestras 3 mejores técnicas para operar a corto plazo el índice JDIA contra un clasificador SVM y en en cada uno de los casos, la IA ha obtenido retornos muy superiores demostrando que los mercados financieros a pesar de ser procesos estocásticos no lineales se pueden crear modelos predictivos rentables a largo plazo e incluso generar alphas mediante técnicas de inteligencia artificial acompañada de un apalancamiento correcto. Por otro lado, se demuestra también el potencial que tiene la inteligencia artificial para optimizar modelos y donde seguiremos estudiando esta capacidad en los puntos siguientes e incluso comprobando si la IA es capaz mediante algoritmos no supervisados de detectar por si misma otros patrones nuevos que sumen valor al sistema.

5.7 Clasificador long/short Gaussian Mixture + SVM-RBF

En este apartado, se va a tratar de desarrollar un algoritmo ML de compra/venta para operar sobre diversos derivados financieros como podrían ser cfds, futuros, opciones, etc sobre el índice DJIA, donde dos algoritmos ML diferentes aportaran información del patrón subyacente, la idea ya ha sido recogida y probada en múltiples tesis *[28]* con muy buenos resultados, en este apartado se utilizará la misma idea, pero empleando nuestras variables seleccionadas.

Para ello se creará un algoritmo basado en dos inteligencias artificiales que clasificaran el contexto subyacente en el mercado. En primer lugar, la mixtura Gaussiana en función del precio de apertura, precio más alto y bajo del día unido al retorno respecto al día anterior, clasificará cada día en una región o régimen hasta el número máximo de regiones que hayamos indicado, en nuestro caso 10, este número junto con otras variables serán modificadas más adelante para comprobar si el algoritmo puede ser mejorado.

Figura 68: Clasificación de diferentes regiones mediante algoritmo mixtura Gaussiana

En segundo lugar, el clasificador RBF-SVM utilizara estas regiones junto al precio, mínimos y máximos diarios y 4 nuevos retornos para crear una señal de compra/venta.

Si la señal es -1 el clasificador nos indicara que espera que el retorno resultante del dia sea negativo, 1 si espera que sea positivo. Por ende, si es -1 realizaremos una operación “short” o de apuesta en contra del mercado, si es 1 realizaremos una apuesta a favor del mercado. Por otro lado, los cuatro nuevos retornos están calculados respecto al día anterior hasta 4 lags.

Para comprobar el correcto de funcionamiento de las predicciones se utilizara el sistema “time series cross validation” ya explicado anteriormente, con 10 particiones, es decir un 10% de test respecto al total disponible de datos para observar el desempeño global del algoritmo.

Tras ejecutar el script programado en Python obtenemos los siguientes resultados en precisión y retorno porcentual obtenido para cada una de las 10 observaciones.

Nº	Precission (%)	Strategy Comulative Returns (%)	Market Comulative Returns (%)	Sharpe Ratio
1	0.4766	13.45	6.31	0.833
2	0.5093	9.75	18.29	0.823
3	0.4813	8.64	1.7	0.387

4	0.4906	-2.81	15.9	-0.434
5	0.5327	10.14	8.3	0.953
6	0.5046	1.02	9.45	0.013
7	0.5046	-4.98	0.63	-0.377
8	0.4953	-6.18	7.72	-0.493
9	0.514	-0.8	15.41	-0.169
10	0.6261	33.36	14.25	2.654
TOTAL	51.355	61.61	98.01	0.42

Tabla 10: Resultados desagregados de los 10 test realizados con el clasificador long/short.

Como podemos observar genera un retorno total inferior al mercado, generando solo un retorno Alpha en 4 de las 10 ocasiones. Sin embargo, no buscamos batir al mercado en retornos a largo plazo, puesto que se necesitaría un sistema mucho más complejo, sino crear un sistema que genere señales de compra/venta que permita utilizar derivados financieros que permita apalancarnos y maximizar nuestras ganancias diarias.

En los siguientes gráficos se simula la utilización de un único cfd en todo el dataset, el cfd permite en resumen apostar 1€ por cada punto del índice, en una posición long si sube 10 puntos ganaras 10 € si bajase perderías 10€, de la misma forma a la inversa con un short.

Los resultados acumulados del algoritmo, como así su función de distribución y retornos diarios son los siguientes:

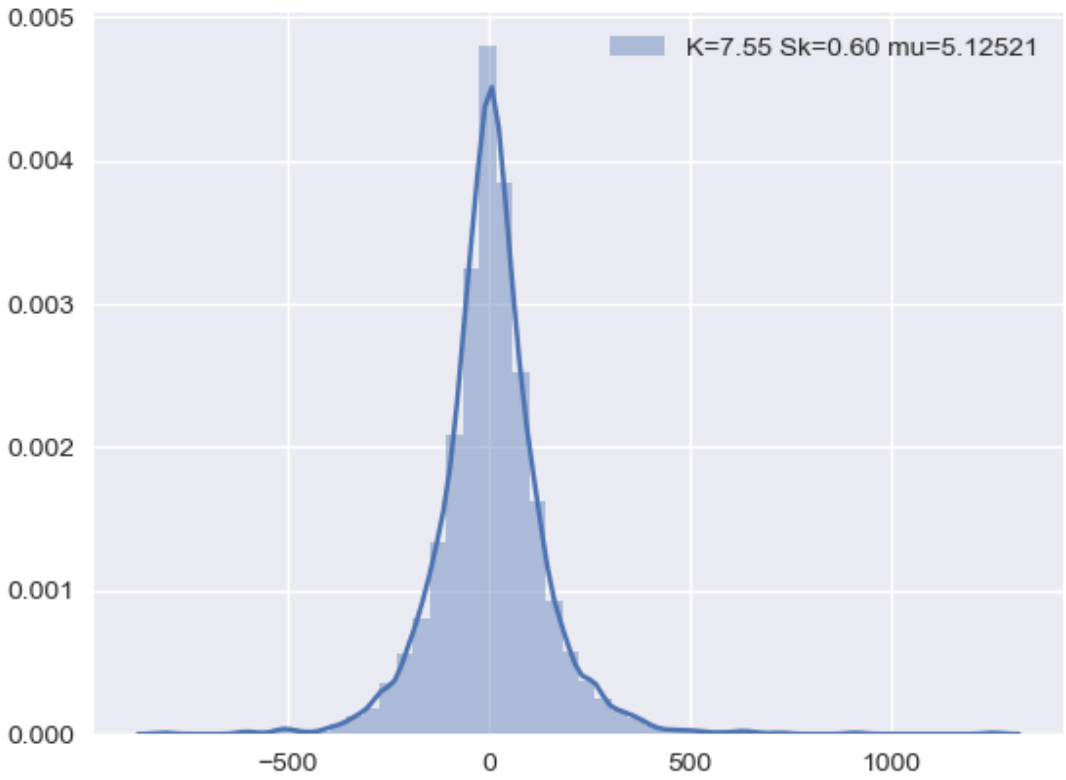

Figura 69: Función de distribución de los retornos obtenidos con el modelo IA y con IF-ELSE

Como podemos observar tiene una kurtosis de resultados alta, es positivo ya que reduce la varianza o volatilidad del retorno. Por otro lado, se puede observar una skewness positiva, como ya hemos explicado anteriormente nos indica que con una frecuencia superior obtendremos retornos negativos sobre positivos, pero también obtendremos retornos positivos atípicos con mayor frecuencia, la media de retorno diario es de 5 puntos sobre el índice.

Una observación que es posible realizar es observar que el patrón de retornos inferido por las dos AIs es inverso al patrón de retornos del propio índice que destacaba por una kurtosis más pequeña y una skewness negativa, este resultado no es ni bueno ni malo, simplemente indica que nuestro algoritmo es propenso a mayores ganancias, pero también a perdidas pequeñas en mayor frecuencia, la decisión de utilizar un sistema u otro dependerá del perfil de riesgo del inversor.

Aunque aparentemente los retornos obtenidos puedan parecer ruido, debemos recordar que el mercado se comporta como un proceso estocástico por ende existe un componente muy alto de ruido, el trabajo consistía en obtener si fuera posible un patrón que permitiera generar retornos a largo plazo y sin mucha volatilidad de resultados, como vemos en la siguiente gráfica.

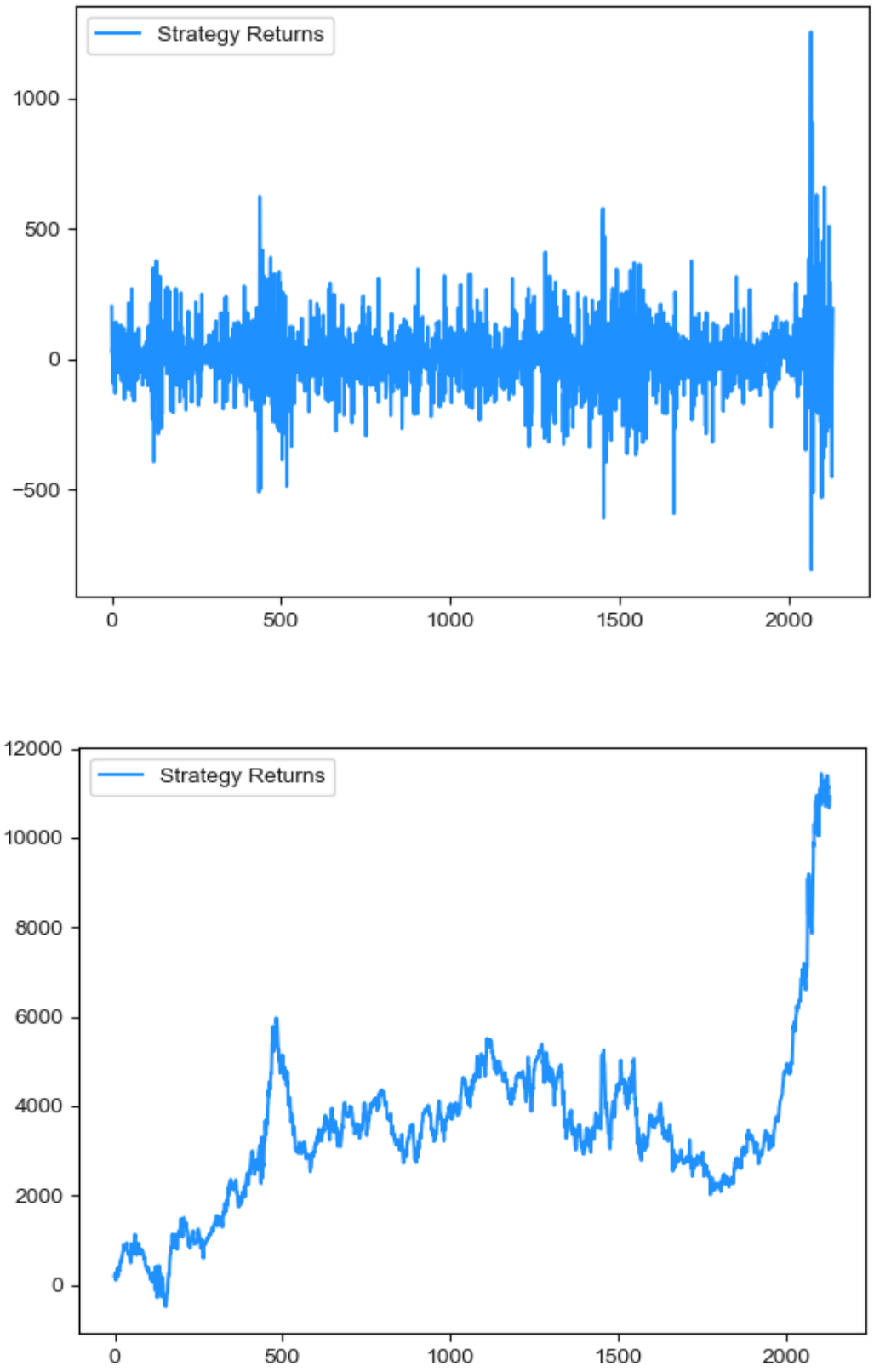

Figura 70: Retornos (imagen superior) y retonos acomulados (imagen inferior) del modelo IA

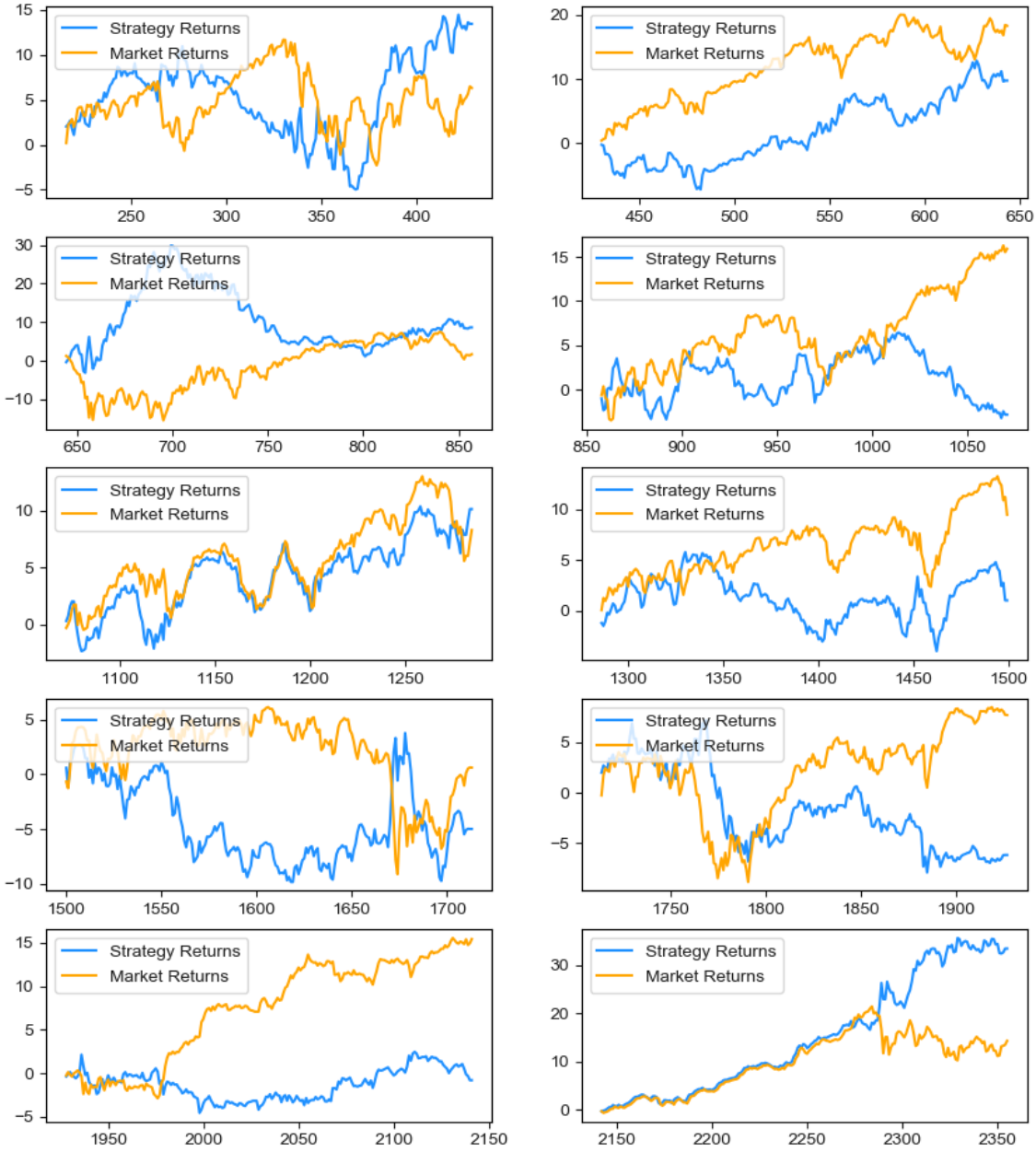

Figura 71: Resultados acumulados desagregados para los 10 test realizados.

5.8 Redes Recurrentes

Las redes recurrentes fueron introducidas por primera vez en 1980, debido a su alta demanda de memoria RAM y capacidad de computo no pasaron a realmente utilizarse hasta la década de 2010, desde entonces se han convertido en la mejor arma que se dispone actualmente para realizar análisis de series temporales, por esta causa se decidió introducirlas en el TFG y poder comparar los resultados con el de otros modelos.

Este tipo de redes son usadas tanto en la predicción de mercados financieros como en análisis de sentimiento, traducción, etc. incluso ya es posible crear música con ellas como se puede observar en el siguiente video *"Generate music with RNN" [23]*.

Este tipo de redes aprenden de una forma similar a los seres humanos, disponemos de una memoria que nos permite aprender de los errores, no partimos desde 0 cuando pretendemos adquirir un conocimiento. Siguiendo esta filosofía se crearon basándose en un sistema recurrente de datos tal y como su propio nombre indica su salida no depende solo de su actual input y peso sino de sus estados pasados.

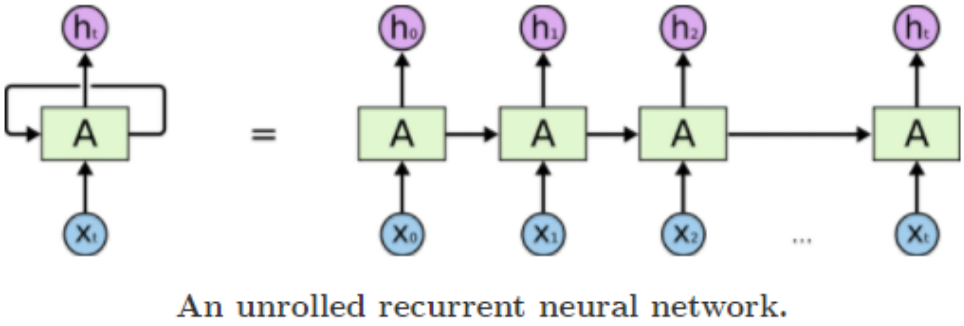

Figura 72: Diagrama de estados de una neurona en una RNN *[22]*.

Este comportamiento es descriptible en la siguiente formula donde como vemos la salida depende de la salida anterior y del peso actual más un componente de entropía o bias y su función de activación $\emptyset$.

$$y_t = \emptyset({W_x}^T \cdot W_x + Y_{(t-1)} \cdot W_y + b) \quad (53)$$

¿A pesar de que aparentemente puedan ser muy útiles tienen un problema, que pasa si necesita memorizar a largo plazo? Para resolver este problema, se desarrollaron las redes neuronales **Long Short Term Memory**.

5.8.1 Long Short Term Memory o LSTM

Este tipo de redes, son un tipo especial de redes neuronales capaces de aprender a largo plazo, fueron introducidas en 1997 y se popularizaron de forma muy rápida gracias a su gran utilidad a la hora de solventar muchos problemas sobre todo en el campo del análisis de series temporales y traducción.

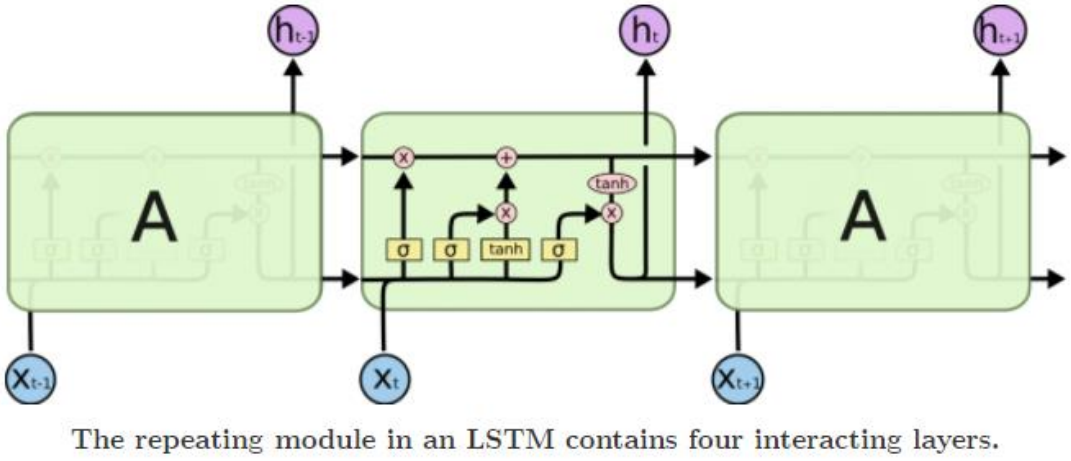

Figura 73: Inside a LSTM neuron *[22]*.

La neurona se compone de diferentes "puertas" o "Gates" donde se trata la información que pasara a la salida. En la primera puerta, llamada "forger layer", la función sigmoid decide cuanto porcentaje de la información del estado anterior y nueva sustituirá al estado "permanente" de la red.

En la segunda, *figura 74*, se decidirá cuanta información de la que no formara parte del estado de la neurona va a ser almacenada. La función sigmoid, llamada "input gate layer" decidirá que valores serán actualizados y la capa tanh decidirá un nuevo vector de candidatos que podrán ser añadidos al nuevo estado de la neurona.

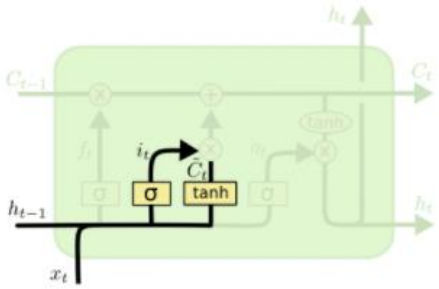

Figura 74: Second Gate *[22]*.

$$i_t = \emptyset(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (54)$$

$$\widetilde{C_t} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (55)$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t} \quad (56)$$

Finalmente actualizamos al nuevo estado de la neurona y decidimos que información formara parte de la salida. El valor final de salida, estará compuesto tanto por información nueva como por el propio estado permanente.

Primero decidimos que parte de la información pasará gracias a la función sigmoid, luego lo combinaremos con un nuevo vector de candidatos del estado de la neurona obtenido mediante la función tanh, el resultado será nuestro output.

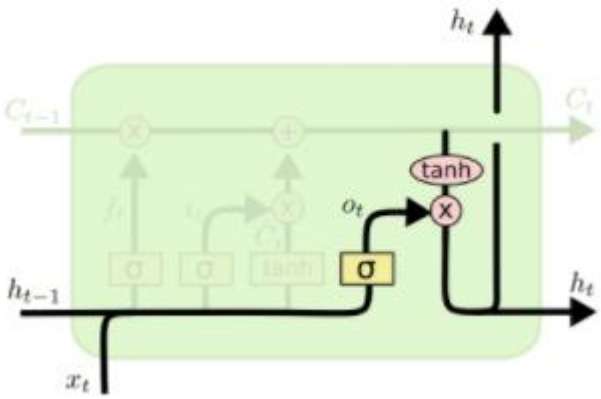

Figura 75: Third Gate *[22]*.

$$O_t = \emptyset(W_0[h_{t-1}, x_t] + b_0) \quad (57)$$

$$h_t = O_t * \tanh(C_{t)} \quad (58)$$

5.8.2 Resultados Down Jones sobre LSTM

Para este apartado se ha realizado dos modelos predictivos diferentes, en primer lugar, un clasificador binario de compra/venta y por otro lado un modelo regresivo para calcular el retorno según la ventana de días que se quiera calcular.

5.8.2.a Clasificador binario LSTM (1D)

Los datos empleados para el clasificador han sido simplemente los retornos calculados respecto a una ventana de 3 días.

La fórmula empleada para el cálculo de estos retornos es:

$$r_{i0} = \frac{price_i}{price_0} - 1 \quad (59)$$

$$r_0^i = [r_{00}, \dots, r_{i0}] \quad (100)$$

Los parámetros de entrenamiento que se pasaron al modelo fueron los retornos salvo el ultimo, donde según la diferencia con el retorno anterior se interpretaba la tendencia resultante de ese día up (compra) 1, down (venta) 0, es importante señalar que todos los parámetros anteriores salvo el dato de tendencia resultante que se emplearía como solución, fueron preprocesados mediante StandardScaler que permite remover la media de los datos y estandarizarla a una varianza única. El modelo planteado se trata de una doble capa de red LSTM con 50 y 100

neuronas respectivamente con funciones de activación tanh que permite computar inputs entre el rango -1,1 junto a una última capa con una única neurona con función de activación sigmoid. Por otro lado, se empleó como función de perdidas "binary cross entropy" un subtipo de la función "cross entropy" mucho más eficiente para clasificaciones binarias y por último como optimizador se empleó "Adam".

Los resultados fueron un 58% de aciertos o posiciones ganadoras (retornos positivos) frente al 53% de retornos positivos del mercado, esto nos indica que la red neuronal LSTM es capaz de clasificar ligeramente mejor que el azar.

Realizando un análisis más detallado podemos observar en la matriz de confusión (eje y "true label", eje x "predicted label") como de las 295 operaciones de tendencia bajista "down" cuantificadas como 0, 160 han sido correctamente predichas suponiendo un 54.9% de precisión, sin embargo, la precisión es ligeramente mayor en las posiciones alcistas, donde de 176 tendencias alcistas diarias clasificadas, 110 han sido correctamente clasificadas y 66 erróneas, suponiendo una precisión del 62.5%.

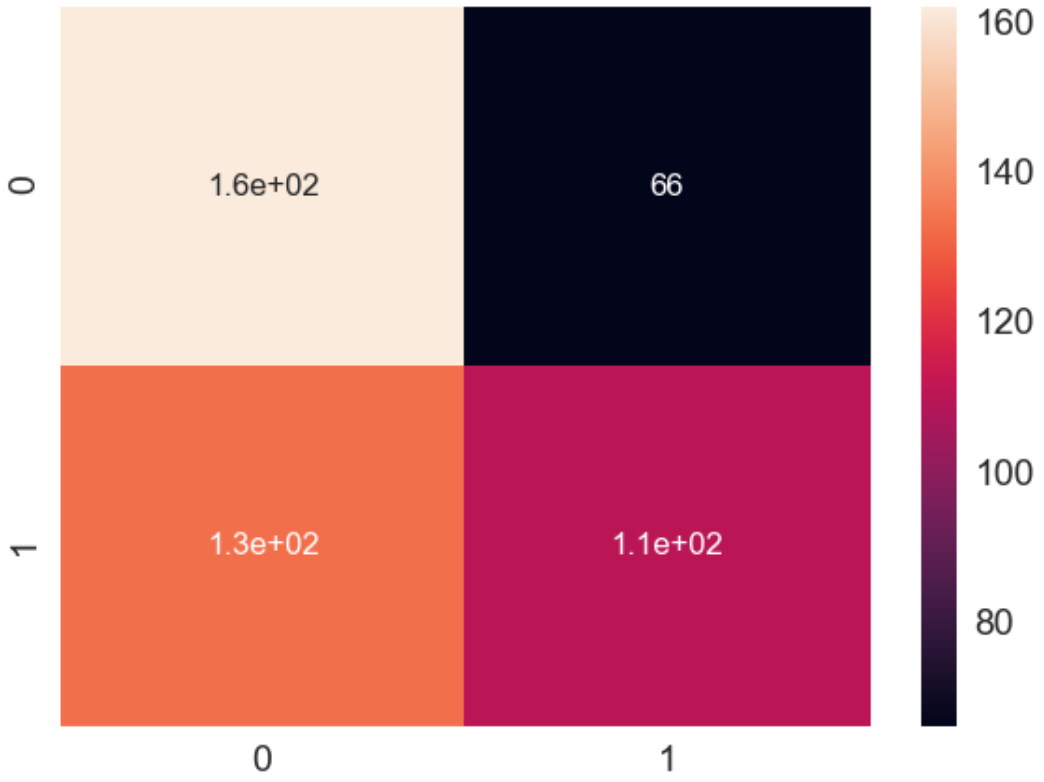

Figura 76: Matriz de confusión del clasificador 1D.

En general se puede observar como la precisión global del modelo es alta, pero ahora se realizara un mayor ajuste los umbrales de decisión con el fin de analizar sus límites en la precisión, por lo tanto estableciendo un nuevo umbral de $p>0.65$ para "up" y $p<0.35$ para "down", la precisión mejora hasta un 64% para tendencias bajistas y un 84% para tendencias alcistas , detectando 21 posiciones alcistas correctas de 25,para un test de 471 días, suponiendo un magnífico resultado, ya que a mayor precisión mayor posibilidad de apalancamiento con derivados.

5.8.2.b Clasificador binario LSTM (2D)

Utilizaremos los retornos calculados con los precios de apertura, pero también esta vez calcularemos el retorno con el precio de cierre, de esta forma obtenemos dos dimensiones a la hora de ver como varían, para una mejor comprensión añado la siguiente imagen donde la escala y la forma el precio en valores absolutos y el eje x lo forma los dias con t0 como dia actual. La arquitectura propuesta es una red LSTM de 4 capas en serie con un numero de neuronas de 70,150,70 y 30 por este orden.

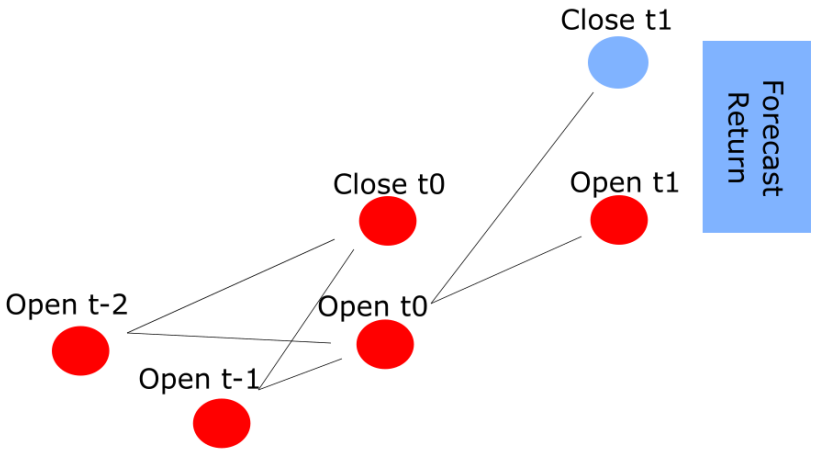

Figura 77: Grafo de retornos.

Utilizando umbrales de decisión sobre la función de activación sigmoid de >0.5 para posiciones alcistas y <0.5 para bajistas, obtenemos una precisión global del 62.6%. Analizando un poco más en profundidad 154 posiciones bajistas acertadas de 250 totales suponiendo así un 61.6% de precisión, por otro lado en las posiciones alcistas observamos como el algoritmo ha acertado 141 de 221 de suponiendo un 63.8% de precisión.

Realizando ahora un mejor ajuste similar al caso anterior con umbrales de decisión del 0.65 y 0.35 respectivamente para posiciones alcistas y bajistas obtenemos: un 80% de precisión para posiciones bajistas y un 74% para posiciones alcistas, lo que supone una precisión global del algoritmo del 78%. Realizando un total de 69 operaciones 54 de ellas correctas sobre 471 días.

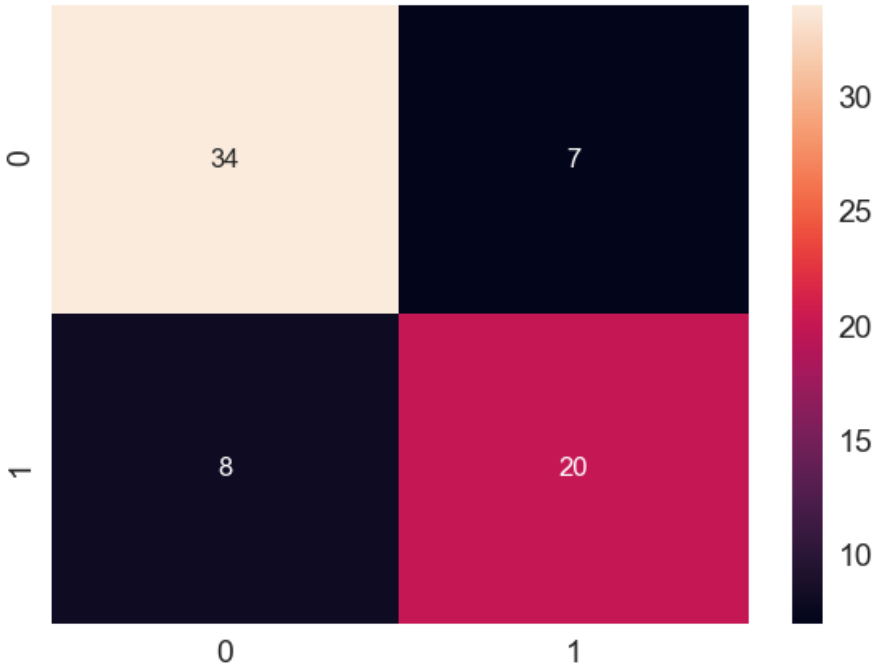

Figura 78: Matriz de confusión del clasificador 2D.

5.8.3 Modelo Regresivo LSTM

Tratando los datos de forma similar al modelo anterior vamos a realizar un modelo regresivo, utilizando esta vez una función de activación lineal junto con una función de pérdidas llamada error cuadrático medio y el optimizador usado de costumbre “Adam”. El objetivo es tratar de obtener el retorno del día siguiente.

Una vez preparados los datos, creamos nuestro modelo de red LSTM, con 4 capas con funciones de activación tangente hiperbólica y una última capa ForwardPass de 1 neurona con función de activación lineal que permitirá obtener el valor final.

```
def create_model():
    model = Sequential()
    model.add(LSTM(150, input_dim=1, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(LSTM(300, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(LSTM(100, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(LSTM(40, return_sequences=False))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='linear'))

    model.compile(loss='mean_squared_error',
                  optimizer="adam")
    return model
```

Figura 79: Código empleado para la construcción de la red.

Diseñamos este modelo para que sea entrenado en 50 epoch con el 70% de dataset, obteniendo en su respectivo 30% de test una precisión del 62% en la tendencia del retorno del día siguiente, frente a un 52% de tendencias alcistas en el propio test.

```
Epoch 50/50
1270/1270 [==============================] - 1s 694us/step - loss: 0.6883 - val_loss: 0.2459
942/942 [==============================] - 1s 571us/step
0.6178343949044586
0.5159235668789809
```

Figura 80: Resultados global del modelo.

El resultado vuelve a ser muy positivo ya que permite obtener un numero de aciertos en la tendencia superior al azar, siendo a pesar de todo un modelo regresivo, añado además un fragmento del test donde se puede ver los valores predichos vs el valor real.

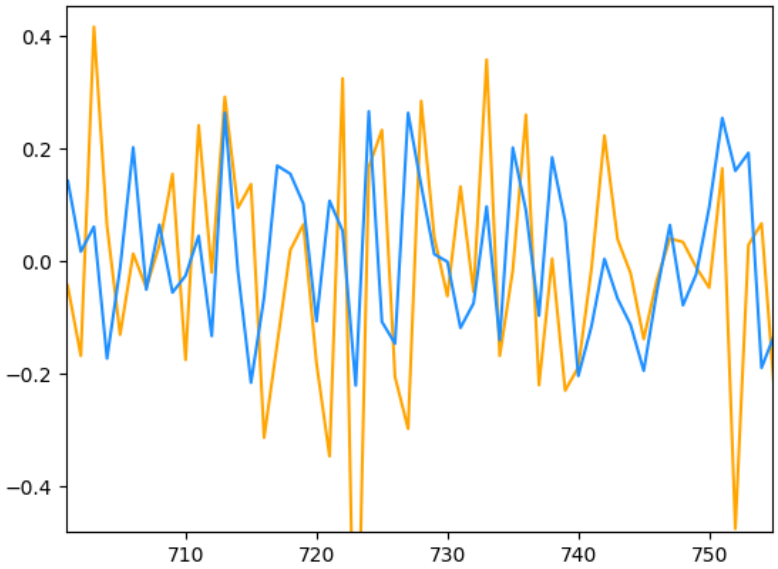

Figura 81: Resultados del test con el modelo regresivo.

5.9 Sentiment analysis NLP

El análisis de sentimiento se basa en computar si el texto analizado tiene un sesgo positivo, neutro o negativo. Conocer de una forma aproximada el estado de ánimo del usuario es vital para las estrategias en diversos sectores que tienen a este como activo, su uso principal en estrategias de marketing se basa en el análisis del potencial nicho de negocio, análisis de campañas y resultados, posicionamiento etc, todas aplicables tanto a actividades empresariales como políticas.

En el trabajo se va a tratar de analizar si de alguna forma ayuda en las predicciones computar el estado de ánimo del usuario de twitter sobre categorías relacionadas con los mercados financieros, dotandonos de herramientas predictivas que mejoren los resultados anteriores.

Los sistemas NLP o de procesado natural del lenguaje se basan en el desarrollo de modelos que permitan interpretar el discurso dado en formato audio o texto de un usuario y extraer de este su estado de ánimo.
Los sistemas se basan en analizar los siguientes aspectos:

Información Semántica: Se trata de comprender el significado de cada palabra utilizada, sin embargo a veces el significado depende del contexto.

Sintaxis: Entender la estructura de la frase u oración para comprender la información aportada.

Contexto: Entender el contexto en el que se desarrolla la frase o las palabras empleadas es vital para entender con totalidad el significado global.

Este análisis puede ser desarrollado mediante reglas que permitan al software analizar la cadena de texto y realizar una puntuación, sin embargo, no suelen ser muy precisos y por ello se decidió desarrollar modelos mediante inteligencia artificial. Estos modelos están desarrollados mediante diferentes técnicas basas en aprendizaje supervisado:

Tokenization: se basa en diferenciar las diferentes palabras empleadas en el texto. En lenguajes derivados del latín, por ejemplo es increíblemente sencillo ya que las palabras van separadas por espacios en blanco, pero en otros idiomas como el chino es bastante más complicado y por ende se necesita de un sistema más complejo.

Speech Tagging (POS): empleado para etiquetar gramaticalmente el texto dentro de las diferentes categorías empleadas en el lenguaje como adjetivos, verbos, adverbios, etc. con el fin de saber los temas desarrollados en el escrito.
Aunque a priori pueda parecer sencillo una misma palabra puede tener diferentes significados dependiendo del contexto y por ende emplear simplemente un diccionario podría generar resultados muy ambiguos. Para evitar esto el algoritmo calcula a partir del segmento seleccionado que secuencia de etiquetados que puede tener asociada posteriormente.

Named Entity Recognition: Se desarrolla posteriormente al speech tagging y es empleado para reconocer etiquetas importantes como nombres de compañías, calles, plazas etc.

Classification Sentiment: Al final de todo el proceso un algoritmo utilizara todos los inputs anteriores para evaluar y medir el grado de emoción positiva, negativa o ausencia (neutro) del texto introducido al principio.

A la hora de la verdad se emplean sistemas híbridos para maximizar la precisión del modelo, tal y como explica LEXALYTICS en su web, una empresa especializada en sentiment analysis.

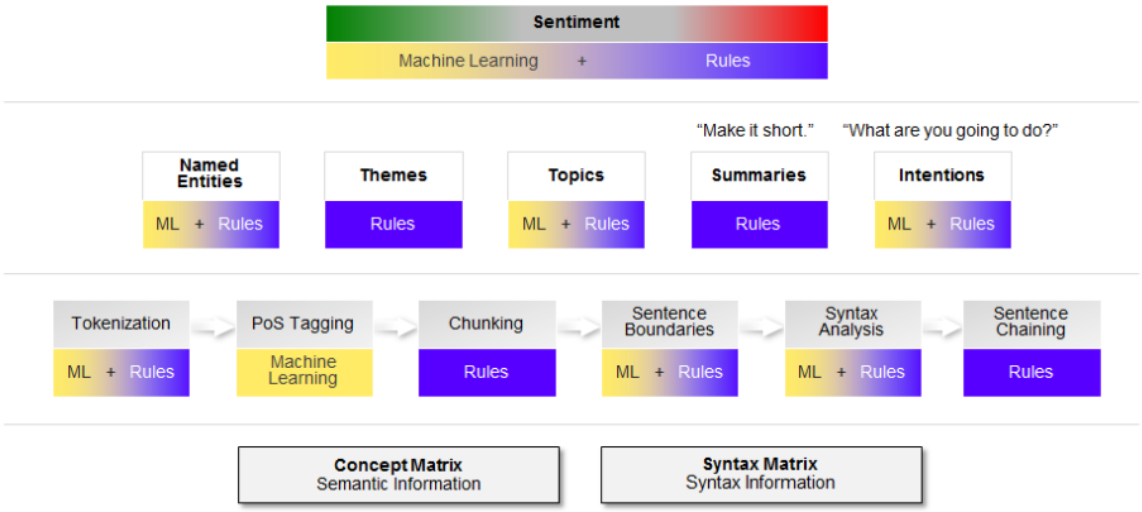

Figura 82: Core Network *[24]*

Vamos a plantear para este apartado un pequeño caso de estudio donde descargaremos tweets que contengan la palabra "dow jones" desde la fecha 01/01/2017 hasta 01/05/2018, posteriormente mediante la librería de Python TextBlob se analizará cada uno de los tweets para clasificar si es positivo(1), negativo(-1) o neutro(0), después se agruparan por mes y se analizara mediante la puntuación total y la media resultante por mes, para observar si existe correlación o no con los retornos de ese mismo mes y del mes siguiente.

En primer lugar, se puede como de los 21874 tweets obtenidos en esa franja temporal, hay un total de 19.2% de tweets positivos, un 12.2% de negativos y un 68.4% de neutros. Por otro lado, los resultados agrupados por mes ordenados cronológicamente son los siguientes, observando como la opinión media suele ser positiva salvo en dos casos concretos en el mes de marzo de 2017 y en el mes de febrero de 2018.

Date	Sum	Mean	Return	Next Month Return
17-01	453	0.1794	0.0025	0.0518
17-02	201	0.1093	0.0518	-0.0139
17-03	-466	-0.1334	-0.0139	0.0143
17-04	118	0.0443	0.0143	0.0032
17-05	103	0.0731	0.0032	0.0172
17-06	38	0.0361	0.0172	0.0266
17-07	26	0.0388	0.0266	0.0009
17-08	64	0.1174	0.0009	0.0200
17-09	361	0.1403	0.0200	0.0454

17-10	46	0.1138	0.0454	0.0367
17-11	141	0.1327	0.0367	0.0207
17-12	102	0.1578	0.0207	0.0513
18-01	51	0.1924	0.0513	-0.0406
18-02	-49	-0.2722	-0.0406	-0.0378
18-03	13	0.1313	-0.0378	0.0016

Tabla 11: Resultados agregados por mes.

Realizando un análisis de correlaciones entre las variables Sum, Mean, Return y Next Month Return, observamos como existe una correlación de 0.578 entre el retorno y la opinión media del mismo mes y un 0.418 con el retorno del siguiente mes, demostrando que el análisis NLP sobre las opiniones de los inversores puede ser efectivo para realizar predicciones.

	Sum	Mean	Return	Next Month Return
Sum	1	0.628	0.327	0.356
Mean	0.628	1	0.57	0.41

Tabla 23: Correlacione lineal entre las diferentes variables.

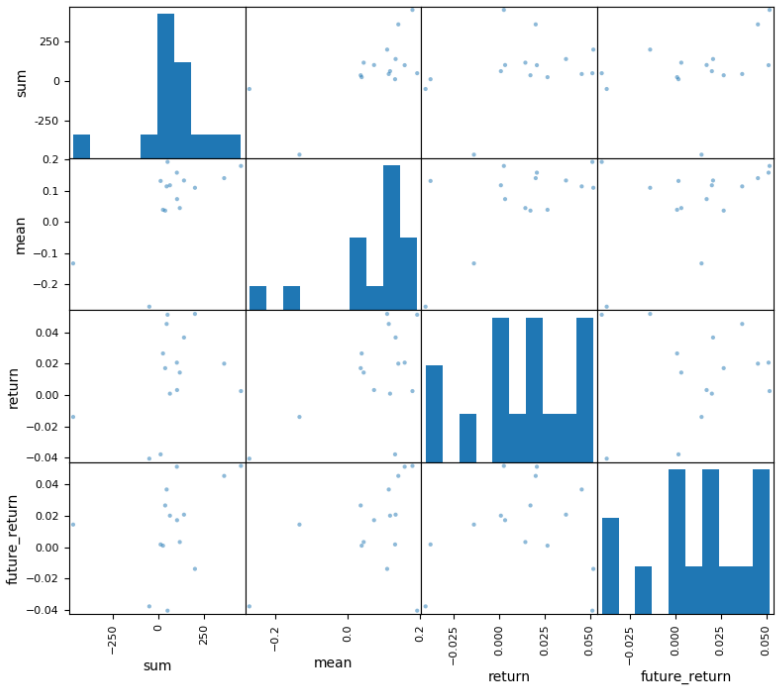

Figura 83: Matriz de correlaciones.

En la *figura 84*, se puede observar esta relación entre la opinión media y el retorno desarrollado a lo largo del mes, observando de forma clara como en los meses donde las opiniones han sido

claramente negativas han terminado con retornos negativos, pudiendo utilizarse así, como un detector de caídas al rebasar un umbral x a estimar de sesgo negativo.

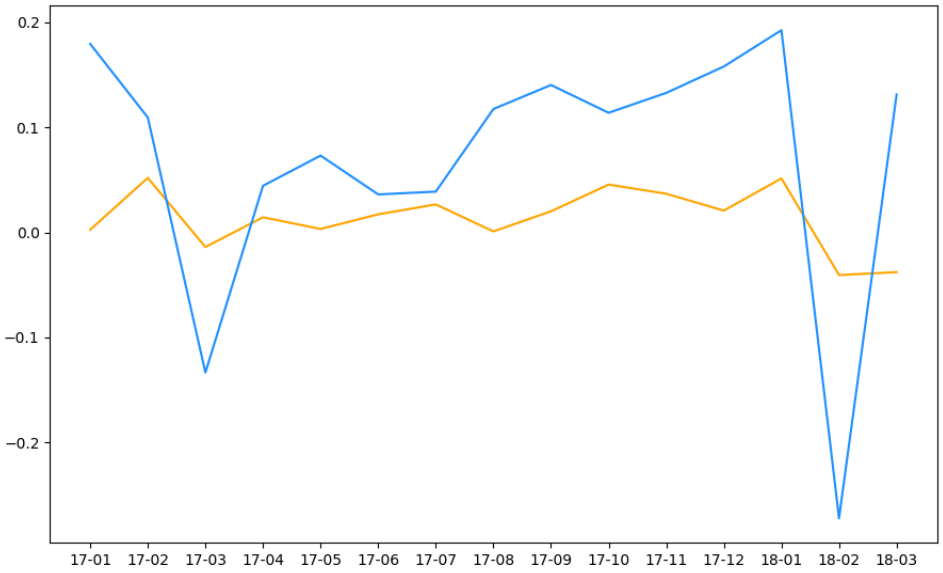

Figura 84: Opinión media SA vs retorno del índice DJIA.

5.10 Algoritmos Genéticos.

Los procesos de optimización nos otorgan múltiples herramientas para modelar y reformular los datasets, en este apartado en concreto vamos a hablar de algoritmos genéticos como método para obtener la mejor combinatoria de variables para nuestro modelo o la optimización de la arquitectura propuesta.

Los algoritmos genéticos o GA surgieron entorno a la década de los 70, fueron bautizados de la siguiente forma debido a su inspiración en la evolución biológica y en su base genético-molecular. El algoritmo busca someter a una población de individuos con unas determinadas características a un contexto especifico en el cual los individuos más aptos podrán sobrevivir y reproducirse, mientras que los menos aptos serán descartados.

Son empleados principalmente en funciones no derivables o de derivación compleja para obtener mínimos, siendo su funcionamiento es el siguiente:

Inicialización: se inicializa la población con unas características aleatorias donde sus diferentes cromosomas o individuos forman una combinación de variables donde puede existir una posible solución óptima.
El carácter aleatorio de la población permite la diversidad suficiente para evitar la convergencia en mínimos locales.

Evaluación: A cada uno de estos individuos se le aplicara una función de “aptitud” o “fitness function” para evaluar su solución.

Condición de finalización: El AG deberá ser detenido cuando se alcance una posible solución óptima si satisface los criterios de detección ya sea por número máximo de iteraciones o cuando la población global ya haya convergido y no cambie, mientras que los casos anteriores no se cumplan el algoritmo sigue ejecutándose:

Selección: Después de evaluar a cada individuo o cromosoma se procede a elegir los mejores para que sean cruzados en la siguiente iteración, a mayor desempeño mayor probabilidad.

Recombinación o cruzamiento: La recombinación es el principal operador genético y representa la reproducción sexual, opera sobre dos individuos a la vez para generar dos descendientes.

Mutación: modifica al azar las variables del individuo y permite alcanzar nuevas combinatorias o posibles soluciones

Remplazo: una vez aplicados los operadores genéticos anteriores, se seleccionan los mejores individuos para la población siguiente.

Gen: Un gen es la posición en el vector de variables del cromosoma o individuo

Allele: Es el valor contenido dentro del gen

Un pseudo código posible para desarrollar el algoritmo podría ser el siguiente:

```
GA()
   initialize population
   find fitness of population

   while (termination criteria is reached) do
      parent selection
      crossover with probability pc
      mutation with probability pm
      decode and fitness calculation
      survivor selection
      find best
   return best
```

Figura 85: Pseudo código para GA

Tal y como ya se ha explicado, los algoritmos genéticos permitirán optimizar la selección de variables o selección de parámetros del modelo y mejorar así los resultados.
En este punto, se va a utilizar un AG para mejorar una arquitectura de red ya propuesta para el clasificador binario realizado en el punto 5.8. Cabe recordar que se consiguió en ese modelo un 62.8% de precisión, mediante una arquitectura con cuatro capas LSTM colocadas en serie con un numero de neuronas por orden de 70,150,70,30.

Para realizar nuestro AG, será necesario en primer lugar, desarrollar las diferentes funciones que permitirían llevar a cabo el proceso evolutivo de la población, considerando que cada modelo predictivo es un individuo de la población que será juzgado por su precisión en el test y por ende aceptado para reproducirse o descartado si su desempeño no es lo suficientemente bueno. Por otro lado, se incluye una función de mutación donde según una determina probabilidad, este podrá sufrir un cambio en el valor de uno de sus parámetros. Estos parámetros corresponderían con un vector de un determinado número de capas, es decir si existen 4 capas, el vector tendrá

una longitud de 4 posiciones y un valor que corresponderá al número de neuronas, los pesos serán inicializados al azar.

En futuros proyectos se podrá considerar el desarrollo de funciones fitness más complejas para optimizar arquitecturas variantes de forma automática mediante AG.

Los parámetros empleados para configurar el algoritmo genético son:

Población	8 individuos o modelos
Mutación	50% de probabilidad de mutar algún parámetro del modelo
Nº epochs	8 (los mismos que en el modelo original)
Selección de más aptos	40%

Tabla 13: Parámetros de configuración del GA.

Al principio, el test fue realizado en una GPU de modelo NVIDIA 745, siendo esta insuficiente debido a la necesidad de realizar sistemáticamente múltiples entrenamientos y testeos con un numero de neuronas por capa de 1 a 128, el algoritmo se volvía demasiado pesado e inviable, por ello se planteó y decidió buscar una forma de testearlo mediante computación en la nube. Tras barajar múltiples opciones como Amazon AWS o Google CLOUD se llegó a la conclusión de utilizar una nueva empresa, llamada FLOYD (floydhub.com) ,dedicada a ofrecer servicios de computación en la nube expresamente para inteligencia artificial mediante Python.
Disponen de un sistema donde subiendo tu código y el dataset te permiten ejecutarlo en una GPU alquilada expresamente para ti, disponen de diferentes precios y planes.

De esta forma, tras varios ciclos de vida , el algoritmo llego a alcanzar una precisión global del 64.11%, un 1.31% superior, como podemos observar el modelo aún es muy mejorable sobre todo en la selección de parámetros de su arquitectura, sin embargo esta forma es altamente ineficiente con los recursos que se disponen actualmente pero con una GPU mucho más potente y otorgando más grados de libertad al AG para actuar, es muy probable que se puedan encontrar arquitecturas mucho más óptimas.

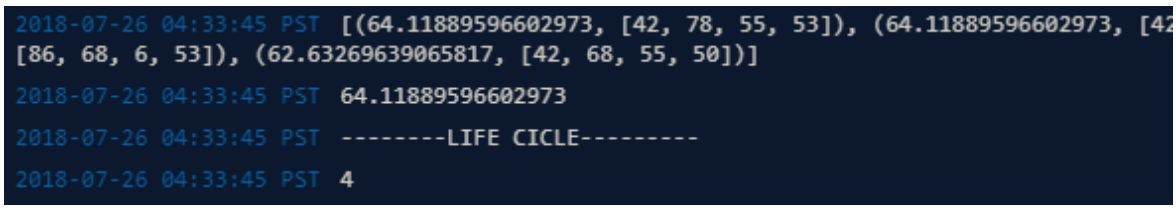

Figura 86: Resultados 4 ciclo de vida del GA.

Capítulo 6. Conclusiones

Al finalizar este proyecto se ha podido cumplir con exito muchos de los objetivos planteados al principio de este TFG.
Se planteó la necesidad de crear clasificadores de imágenes que permitieran en un futuro cuando se disponga del dataset necesario clasificar estas imágenes y los objetos que pudieran aparecer en ellos, con el fin de obtener información de forma automática más allá del contenido en formato texto que fuera introducido por los clientes o pubs. En este sentido,se pudo realizar con éxito clasificadores sobre dataset tan simples como "perros y gatos" o más complejos como Cifar10, abriendo la puerta al desarrollo de futuros clasificadores sobre tipos de bebidas subidas, marcas, etc e incluso deteccion de objetos para poder extraer diferentes métricas de una misma imagen, como por ejemplo si aparecen personas o personajes ficticios anunciando el producto, lugar empleado, etc.

Por otro lado, se planteó la necesidad de desarrollar algoritmos de predicción que actuasen sobre series temporales. Debido al mismo problema anterior, que era la falta del dataset necesario para entrenar y testear los modelos. Se decidió emplear series temporales financieras, siendo estas en un principio de mayor dificultad a la hora de poder realizar predicciones debido a su alto componente aleatorio.
Los resultados fueron magníficos y demostraron el enorme potencial de la inteligencia artificial para inferir patrones no lineales. En un futuro, cuando se dispongan de los datos necesarios, se podrá realizar modelos de predicción sobre el éxito un evento programado, ventas de bebidas o otros productos, etc.

A continuación, se añade una tabla donde se recogen los mejores modelos desarrollados para la predicción de series temporales financieras y sus resultados finales.

Algoritmo	Total Return	Precisión	Sharpe	IA
CR(10,EMA 30)	10530€	54.77%	-	No
Hist UP	8958€	55.72%	-	No
Hist UP	15676€	55.73%	1.035	Si
CR(10,EMA 30)	14691€	54.86%	0.874	Si
Clasificador long/short Gaussian Mixture + SVM-RBF	11946€	51.35%	0.42	Si
LSTM Binary Clasificator	-	63.6%	-	Si
LSTM Binary Clasificator Adjusted	-	80% short / 74% long	-	Si
GA over LSTM Binary Classificator Adjusted	-	64.11%	-	Si

Tabla 14: Comparativa de resultados finales

En adición, también se abre la posibilidad de desarrollar en un futuro trabajo una arquitectura de red compleja basada en la integración de diferentes modelos predictivos, con el fin de crear un Comunity Manager Online basado en inteligencia artificial, que analice en tiempo real todos los

datos disponibles y sea capaz de crear anuncios para aprovechar una oportunidad detectada en el mercado.

En este sentido, Nextnight pretende ofrecer a medio plazo no solo una red de comunicación entre clientes y usuarios, sino presentarse como una start-up que apuesta por integrar la última tecnología en inteligencia artificial en todos sus procesos de control, ya sea para clientes o incluso para el manejo de los propios recursos de la empresa.
Sin embargo, la velocidad de adopción de estas nuevas tecnologías y el desarrollo de plataformas más complejas a nivel técnico vendrá definido por la tasa de crecimiento de la start-up, siendo prioritario en primer lugar, el éxito de la propia red social como vía de comunicación.
La fase en la que esta situada ahora mismo, es en el desarrollo de la infraestructura base a nivel de arquitectura para dar un servicio básico pero fiable a todos los usuarios y clientes que se vayan incorporando. NextNight actualmente incorpora tecnología basada en bases de datos relacionales como MySQL y no-relacionales como Apache Cassandra, utilizando PHP Laravel para el servicio REST que utiliza MYSQL y Java Spring + Spark para el servicio REST que da soporte a la arquitectura BIG-DATA de Cassandra.
Ambas aplicaciones tanto NextNigh BC como NextNight SN utilizaran ambos servicios para ofrecer la mayor calidad de uso a sus usuarios.

Por último, se ha comprobado el enorme potencial de lenguajes como Python para el procesamiento de datos y construcción de modelos mediante librerías de software como Keras, tomándose ésta, en clara consideración para la construcción de los futuros modelos predictivos de NextNight, como así mismo, la utilización de empresas de computación en la nube especializadas en inteligencia artificial como Floyd para el entrenamiento de los modelos.

Capítulo 7. Anexos

7.1 Nomenclaturas.

IA: Inteligencia Artificial.

ML Machine Learning.

NN Neural Network.

FNN ForwardPass Neural Network.

ARIMA Autoregressive Integrated Moving Average .

DJIA Dow Jones Industrial Average.

RNN Recurrent Neural Network.

LSTM Long Short Term Memory.

AG Algoritmos Genéticos.

Capitula 8. Bibliografía

[1] Ian ,G.; Yoshua, B.; Aaron ,C.; "Increasing Accuracy, Complexity and Real-World Impact," *DEEP LEARNING*, pp. 23, September 2016.

[2] Accenture, "Accenture and Frontier Economics," https://www.accenture.com/us-en/insight-artificial-intelligence-future-growth. [Online].

[3] tekcrispy.com, "IBM utiliza la Inteligencia Artificial para identificar el riesgo de psicosis en pacientes" , https://www.tekcrispy.com/2018/02/06/ibm-inteligencia-artificial-riesgo-psicosis/

[4] marutitech.com, "How business can leverage reinforcement learning?", https://www.marutitech.com/businesses-reinforcement-learning/

[5]medium.com, "what is underfitting and overfitting in machine learning and how to deal with it", https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

[6] Diederik P. "A Method for Stochastic Optimization", Cornell university Library

[7] stats.stackexchange.com, "What is the batch size in neural network?", https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network, answer 2.

[8] analyticsvidhya.com, "Understanding Support Vector Machine Learning". https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[9] people.eecs.berkeley.edu, "The Kernel Trick", figure 1, https://people.eecs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf

[10] robjhyndman.com, "Cross validation for time series", https://robjhyndman.com/hyndsight/tscv/

[11] es.wikipedia.org, "Curva Roc", https://es.wikipedia.org/wiki/Curva_ROC

[12] mattmazur.com, "**A Step by Step Backpropagation Example**," https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example. [Online].

[13] Accenture,"Introducign convolucional neural networks", http://neuralnetworksanddeeplearning.com/chap6.html. [Online].

[14] adeshpande3.github.io," A beginners guide to understanding convolutional neural networks", https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

[15] http://cs231n.github.io ,"Convolutional Neural Networks", general pooling,http://cs231n.github.io/convolutional-networks/

[16] Youtube, "**Visual Cortex Cell Recording**" , https://www.youtube.com/watch?v=Cw5PKV9Rj3o. [Online].

[17] toronto.edu," Cifar10 dataset" , https://www.cs.toronto.edu/~kriz/cifar.html.

[18] Tamraparni, Dasu.; Theodore Johnson. Exploratory Data Mining and Data Cleaning, 2003

[19] Ray Dalio; "How economic machine works", 2008.

[20] An Introduction to Omega, Con Keating and William Shadwick, The Finance Development Center, 2002

[21] Maria Pilar G; "Analisis de series temporales ARIMA" , Universidad del Pais Vasco, 2009

[22] colah.github.io, “Understanding LSTM Networks”, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[23] youtube.com, “**Generate music with RNN**,”https://www.youtube.com/watch?v=A2gyidoFsoI. [Online].

[24] lexalytics.com, “ Machine Learning vs Natural Language Processing”, https://www.lexalytics.com/lexablog/machine-learning-vs-natural-language-processing-part-1.

[25] appliedmachinelearning.wordpress.com, “Achieving 90% accuracy in object recognition on Cifar10”, https://appliedmachinelearning.wordpress.com/2018/03/24/achieving-90-accuracy-in-object-recognition-task-on-cifar-10-dataset-with-keras-convolutional-neural-networks/

[26]cs231n.github.io, “Convolution operation animation with shared weights”, http://cs231n.github.io/assets/conv-demo/index.html

[27] medium.com, “explanation of convolutional neural networks”, https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8

[28] blackarbs.com “ Can we use mixture models to predict makets”, http://www.blackarbs.com/blog/can-we-use-mixture-models-to-predict-market-bottoms/4/1/2017