Facultad de Empresariales

Impacto de la volatilidad en un robo-trader de Deep Reinforcement Learning

Autor: Luis García Castro
Director: Eduardo César Garrido Merchán

MADRID | Abril 2024

Índice de contenidos

Índice de figuras

Índice de tablas

Índice de ecuaciones

Resumen

Este estudio investiga el impacto de la volatilidad en el rendimiento de un robo-trader basado en un modelo de Deep Reinforcement Learning (DRL). Tradicionalmente, los modelos de selección y asignación de activos se han basado en supuestos que pueden no ser aplicables en el mundo real y la introducción de la Inteligencia Artificial (IA) y el Machine Learning (ML) ha abierto nuevas perspectivas para los inversores. Dado el escaso enfoque en el estudio de la volatilidad en años recientes, se propuso investigar cómo esta variable afecta a un robo-trader basado en DRL. Para ello, se crearon tres carteras diferentes según el nivel de volatilidad de los activos. Posteriormente, se desarrolló un código en Python (enlace) para automatizar el proceso, desde la descarga de datos hasta la implementación del algoritmo de DRL.

Los resultados revelaron que la cartera de alta volatilidad demostró el mejor rendimiento en términos del ratio de Sharpe, mientras que las carteras de volatilidad media y baja mostraron rendimientos positivos pero insuficientes para compensar el riesgo asumido. Bien es cierto que la hipótesis alternativa (a mayor volatilidad mejor ratio de Sharpe) fue confirmada. Sin embargo, en comparación con la estrategia de mercado, el modelo de DRL superó en rendimiento solo en la cartera de alta volatilidad, mientras que para las carteras de media y baja volatilidad los rendimientos obtenidos en relación con el riesgo asumido fueron peores cuando se utilizó el modelo de DRL. Es crucial destacar que estos resultados representan una prueba de concepto y no reflejan completamente el potencial del DRL debido a limitaciones computacionales.

Palabras clave: Volatilidad, Machine Learning (ML), Deep Reinforcement Learning (DRL), Proximal Policy Optimization (PPO), Ratio de Sharpe, Python

Abstract

This study investigates the impact of volatility on the performance of a robo-trader based on a Deep Reinforcement Learning (DRL) model. Traditionally, asset selection and allocation models have relied on assumptions that may not be applicable in the real world, and the advent of Artificial Intelligence (AI) and Machine Learning (ML) has opened new perspectives for investors. Given the limited focus on volatility studies in recent years, the aim was to examine how this variable affects a DRL-based robo-trader. To achieve this, three different portfolios were created based on the level of asset volatility. Subsequently, a Python code (link) was developed to automate the process, from data downloading to DRL algorithm implementation.

The results revealed that the high-volatility portfolio demonstrated the best performance in terms of the Sharpe ratio, while the medium and low volatility portfolios showed positive but insufficient returns to offset the assumed risk. It is worth noting that the alternative hypothesis (higher volatility leads to a better Sharpe ratio) was confirmed. However, compared to the market strategy, the DRL model outperformed only in the high-volatility portfolio, whereas for the medium and low volatility portfolios, the returns obtained in relation to the assumed risk were worse when using the DRL model. It is crucial to emphasize that these results represent a proof of concept and do not fully reflect the potential of DRL due to computational limitations.

Keywords: Volatility, Machine Learning (ML), Deep Reinforcement Learning (DRL), Proximal Policy Optimization (PPO), Sharpe Ratio, Python

1. Introducción

La gestión de carteras, una práctica vital en el mundo de las finanzas despierta un interés genuino tanto entre inversores como académicos. Construir una cartera óptima que maximice los rendimientos mientras se controlan los riesgos es todo un reto. Uno de los mayores dilemas con los que nos encontramos a la hora de invertir reside en la selección y asignación de activos (Pagdin & Hardy, 2017). Qué activos hay que seleccionar y cuanto se debe invertir en cada uno es la pregunta clave que se hace cada inversor (Kevin, 2022). Desde Markowitz, Sharpe o Merton, muchos son los economistas que han dedicado gran parte de su vida a estudiar esta pregunta y han ido de esa manera construyendo poco a poco lo que se conoce como Teoría de Carteras o Portfolio Management Theory (Baker & Filbeck, 2013).

Harry Markowitz es considerado el padre de la Teoría Moderna de Carteras (Elton & Grubre, 1997). El modelo de Markowitz, también conocido como modelo Media-Varianza, utiliza el retorno esperado (media) y la volatilidad o riesgo (varianza) como ejes principales para la selección de porfolios (Markowitz, 1999). El objetivo final del modelo de Markowitz consiste en encontrar la combinación óptima de los activos que maximicen el retorno esperado para un nivel de riesgo determinado (Ozbayoglu et al., 2020). Todas estas carteras óptimas se reflejan en lo que se denomina frontera eficiente. Mirando la frontera eficiente, el inversor podrá elegir cual es la que más le conviene en función de sus preferencias.

El modelo de Markowitz, que introdujo el concepto de “Modern Portfolio Theory (MPT)”, supuso un antes y un después en la gestión de carteras y le hizo merecedor del premio Nobel. Muchos otros economistas, como Sharpe, Linter o Tobin, han utilizado el trabajo de Markowitz para mejorar y expandir la Teoría Moderna de Carteras (Chen & Chen, 2016).

El modelo de Markowitz y sus derivados se pueden aplicar a ciertas situaciones, pero aun así sus hipótesis hacen que su uso en la realidad se vea muy limitado. Por ejemplo, Markowitz asume que todos los retornos esperados son conocidos y siguen una distribución normal o que no hay costes de transacción o impuestos (Beste et al., 2002). Además, estos métodos tradiciones utilizan datos históricos para estimar los retornos futuros (Jin & Yuan, 2021), lo que provoca que en ciertos casos el uso de estos modelos conlleve resultados inexactos o no se pueda aplicar en ciertos contextos. Sin embargo, la

llegada de la IA y el Machine Learning (ML) ha traído consigo nuevas posibilidades y herramientas para los inversores ya que métodos como el DRL se basan en la experiencia y no en datos históricos, permitiendo la adaptación de estos nuevos modelos a distintas situaciones de los mercados y obteniendo mejores resultados que los modelos tradicionales (Sutton & Barto, 2018).

Se han hecho varios estudios alrededor de la aplicación de técnicas de ML al entorno de las finanzas. Principalmente se ha puesto el foco en aplicaciones para predecir los precios de las acciones y para crear estrategias de trading. También se ha analizado en gran cantidad como utilizar estar herramientas tecnológicas desde el punto de vista del retorno. Sin embargo, el estudio del riesgo o la volatilidad ha recibido muy poca atención estos últimos años, siendo este una variable de gran importancia para el análisis de inversiones (Muñoz & Castañeda, 2023). Por ello, he considerado que era una buena oportunidad relacionar dos conceptos que son el presente y futuro de las finanzas: el Deep Reinforcement Learning (DRL) y la volatilidad. En este trabajo se estudiará si la volatilidad afecta o no a un robo-trader que utiliza DRL para tomar decisiones de inversión.

El trabajo se va a estructurar de la siguiente manera. Se empezará realizando un estado del arte para ver qué es lo que se ha escrito del tema a tratar y en qué punto nos encontramos ahora mismo. Más tarde, se elaborará un marco teórico que ayudará a asentar las bases teóricas necesarias para una buena comprensión del trabajo. Después, se realizarán tres experimentos con tres carteras distintas. Una cartera compuesta por activos de volatilidad baja, otra cartera compuesta por activos de volatilidad media y una última compuesta por activos de volatilidad alta. Los activos se han seleccionado utilizando la beta, que representa la volatilidad de los retornos de un activo frente al mercado. Se ejecutará el código de DRL aplicado a estas carteras y se utilizará el ratio de Sharpe para realizar las comparaciones. Es importante mencionar que también se obtendrá el ratio de Sharpe que hubiera resultado si simplemente hubiéramos mantenido los activos de las distintas carteras en nuestra posesión sin realizar ninguna operación. Luego se interpretarán los resultados obtenidos y se realizará una conclusión.

2. Estado del arte

En esta sección se pretende explorar las investigaciones y los desarrollos más importantes que se han llevado a cabo en torno al Deep Reinforcement Learning (DRL) y las finanzas. Con esto podremos situar el proyecto dentro del panorama actual, identificando las contribuciones previas y posibles huecos que hay en la literatura académica.

Markowitz (1999) fue el primero en intentar construir un modelo matemático para la optimización de porfolios. Para él, la gestión de carteras es el proceso de toma de decisiones que implica la reasignación continua de una cantidad de fondos en diversos productos financieros con el objetivo de maximizar el rendimiento y al mismo tiempo limitar el riesgo. El modelo de Markowitz, como ya se ha mencionado antes, busca encontrar los porfolios que maximicen el retorno para un determinado nivel de riesgo (Markowitz, 1999). Sin embargo, sus hipótesis y la de los modelos que vinieron después hacen que su aplicación en el mundo sea muy difícil. Por ello, la llegada de las nuevas tecnologías permitió a los analistas financieros usar nuevas herramientas que les permitieran construir de forma más eficiente porfolios óptimos.

Con la llegada del ML se empezaron a elaborar numerosos modelos, principalmente árboles de decisión o redes neuronales, enfocados en resolver problemas en el mundo financiero. La gran mayoría de ellos se enfocaron en predecir tendencias o movimientos de precios, donde, utilizando precios históricos como input, una red neuronal devuelve como output un vector con los precios de los activos en un periodo futuro. Luego, un agente de bolsa puede utilizar estas predicciones para tomar una decisión y construir la cartera (Chakraborty, 2019). La implementación de estos modelos es bastante sencilla ya que se trata de aprendizaje supervisado, es decir, se utiliza un conjunto de datos etiquetados. El principal problema que presentan estos modelos es que dependen en gran medida en su precisión de predicción, pero predecir precios futuros y sus movimientos es una tarea altamente compleja. Además, estos modelos no devuelven acciones de mercado (comprar, vender o mantener un activo), sino que simplemente devuelven precios. Cuando los investigadores intentaron que los modelos tomasen una decisión por su cuenta para automatizar el proceso, se dieron cuenta que necesitaban una capa más de lógica en las redes neuronales. Cuando una red neuronal utiliza dos o más neurona, ya se hace referencia al término de aprendizaje profundo. Otro problema que presentaban estos algoritmos predictivos es que, al predecir precios o tendencias no tienen en cuenta

distintos factores como impuestos o costes de transacción, que son elementos que pueden influir altamente en la toma de decisión de un agente.

Por ello, viendo los problemas que estaban presentando estos algoritmos de aprendizaje supervisado, se empezó a tratar este problema como un problema de aprendizaje por refuerzo (este concepto se explicará más adelante), donde se puede integrar la predicción de precios y la consecuente construcción del porfolio en un solo paso. Al tratar este problema de esta manera, se puede formular este problema como un Proceso de Decisión de Markov y luego resolverse utilizando modelos como Monte Carlo, SARSA o Q-Learning (Chakraborty, 2019).

Más en profundidad dentro del RL aplicado al problema de optimización de carteras, hoy en día se utilizan tanto modelos basados en valor (modelos que asignan un valor a cada estado o acción en función del beneficio esperado a largo plazo, es decir, utilizando la función valor) como los mencionados Q-learning o SARSA, como modelos basados en políticas (encontrar directamente la política óptima tuneando un vector de parámetros) (Nachum et al, 2017), como DPG o DDPG. Las principales variables que componen estos modelos suelen ser el tiempo y el precio pasado de los activos. Por otro lado, como recompensa se utiliza el rendimiento de la cartera, el ratio de Sharpe o el ratio de Sortino. Además, algunos de estos modelos incluyen características que los modelos basados en Markowitz ignoran como pueden ser los costes de transacción o la inversión en activos libres de riesgo (Hambly et al., 2023). También se han realizado comparaciones entre estos dos tipos de modelos. En 2016 Du, Zhai y Lv utilizaron un problema de optimización de carteras que incluía un activo con riesgo y otro activo libre de riesgo para comparar el rendimiento de un algoritmo Q-learning y un algoritmo de Aprendizaje por Refuerzo Recurrente (RRL) utilizando tres funciones valor diferentes. Descubrieron que el algoritmo Q-learning es menos estable y que se debe usar el ratio de Sharpe como medida de recompensa.

Sin embargo, a pesar del éxito de los modelos de RL, se ha observado que estos modelos sufren cuando se intentan escalar y no pueden gestionar problemas de alta dimensionalidad principalmente porque las bases de datos financieras son muy grandes y dependen en gran cantidad de la variable tiempo (Mosayi et al., 2023). Por ello, los investigadores llegaron a la conclusión de que aplicar DRL a un problema de optimización de carteras que estuviera modelado como un Proceso de Decisión de Markov podría ser muy beneficioso. Además, si se incorpora el uso de modelos de DRL,

las carteras se pueden ir ajustando cada cierto tiempo en función de la estrategia tomada por el modelo. Actualmente los modelos incluyen tres estrategias: vender el activo, comprar el activo o mantener el activo

El DRL empezó a llamar la atención alrededor del 2015 debido a los grandes logros que se obtuvieron cuando se empezó a aplicar a videojuegos (François-Lavet et al., 2019). Google Deepmind fue el primero en desarrollar uno de estos modelos que era capaz de jugar al Atari a un nivel similar al de jugadores profesionales. Más tarde, utilizando este modelo como base, Google desarrolló el AlphaGo, que se convirtió en el primer modelo en derrotar a un humano en el juego Go. Como afirma François-Lavet (2019), esto fue un gran avance ya que se trata de un juego con una cantidad enorme de escenarios y movimientos diferentes posibles. Sin embargo, estos juegos se tratan como un problema de aprendizaje por refuerzo donde el espacio de acciones es discreto, es decir, podemos contar el número de acciones que puede llevar a cabo el agente. Básicamente, todas las acciones posibles que se pueden realizar se pueden representar en un conjunto finito. Pero cuando nos vamos a problemas relacionados con las finanzas, nos encontramos con que hay que tratarlos con un espacio de acciones continuo porque la cantidad de un activo que se puede comprar, vender o mantener es continua. Es cierto que se pueden discretizar las acciones de mercado (por ejemplo, tomar por acción invertir todo el capital en un activo) pero se ha visto que esto no aporta mucho valor.

Dentro de la aplicación de DRL a la gestión de carteras, los modelos que más se utilizan actualmente son el de Proximal Policy Optimization (PPO), Deep Deterministics Policy Gradient (DDPG) y modelos de Policy Gradient (PG). Estos modelos permiten obtener una estrategia de optimización de carteras en un espacio de acciones continuo. Es cierto que muchos de estos modelos se han probado con criptomonedas y otros activos de alta volatilidad. También se han probado Redes Neuronales Convolucionales (CNN) obteniendo muy buenos resultados (Mosayi et al., 2020). Aun así, la aplicación del DRL a las finanzas es algo muy novedoso y, aunque empieza a haber cada vez más información, la investigación acerca del tema es por ahora muy limitada.

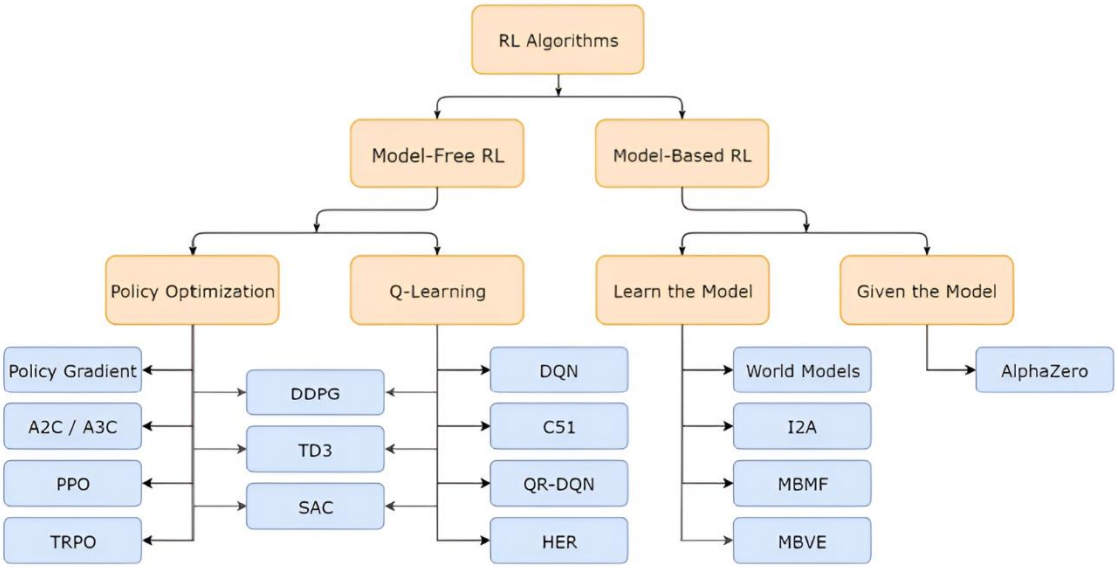

Figura 1: Taxonomía de algoritmos de RL. Adaptado de OpenAI. Part 2: Kinds of RL Algorithms

En resumen, se ha visto que los modelos clásicos de optimización de carteras no funcionan de forma eficiente en el mundo real debido a sus hipótesis. Por ello, investigadores y analistas están buscando nuevas herramientas que les permitan atacar este problema. Una de esas maneras es utilizando modelos de DRL. Se ha visto que algunos investigadores han obtenido muy buenos resultados utilizando estos algoritmos frente a modelos más clásicos. Sin embargo, al ser un tema tan novedoso y de reciente creación, más experimentos son necesarios para ver si realmente se obtienen mejores resultados utilizando modelos de DRL. Este trabajo intenta contribuir centrándose en algo tan relevante como la volatilidad. En este trabajo realizaremos tres experimentos que definiremos más adelante para ver si la volatilidad afecta o no a un robo-trader de DRL.

3. Definición del proyecto

Como bien se ha mencionado anteriormente, el trabajo pretende averiguar si la volatilidad afecta o no a un robo-trader que utiliza DRL para tomar decisiones de inversión. Para ello se va a aplicar el código de DRL a tres carteras distintas con distintas volatilidades. Para la selección de las carteras se ha utilizado como criterio la beta de los activos. De esa manera la composición de las carteras es la siguiente:

- Cartera 1: activos de volatilidad (beta menor que 1) seleccionados del S&P low volatility index. Los activos son: McDonalds, Coca-Cola, Kimberly-Crack y Procter & Gamble.
- Cartera 2: activos de volatilidad media (beta alrededor de 1) seleccionados del DJIA. Los activos son: Home Depot, Intel, 3M, Exxon y Honeywell.
- Cartera 3: activos de volatilidad alta (beta mayor que 1) seleccionados del NASDAQ 100. Los activos son: Apple, Tesla, Advanced Micro Devices, Nvidia y Qualcom.

Al ejecutar el código en cada una de estas carteras obtendremos varios ratios de Sharpe que compararemos entre sí para ver si la volatilidad afecta al rendimiento. También se obtendrá el ratio de Sharpe que obtendríamos si hubiéramos mantenido los activos en nuestro poder sin haber realizado ninguna operación para una mejor comparación.

3.1 Hipótesis

- Hipótesis nula: el ratio de Sharpe será igual en cada uno de los experimentos, es decir, obtendremos los mismos rendimientos en cada una de las carteras propuestas.
- Hipótesis alternativa: a mayor volatilidad, mejor será el ratio de Sharpe obtenido. La volatilidad sí que afecta al robo-trader que utiliza DRL y por tanto los ratios de Sharpe serán diferentes en cada uno de los experimentos.

3.2 Objetivos

1) Objetivo A: Comparar el rendimiento de modelos de DRL en función de distintas volatilidades.

2) Objetivo B: Realizar tres experimentos para obtener evidencias.
3) Objetivo C: Analizar la sensibilidad de los modelos utilizados a distintos parámetros.
4) Objetivo D: Estudiar la aplicabilidad real del código de DRL utilizado.

3.3 Asunciones

1) Asunción A: El comportamiento futuro de los activos tendrá un comportamiento similar al comportamiento pasado de los activos.
2) Asunción B: Los parámetros son estables a lo largo del tiempo
3) Asunción C: Los datos históricos son suficientes para llevar a cabo los experimentos de manera exitosa

3.4 Restricciones

1) Restricción A: Mi conocimiento sobre el tema. Aunque he invertido gran cantidad de tiempo en estudiar, mi conocimiento sobre el DRL en ciertas áreas puede suponer una limitación.
2) Restricción B: Acceso limitado a datos de los precios de los activos.
3) Restricción C: Falta de datos de precios en algunos días.
4) Restricción D: Sesgo en la selección de la muestra temporal.

4. Marco teórico

Para poder comprender bien este trabajo y los experimentos que se van a realizar, es necesario proporcionar unas bases conceptuales y unas definiciones esenciales. Esta sección se dividirá en dos partes, una primera que hablará de la volatilidad y su modelado, y una segunda parte que hablará de los conceptos fundamentales para entender el DRL, y más en concreto el algoritmo que se va a utilizar, el Proximal Policy Optimization.

4.1 Volatilidad

El nivel de riesgo es una de las características más importantes de un activo. Este riesgo total se puede dividir en dos partes: riesgo sistemático y riesgo no sistemático o específico. El riesgo sistemático es el riesgo que no se puede diversificar porque es el riesgo inherente al propio mercado y que afecta a todos y cada uno de los activos de dicho mercado. Por otro lado, el riesgo no sistemático es el riesgo que puede reducirse a través de la diversificación ya que es el riesgo propio de una empresa en particular (Novales, 2017).

Sin embargo, la gran pregunta que surge es como podemos medir ese riesgo total de las acciones. Una manera de hacerlo es a través de las fluctuaciones de los precios de un activo durante un periodo de tiempo concreto. Es por ello por lo que utilizamos la volatilidad como medida del riesgo de un activo. La volatilidad se entiende como la variabilidad del rendimiento de un activo frente a un rendimiento esperado (Rossi, 2013). Esto no quiere decir que un cambio en el precio o rendimiento signifique inmediatamente un cambio en la volatilidad. Si no que, cuando hacemos referencia a un incremento de la volatilidad, estamos haciendo referencia a una mayor dispersión del rendimiento de un activo frente a un rendimiento medio esperado.

Para medir la volatilidad se suele utilizar la desviación típica:

$$\sigma = \sqrt{\frac{\sum_{t=1}^{T}(R_t - \overline{R})}{T-1}} \quad (1)$$

Donde R_t es el retorno en el momento t y $\overline{R}$ la media de los retornos. Al final la desviación típica nos proporciona una medida de dispersión de los retornos que podemos utilizar para medir la volatilidad de un activo.

Cuando hablamos de volatilidad también es importante hablar de la beta de un activo. La beta es una medida de la volatilidad de un activo en relación con la variabilidad del mercado. Una beta mayor que 1 indica que el activo es más volátil que el mercado en general y por lo tanto el retorno esperado es superior al del mercado. Una beta igual 1 indica que la volatilidad de los rendimientos de un activo es igual a la volatilidad del mercado y se esperan retornos iguales o muy similares a los del mercado. Si la beta es menor que 1 quiere decir que el activo es menos volátil que el mercado y por lo tanto el retorno esperado es menor al del mercado. Es importante mencionar que se utilizará la beta para seleccionar nuestras carteras ya que los activos se seleccionarán de distintos índices y así podremos realizar una mejor comparación.

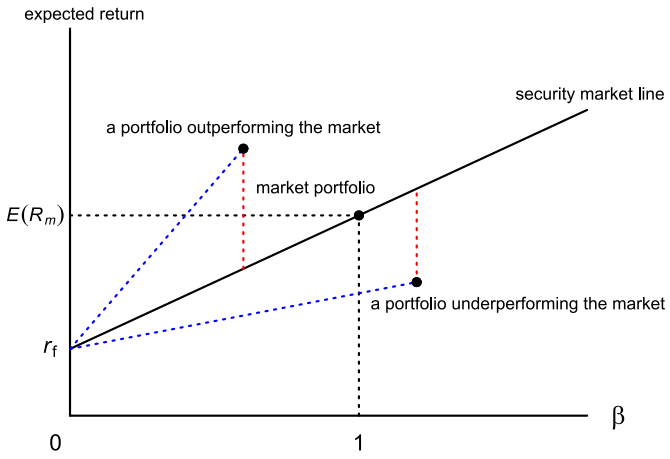

Figura 2: CAPM que describe la relación entre las betas y el retorno esperado. Adaptado de: Munasca. (2015). Capital Allocation Line (Modern Portfolio Theory).

4.2 Deep Reinforcement Learning

Para entender el DRL es necesario entender primero que es el RL, que es el DL y como se combinan ambos para formar el DRL.

Según Lasse Rouhianinen (2018) en su libro Inteligencia Artificial, esta se puede definir como “la habilidad de los ordenadores para hacer actividades que normalmente requieren de inteligencia humana. Más detalladamente se podría decir que la IA es la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano.”

Una de las ramas más importantes de la IA, es el ML que se puede definir como el proceso por el cual se programa a un ordenador para que aprenda de una serie de datos o de experiencias pasadas. La gran ventaja del ML es que sus modelos pueden aprender sin la necesidad de la intervención de un ser humano ya que, como hemos dicho anteriormente, aprende de los datos y de experiencias. Dentro de ML se distinguen cuatro tipos principales de aprendizaje: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje por refuerzo y aprendizaje profundo (Deep learning). Para este trabajo se pondrá el foco en aprendizaje por refuerzo y en el Deep learning.

4.2.1 Reinforcement Learning

En el aprendizaje por refuerzo o reinforcement learning (RL) el agente tiene que tomar una serie de decisiones con el objetivo maximizar la recompensa a largo plazo. El agente realiza una acción en un determinado entorno en función de un estado (que representa lo que el agente sabe del entorno) y recibe feedback en forma de una recompensa y nuevo estado. Con esta nueva información el agente realizará una nueva acción. Como bien se puede observar, la recompensa dependerá de las acciones que decida tomar el agente. Por ello, el algoritmo debe encontrar la política óptima, es decir, el conjunto o estrategia de acciones que maximicen la recompensa obtenida a largo plazo. La siguiente imagen muestra el proceso descrito:

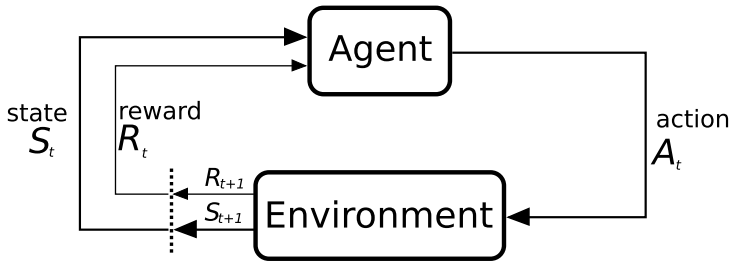

Figura 3: Proceso seguido por un algoritmo de RL y como se relacionan las distintas partes entre sí. Adaptado de: Gahete Díaz, J. L., Muñoz San Roque, A., Pizarroso Gonzalo, J., & Portela González, J. (2023). Machine Learning I

La interacción agente-entorno mostrada en la imagen sigue la propiedad de Markov, que nos dice que el futuro solo depende de la información actual que posea el agente. Es decir,

el efecto de una acción sobre un estado solo depende de la acción y del propio estado (Chakraborty, 2019).

Como el agente selecciona las acciones se conoce como política. Entonces, lo que tendrá que hacer un agente de RL es encontrar la política que optimice el retorno esperado a largo plazo.

Para encontrar dicha política óptima es indispensable hallar la función valor. Para ello, podemos recurrir a la ecuación de Bellman, ideada por Richard Bellman en 1964. Para ello, empezaríamos obteniendo la siguiente ecuación:

$$V(s) = \max\left(R(s,a) + \gamma * V(s`)\right) \quad (2)$$

Los elementos que componen la ecuación son los siguientes:

- s → define el estado.
- a → define la acción llevada a cabo por el agente.
- R → define la recompensa.
- γ → define el factor de descuento.

De esta forma, la función valor nos estima la recompensa que se va a recibir partiendo de un cierto estado y realizando una determinada acción. El factor de descuento es clave para ajustar ya que, cuanto más nos alejamos, menor será la recompensa.

Si seguimos descomponiendo la ecuación antes mencionada, llegamos a la ecuación de Bellman:

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma * V_\pi(S_{t+1}) | S_t = S] \quad (3)$$

La ecuación de Bellman nos dice el retorno o recompensa esperada empezando en el estado S y siguiendo la política π. Además, se puede ver que sigue la propiedad de Markov.

Esta ecuación la podemos llevar a un siguiente nivel al sustituir la V por la Q. La V nos indica el valor del estado en el que nos encontrábamos, pero la Q nos indica la calidad de la acción seleccionada en función del estado. Entonces la ecuación quedaría de la siguiente manera (Brea et al., 2023):

$$New\ Q(s,a) = Q(s,a) + \alpha * [R(s,a) + \gamma * maxQ'(s',a') - Q(s,a)] \quad (4)$$

Por otro lado, uno de los desafíos más importantes que presenta el RL es el trade off entre explotación y exploración. El modelo debe explotar las acciones que le han funcionado en el pasado, pero también debe explorar nuevas acciones. Hacer únicamente una de las dos es un error por lo que encontrar el equilibrio entre ambas es de gran importancia.

4.2.2 Deep Learning

Como bien define Torres, el Deep Learning "es un subconjunto de Machine Learning (ML), que es solo una parte de la inteligencia artificial (IA).

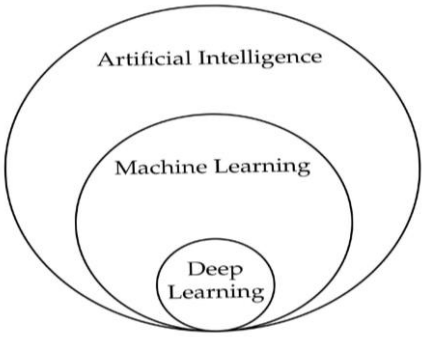

Figura 4: Clasificación visual de la IA, ML y DL . Adaptado de: Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

El Deep learning busca crear algoritmos que puedan aprender a través de redes neuronales profundas. Estas redes neuronales no abordan los problemas de manera secuencial, sino que lo hacen de forma paralela de la misma manera que lo hace un humano. La gran ventaja del Deep learning frente a los otros tipos de aprendizaje es que se pueden usar los datos en su forma original y permite obtener soluciones que no sean lineales. Otras técnicas de ML requieren convertir los datos a otras formas o crear nuevas variables. Sin embargo, el Deep learning requiere de una cantidad de datos y una capacidad de computación mucho mayor que otras técnicas (Chauhan & Singh, 2018).

Las redes neuronales están basadas en el cerebro humano y se componen de varias capas de neuronas (Balhara et al., 2022). En cada neurona, una función de activación determina la salida en función de los datos de entrada que ha recibido.

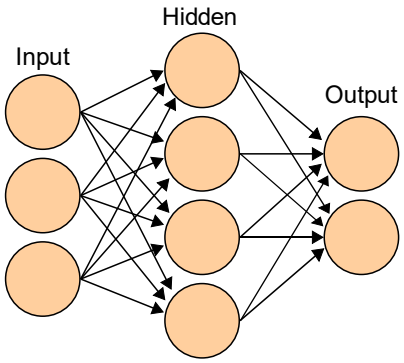

Figura 5: Esquema de una red neuronal con una capa oculta. Adaptado de: Colin M.L. Burnet

La primera capa recibe los valores de entrada. Los siguientes valores de la capa oculta son una transformación de estos valores mediante una función paramétrica no lineal (aquí es donde entra en juego la función de activación). Estos datos se pueden seguir transformando hasta obtener los valores de salida (Francois-Lavet et al., 2018).

Si se analiza más en detalle una sola neurona, se puede ver que el perceptrón es la unidad matemática más simple. Admite una serie de inputs con sus respectivos pesos y la suma ponderada de estos inputs pasa a través de la ya mencionada función de activación para obtener una serie de outputs. Entonces básicamente lo que ocurre en una neurona es la siguiente formula:

$$Y = f(\sum_{i=1}^{n} X_i * W_i) \quad (5)$$

Donde Y son los valores de salida de la neurona, f la función de activación, Xi los valores de entrada y Wi representa los pesos. Como se ha dicho anteriormente, el Deep Learning se caracteriza por tener varias capas de neuronas, por lo que esta fórmula se aplicará tantas veces como capas de neuronas haya.

Cuando una red neuronal está aprendiendo va ajustando los pesos asignados a los datos de entrada (que inicialmente son asignados de forma aleatoria) de manera que la diferencia entre el output de la red neuronal y la salida real sea la más pequeña posible y de esa forma minimizar el error.

Es por ello por lo que se puede observar la importancia de la función de activación en las redes neuronales. No hay una sola función de activación, sino que hay varias como puede

ser la función ReLU, que anula los valores negativos, o la función Sigmoide que devuelve valores entre 0 y 1.

4.2.3 Deep Reinforcement Learning

Recientemente se ha abierto un nuevo campo con la combinación del Deep Learning y el Reinforcement Learning dando lugar al Deep Reinforcement Learning (DRL). La siguiente imagen muestra como es una estructura de un modelo de DRL.

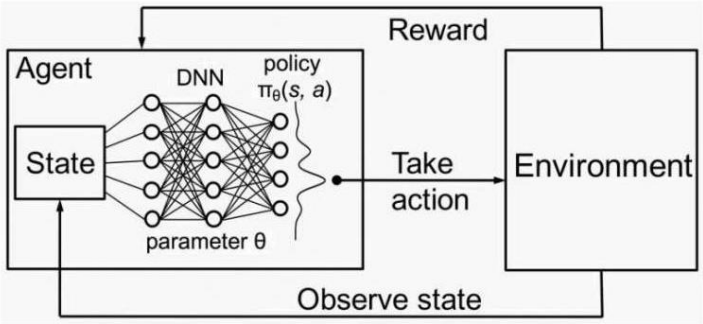

Figura 6: Estructura de un modelo de DRL y como se relacionan todas las partes entre si. Adaptado de: Fuentes Brea, J. P., López López, G., Palacios Hielscher, R., & Sánchez Úbeda, E. F. (2023). Machine Learning/Analítica Avanzada.

En el DRL el agente (una red neuronal profunda) interactúa con el entorno y aprende la política asociando acciones a determinados estados (Balhara et al., 2022). Gracias al DRL podemos mejorar el aprendizaje cuando nos encontramos con muchos datos y grandes espacios de estados y acciones como puede ocurrir con los datos financieros (Chakraborty, 2019). Hay muchos algoritmos y modelos dentro del DRL, pero nosotros nos vamos a centrar en el que vamos a utilizar en nuestro código, el Proximal Policy Optimization.

4.2.4 Proximal Policy Optimization

El algoritmo Proximal Policy Optimization (PPO) es una variante del algoritmo Trust Region Policy Optimization (TRPO) que a su vez se clasifica dentro de lo que se conoce como métodos basados en políticas. Estos métodos funcionan muy bien en espacios de acciones continuos (como son las finanzas) y, en vez de utilizar una función de valor separada, tratan de encontrar la política óptima directamente (los métodos basados en valor utilizan directamente la recompensa para seleccionar la siguiente acción). Estos

métodos empiezan con una determinada política que van mejorando poco a poco a través de métodos de políticas de gradiente.

Lo que hacen es ir seleccionando políticas aleatorias en cada paso y, si la nueva política es mejor que la anterior, ajustar los parámetros de la política anterior (por ejemplo, los pesos de la red neuronal) en la dirección de esa política nueva. Para ello, podemos utilizar la función valor como medida de la calidad de una política. La ecuación que representa todo esto es la siguiente:

$$\theta_{t+1} = \theta_t + \alpha * \nabla_\theta V^\pi(s_0) \quad (6)$$

Se puede ver como se empuja el conjunto de parámetros θ en la dirección de la mejor acción. Cada paso t el algoritmo realizará esta actualización de los parámetros utilizando la derivada de la función valor (ya que queremos maximizar) y una tasa de aprendizaje α.

Como se puede observar, estos métodos generan una política aleatoria en cada iteración antes de evaluar la calidad. Esto hace que haya una alta varianza ya que la trayectoria del modelo, que en los métodos basados en valor utiliza la recompensa como guía, es completamente aleatoria.

Uno de los métodos para intentar reducir la varianza de los métodos basados en políticas es el TRPO. El TRPO utiliza una función de pérdida para para reducir la varianza en los parámetros de la política. Es importante mencionar que no se pueden cambiar los parámetros de una política a otra en grandes magnitudes, sino que hay que ir poco a poco y esto hace que ir probando políticas aleatoriamente lleve mucho tiempo.

Es por eso por lo que también se busca acelerar el proceso del algoritmo. Para ello se utiliza lo que se conoce como regiones de confianza. Los algoritmos calculan la calidad de la aproximación de los parámetros antes comentada y si es buena, la región de confianza es aumentada. El TRPO compara la antigua y la nueva política para realizar el mayor ajuste posible a los parámetros. Es aquí cuando entra en juego la función de pérdida antes mencionada. Para poder hacer el mayor ajuste posible a los parámetros, el TRPO intenta maximizar la función de pérdida. En decir, lo que hace el TRPO es usar regiones de confianza para limitar el ajuste máximo de una política a otra.

El PPO tiene alguno de los beneficios del TRPO y el DDPG, pero es más fácil de implementar y es más barato. El PPO intenta evitar que la nueva política se aleje mucho

de la antigua limitando el gradiente. En el fondo sigue el mismo razonamiento que el TRPO, hacer la mejora más grande posible en los parámetros de la política sin que esto provoque un colapso (Bellemare et al., 2023).

5. Experimento

Una vez definido el proyecto, se va a describir formalmente el experimento y el código a utilizar para poder obtener los resultados necesarios para después realizar un análisis de estos. Es importante mencionar que el código utilizado se va a ejecutar en el entorno de Google Colab para que los comandos "pip" puedan funcionar. Básicamente lo que se necesita es un código que descargue datos financieros, realice ciertas modificaciones a esos datos, entrene un modelo de DRL y lo pruebe a través de una serie de datos reservado para testing.

5.1 Entorno

- **FinRL**: Se trata de una librería específica para finanzas y DRL en finanzas. Proporciona las herramientas y modelos necesarios para entrenar agentes de DRL en entornos financieros.
    - De esta librería se han importado varias funciones y módulos específicos. Los más importantes son los siguientes:
        - Yahoo Downloader: Se utiliza para descargar datos financieros de Yahoo Finance. Proporciona métodos para descargar datos históricos de precios de acciones, volumen de operaciones y otros datos relevantes.
        - FeatureEngineer: Se utiliza para preprocesar los datos y diseñar las características necesarias para el modelo. Estas características pueden ser volumen de negociación, indicadores técnicos o información fundamental entre otros.
        - StockPortfolioEnv: Este módulo implementa el entorno específicamente diseñado para simular el entorno de inversión en acciones. Define las observaciones, acciones y recompensas disponibles para un agente DRL que busca tomar decisiones de inversión.
        - DRLAgent: Proporciona una interfaz para entrenar y evaluar agentes de RL (como DQN, A2C, PPO, etc.) en entornos financieros. Como se ha mencionado anteriormente en este

experimento se va a utilizar el Proximal Policy Optimization (PPO).

- **Condacolab**: Es una librería que permite la integración de Colab con Conda para facilitar la instalación de paquetes a través de Conda en Colab.
- **Gym**: Esta librería proporciona una variedad de entornos que pueden ser utilizados para entrenar algoritmos de DRL.
- **Otras librerías estándar** de Python para el procesamiento de datos y visualización como pueden ser:
    - Pandas: Es una librería que permite el análisis y manipulación de datos.
    - Numpy: Facilita la realización de distintas operaciones numéricas con datos estructurados como cálculos estadísticos o manipulación de matrices.
    - Matplotlib: Proporciona herramientas para crear gráficos.

5.2 Código

Una vez visto el entorno y cuales son las librerías más importantes que se van a utilizar, es relevante realizar una breve descripción del código que se va a utilizar en este experimento. El código se puede consultar en el siguiente enlace. Como bien se ha mencionado anteriormente, el código se ha ejecutado en el entorno de Google Colab ya que en local no se tenían los suficientes recursos para ejecutarlo y los comandos "pip" no iban a funcionar.

En primer lugar, se procede a la instalación de todas las librerías necesarias mencionadas anteriormente y a la creación de lo que se conoce como "folders". Gracias a ello, el código verifica que existen ciertos directorios importantes. Si dichos directorios no existen, el código los crea automáticamente. De esta manera nos aseguramos de que los directorios necesarios están disponibles para poder guardar datos, modelos entrenados, gráficos, etc.… y evitar así que el programa de algún tipo de fallo. Más tarde se procede a crear las tres variables que almacenan los tickers de las tres carteras creadas en función de la volatilidad.

Una vez realizados estos pasos, se procede a diseñar el dataframe. Para ello se descargan los datos de Yahoo Finance y se procesan. Además, también se calcula la matriz de covarianza y el rendimiento de las acciones para el intervalo de tiempo establecido (252

días, que son los días hábiles en un año). La matriz de covarianza nos proporciona información sobre como se mueven los precios de las acciones en conjunto, lo que es crucial para entender el riesgo de las tres carteras creadas. Adicionalmente, gracias al módulo FeatureEngineer, podemos añadir algunas características como indicadores técnicos. Después, como ya se han obtenido y procesado los datos necesarios, se puede dividir el dataset en dos partes, una para el entrenamiento y otra para test. El dataset de entrenamiento se compone de los datos que van desde el 1 de enero de 2008 al 31 de diciembre de 2022. Por otro lado, el dataset de prueba se compone de los datos que van desde el 1 de enero de 2023 hasta el 31 de diciembre de 2023.

Para poder entrenar a nuestro modelo es necesario definir un entorno de aprendizaje (utilizando OpenAI Gym). Para ello, tenemos que inicializar dicho entorno con una serie de parámetros como los datos históricos, la dimensión de las acciones, la cantidad de dinero inicial disponible para operar o el coste de cada transacción. Además, hay que definir el espacio de acciones que puede realizar el algoritmo y el espacio de observación, que representa la información sobre el entorno que posee el modelo y que utilizará para decidir que acciones tomar en cada paso. Es importante mencionar también que a través de la función "step" se puede definir que es lo que ocurre en cada paso de tiempo en el entorno. Se recibe una acción como entrada, se actualiza el estado del entorno en función de esta acción y se calcula la recompensa en función de la acción tomada y el estado actual, devolviendo un nuevo estado, es decir, una nueva información que el agente puede observar. Además, se han creado diversas funciones para ir guardando información sobre el rendimiento de las carteras y las acciones realizadas durante la ejecución del entorno. En resumen, en esta parte del código se crea una clase que simula un entorno de trading que utilizaremos para entrenar a nuestro modelo. Dentro de esta clase se han definido distintos parámetros clave, que acciones se pueden realizar con cada paso del tiempo y como se guarda la información sobre el rendimiento de las carteras y las acciones realizadas. Como tenemos tres carteras de acciones tenemos que configurar los argumentos y parámetros del entorno definido anteriormente para cada una de las tres carteras.

Una vez establecidos los entornos, se crean tres agentes de DRL, uno para cada una de las tres carteras de acciones. Estos agentes se encargarán de aprender como tomar decisiones en función de la información del mercado. Este experimento se realizará varias veces con diferentes configuraciones aleatorias para entender mejor como varían los

modelos. Para ello es indispensable crear una serie de semillas. También hay que establecer el número de pasos de tiempo que se utilizarán para entrenar cada agente de DRL. Cuantos más pasos de tiempo se utilicen, más entrenamiento recibirán los modelos, pero también aumentará el tiempo de ejecución.

Finalmente, se crea la parte del código donde se itera a través de un bucle “for” sobre una serie de métodos, donde para cada método se entrena un modelo PPO con diferentes parámetros. Luego, iteras sobre un conjunto de semillas y entrenas el modelo con cada semilla. Después del entrenamiento, evalúas el rendimiento del modelo usando el ratio de Sharpe. Esto se hace para cada agente creado en función de las tres carteras. Después, imprimimos una gráfica con los resultados para su mejor comprensión y análisis, y para poder comparar los resultados obtenidos con las tres carteras.

6. Resultados

Una vez ejecutado el código, se han obtenido los resultados para cada una de las tres carteras creadas: alta volatilidad, volatilidad media y baja volatilidad. Es importante mencionar que estos resultados obtenidos son una mera prueba de concepto que en ningún caso refleja todo el potencial de comportamiento del DRL puesto que existía una limitación de capacidad computacional a la hora de ejecutar el código.

Como se ha mencionado anteriormente, la métrica seleccionada para interpretar los resultados es el ratio de Sharpe ya que es una medida que evalúa el rendimiento de un activo en relación con su riesgo. Aunque se ha hecho de manera automática en la última parte del código, el ratio de Sharpe sigue la siguiente fórmula:

$$Ratio\ de\ Sharpe = \frac{\bar{R}_p - R_f}{\sigma_p} \quad (7)$$

Por ello, una vez ejecutada el código los resultados obtenidos son los siguientes:

- Ratio de Sharpe de la cartera de volatilidad baja: 0.163
- Ratio de Sharpe de la cartera de volatilidad media: 0.229
- Ratio de Sharpe de la cartera de volatilidad alta: **2.678**

La siguiente imagen muestra de manera visual los resultados obtenidos:

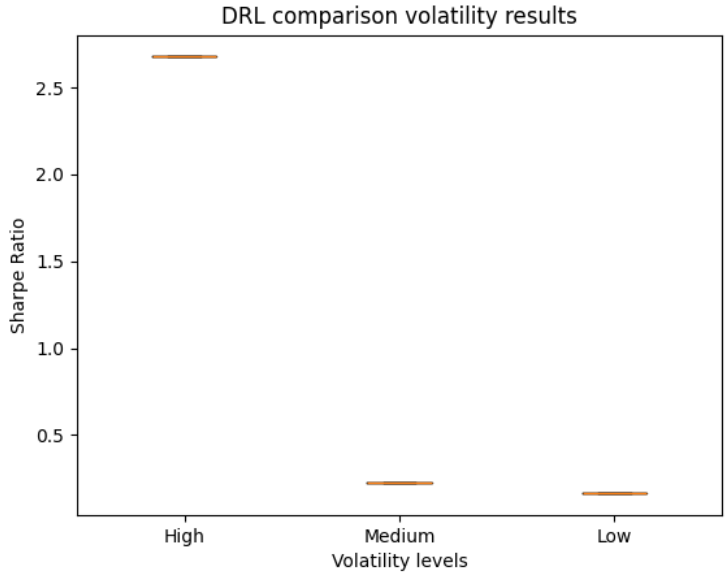

Figura 7: Resultados obtenidos con DRL. Elaboración propia

Como bien se puede observar, la cartera de alta volatilidad obtiene un mejor rendimiento con relación al riesgo en comparación con las carteras de media y baja volatilidad. Un ratio de Sharpe de 2.678 indica que esta cartera ha generado retornos por encima de la tasa libre de riesgo de 2.678 unidades por cada unidad de riesgo, es decir, se están obteniendo retornos más altos en comparación con el riesgo asumido. Por otro lado, aunque los ratios de Sharpe de las carteras de media y baja volatilidad siguen siendo positivas, se puede observar que están generando rentabilidades que son solo marginalmente superiores al riesgo que asumen por lo que se podría considerar que el rendimiento no es lo suficientemente alto como para compensar el riesgo asumido en estas dos carteras. Esto también indica una menor eficiencia en la obtención de retornos en relación con el riesgo cuando el modelo opera con carteras de medio o baja volatilidad. Aun así los resultados obtenidos verifican la hipótesis alternativa que decía que a mayor volatilidad, mejor sería el ratio de Sharpe.

Para poder realizar una mejor comparación sobre los resultados obtenidos utilizando el modelo de Deep Reinforcement Learning, se ha obtenido el ratio de Sharpe de cada una de las tres carteras si siguieran la estrategia de mercado durante 2023. Para ello, se ha calculado el retorno diario de cada una de las acciones para después poder obtener el retorno promedio y la volatilidad de dichos retornos diarios. Con estos cálculos se puede aplicar luego la fórmula del ratio de Sharpe antes mencionada.

Una vez realizado esto se obtienen los siguientes resultados:

- Ratio de Sharpe para la cartera de volatilidad baja: **2.136**
- Ratio de Sharpe para la cartera de volatilidad media: 1.357
- Ratio de Sharpe para la cartera de volatilidad alta: 0.694

La siguiente tabla muestra la comparación de ambos modelos:

	Modelo de DRL	Estrategia de mercado
Volatilidad baja	0.163	**2.136**
Volatilidad media	0.229	**1.357**
Volatilidad alta	**2.678**	0.694

Tabla 1: Ratios de Sharpe obtenidos utilizando el modelo de DRL y la estrategia de mercado. Elaboración propia.

Estos resultados nos indican que el modelo de DRL elaborado parece tener un desempeño superior a la estrategia de mercado cuando se opera con activos de alta volatilidad, pero no logra superarla cuando los activos poseen un beta cercana o menor que uno. Esto nos dice que el modelo es más eficiente cuando opera con activos de altas volatilidades. Por ello, puede ser que nuestro modelo obtenga buenos rendimientos operando con activos como las criptomonedas o divisas que fluctúan mucho.

7. Conclusión

El objetivo de este trabajo era examinar como la volatilidad podía afectar a un robo-trader que utiliza un modelo de Deep Reinforcement Learning para operar. Se ha visto que los modelos tradiciones para la selección y asignación de activos se apoyan en muchas hipótesis que hacen que su aplicación en el mundo real no sea del todo adecuada. Por ello, la llegada de la IA y el ML ha abierto nuevas posibilidades para los inversores ya que, aparte de no basarse en ninguna de las hipótesis en las que se basan los modelos tradicionales, los modelos de DRL utilizan la experiencia para su entrenamiento, permitiendo así su adaptación a distintas situaciones de los mercados.

Por ello, viendo que el estudio de la volatilidad ha recibido poca atención estos últimos años, se consideró relevante realizar un estudio que uniera el DRL y la volatilidad. Para ello, se crearon tres carteras distintas: una que incluía activos con betas menores a 1, otra con activos con betas alrededor de 1 y una última con activos cuyas betas era mayores que 1. Más tarde se desarrolló un código (enlace) en python para poder realizar este estudio. Este código descarga los datos de los activos seleccionados desde Yahoo Finance y, después de procesar los datos, crea un dataset con toda la información necesaria. Después se desarrolla todo lo necesario para poder utilizar el algoritmo PPO, un algoritmo de DRL que es una variante del TRPO. Una vez desarrollado todo el código, se pasó por cada una de las carteras para poder obtener los tres ratios de Sharpe, que es la métrica que ha sido seleccionada para poder comparar los distintos resultados.

Se ha visto que la cartera de alta volatilidad es la que obtiene el ratio de Sharpe más alto y que, aunque, el ratio de Sharpe para las carteras de media y baja volatilidad sigue siendo positivo, puede considerarse que su rendimiento no es lo suficientemente alto como para compensar el riesgo asumido. Además, cuando se compararon estos resultados con el ratio de Sharpe obtenido si seguía la estrategia de mercado, se pudo observar que la estrategia de mercado obtenía mayores rendimientos en relación con el riesgo asumido en las carteras de media y baja volatilidad. Sin embargo, para la cartera de alta volatilidad el ratio de Sharpe seguía siendo mucho más grande cuando se utilizaba el modelo de DRL. Esto permite afirmar que nuestro modelo obtiene mejores rendimientos con volatilidades muy altas, sugiriendo su aplicación en activos como las criptomonedas o monedas de mucha fluctuación.

Sin embargo, es importante mencionar que estos resultados son una mera prueba de concepto que en ningún caso refleja todo el potencial del DRL ya que existía una cierta limitación en cuanto a la capacidad computacional a la hora de ejecutar el código. Por ello, para futuras investigaciones se sugiere aumentar dicha capacidad computacional para exprimir este algoritmo creado lo máximo posible. Además, para estudiar todavía más en profundidad el impacto de la volatilidad en modelos que utilizan DRL, se aconseja la creación de muchas más carteras con muchas volatilidades diferentes, y para un estudio más completo, estas carteras podrían crearse siguiendo otras medidas de volatilidad y no solo teniendo en cuenta las betas de los activos.

8. Declaración de Uso de Herramientas de Inteligencia Artificial Generativa en Trabajos Fin de Grado

Por la presente, yo, Luis García Castro, estudiante de ADE + Business Analytics de la Universidad Pontificia Comillas al presentar mi Trabajo Fin de Grado titulado Impacto de la volatilidad en un robo-trader de Deep Reinforcement Learning ", declaro que he utilizado la herramienta de Inteligencia Artificial Generativa ChatGPT u otras similares de IAG de código sólo en el contexto de las actividades descritas a continuación:

1. **Interpretador de código:** Para realizar análisis de datos preliminares.
2. **Corrector de estilo literario y de lenguaje:** Para mejorar la calidad lingüística y estilística del texto.
3. **Sintetizador y divulgador de libros complicados:** Para resumir y comprender literatura compleja.
4. **Traductor:** Para traducir textos de un lenguaje a otro.

Afirmo que toda la información y contenido presentados en este trabajo son producto de mi investigación y esfuerzo individual, excepto donde se ha indicado lo contrario y se han dado los créditos correspondientes (he incluido las referencias adecuadas en el TFG y he explicitado para que se ha usado ChatGPT u otras herramientas similares). Soy consciente de las implicaciones académicas y éticas de presentar un trabajo no original y acepto las consecuencias de cualquier violación a esta declaración.

Fecha: Abril 2024

Firma: Luis García Castro

9. Bibliografía

Badr, H., Ouhbi, B., & Frikh, B. (2020, November). Rules based policy for stock trading: a new deep reinforcement learning method. In *2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech)* (pp. 1-6). IEEE.

Baker, H. K., & Filbeck, G. (Eds.). (2013). *Portfolio theory and management*. Oxford University Press, USA.

Balhara, S., Gupta, N., Alkhayyat, A., Bharti, I., Malik, R. Q., Mahmood, S. N., & Abedi, F. (2022). A survey on deep reinforcement learning architectures, applications and emerging trends. *IET Communications*.

Bellemare, M. G., Dabney, W., & Rowland, M. (2023). *Distributional reinforcement learning*. MIT Press.

Beste, A., Leventhal, D., Williams, J., & Lu, Q. (2002). The Markowitz Model. *Selecting an Efficient Investment Portfolio". Lafayette College, Mathematics REU Program.*

Briola, A., Turiel, J., Marcaccioli, R., Cauderan, A., & Aste, T. (2021). Deep reinforcement learning for active high frequency trading. *arXiv preprint arXiv:2101.07107*.

Chakraborty, S. (2019). Capturing financial markets to apply deep reinforcement learning. *arXiv preprint arXiv:1907.04373*.

Chen, J. M., & Chen, J. M. (2016). *Modern portfolio theory* (pp. 5-25). Palgrave Macmillan US.

Elton, E. J., & Gruber, M. J. (1997). Modern portfolio theory, 1950 to date. *Journal of banking & finance*, *21*(11-12), 1743-1759.

Fischer, T. G. (2018). *Reinforcement learning in financial markets-a survey* (No. 12/2018). FAU discussion papers in economics.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018). An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, *11*(3-4), 219-354.

Fuentes Brea, J. P., López López, G., Palacios Hielscher, R., & Sánchez Úbeda, E. F. (2023). Machine Learning/Analítica Avanzada.

Gahete Díaz, J. L., Muñoz San Roque, A., Pizarroso Gonzalo, J., & Portela González, J. (2023). Machine Learning I.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Hambly, B., Xu, R., & Yang, H. (2023). Recent advances in reinforcement learning in finance. *Mathematical Finance*, *33*(3), 437-503.

Harry M. Markowitz (1999) The Early History of Portfolio Theory: 1600–1960, Financial Analysts Journal, 55:4, 5-16, DOI: 10.2469/faj.v55.n4.2281

Jin, M., Li, Z., & Yuan, S. (2021, December). Research and analysis on Markowitz model and index model of portfolio selection. In *2021 3rd International Conference on Economic Management and Cultural Industry (ICEMCI 2021)* (pp. 1142-1150). Atlantis Press.

Kevin, S. (2022). *Security analysis and portfolio management*. PHI Learning Pvt. Ltd.. Mosavi, A., Faghan, Y., Ghamisi, P., Duan, P., Ardabili, S. F., Salwana, E., & Band, S.

Lei, K., Zhang, B., Li, Y., Yang, M., & Shen, Y. (2020). Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading. Expert Systems with Applications, 140, 112872.

Liu, X. Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., & Wang, C. D. (2020). FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. *arXiv preprint arXiv:2011.09607*.

Liu, X. Y., Xia, Z., Rui, J., Gao, J., Yang, H., Zhu, M., ... & Guo, J. (2022). FinRL-Meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, *35*, 1835-1849.

Maeda, I., DeGraw, D., Kitano, M., Matsushima, H., Sakaji, H., Izumi, K., & Kato, A. (2020). Deep reinforcement learning in agent based financial market simulation. *Journal of Risk and Financial Management*, *13*(4), 71.

Mosavi, A., Faghan, Y., Ghamisi, P., Duan, P., Ardabili, S. F., Salwana, E., & Band, S. S. (2020). Comprehensive review of deep reinforcement learning methods and applications in economics. *Mathematics*, *8*(10), 1640.

Muñoz, J. E. M., & Castañeda, R. (2023). El uso de machine learning en volatilidad: una revisión usando K-means. *Universidad & Empresa*, *25*(44), 1.

N. K. Chauhan and K. Singh, "A Review on Conventional Machine Learning vs Deep Learning," *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, Greater Noida, India, 2018, pp. 347-352, doi: 10.1109/GUCON.2018.8675097.

Nachum, O., Norouzi, M., Xu, K., & Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, *30*.

Novales, A. (2017). Midiendo el riesgo en mercados financieros. *Departamento de economía cuanfitafiva. Universidad Complutense*.

Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Applied Soft Computing*, *93*, 106384.

Pagdin, I., & Hardy, M. (2017). *Investment and Portfolio Management: A Practical Introduction*. Kogan Page Publishers.

Rossi, G. D. (2013). La volatilidad en mercados financieros y de commodities: Un repaso de sus causas y la evidencia reciente. *Invenio*, *16*(30), 59-74.

Rouhiainen, L. (2018). Inteligencia artificial. *Madrid: Alienta Editorial*, 20-21.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Schmitt, S., Hudson, J. J., Zidek, A., Osindero, S., Doersch, C., Czarnecki, W. M., ... & Eslami, S. M. (2018). Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*.

Yu, P., Lee, J. S., Kulyatin, I., Shi, Z., & Dasgupta, S. (2019). Model-based deep reinforcement learning for financial portfolio optimization. In *RWSDM Workshop, ICML* (Vol. 1, p. 2019).

Zai, A., & Brown, B. (2020). *Deep reinforcement learning in action*. Manning Publications.

Zhang, Z., Zohren, S., & Roberts, S. (2019). Deep reinforcement learning for trading. *arXiv preprint arXiv:1911.10107*.