Deep Reinforcement Learning, Análisis Técnico y Análisis de Sentimientos para Operar sobre Bitcoin

Alumno: Franco M. Ferrari
Tutor: Ramiro H. Gálvez
Año: 2021

Abstract

In the financial world, more specifically in the asset trading sector, different types of analysis are usually carried out and different strategies are followed to obtain the highest possible net profit. This type of analysis can be classified into two: fundamental analysis and technical analysis. Both, with completely different approaches, seek to provide the greatest amount of information for making informed decisions about a potential investment. The first one, seeks to understand political, social, market issues, etc. that impact on the price of an asset. The second one, regardless of the asset, seeks to understand the behavior of its price based on concrete data from the past. Some years ago, cryptocurrencies appeared. A new niche and investment opportunity for traders. One of the many complexities that these assets have, is that since they are based on blockchain technology, the anonymity and lack of support for them by formally recognized and regulated entities such as banks or companies, make it impossible to carry out the classic fundamental analysis as it is. This thesis uses the latest technologies in the field of data science, such as Deep Reinforcement Learning. Different performance improvement techniques are tested, and also data that is believed could add information of the type used in fundamental analysis, to develop a trading Bot. This one operates and learns about one of the most famous cryptocurrencies, Bitcoin, with the objective of generating the highest possible net profit. The impact of each of the alternatives tested on this work and their contribution to improving performance and feasibility is discussed. Also, at the end it is discussed how these models can be applied to generate value in the daily work of human traders.

Resumen

En el mundo financiero, más específicamente en el sector del trading de activos, se suelen realizar distintos tipos de análisis y seguir distintas estrategias para obtener la mayor ganancia neta posible. Estos tipos de análisis se los puede clasificar en dos: análisis fundamental y análisis técnico. Ambos con enfoques completamente distintos buscan aportar la mayor cantidad de información para la toma de decisiones informada sobre una potencial inversión. El primero busca entender cuestiones políticas, sociales, de mercado, y otras que impacten sobre el precio de un activo. El segundo, independientemente de cual sea el activo, busca comprender el comportamiento de su precio basándose en datos concretos del pasado. Ya hace algunos años aparecieron las criptomonedas. Un nuevo nicho y oportunidad de inversión para los traders. Una de las tantas complejidades que tienen estos activos es que, al estar basadas en tecnología blockchain, el anonimato y la falta de respaldo de las mismas por entidades formalmente reconocidas y reguladas como bancos o empresas, imposibilitan realizar el análisis fundamental clásico como tal. En esta tesis se utilizan las últimas tecnologías en el campo de la ciencia de datos, como el Deep Reinforcement Learning. Se prueban distintas técnicas de mejora de performance, y se añaden datos que se cree, podrían sumar información del tipo que se utiliza en el análisis fundamental, para desarrollar un Bot de trading. Este bot opera y aprende sobre una de las criptomonedas más conocidas, el Bitcoin, con el objetivo de generar la mayor ganancia neta posible. Se discute el impacto de cada una de las alternativas probadas en el trabajo y el aporte de las mismas a la mejora de la performance y factibilidad. Al final, se discute cómo estos modelos pueden ser aplicados para generar valor en el trabajo diario de los traders humanos.

Capítulo 1: Introducción

En el ámbito financiero, ámbito definido como el área de la economía que estudia la obtención y administración de recursos financieros, en los últimos años han surgido nuevas técnicas y tecnologías de análisis que están revolucionando la industria en la que se desenvuelve. Particularmente en el desarrollo de estrategias de *trading*. Se denomina trading a un nicho del dominio de las finanzas. El mismo consiste en la compraventa de activos cotizados con gran liquidez en el mercado financiero eléctronico que hasta cierto punto se encuentra regulado. Hoy en día, existen gran cantidad de tipos diferentes de análisis, tanto técnicos como fundamentales, que realizan los llamados “traders” (personas que trabajan haciendo venta y compra de acciones). El análisis fundamental se basa comúnmente en comprender los rendimientos de las empresas o sectores del mercado en el que se desea invertir: cuestiones empíricas, implicancias socio-políticas o económicas y comunicados de prensa que los involucran, entre otras. Además, también se realiza lo que se conoce como análisis técnico. Este tipo de análisis hace uso de estimaciones y predicciones basadas en datos del pasado, que surgen a partir de resultados obtenidos desde sencillos modelos matemáticos hasta complejos algoritmos, identificando así patrones de comportamiento sobre el futuro de una acción (Baggini, 2017). Ya sea con análisis fundamental o técnico, el resultado de dichas intuiciones o predicciones se utiliza para hacer provecho de la información imperfecta, teniendo en cuenta la pertinencia de la información, para sacar el mayor rédito (ganancia) posible, tanto en la compra como en la venta de activos. Por lo tanto, esta información termina convirtiéndose en una herramienta de soporte a la toma de decisiones para el dia a dia de los traders.

En los últimos años y con la aparición de las criptomonedas se abrió un nuevo nicho para las inversiones y los traders. Esto es porque el valor de las mismas suele tener cierta volatilidad a lo largo del tiempo. El desafío aquí se encuentra en que la mayoría de estas criptomonedas, o por lo menos las más conocidas, no se encuentran respaldadas por una institución financiera, empresa o mercado, ya que una de sus principales características es la anonimidad que brindan. Por lo tanto, los análisis que comúnmente realizan los traders para la compra y venta de acciones (principalmente el análisis fundamental) no aplican directamente sobre las criptomonedas, ya que no siempre se tiene certeza de la razón por la cual su precio varía. Por ende, no se puede sacar provecho de lo que se conoce como “información imperfecta” en el ámbito financiero. Entonces, surge la pregunta ¿Qué técnicas o análisis utilizan hoy en día los traders para trabajar sobre inversiones en criptomonedas? Si bien hay muchas discusiones al respecto, una de las técnicas que se podrían encuadrar dentro de lo que se conoce como análisis técnico es la utilización de modelos de Machine Learning y Deep Learning para identificar patrones y predecir el precio futuro de estos activos. Si bien estos análisis sirven como una herramienta para dar soporte a la toma de decisiones en la compra y venta de criptomonedas, en esta tesis se busca dar un paso más allá.

En este trabajo se exponen algunas estrategias de inversión clásicas y un bot de Reinforcement Learning básico ya existente, entrenado por los creadores de la librería TensorTrade[1], que no solo predice el valor futuro de una criptomoneda, sino que además toma la decisión, opera comprando/vendiendo y aprende de su accionar (King, 2019). Una de las críticas que reciben los métodos estadísticos y dichos bots, es que estarían realizando únicamente análisis técnico y dejando de lado por completo el análisis fundamental. El problema real sobre esto es: ¿Sobre qué información se podría realizar técnicas de análisis fundamental para criptos? Para responder esta pregunta, hay que tener en cuenta que el análisis fundamental es el método que se utiliza para valorar un activo dentro de un contexto determinado, y que el mundo de las criptomonedas se desarrolla en un contexto que depende de la opinión de desarrolladores, mineros, emprendedores, traders, inversores y gente influyente que comenta al respecto en las redes sociales. De la misma manera, se debe tener en cuenta el impacto por la aparición de nuevas regulaciones gubernamentales en cada país y cómo son percibidas por el resto de los agentes anteriores.

Esta tesis parte de utilizar el bot anterior para marcar un *benchmark*[2], con el objetivo de experimentar y revelar a lo largo de varias pruebas, distintas técnicas y metodologías que mejoren y superen al mismo. Para ello se explora un campo de estudio conocido como Deep Reinforcement Learning, con el cual se entrena un nuevo bot de trading a partir de variables técnicas. Además, se utilizan otras herramientas enmarcadas en el campo de la ciencia de datos, tal como la ingeniería de atributos. Por último, se pretende incluir variables de análisis de sentimientos provenientes de redes sociales, en la búsqueda de cubrir el déficit de información fundamental. (Badiola Ramos, 2019). Al comienzo del trabajo, se asumen ciertos supuestos sobre cómo medir e identificar si un modelo de estas características se encuentra performando bien; es decir, que esté aprendiendo realmente sobre la estrategia que se le está enseñando. También se define cómo medir la factibilidad de estos modelos, refiriéndose a que la estrategia aprendida genere ganancias al ser utilizada en las ruedas de operaciones.

Se trabaja con dos fuentes de datos. La primera con información histórica del valor de apertura/cierre, variaciones y volumen por hora de la criptomoneda Bitcoin proveniente de exchanges[3] oficiales. De esta fuente se extraen y crean todas las variables necesarias relacionadas al análisis técnico. La segunda fuente de información es el conjunto de datos creado, procesado y utilizado por Jaime Badiola Ramos en su trabajo (Badiola Ramos, 2019). Estos datos provienen de la mundialmente conocida red social Twitter. De esta plataforma se tomaron aproximadamente 17,7 millones de tweets que fueron analizados por un clasificador de sentimientos.

Utilizando los supuestos y definiciones de performance y factibilidad, en esta tesis se quiere probar la hipótesis de que es posible realizar análisis técnico y fundamental sobre Bitcoin de una manera mayormente automática, mediante algoritmos matemáticos de Deep Reinforcement Learning, ingeniería de atributos y variables de análisis de sentimientos. De esta forma, se obtiene una herramienta que hora a hora puede recomendar el mejor modo de operar para mantener la rentabilidad de los activos en Bitcoin y generar ganancias.

---

[1] https://www.tensortrade.org/en/latest/
[2] Cualquier índice que se utilice como referencia para hacer comparaciones en el ámbito financiero.
[3] Un exchange de criptomonedas es el punto de encuentro donde se realizan los intercambios de estas a cambio de dinero efectivo o de otras criptomonedas.

Capítulo 2: Marco teórico

En este capítulo se elaborará el marco teórico compuesto por antecedentes, investigaciones previas, consideraciones teóricas y breves explicaciones de ciertas técnicas. Marco que dará soporte a las bases de las cuales se parte y desarrolla este trabajo. Estas bases estarán compuestas por variados dominios y nichos pertenecientes a criptomonedas, ciencia de datos y finanzas.

2.1 Mundo de las criptomonedas

En esta sección se busca dar una mirada general a distintos conceptos relacionados al mundo de las criptomonedas, que se considera importante tener en cuenta para interpretar el problema y enfoque que se le quiere dar a la solución.

2.1.1 Criptomonedas

Una criptomoneda es un sistema de moneda virtual que funciona de manera muy similar a una moneda estándar, lo que permite a los usuarios realizar pagos virtuales por bienes y servicios sin una autoridad central de confianza. Estas se basan en la transmisión de información digital utilizando métodos criptográficos para garantizar transacciones legítimas y únicas (Farell, 2015).

2.1.2 Bitcoin

Si bien el concepto de *electronic currency* existe desde la época de 1980, Bitcoin fue lanzado en 2009 bajo el pseudónimo de Satoshi Nakamoto[4], convirtiéndose en la primera criptomoneda descentralizada utilizando la tecnología blockchain (Farell, 2015). Este hecho es el que generó que hoy, el BTC sea una de las criptomonedas más famosas, en conjunto con otras pioneras como Ether (ETH), la cual se utiliza en los contratos inteligentes, o Tether (USDT) utilizada como una criptomoneda estable que guarda una relación 1 a 1 con el dólar americano. Por ser el Bitcoin la criptomoneda más famosa, existe una cantidad considerable de aficionados a estas nuevas tecnologías o “modas de invertir”, que lo hacen de manera totalmente especulativa, sin realizar ningún tipo de análisis profesional, solo por el hecho de *seguir a la masa*[5].

2.1.3 Blockchain

El blockchain es un sistema de contabilidad pública que mantiene la integridad de los datos de transacciones realizadas. Uno de sus primeros usos fue convertirse en el sistema que mantiene las transacciones de la criptomoneda Bitcoin (Swan, 2015). El Bitcoin utiliza un mecanismo de clave pública, mejor conocido como PKI (Housley, 2004). Este mecanismo se basa en utilizar la clave pública para identificar la billetera virtual del usuario y la clave privada para autenticar al usuario. Una transacción de Bitcoin consiste en la clave pública del remitente, múltiples claves públicas del receptor y un valor transferido. Luego,

[4] Al día de hoy, todavía no se tiene certeza si fue una persona o un grupo de personas.
[5] El hecho de moverse como en un rebaño o seguir a la masa (herd behaviour en inglés) es un fenómeno con muchos ejemplos en el campo financiero. Consiste en renunciar a la propia voluntad para seguir lo que hacen los demás y con frecuencia se asocia con las burbujas especulativas o ventas masivas provocadas por el pánico.

esta transacción queda escrita en un bloque. El proceso continúa vinculando dicho bloque con un bloque escrito anteriormente. Todos estos bloques contienen la información de todas las transacciones realizadas y son almacenados en algún tipo de memoria, como por ejemplo, un disco rígido. A estas memorias se las llama nodos. Todos los nodos almacenan todos los bloques y los utilizan para verificar la exactitud de cada nueva transacción registrada en la red de Bitcoin, utilizando información de los bloques anteriores. Habiendo realizado este proceso los nodos son recompensados. A este método se lo conoce como minería[6], el cual es uno de los principales conceptos de la tecnología blockchain. De esta manera, cuando las transacciones son confirmadas con éxito, se llega a un consenso entre todos los nodos. Esto es posible gracias a que los nuevos bloques, están vinculados a los bloques anteriores y alineados en una cadena continua. Esta cadena de bloques, es la técnica de contabilidad pública de Bitcoin, llamada blockchain (Yli-Huumo, Ko, Choi, Park, Smolander, 2016).

2.2 Deep Reinforcement Learning

En esta sección se describen todos los conceptos y métodos correspondientes al campo de la ciencia de datos que se utilizan a lo largo del capítulo 3 para desarrollar el bot. La tecnología descrita a continuación, es la responsable de automatizar mayormente la labor diaria del trader humano.

2.2.1 Deep Learning

2.2.1.1 Machine learning y redes neuronales artificiales

El aprendizaje estadístico o machine learning se refiere a una gran cantidad de herramientas que se utilizan para interpretar datos. A estos se los suele conocer como modelos de aprendizaje estadístico o de machine learning. Estos modelos se pueden clasificar en supervisados y no supervisados. Los modelos que se encuentran clasificados como supervisados, se suelen utilizar para realizar predicciones o estimaciones basadas en datos. En el aprendizaje no supervisado, se utilizan datos como entrada pero no se obtienen necesariamente predicciones o estimaciones, sino que son técnicas que brindan como salida información sobre la relación y estructura de los datos de entrada (James, Witten, Hastie, Tibshirani, 2013).

Las redes neuronales artificiales son modelos de machine learning inspirados por las redes de neuronas biológicas que se encuentran en el cerebro. Se los suele enmarcar en un nicho particular de machine learning conocido como deep learning. Este tipo de modelos se suele utilizar para resolver los problemas más complejos del mundo de machine learning. Esto es así, ya que según el teorema de aproximación universal (Hornik, Stinchcombe, White, 1989) los modelos basados en deep learning, o más específicamente las redes *feed-foward*[7], son capaces de aproximar con una precisión arbitraria cualquier función con un número finito de discontinuidades, siempre y cuando las funciones de activación de las neuronas ocultas sean no lineales.

[6] El acto de *minar* consiste en resolver un desafío criptográfico de proof-of-work (PoW) de complejidad variable.
[7] Modelo de deep learning donde la función de activación solo fluye hacia adelante desde la primera capa hasta la última.

2.2.1.2 Perceptrón Multi Capa

Uno de los modelos de deep learning que cumplen con el teorema anterior y son *feed-forward*, es el perceptrón multicapa o MLP. Los modelos de deep learning basados en redes neuronales, como el perceptrón multicapa, están definidos bajo una topología de capas y nodos que intentan replicar el mecanismo que existe entre las neuronas biológicas y sus conexiones entre sí. En particular, el perceptrón multicapa es un modelo compuesto por una capa de entrada, una o más capas conocidas como capas escondidas y una capa final conocida como la capa de salida. Se dice que este es un modelo totalmente conectado, ya que para cada una de todas las capas todas sus neuronas se encuentran conectadas con las neuronas de la capa anterior (Gerón, 2019).

2.2.1.3 Redes neuronales recurrentes y LSTM

La principal diferencia entre los modelos de redes neuronales artificiales *feed-foward*, como el perceptrón multicapa, y las redes neuronales artificiales recurrentes, es que éstas permiten que algunas capas tengan conexiones con capas anteriores. Cuando esto sucede y la salida de una capa en un paso t es la entrada de alguna de las capas t-n anteriores, se puede decir que se obtiene una especie de memoria. La parte de una red que mantiene un estado a lo largo del tiempo se llama celda de memoria. Estas celdas de memoria suelen ser muy útiles para realizar predicciones de series de tiempo (Gerón, 2019). Una de las redes recurrentes más utilizadas para resolver problemas de forecasting o predicciones sobre series de tiempo son las LSTM (Long Short Term Memory). Una celda LSTM puede aprender a reconocer una entrada importante, guardarla en un estado "Long-term", y aplicarla durante un estado "Short-term". De esta manera, no se pierden patrones importantes sucedidos en el pasado, y de ser necesario se los utiliza en las predicciones más cercanas (Hochreiter, Schmidhuber, 1997).

2.2.2 Reinforcement Learning y Deep Reinforcement Learning

2.2.2.1 Reinforcement Learning

Reinforcement Learning es uno de los campos dentro de Machine Learning que se puede considerar de los más antiguos (Sutton, Barto, 1998). En 2013, tomó una alta relevancia cuando investigadores del startup británica DeepMind[8], aplicaron el poder del Deep Learning en el campo del Reinforcement Learning, y crearon un sistema capaz de aprender a jugar cualquier juego de Atari desde cero. No solo eso, sino que en varias ocasiones superaron a los jugadores humanos utilizando únicamente como entrada los píxeles de las pantallas (Mnih, Kavukcuoglu, Silver, Graves, Antonoglou, Wierstra, Riedmiller, 2013).

Los modelos basados en Reinforcement Learning cambian en cierta medida la dinámica o forma en la que son entrenados con respecto a los clásicos modelos de machine learning. Se comienza por crear un agente, el cual realiza observaciones y acciones a través de un ambiente. En respuesta a dichas acciones, recibe recompensas. Estas recompensas, pueden ser positivas y vistas como premios, o negativas y vistas como

[8] DeepMind fue adquirida por Google en 2014. La compañía tiene base en Londres e investigadores en centros establecidos en Canadá, Francia y Estados Unidos. https://deepmind.com/

castigos. Entonces, el agente interactúa con el ambiente y aprende a prueba y error, intentando maximizar las recompensas positivas y minimizando los castigos (Gerón, 2019).

2.2.2.2 A2C

Existen distintos tipos de agentes que utilizan diferentes técnicas para aprender de sus interacciones con los ambientes. Uno de ellos es A2C, que proviene de la familia de agentes Actor-Critic, un paradigma de RL que cuenta con dos agentes, uno "crítico" que evalúa las decisiones tomadas y otro "actor" que controla cómo reacciona el agente. El problema principal de estos modelos, es encontrar una función de valor que evalúe correctamente la política que se está buscando aprender. A2C, Actor Crítico con ventaja, es una variante que utiliza una función de valor diferente a AC. Esta logra controlar mejor la variabilidad que sufre este tipo de modelos (Mnih, Puigdomènech Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu, 2016).

2.2.2.3 PPO

PPO, Proximal Policy Optimization (Schulman, Wolski, Dhariwal, Radford, Klimov, 2017), es un tipo de agente que intenta combinar las ideas de A2C explicado anteriormente y TRPO (Schulman, Levine, Moritz, Jordan, Abbeel, 2015). TRPO (Trust Region Policy Optimization), es un tipo de agente que, en cada iteración en la que se busca optimizar la política que se encuentra aprendiendo, lo hace intentando garantizar siempre una mejora monótona. Entonces, la idea principal de PPO, es que después de cada actualización por paso, la nueva política no se encuentre demasiado lejos de la vieja política. Para esto, PPO utiliza una técnica llamada clipping, que evita actualizaciones demasiado lejanas entre sí. PPO1 es la implementación de este algoritmo sobre la librería OpenAI. PPO2 es la variante técnica de PPO1 que utiliza GPU para entrenar[9].

2.2.2.4 Deep Reinforcement Learning

Deep Reinforcement Learning, es un concepto que hace alusión a la combinación del Reinforcement Learning tradicional con el Deep Learning. Los agentes mencionados en esta misma sección toman decisiones. Esas decisiones están basadas en una predicción hecha por algún tipo de política. Cuando estos agentes hacen uso de algún algoritmo de deep learning como política de predicción, es cuando se está trabajando bajo el dominio de Deep Reinforcement Learning. (Li, 2018).

2.3 Análisis de inversión y estrategias

En esta sección se pretende explicar brevemente en qué consiste el análisis fundamental y el análisis técnico. Además, se explican dos indicadores clásicos en el mundo de las inversiones que se utilizan como *estrategias baseline*; es decir, estrategias base que sirven para realizar comparaciones.

[9] Implementación técnica de PPO por la librería stable baselines: https://stable-baselines.readthedocs.io/en/master/modules/ppo2.html#

2.3.1 Análisis fundamental

Es posible definir al análisis fundamental como una metodología que se utiliza para identificar y definir el valor de un bien basándose en los aspectos fundamentales internos y externos del mismo, como puede ser una compañía. Esta metodología nace de la idea de buscar, a partir de pasos lógicos, llegar al valor intrínseco de un bien, estudiando la composición del mismo y su interacción con el entorno. En base al resultado obtenido por este análisis y al valor formal del bien, es que luego se puede concluir si es conveniente invertir o no en el mismo (Graham, Dood, 2009). Con respeto a las criptomonedas en general, los aspectos fundamentales externos son publicados en foros y redes sociales. Es aquí, donde el trader humano realiza su investigación basada en su experiencia de forma manual (leyendo) a la hora de querer invertir en criptomonedas.

2.3.2 Análisis técnico

A diferencia del análisis fundamental, el análisis técnico utiliza metodologías, gráficos y técnicas estadísticas sin tener en cuenta específicamente cual es el bien que se está evaluando (Murphy, 2007). Para desarrollar este tipo de análisis, primero se suele buscar identificar patrones y establecer hipótesis futuras sobre el precio de un activo. Luego, realizando análisis cuantitativos mediante técnicas estadísticas (ej: medias móviles) basadas en las variables comunes a cualquier activo, como es el precio, volumen, máximos y mínimos, es que se intenta arribar a indicadores, que ayuden a aproximar el valor futuro o comportamiento futuro del precio del bien en cuestión. Este tipo de análisis, suele ser más sencillo y automático que el fundamental, ya que es la computadora la que procesa la información en este caso. De todas formas, no deja de exigir el expertise del trader para interpretar las señales de los indicadores y tomar una decisión.

2.3.2.1 RSI divergence

RSI, Relative Strenght Index, es un indicador de momentum que se suele utilizar en el análisis técnico para medir la magnitud de los cambios de precio recientes, con el objetivo de evaluar las condiciones de sobrecompra o sobreventa del precio de un activo. El resultado de este indicador oscila entre 0 y 100. Cuando la tendencia del precio es hacia abajo, la señal del indicador suele tener valores más cercanos al cero. Lo contrario sucede con valores más cercanos al 100. Se le llama divergente (Cory, 2019) a un indicador técnico, cuando el precio de un bien se mueve de forma opuesta a la dirección del indicador en cuestión. Las situaciones de RSI divergence, se suelen dar cuando el activo que se está analizando no contiene tendencias estables a largo plazo (Brown, 1999).

2.3.2.2 Simple Moving Average (SMA) Crossover

SMA o Simple Moving Average, es un indicador basado en dos medias móviles, calculadas con periodos de tiempo distintos sobre el precio de un activo. Siempre, una de ellas es calculada con un valor a largo plazo y la otra con un plazo más corto. El indicador SMA Crossover, indica el momento en el que estas medias móviles se cruzan, lo cual estarían señalando un cambio de tendencia en el precio del activo (Attia, 2019).

2.4 Ratios de riesgo de retorno

En esta sección se presentan 3 ratios que comúnmente se utilizan para hacer evaluación de riesgo sobre una inversión. Estos ratios son utilizados en este trabajo cuando se intenta mejorar la forma en la que el agente es recompensado por sus acciones.

2.4.1 Sharpe ratio

Sharpe ratio, originalmente creado por William F. Sharpe en 1966 y nombrado como "reward-to-variability" ratio, que luego se conoció con el nombre que lleva actualmente en referencia al autor del mismo, es un indicador que mide la relación de la rentabilidad sobre la volatilidad histórica de una inversión. Cuanto mayor es el resultado, mejor es la rentabilidad de la inversión en relación al riesgo asumido al realizarla (Sharpe, 1966).

2.4.2 Sortino ratio

Sortino ratio, es un indicador derivado del Sharpe ratio. En términos matemáticos cambia en que la volatilidad histórica de la inversión o el desvío estándar de la misma, es calculado únicamente teniendo en cuenta las variaciones hacia abajo. De esta manera, se penaliza solamente en base a la variación de los retornos hacia abajo, sin tener en cuenta la volatilidad hacia arriba del valor histórico del activo (Sortino, Price, 1994).

2.4.3 Omega ratio

Omega ratio, también es un indicador que mide la performance de una inversión en base al riesgo de retorno de la misma. Esta es una alternativa totalmente distinta a los ratios mencionados anteriormente. Este ratio, se encuentra definido como la probabilidad de ganancias sobre la probabilidad de pérdidas. Dicho de otra manera: cuántas unidades de la divisa en cuestión fueron ganadas en las subidas del precio del activo, sobre la cantidad de unidades arriesgadas en las bajadas del mismo (Benhamou, Guez, Paris, 2019).

2.5 Análisis de sentimientos

En esta sección se explica qué es el análisis de sentimientos, el modelo utilizado para realizar este análisis y la fuente de datos proveedora para la entrada para este modelo. En esta tesis se utiliza el análisis de sentimientos para brindar un soporte mayormente automatizado al proceso de análisis fundamental dentro del método propuesto.

2.5.1 Análisis de sentimientos

El análisis de sentimientos es un tipo de análisis que busca identificar la polaridad del sentimiento predominante en un texto. Para ello, existen principalmente dos enfoques: el primero basado en recursos léxicos y procesamiento del lenguaje natural, y el segundo basado en algoritmos de machine learning (Denecke, 2008). Más allá de la técnica, modelo o procesamiento que se realice sobre el corpus bajo análisis, la finalidad del análisis generalmente es la misma: poder identificar si un texto expresa un sentimiento que se inclina a lo que se defina o califique como positivo, negativo o neutro.

2.5.2 VADER

VADER es un software desarrollado para realizar análisis de sentimientos especialmente sobre redes sociales. El mismo, fue probado en una investigación de los mismos autores sobre la plataforma Twitter (Hutto, Gilbert, 2014). Este sistema funciona al utilizar una combinación de léxicos predeterminados y de reglas, para detectar la polaridad de los sentimientos en un texto. En dicha investigación se lo compara con otras 11 herramientas de análisis de sentimientos, y se concluye que es uno de los mejores entre todas ellas, ya que era capaz de generalizar mejor que el resto y contaba con mayor precisión.

El software[10] funciona recibiendo un texto como entrada y presentando 4 valores resultantes como salida. Los valores de salida corresponden a la polaridad positiva, negativa y neutra del sentimiento, y un score compuesto por el promedio de ambas tres.

2.5.3 Twitter

Twitter hoy en día es una de las redes sociales más grandes del mundo, contando con más de 353 millones de usuarios. Es una de las plataformas de microblogging más populares que existe en la actualidad. En la misma se comparten tweets, que suelen ser fragmentos cortos de texto, donde se pueden añadir enlaces, imágenes, videos, etc. cuyo límite de extensión son 280 caracteres. A partir de estos tweets es que se generan hilos de conversación y discusión sobre temas variados. (Webempresa, 2018). Esta plataforma brinda un servicio de *API Streaming*[11], del cual se pueden descargar tweets filtrados por ciertas características y limitaciones en tiempo real y de manera gratuita. La razón de elegir Twitter por sobre otras redes sociales, como la plataforma que brinda datos para cubrir el déficit de análisis fundamental queda explicado en la sección 3.1.2.

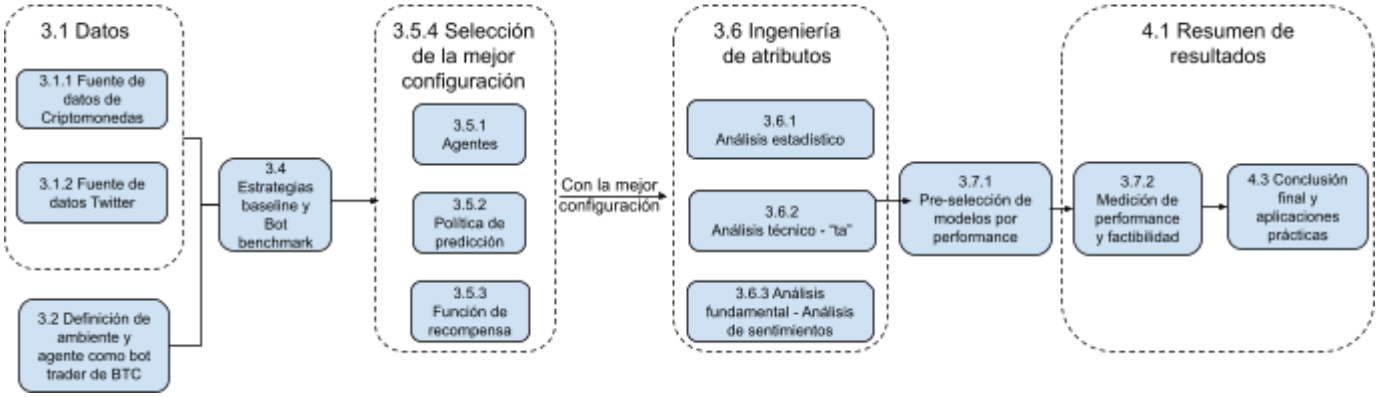
[10] Repositorio del software OpenSource de VADER: https://github.com/cjhutto/vaderSentiment
[11] https://developer.twitter.com/en/docs/tutorials/stream-tweets-in-real-time

Capítulo 3: Metodología

En este capítulo se presentan los datos con los que se realizó la investigación y se expone todo el procedimiento de evaluación, experimentación realizada y alternativas que se fueron utilizando en búsqueda del objetivo final: obtener un bot basado en Deep Reinforcement Learning que pueda operar con Bitcoin, realizando análisis técnico y fundamental para generar la mayor cantidad de ganancias posibles en un determinado lapso de tiempo.

A fin de facilitar la comprensión del proceso adoptado para el desarrollo de la metodología de trabajo, se comparte un mapa conceptual para visualizar cada parte del procedimiento, indicando el papel de cada título y sección en un diagrama de bloques. (Figura 3.1).

Figura 3.1: Mapa conceptual del proceso de la metodología desarrollada



3.1 Datos

Para realizar el análisis técnico y fundamental, se requiere de distintos tipos de datos. Esto lleva a utilizar dos tipos de fuente de datos distintas para el propósito de este trabajo.

3.1.1 Fuente de datos de criptomonedas

Para cubrir el análisis técnico se utilizaron variables numéricas que explican hora a hora las características que definen el estado o valor de un activo: Apertura (Open), Cierre (Close), Máximo (High), Mínimo (Low), Volumen BTC, Volumen USD. En este caso en particular, se trabajó con un conjunto de datos proveniente de https://www.cryptodatadownload.com/ , sitio creado con el fin de proporcionar datos históricos sobre criptomonedas de forma gratuita, con el objetivo de ser usada para investigación, además de compartir análisis y reportes sobre criptomonedas y riesgo de mercado. Del mismo, se extrajeron datos a nivel horario con información técnica sobre Bitcoin a lo largo de un periodo determinado (1/7/2017 al 17/5/2019).

3.1.2 Fuente de datos de Twitter

Por otra parte, el análisis fundamental, se suele basar en información intrínseca del activo que se está analizando. Como el Bitcoin no está respaldado por una empresa o entidad financiera, la única información intrínseca sobre el mismo proviene del comentario popular.

Hoy, en el año 2021 y ya hace algunos años más, las redes sociales han contribuido a centralizar ciertas interacciones entre las personas de todo el mundo. Algunas fueron creadas en búsqueda de centralizar un propósito claro, como Linkedin: red social de profesionales y búsqueda laboral. Otras fueron creadas sin un propósito claro o bien definido como lo es Facebook, la red social más grande hasta el día de la fecha, que hoy cuenta con más de 2.740 millones de usuarios.

Para el propósito de este trabajo y en búsqueda de realizar un análisis fundamental de manera diferente a como se suele realizar (teniendo en cuenta que para el mismo es necesario analizar qué comentan las personas sobre un determinado tema), se considera que la plataforma más grande que se ajusta a dicha necesidad es Twitter. Esta red social, hoy cuenta con 353 millones de usuarios, en donde su propósito es el microblogging e intercambio de opiniones sobre variados tópicos. Si bien la fuente original del conjunto de datos con el que se trabajó es Twitter, el mismo ya se encontraba procesado, agrupado y clasificado a nivel de análisis de sentimientos, con una granularidad horaria, cubriendo el periodo de tiempo desde 1/8/2017 al 21/1/2019. El conjunto de datos fue armado por Jaime Badiola Ramos (Badiola Ramos, 2019) y utilizado en su trabajo: *¿Podemos comerciar Bitcoin usando análisis de sentimiento sobre Twitter?.* Todo lo anteriormente mencionado en este párrafo se desarrollará y explicará en detalle cuando sea utilizado en la sección 3.6.3.

3.1.3 Análisis exploratorio de datos

En esta sección se realizará una descripción y explicación más profunda sobre las dos fuentes de datos a utilizar en este trabajo.

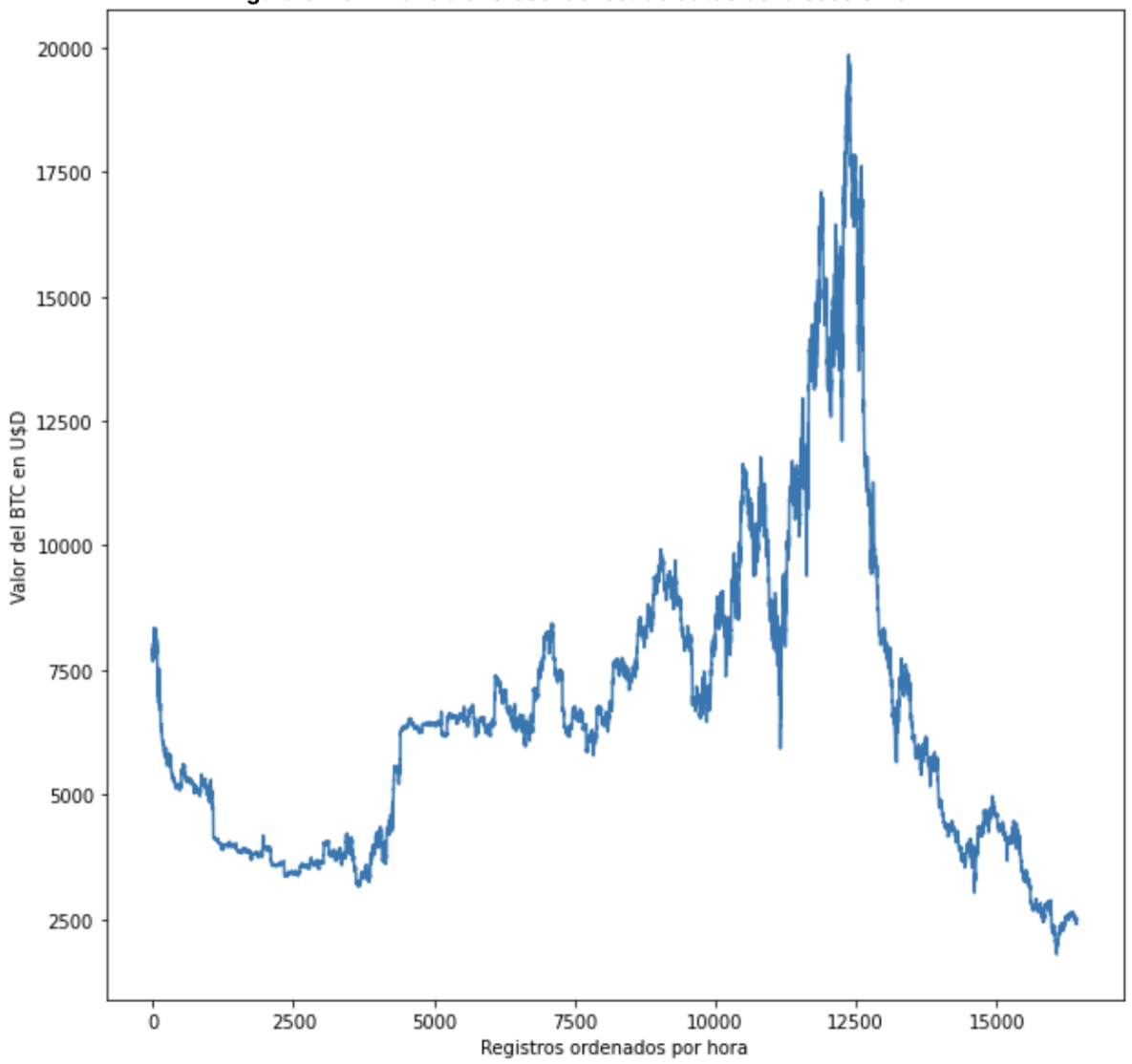
Comenzando con el conjunto de datos de la sección 3.1.1, del cual se extraen las variables técnicas, se cuenta con registros por hora con 7 columnas, de las cuales 2 son de tipo categóricas y 6 son de tipo numéricas. Las variables de tipo categórico son: “Symbol”, que indica el tipo de comparación entre divisas que se está realizando y que cuenta con un único valor, “BTCUSD”, para todos sus registros (razón por la cual no se tuvo en cuenta), y “Date” que indica la fecha y hora de la observación de cada registro. Analizando la columna “Date”, se encuentra que cuenta con datos desde el 1 de julio del 2017 11a.m. hasta el 17 de mayo del 2019 1a.m. Sumando un total de 16.431 registros. Las variables numéricas cuentan con las características estadísticas que se pueden ver en la Figura 3.1.3.1.

Figura 3.1.3.1: Descripción estadística de las variables numéricas

	Open	High	Low	Close	Volume BTC	Volume USD
count	16431.000000	16431.000000	16431.000000	16431.000000	16431.000000	1.643100e+04
mean	6609.643980	6649.279040	6565.967278	6609.974787	569.713472	4.241967e+06
std	3144.197242	3179.350555	3103.714099	3144.047541	676.304487	6.953876e+06
min	1785.010000	1858.270000	1758.200000	1785.010000	0.000000	0.000000e+00
25%	4010.015000	4024.990000	3994.370000	4010.085000	203.425000	1.026741e+06
50%	6360.520000	6379.990000	6343.690000	6360.530000	363.140000	1.964696e+06
75%	7960.490000	8020.000000	7890.000000	7960.490000	671.060000	4.643561e+06
max	19847.110000	19891.990000	19725.980000	19847.110000	10464.320000	1.438437e+08

Observando los valores máximos y mínimos, se puede interpretar que probablemente el periodo tomado en cuenta para el análisis, tiene una alta variación a lo largo del tiempo. Se comprueba lo anterior observando el gráfico de la Figura 3.1.3.2:

Figura 3.1.3.2: Variable 'Close' del set de datos de la sección 3.1.1



Es de público conocimiento que el Bitcoin es una de las criptomonedas más volátiles, y que en 2018, alcanzó máximos históricos (para aquel entonces) llamando fuertemente la atención a los sectores financieros de todo el mundo. Estas fechas de análisis fueron especialmente elegidas, para poder observar el comportamiento general del agente operando en un contexto extremadamente volátil y cambiante.

Continuando con el análisis de datos exploratorio de la segunda fuente de datos presentada en la sección 3.1.2, de la cual se extrajo las variables fundamentales e intrínsecas de Twitter, se cuenta con un total de 16 columnas. Una buena parte de estas, se encuentran repetidas por el conjunto de datos anterior (3.1.1). Esto es a razón de que el trabajo original del cual proviene este conjunto de datos tenía un enfoque similar, en el que se buscaba relacionar el precio del Bitcoin con variables de análisis de sentimientos de tweets que hablen del tema. Para este trabajo se utilizó únicamente las columnas listadas en la siguiente tabla (Figura 3.1.3.3):

Figura 3.1.3.3: Tabla descriptiva de las variables de análisis de sentimientos

Columna	Descripción
Count_Bots	Volumen de mensajes clasificados como bots
Count_News	Volumen de mensajes que contienen al menos un link
Sent_Positives	Promedio de tweets clasificados por sentimientos únicamente positivos.
Sent_Negatives	Promedio de tweets clasificados por sentimientos únicamente negativos.
Count_Neutrals	Volumen total de tweets clasificados como sentimiento neutral.
Count_Positives	Volumen total de tweets clasificados por sentimientos únicamente positivos.
Count_Negatives	Volumen total de tweets clasificados por sentimientos únicamente negativos.
Total volume of tweets	Volumen total de tweets
Compound_Score	Promedio de todos los scores de sentimientos clasificados

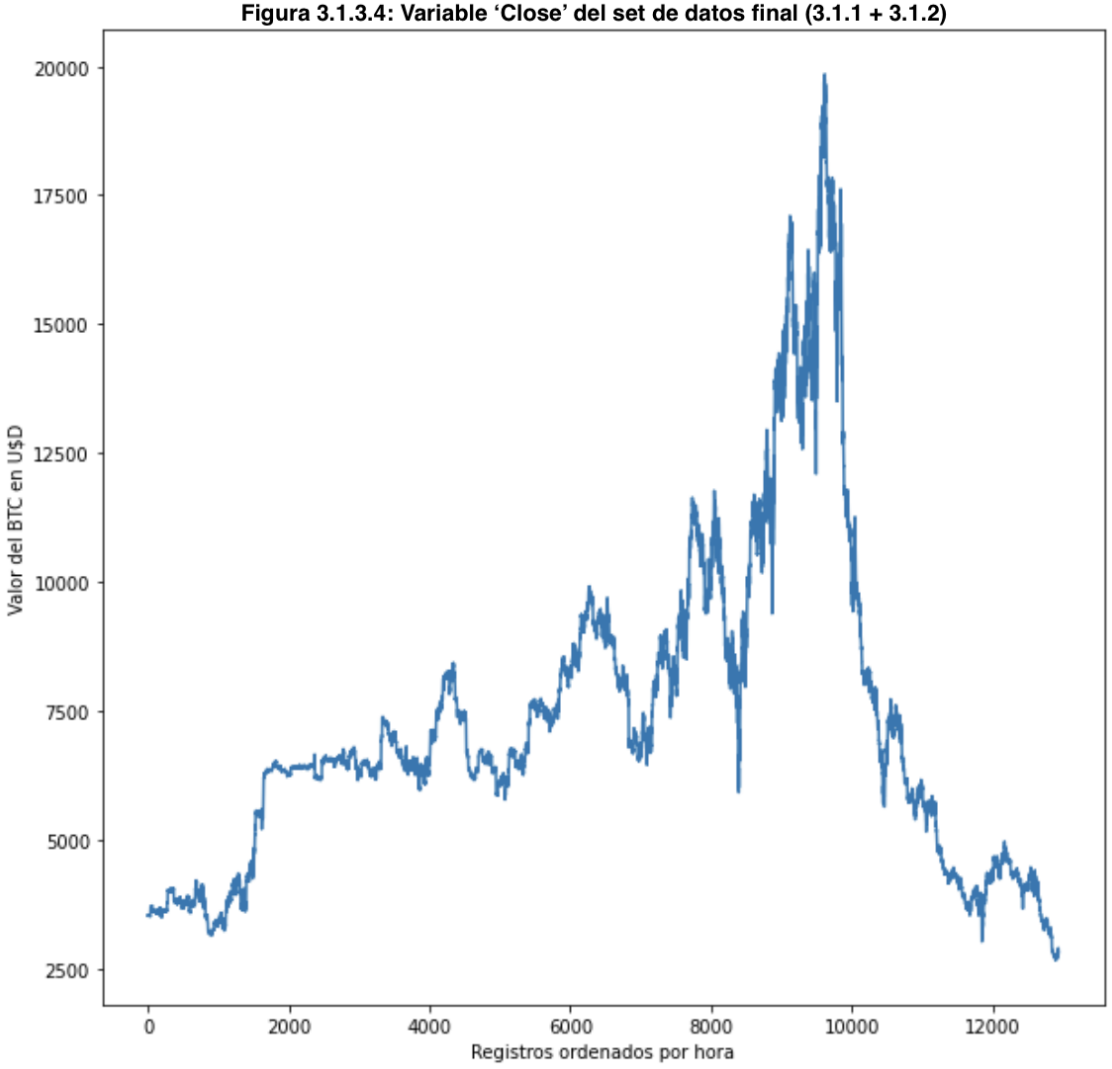
Según las notas del autor del conjunto de datos sobre cómo fue creado, él describe que utilizó la librería “GetOldtweets” para descargar el contenido de Twitter, y que consiguió descargar 17.7 millones de tweets en idioma inglés, bajo el criterio de que estos contengan la palabra “Bitcoin”. También cuenta que, en esa descarga, se encontró con algunos valores nulos en determinadas horas, lo cual consideró no relevante por tratarse de muy pocos casos. Luego, utilizó la librería VaderSentiment para clasificar los textos de los tweets por sentimientos y que agregó 30 expresiones y palabras al diccionario del mismo. Agrega que después de analizar los resultados, encontró mensajes que no hablaban del bitcoin en sí, o que mencionan otras criptomonedas como “Bitcoin Gold”, o usuarios promocionando su

propio contenido. Otros mensajes fueron identificados y clasificados como bots, lo cual era indicado por los mismos *hashtags* (#) que componían los tweets. Finaliza indicando que todos estos mensajes fueron filtrados o clasificados en base a lo que eran, y que el sentimiento resultante de esos mensajes no fue considerado dentro del conjunto de datos, pero que sí se mantuvo la información del volumen de tweets en la columna " Count_bots".

En este caso, a partir de la descripción presentada se puede observar que cierta cantidad de registros, contienen datos nulos en las columnas relacionadas al análisis de sentimientos. La manera de tratar y procesar estos valores nulos fue utilizando el método "bfill" de la función "fillna()", el cual completa los valores nulos con el valor inmediatamente anterior. Para este conjunto de datos (3.1.2) se cuenta con un total de 12.936, que equivalen al total de horas entre el 1 de agosto de 2017 12a.m. hasta el 21 de enero de 2019 11p.m. Esto quiere decir que se cuenta con aproximadamente 539 días de datos.

Utilizando ambos conjuntos de datos (3.1.1 y 3.1.2), se elaboró un "conjunto de datos final", el cual respeta las fechas del conjunto de datos de análisis de sentimientos (3.1.2), por contener menos registros y ser un subintervalo del primero (3.1.1). Con este conjunto de datos final se explora y trabaja a lo largo de todo el proyecto (Figura 3.1.3.4).

Figura 3.1.3.4: Variable 'Close' del set de datos final (3.1.1 + 3.1.2)



3.2 Definición de ambiente y agente como bot trader de BTC

Para el problema en cuestión y la solución que se desea dar basada en Reinforcement Learning, es necesario definir el ambiente o environment en el que operará el bot, la definición del agente que representará al bot y las distintas posibilidades o acciones que el mismo puede tomar.

3.2.1 Ambiente

En este caso, el ambiente debe contar con todas las funcionalidades necesarias para que el agente pueda recorrer y aprender del mismo. Para poder realizar esto, se crea un ambiente de trading de criptomonedas y se entrena un agente para que se convierta en un Bot rentable de Bitcoin. A nivel técnico, en esta tesis se utiliza la librería *gym*[12] en *python*[13] para establecer el ambiente.

Lo primero que se debe entender es: cómo un trader humano percibe este tipo de ambientes y qué decisiones puede tomar al respecto. El trader humano probablemente observe gráficos y otro tipo de indicadores, más todo su conocimiento y experiencia sobre el precio de un activo, para luego poder tomar una decisión informada. En base a lo anterior, se puede decir que principalmente el ambiente se encuentra definido por un “espacio de observación” y un “espacio de acción”. El “espacio de observación” se lo puede definir como el contenedor de todas las variables y datos que se quiere que el agente tome en cuenta a la hora de evaluar y analizar el precio de un activo. Más específicamente, esto sería las variables que componen el conjunto de datos que se utiliza como entrada, sumando ciertas variables que se llamarán “variables de estado”. Las “variables de estado”, son aquellas que informen en todo momento el balance de la cuenta en USD, la cantidad de BTC que se tiene al momento, la recompensa neta histórica adquirida por el agente, el “paso en el tiempo” actual, el histórico de los estados de balance y el histórico de las transacciones realizadas. De esta manera, a cada “paso en el tiempo” que realice el agente, este podrá medir y aprender a modo prueba y error de su propio accionar. Al mismo tiempo las “variables de estado” son actualizadas en cada paso.

Una vez que el trader humano es consciente del ambiente en el que se encuentra, es decir, realizó los análisis que consideró necesarios, debe tomar una decisión. Estas decisiones son las que componen el “espacio de acción”. En este caso el “espacio de acción” está compuesto por dos partes. La primera parte es discreta y trata tres posibilidades claras: Comprar, Vender o Mantener. La segunda parte de este espacio es más compleja, ya que es continua y está compuesta por el porcentaje de 0% a 100% de cuánto comprar o vender, en base al balance y precio que se tiene en un momento determinado en el tiempo. Esta última fue discretizada en 10 opciones (1/10… 10/10).

Por último y no menor, es necesario definir el “Reward” o “Recompensa” que recibe el agente, para entender y aprender si las acciones que está tomando, lo están beneficiando o por el contrario lo están castigando. Este mecanismo de recompensa y castigo es evaluado en cada paso, hora a hora, en base a las decisiones tomadas por el agente. El

[12] https://gym.openai.com/
[13] Lenguaje de programación

mismo en un principio es definido de manera muy sencilla como se puede ver en la figura 3.2.1.1. Esta función de recompensa o reward se la llamó *profit*.

Figura 3.2.1.1: Función de recompensa obtenida en cada paso

$$Recompensa = Balance\ neto\ actual - Balance\ neto\ paso\ anterior$$

En este punto es importante diferenciar el significado de las palabras *recompensa* y *ganancia* que se utilizan a lo largo de todo el trabajo. Cuando se hable de recompensa, se estará haciendo referencia al premio o castigo que el bot de reinforcement learning estará recibiendo. Cuando se hable de ganancia, se estará haciendo referencia al dinero obtenido por el bot de reinforcement learning al operar. Si bien, se profundiza sobre este tema en la sección 3.3.3, vale aclarar que la performance será medida en términos de recompensa y la factibilidad será medida en términos de ganancia.

3.2.2 Bot y Agente

El Bot es definido como un conjunto de funciones, que se irán ejecutando en base a las predicciones y decisiones del agente. Para ello, se cuenta con funciones que determinan la ejecución de las acciones que el agente decida tomar. Al instanciar una clase del Bot, se inicializan las variables de estado de la siguiente manera:

- Balance = 10.000
- Recompensa neta = 0
- BTC retenidos = 0
- Paso actual = 0
- Transacciones = Vector vacío que contendrá cada transacción.
- Histórico de transacciones = Vector inicializado con el valor de las variables anteriores.

Se puede observar que se comienza con un balance de 10.000 USD. Esto se definió de esta manera para respetar el benchmark propuesto, en el cual las estrategias baseline y el Bot básico que se utilizan como benchmark comienzan con este valor de balance. Además de las variables de estado, se incluyen ciertas variables para añadir realismo al ambiente, como por ejemplo el costo de la comisión que tiene operar. Estas se mantienen constantes y no son significativamente influyentes para el objetivo al que se pretende arribar. Una vez inicializado el agente, se procede a pasarle como parámetro el ambiente definido en la sección anterior. El ambiente, además de contener el “espacio de acción” y el “espacio de observación”, contiene una porción del conjunto de datos destinado a ser utilizado para entrenar el agente. Cada registro del mismo, es un paso de entrenamiento para el agente. Se explica con más profundidad sobre la partición y preparación de los datos en la sección 3.3.

En cada paso se selecciona una observación en orden determinada por la función *“step()”*, se actualiza la variable de estado *current_step,* se toma la observación y se le realizan todas las transformaciones necesarias dentro de una función *“_next_observation().* Con dicha información, el modelo realiza una predicción, toma una decisión y ejecuta una acción. Seguidamente se calcula el retorno o recompensa (*premio* o *castigo*) obtenida en

base a una regla predefinida, la cual es computada por una función llamada *"reward".* Antes de seguir al próximo paso, se actualizan las variables de estado. Una vez finalizado el entrenamiento al transitar por todos los registros, se desactiva la opción de seguir aprendiendo en cada paso, se calcula la recompensa total que se obtendría sobre un conjunto de datos de validación, y se reinicia el ambiente con una función *"reset()",* que deja a las variables de estado en los valores expuestos al principio de esta sección.

Se repite este proceso diez veces, consiguiendo como resultado diez agentes distintos, los cuales van aprendiendo de sus predecesores y corrigiendo la toma de decisiones informada. Se definió que se trabaja con diez iteraciones porque, en pruebas de concepto, se observaron cambios suficientes entre esta cantidad para el desarrollo y las conclusiones que se proponen en este proyecto.

3.2.3 Analogía del método para la solución del problema

Si se presta atención cuidadosamente a los detalles técnicos de como fue definido el ambiente, el agente y la extracción de datos, se puede decir que se está describiendo el proceso clásico y manual que realiza el trader humano a la hora de realizar *trading*. Primero, busca y lee información para realizar análisis fundamental, que en este caso queda cubierto por la descarga de tweets que contienen ciertos identificadores. Luego, con la información del estado del activo y el análisis técnico, el trader busca encuadrar y comprender el contexto y evolución del valor del mismo, lo cual en este trabajo se define como ambiente. Por último, en base a los resultados del análisis técnico y fundamental, más su experiencia y conocimiento en el tema, el trader interpreta dicha información y decide si comprar, mantener o vender y en qué cantidad de ser el caso. La interpretación, comprensión y toma de la decisión informada en este caso, quedan bajo la responsabilidad del bot entrenado.

Repasando las ventajas que aporta este método, se encuentra la extracción de datos técnicos y fundamentales de manera estandarizada, automática y en tiempo real, sin necesidad de la intervención del trader humano. Este proceso conlleva tecnicismos que escapan a este trabajo, pero que se consideran poco significativos en términos de desarrollo y complejidad. La clave se encuentra en la interpretación, comprensión y toma de la decisión informada, que se consigue realizar de una forma mayormente automatizada y basada en criterios matemáticos exactos, evitando el *sesgo psicológico del trader*[14] y el error humano. De esta manera, el método, puede ayudar a acelerar el trabajo del trader humano, haciéndolo más eficiente en términos de la cantidad y complejidad de decisiones que pueden tomarse en un corto plazo de tiempo.

Como se expuso en el párrafo anterior, la *piedra angular*[15] de este método, se encuentra en obtener un bot capaz de tomar todos estos datos, interpretarlos y efectuar la mejor decisión informada posible. Por esta razón, el resto de este capítulo se centra en la experimentación y desarrollo de la esencia de esta herramienta, y en cómo medir el éxito de la misma.

---

[14] Psicología del trader: La psicología del trading son todos aquellos aspectos mentales que afectan a la inversión en los mercados financieros. https://economipedia.com/definiciones/psicologia-del-trading.html

[15] Término usado comúnmente para referirse a cosas o personas que fueron muy importantes para que algún hecho o situación pudiese ser llevada a cabo.

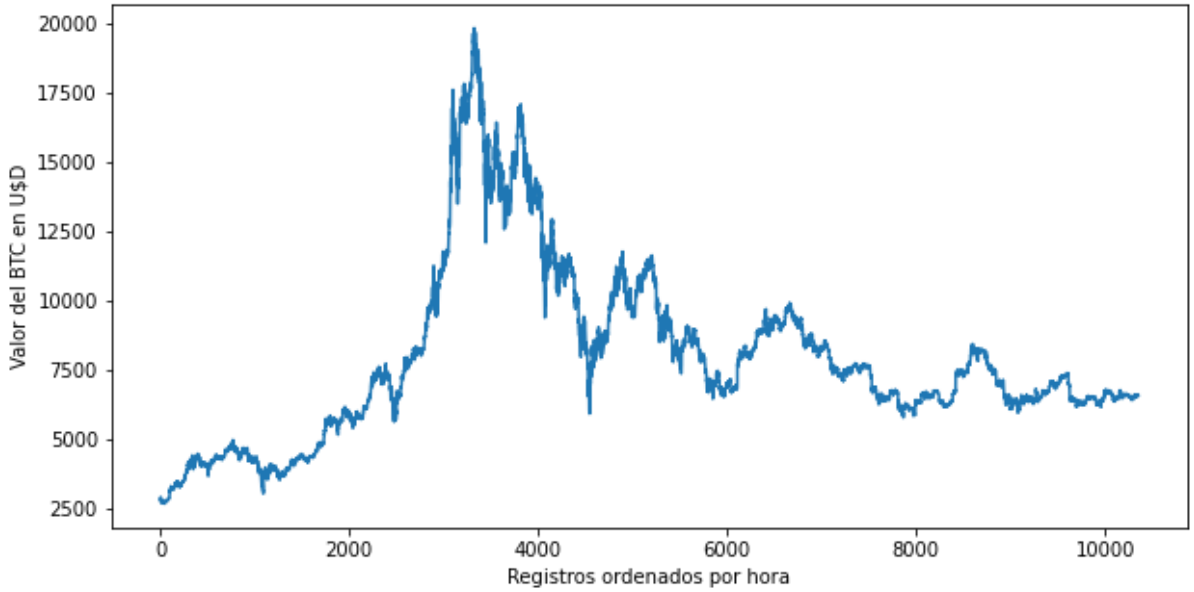
3.3 Estrategia de entrenamiento, validación y testeo de modelos

En esta sección se expone cómo es el proceso de evaluación de los modelos, de qué manera se utilizan los datos disponibles para llevar a cabo este proceso, y se explicarán ciertos supuestos y definiciones necesarias para poder realizar la evaluación de cada uno, a fin de luego arribar a las conclusiones finales.

3.3.1 Preparación de datos para el entrenamiento y evaluación

Partiendo del conjunto de datos que fue descrito en la sección 3.1, este fue separado en tres partes. La primera parte se utilizó para conseguir un conjunto de datos de entrenamiento, con la finalidad de entrenar los modelos. Este está conformado por un total de 10.350 registros, contemplando un periodo desde el 8 de agosto del 2017 hasta el 6 de octubre de 2018 (80% del conjunto de datos original).

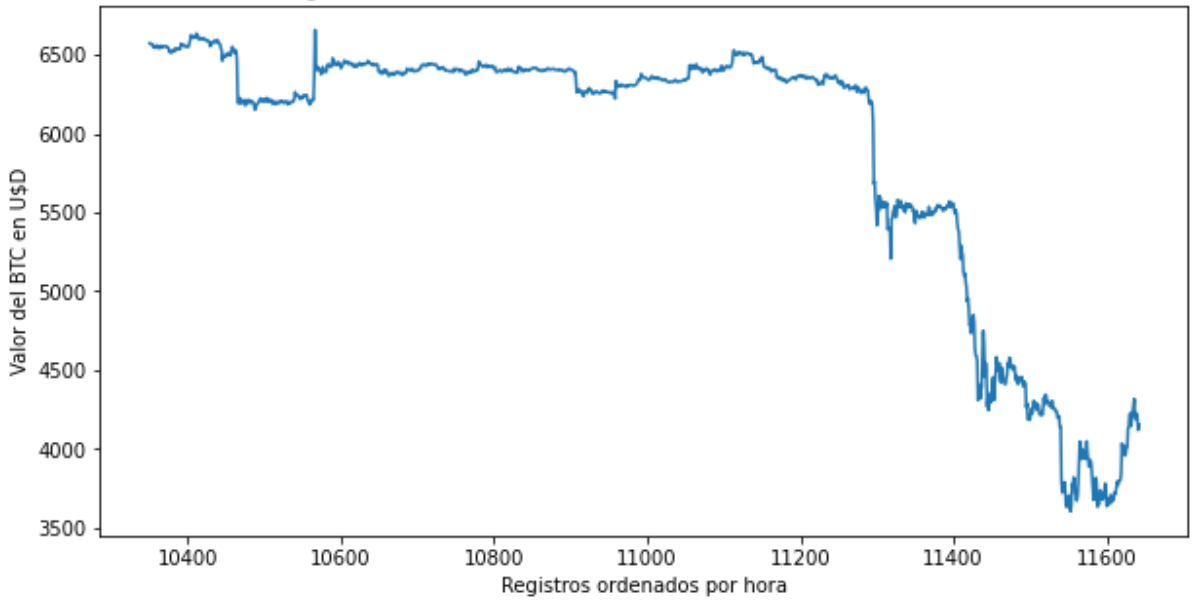
Figura 3.3.1.1: Variable 'Close' del set de datos de entrenamiento



El segundo es un conjunto de datos que se utilizó como validación de los modelos entrenados, que se suele utilizar comúnmente para validar el *trade-off*[16] entre varianza y desvío. Cuenta con un total de 1.293 registros (10% del conjunto de datos original). Cubriendo un periodo desde el 7 de octubre de 2018 hasta el 29 de de noviembre de 2018.

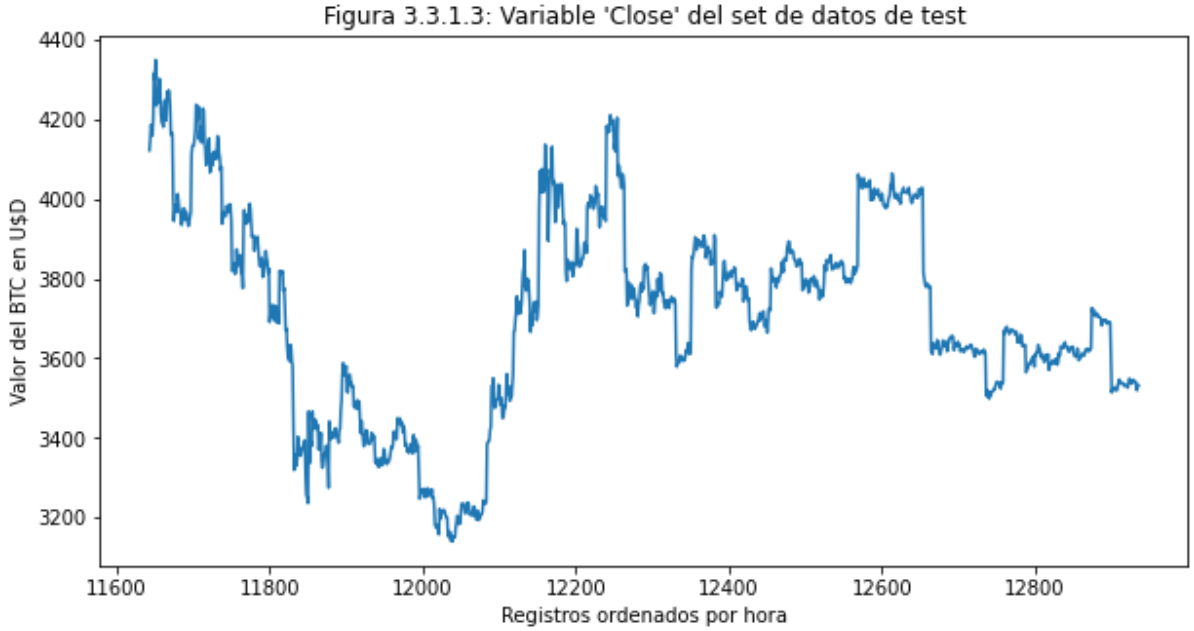
[16] Es una decisión situacional que implica disminuir o perder una calidad, cantidad o propiedad de un conjunto o diseño a cambio de ganancias en otros aspectos. En términos simples, una compensación es donde una cosa aumenta y otra debe disminuir.

Figura 3.3.1.2: Variable 'Close' del set de datos de validación



Por último, se tiene un conjunto de datos de test, nunca antes visto por ningún modelo, que se utilizó para evaluar que los hallazgos de los experimentos, hipótesis y conclusiones sean correctos y genéricos para otros periodos de tiempo. También cuenta con un total de 1.293 registros (10% del conjunto de datos original). Cubriendo un periodo desde el 30 de noviembre de 2018 hasta el 21 de enero de 2019.

Figura 3.3.1.3: Variable 'Close' del set de datos de test



3.3.2 Proceso de evaluación de modelos

Contando con los tres conjuntos de datos mencionados anteriormente, se procede a entrenar los modelos haciendo uso de los datos de entrenamiento. Luego de finalizar cada iteración al entrenar un modelo en particular, se suspende el proceso de aprendizaje, se corre el bot de la iteración correspondiente sobre los datos de validación, y se calcula el

resultado de recompensa obtenido. Dicho resultado en conjunto nuevamente con los datos de entrenamiento, son la entrada hacia el entrenamiento de la siguiente iteración. A lo largo del trabajo se realizan observaciones sobre la performance (término que será definido en la siguiente sección), de los resultados obtenidos de cada iteración sobre los datos de validación en relación con ciertos supuestos que serán definidos en la próxima sección. Estos supuestos indican qué tipo de performance se espera encontrar y alcanzar en los modelos entrenados. En base a los modelos entrenados y los resultados obtenidos en los datos de validación sobre la performance de los mismos, se realiza una preselección de modelos. Luego, se procede a medir y comparar la performance de la mejor iteración de los modelos preseleccionados, tanto en datos de validación como en datos de test nunca antes vistos por ningún modelo. Por último, se mide la factibilidad (término que será definido en la siguiente sección) de los modelos preseleccionados utilizando los datos de test. A partir de los resultados de factibilidad obtenidos, es que se arriba a las conclusiones y resultados presentados al final de este trabajo.

3.3.3 Evaluación de performance y factibilidad

La tecnología actualmente utilizada para Reinforcement Learning pertenece a un campo de estudio muy reciente y complejo, por lo que actualmente existen diferentes técnicas y opiniones encontradas de cómo medir si un Bot o robot es bueno o no. A esto se le suma la cantidad de aplicaciones y enfoques que dificultan aún más la elección de una métrica de evaluación, ya que la misma debería depender del resultado en el contexto real para el problema que se pretenda resolver. Entonces, es posible decir que una manera de juzgar un modelo de reinforcement learning, podría ser observar que tan bien aprende sobre la política que se le está enseñando o sobre cuánta recompensa en total consigue cuando interactúa con el ambiente. Lo que es aún más importante, es medir como luego el agente se termina desenvolviendo en el contexto para el que fue preparado, o como se suele decir, puesto en producción[17] (Poole, Mackworth, 2017). Por lo tanto, se debe medir si el Bot está aprendiendo bien de la forma en la que se lo está entrenando: performance basado en ambiente, agente y política de recompensa. Y por otro lado, se debe medir el éxito obtenido en el contexto en el que se desenvuelve y el resultado de su manera de accionar: factibilidad obtenida en producción.

Para este trabajo, un ejemplo deseado podría ser obtener un Bot que técnicamente haya aprendido bien la estrategia de inversión que se le enseñó. Es decir, que performe bien y que luego sea factible cuando se ponga en producción. Otro ejemplo, podría ser un Bot que aprenda bien la estrategia que se le pretendió enseñar. Es decir que performe bien, pero que luego no consiga ser factible o mantenerse rentable cuando está en producción. Esto puede suceder porque dicha estrategia no es suficiente para conseguir el objetivo de mantener y/o incrementar la rentabilidad. Un último ejemplo podría ser arribar a un modelo que directamente no performe bien. Es decir, que no haya aprendido bien lo que se le pretende enseñar y que por lo tanto se espere que no sea factible en producción. Entonces, es importante antes de seguir adelante con el proyecto, entender y definir cómo se midió la performance de los Bots que se entrenaron, cómo se midió la factibilidad de los mismos a la hora de operar, y remarcar la diferencia entre una buena performance y un modelo factible. Esto se considera necesario por lo que expresa Andrej Karpathy, actual director del área de

[17] Puesta en producción: expresión que refiere a una solución de software operando en la realidad y del cual se espera que funcione como se esperaba. Es decir, que sea factible.

inteligencia artificial de la compañía Tesla y ex investigador de OpenAI, en un comentario[18] referido a un paper particular: "*[Aprendizaje supervisado] quiere funcionar. [...] RL debe ser obligado a funcionar.*" Por esta razón, a continuación se realizan ciertos supuestos sobre cómo se medirán estos Bots tanto en performance como en factibilidad.

En este trabajo se mide la performance de un modelo, en base a la recompensa neta acumulada obtenida por la función de recompensa (Géron, 2019) sobre los datos de validación. Luego, según la pre-selección de modelos mencionada en la sección 3.3.2, se mide la performance de los mismos sobre datos de test. Como se adelantó, se realizaron ciertos supuestos sobre cómo debería resultar la evolución de la recompensa acumulada. Se asume una buena *performance* cuando esta recompensa neta acumulada sobre datos de validación y datos de test es:

- **Positiva creciente:** Recompensa siempre positiva creciente indica que el Bot no está sobre-entrenado sobre los datos de entrenamiento. Esto es así, ya que se está evaluando la recompensa obtenida independientemente de cual sea la acción tomada paso a paso (comprar, vender o mantener). Si la recompensa es siempre positiva, esto quiere decir que el modelo está tomando decisiones acertadas en base a la estrategia en la que se esté basando.

- **Estable:** Entre cada paso, la recompensa obtenida tiene que ser de tamaño similar y constante. Esto se debe a que el problema particular al que se está enfrentando cuenta con alta volatilidad (evolución del precio del BTC). De esta manera se evitan problemas de *catastrophic forgetting* y otros relacionados a *reward shape*. El efecto de catastrophic forgetting sucede cuando un agente aprende muy rápidamente mientras explora el ambiente y luego aprende algo nuevo que rompe con lo anterior, por lo que termina siendo fuertemente castigado (Géron, 2019). El concepto de reward shape, refiere a que se busca que la recompensa que se le dé al agente sea gradual y no con grandes escalones. De esta manera, el agente puede ir explorando poco a poco el ambiente, aprendiendo lentamente de sus errores y no se lo premia de forma repentina y desmedida en un solo paso al alcanzar el objetivo. Caso contrario, se obtendría un agente que solo respondería ante una recompensa extremadamente abundante y segura. (BonsaiAI Microsoft team, 2017).

- **Promedio y mediana alta y similar:** Por último, se considera la recompensa neta de la función de recompensa neta promedio y mediana entre cada iteración, buscando que sean mayores y similares respectivamente posible. Siendo más importantes las iteraciones finales, ya que deberían ser las mejores en términos de aprendizaje. En Reinforcement Learning, cada iteración aprende de la anterior.

Al cumplirse estos tres supuestos, se espera que el Bot entrenado *performe* bien sobre los datos de test. Es decir, que se comporte de manera similar sobre datos de validación, al momento de compararlo con datos nunca antes vistos por el modelo.

[18] Discusión entre Karpathy y autores del paper "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer": https://news.ycombinator.com/item?id=13518039

La *factibilidad* del modelo se mide observando la evolución de cómo el Bot opera y los resultados obtenidos a lo largo del periodo de test. La factibilidad se traduce como: durante cuánto tiempo el bot se mantiene rentable y que tan rentable se mantiene, independientemente del resultado del balance final. Para ello, se tiene en cuenta las siguientes métricas:

- **Porcentaje de volumen en ganancias:** Independientemente de cómo termine el modelo en términos de ganancia, se mide el volumen de USD ganados y perdidos durante cada paso en el periodo de análisis. Con dichos volúmenes se calcula el porcentaje de volumen de USD ganados.

$$\frac{Cantidad\ de\ USD\ ganados\ por\ paso}{Cantidad\ de\ USD\ ganados\ por\ paso + Cantidad\ de\ USD\ perdidos\ por\ paso}$$

- **Porcentaje de pasos rentable:** Se computa la cantidad de pasos en los que el modelo se mantuvo con un balance igual o mayor al inicial. También se calcula la cantidad de pasos en los que se mantuvo con un balance menor al inicial. Con ambos valores, se computa el porcentaje de tiempo durante el cual el modelo se mantuvo rentable en el periodo en el que esté operando.

$$\frac{Cantidad\ de\ pasos\ Rentables}{Cantidad\ de\ pasos\ Rentables + Cantidad\ de\ pasos\ NO\ rentables}$$

- **Volumen de ganancias y pérdidas promedio por paso:** Es el promedio del volumen de las ganancias cuando el modelo se mantiene rentable en cada paso, y el volumen de pérdidas cuando no es rentable a cada paso respectivamente. Con estas métricas, se mide que tan rentable o no rentable fue, en promedio, el resultado de las operaciones del bot.

$$Ganancias\ promedio = \frac{Cantidad\ de\ USD\ ganados\ por\ paso}{Cantidad\ de\ pasos\ Rentables}$$

$$Pérdidas\ promedio\ \frac{Cantidad\ de\ USD\ perdidos\ por\ paso}{Cantidad\ de\ pasos\ NO\ Rentables}$$

- **Factibilidad:** Esta métrica, es la que se toma finalmente para determinar si el modelo es factible o no durante el periodo bajo análisis, dependiendo de si es un valor positivo o negativo. De ser positivo, sirve para medir qué tan factible es el modelo. Por el contrario, de ser negativo, sirve para medir que tan no factible es el modelo durante el periodo donde el mismo se encuentra operando.

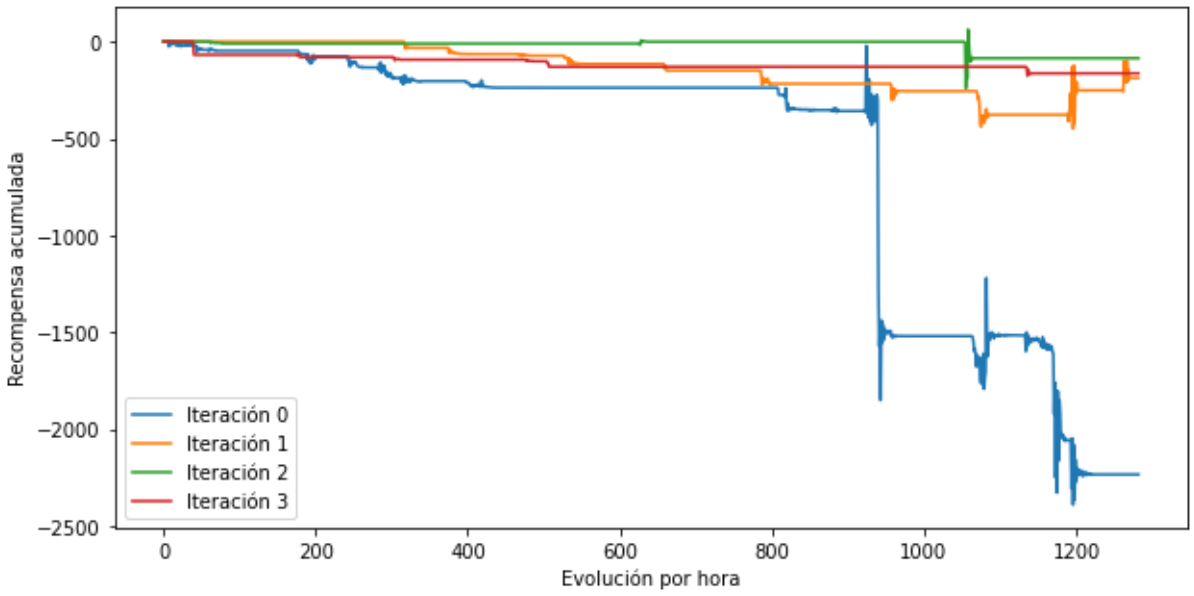
$$Factibilidad = Ganancias\ promedio * \%\ pasos\ rentable - Pérdidas\ promedio * (1 - \%\ pasos\ rentables)$$

A partir de estas métricas, se realizan observaciones sobre la definición de factibilidad anterior, al analizar las operaciones resultantes de los modelos preseleccionados. Luego, se puede determinar finalmente la factibilidad de un modelo en base a la métrica de *factibilidad* sobre datos de test. Nuevamente, se recuerda no confundir los resultados de la *recompensa neta* de la función de recompensa al analizar performance, con los resultados de *ganancias* medidas al analizar la factibilidad al operar.

3.4 Estrategias baseline y Bot benchmark

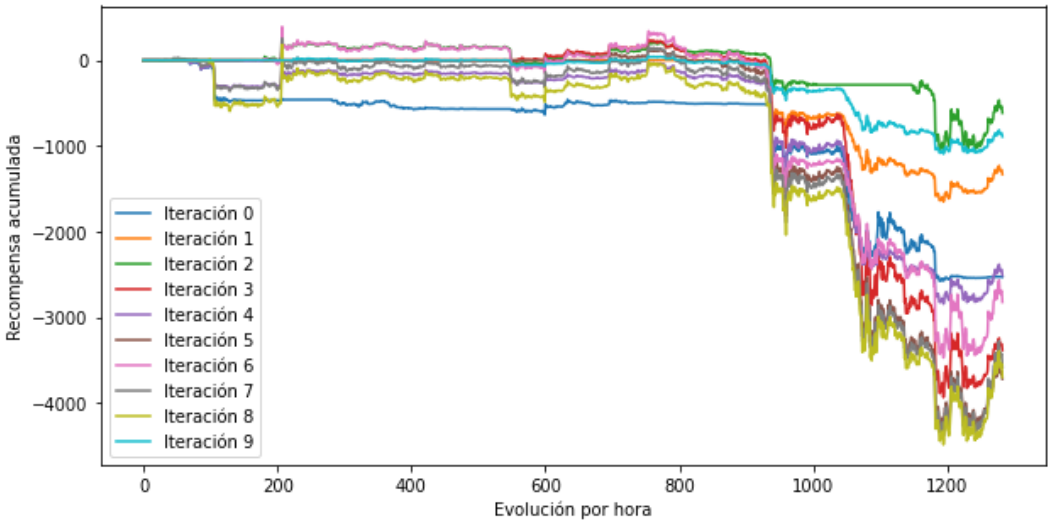
Para poder medir el éxito del objetivo propuesto, es necesario determinar un "piso" o *baseline*, que permita comparar los resultados obtenidos a lo largo de la experimentación y el impacto de cada una de las alternativas utilizadas. Para ello, se escogieron ciertas estrategias de inversión básicas y clásicas, más un Bot benchmark coincidente con el ambiente y agente descrito en la sección 3.2. Tanto las estrategias de inversión, que a lo largo del trabajo serán llamadas estrategias baseline, y el Bot benchmark, utilizan como materia prima información técnica y no fundamental (variables de apertura, cierre, máximos, mínimos y volumen). Las estrategias baseline que se proponen utilizar son: Buy and hold, RSI divergence y SMA Crossover. Si bien la primera estrategia (Buy and Hold) es la más simple de todas, que se traduce en únicamente comprar y mantener, si se piensa en retrospectiva hasta la actualidad, donde el BTC tocó sus máximos históricos en ~60.000 USD, esta estrategia debería ser por lejos la que mayor ganancia hubiese generado desde el inicio de los tiempos del Bitcoin hasta el día de hoy. De la misma manera, si se tomara el periodo bajo análisis de esta tesis, el resultado con Buy and Hold sería totalmente desfavorable (Figura 3.1.3.2). Las últimas dos estrategias, también son simples pero se manifiestan que son efectivas, utilizadas y bien conocidas en el ámbito de las finanzas (Attia, 2019; Brown, 1999). El Bot benchmark de Reinforcement Learning, está basado en un agente A2C, función de recompensa *profit*, una política de predicción MLP y utiliza un ambiente de reinforcement learning como el descrito en la sección 3.2. La regla que utiliza para recompensar es también la expresada anteriormente como balance actual menos balance del paso anterior (figura 3.2.1.1), y utiliza únicamente las 6 variables presentadas al comienzo de este proyecto, sin realizarles ninguna modificación. Al replicar el armado y experimentación del Bot benchmark, se obtienen los siguientes resultados en términos de recompensa neta sobre datos de validación (Figura 3.4.1). Los resultados del entrenamiento prácticamente violan todos los supuestos tomados en la sección 3.3.3. Por esta razón, se esperaría que el modelo no tenga buena factibilidad.

Figura 3.4.1: Bot benchmark (-667 prom. & -175 mediana)



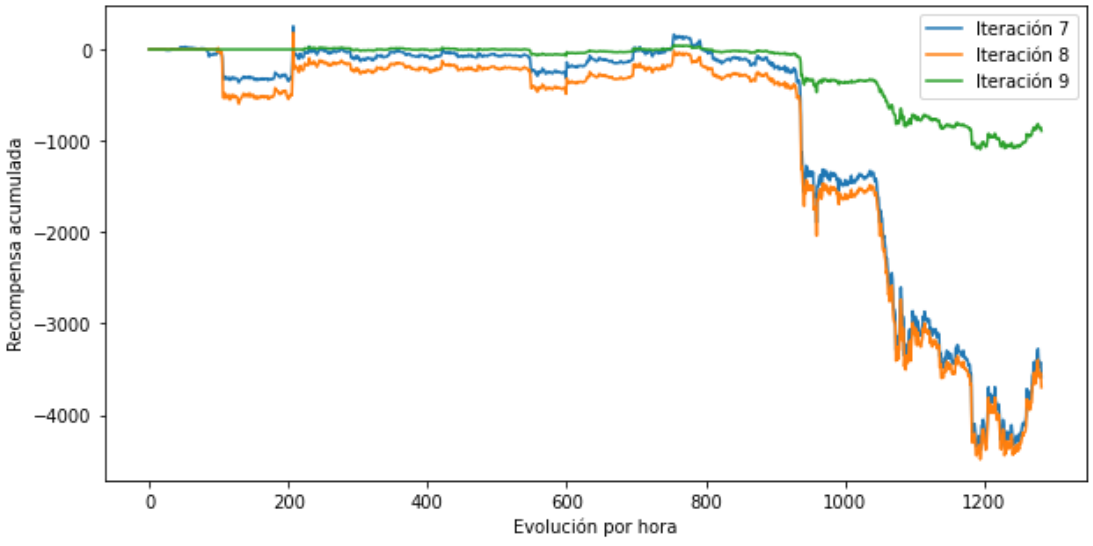
Lamentablemente, se encontró que la forma en la que estaba ordenando los registros el desarrollador del Bot benchmark original, provocan un data leakage o fuga de datos. Esto se dió porque el tipo de formato que tiene el campo “Date” en su vista minable, se encuentra definido como caracteres y contiene las horas en AM y PM. Este formato lleva a que el ordenamiento se termine realizando por los caracteres de AM y PM, lo cual provoca que los modelos y estrategias baseline “vean 12 horas del futuro” en los registros de 12-AM y 12-PM. Se cambió el formato y se volvió a realizar la misma experimentación (Figura 3.4.2):

Figura 3.4.2: Bot benchmark re-entrenado (-2510 prom. & -2677 mediana)



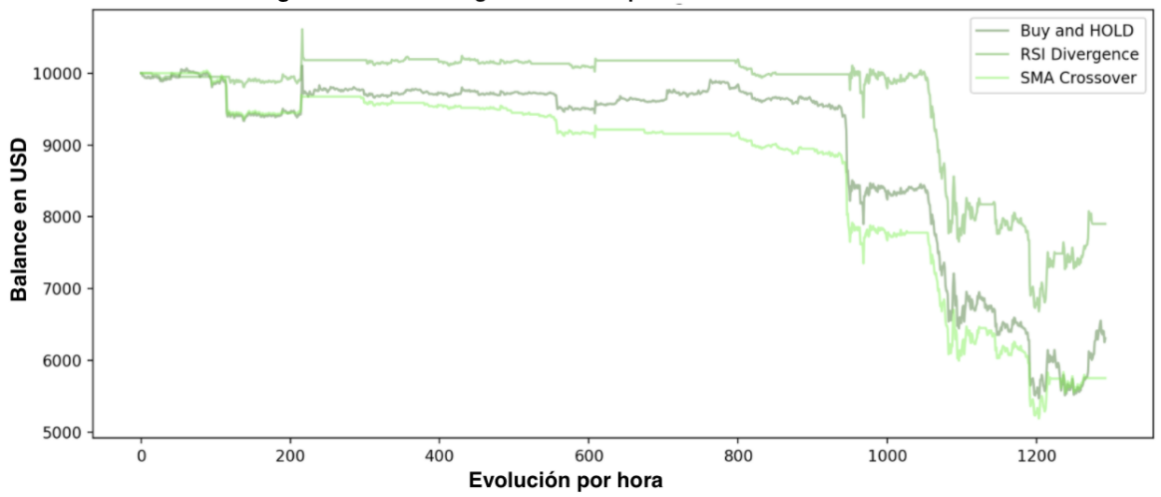
En la figura 3.4.2, es posible ver que ninguna iteración del Bot tiene una recompensa neta creciente positiva, ya que comienzan con 0 y ninguno termina generando grandes recompensas, más bien pérdidas. A continuación, se presenta la performance en términos de recompensa de las 3 últimas y por consiguiente mejores iteraciones, ya que las últimas son entrenadas sumando el aprendizaje de las anteriores. (Figura 3.4.3). Inmediatamente se puede observar que estas mejores iteraciones siguen el patrón de ser fuertemente castigadas luego del paso 900.

Figura 3.4.3: Bot benchmark re-entrenado - Mejores 3 it. (-2726 prom. & -3585 mediana)



A continuación, se presentan las estrategias baseline solo a modo ilustrativo con el fin de mostrarlas operando (ya que las mismas no se entrenan) sobre los datos de validación (Figura 3.4.4). Más adelante en este trabajo se podrá observar operar a los Bots en datos de test y se los comparará con estas estrategias baseline.

Figura 3.4.4: Estrategias baseline operando sobre datos de validación



La caída del precio durante este periodo provoca que para cualquier estrategia (Buy and Hold, RSI y SMA) sea complicado generar rentabilidad comprando BTC, ya que al comienzo de la iteración tienen un balance de 10.000 usd y 0 cantidad de BTC, por lo que lo único que se puede realizar al principio es comprar o mantener. Por supuesto, a la estrategia Buy and Hold, la caída del precio de BTC no le favorece. La señal de SMA parece responder tarde. Esto es un comportamiento esperable, ya que la caída del precio se da de manera repentina y escalonada. Por último, RSI divergente pareciera ser la estrategia que mejor se defiende de los cambios de precio repentinos hasta el momento. Esto último también cobra sentido por la definición comentada en el marco teórico sobre este indicador, el cual al ser divergente se espera que responda rápidamente a este tipo de volatilidad. “Buy and hold” y “SMA Crossover” terminan con un balance similar, mientras que RSI divergente termina con una diferencia de balance mayor de aproximadamente 1.000 dólares. Sin embargo, las tres estrategias terminan con un balance total menor que con el que comenzaron y no pareciera que hayan generado ganancias significativas durante su desempeño en todo el período.

Más allá de las observaciones realizadas, se utilizará el Bot benchmark re-entrenado, con la fuga de datos corregida y las estrategias presentadas, con el fin de utilizarlas como baselines para comparar los resultados de los modelos que se desarrollarán a lo largo del trabajo.

3.5 Seleccionando los mejores modelos

Con respecto a cómo se define a los modelos, hay ciertas variaciones y alternativas que podrían incrementar la performance del Bot benchmark. Se espera que las mejoras de performance se traduzcan en mayor factibilidad. En esta sección se estudiará y explorará aspectos relevantes dentro del mundo del Reinforcement Learning, los cuales se deben tener en cuenta a la hora de definir las características de un modelo. Por esto, las pruebas, conclusiones y decisiones de esta sección, se tomarán en base a los resultados de performance. Entre ellas se encuentran: el agente que se seleccionará para aprender, la política de aprendizaje que utilizará el agente para predecir y las reglas de recompensa y castigo. Luego, se buscará encontrar la mejor configuración de estos tres aspectos entrenando y comparando los bots resultantes.

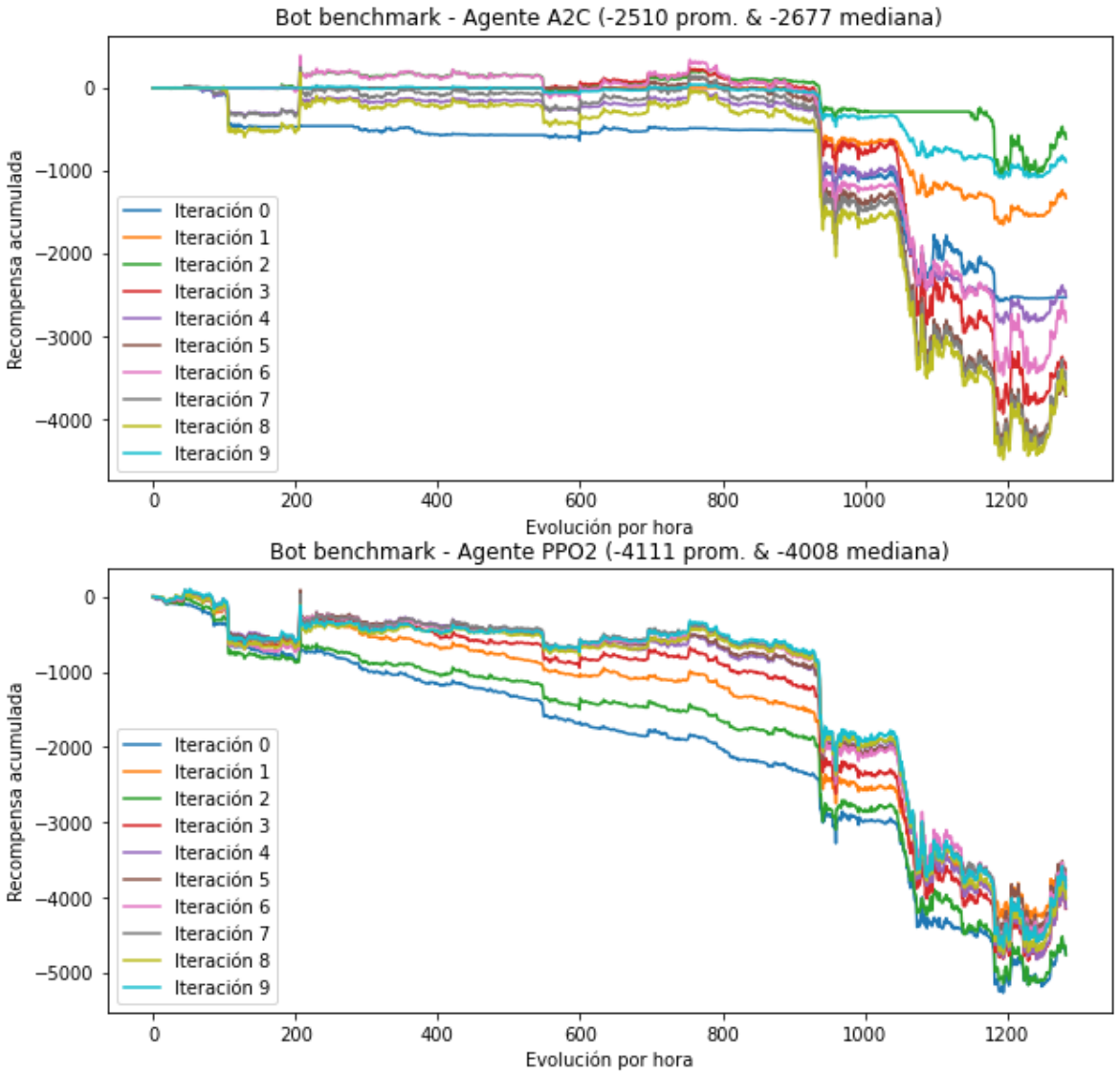
3.5.1 Agente

A la hora de definir un agente para realizar un bot de trading existen varias opciones. El Bot benchmark utiliza un agente A2C perteneciente a la familia de los agentes Actor-Critic, una variante determinística de Asynchronous Advantage Critic (A3C). A2C es similar a A3C pero sin la parte asincrónica, lo cual permite mantener una performance similar siendo mucho más eficiente. El algoritmo A3C, se puede describir esencialmente como la utilización de políticas de gradientes con una función de aproximación, donde ésta es un modelo de deep learning. Los autores utilizaron un método innovador para intentar garantizar que el agente explore mejor el espacio de búsqueda de estados (Mnih, Puigdomènech Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu, 2016). Este tipo de agente es comúnmente elegido para resolver la mayoría de los problemas de Reinforcement Learning. Esto no quita que se prueben otros tipos de agente. Por el contrario, siempre es conveniente y aconsejable probar distintos tipos de algoritmos.

Para este trabajo se probará una variante de agente, el algoritmo PPO2, Proximal Policy Optimization. Como se explica en el marco teórico, la idea principal de utilizar PPO es que después de actualizar la política en un paso determinado, la nueva política no se encuentre demasiado lejos de la vieja. Para ello, PPO hace uso de clipping evitando actualizaciones de políticas demasiado alejadas entre sí. Esta variante a priori es una buena opción a la hora de buscar estabilidad, lo cual es relevante para un contexto con tanta volatilidad. Para trabajar con distintos agentes se utilizó la librería stable-baselines[19] en *python*, la cual contiene una importante variedad de algoritmos de Reinforcement Learning basados en OpenAI Baselines[20]. Se procede a entrenar diez iteraciones de Bots para cada tipo de agente: PPO2 y A2C. Se compararon los resultados en promedios y medianas de la recompensa neta obtenida para determinar cual performa mejor (Figura 3.5.1.1).

[19] https://stable-baselines.readthedocs.io/en/master/index.html
[20] https://openai.com/about/

Figura 3.5.1.1: Agente A2C vs Agente PPO2



Se pueden ver en el gráfico superior los Bots con agentes A2C y en el inferior, los que fueron entrenados con PPO2. La recompensa neta promedio del Bot que utiliza A2C es de -2.510 y una mediana de -2.677, mientras que el promedio del Bot que utiliza PPO2 es de -4.111 y una mediana de -4.008. En ninguno de los dos tipos de Bot se observa recompensa neta acumulada "Positiva Creciente" ni "Estabilidad". Por los valores resultantes del promedio y la mediana, y la evolución que se observa en el gráfico, pareciera ser que los agentes A2C son menos castigados que los agentes PPO2. Por esta razón, se podría decir que para este ambiente y agente, A2C pareciera adaptarse mejor que PPO2. Una posibilidad es que, por la naturaleza de PPO2 de utilizar *clipping* para mantener acortada la distancia entre cada actualización, el algoritmo tardó en encontrar estrategias para no perder recompensa. También se puede pensar que A2C, gracias a su mayor eficiencia algorítmica, aprendió más rápido cuál era una mejor estrategia. Ya sea por la primera o la segunda razón, A2C pareciera desenvolverse mejor que PPO2. Por lo tanto, se continúa trabajando con el agente A2C.

3.5.2 Política de predicción

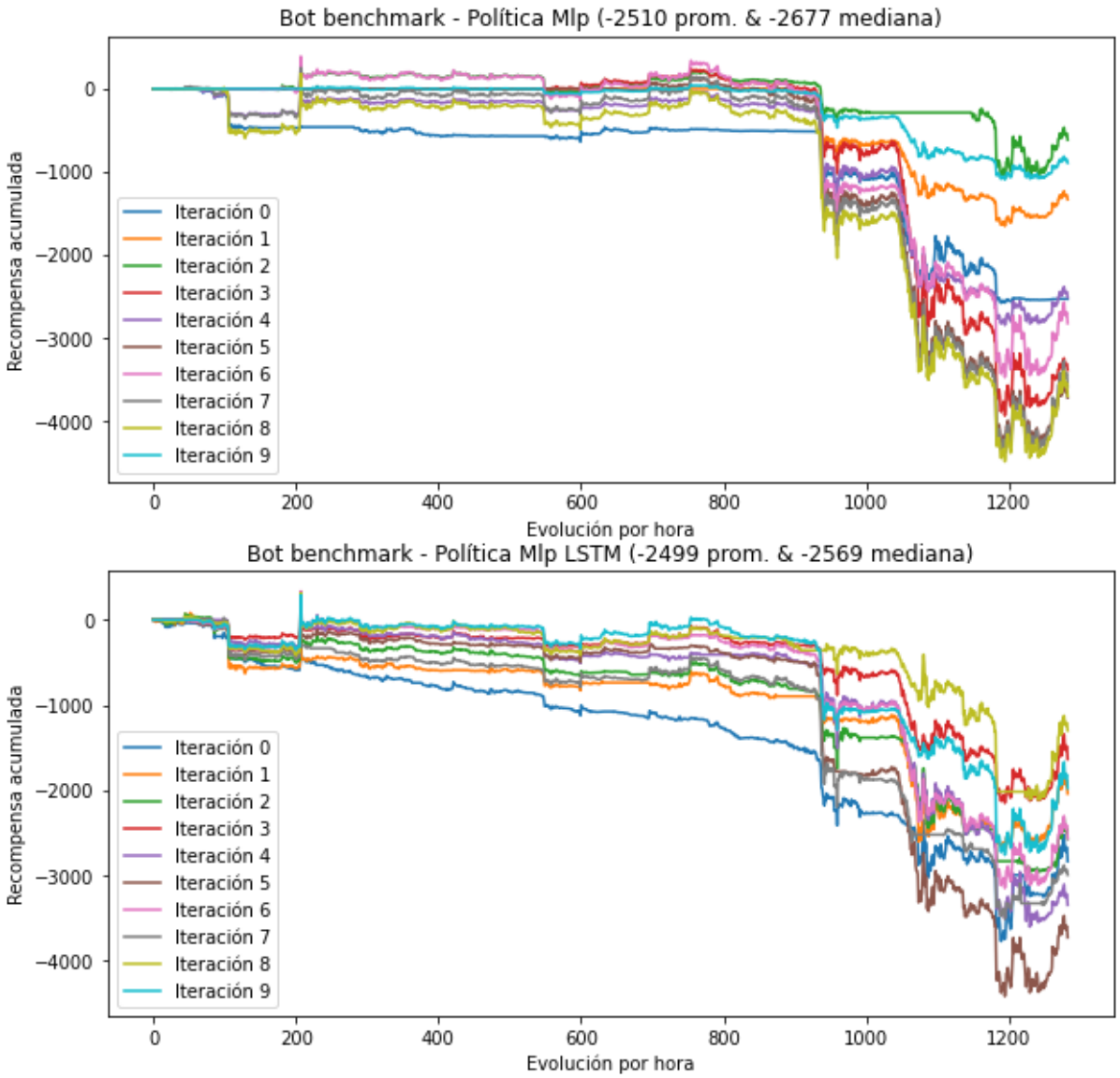
Hasta el momento se mencionó en gran medida sobre los agentes y el ambiente, pero no se trató el tema de la política o topología que utilizan los bots de Reinforcement Learning para realizar las predicciones. Estas predicciones son las que desatan la toma de decisiones informadas por el agente. Las políticas son las que definen qué estructura tiene el modelo que utiliza el bot para realizar las predicciones. El Bot benchmark utiliza una política: Mlp Policy. Esto significa que el Bot utiliza un modelo basado en un perceptrón multicapa para realizar las predicciones. En particular, la librería stable-baselines, contiene distintas políticas que definen la topología completa que utilizan los agentes. En el caso de la política Mlp Policy, la misma está definida por 2 capas de 64 nodos.

Como el objetivo se centra en un problema relacionado con el tiempo y predicciones de series de tiempo, una buena idea es seleccionar una política acorde. En el mundo de Deep Learning, es decir cuando se trabaja con redes neuronales artificiales, se suele automáticamente relacionar los problemas de series de tiempo con las redes neuronales artificiales recurrentes. En específico, con las LSTM o Long Short Term Memory, las cuales presentan una topología que permite a los modelos tomar patrones del pasado lejano (Long) y realizar predicciones haciendo ajustes a partir del pasado más cercano (Short). Relacionadas a este tipo de problemas, se encuentra dentro de la librería stable-baselines, dos políticas que podrían aportar valor al Bot. Ellas son: MlpLstmPolicy y MlpLnLstmPolicy[21]. La primera se podría decir que es una mejora o agregado a la política que viene utilizando el Bot benchmark, MlpPolicy. Esta topología está conformada por una red neuronal artificial recurrente del tipo LSTM y realiza la extracción de características con un Perceptrón multicapa. La segunda política mencionada, MlpLnLstmPolicy, es exactamente igual a MlpLstmPolicy, solo que además, agrega una capa de normalización al principio de la arquitectura de la red. Esto último, suele ayudar a converger más rápido a las neuronas y también se suele utilizar para regularizar modelos basados en redes neuronales artificiales (Géron, 2019).

Por consiguiente, se realiza un entrenamiento de diez bots idénticos al benchmark, con la salvedad de que se cambia la política de MlpPolicy a MlpLstmPolicy. Cabe mencionar, que este único cambio lleva al proyecto a un nuevo campo de estudio. A partir de este punto, se deja el campo de *Reinforcement Learning* y se pasa a explorar modelos de *Deep Reinforcement Learning*. Esto es a consecuencia del cambio de tipo de política de predicciones. A continuación, se presentan los resultados comparando los diez bots benchmark con los nuevos diez bots de Deep Reinforcement Learning utilizando la política MlpLstmPolicy (Figura 3.5.2.1).

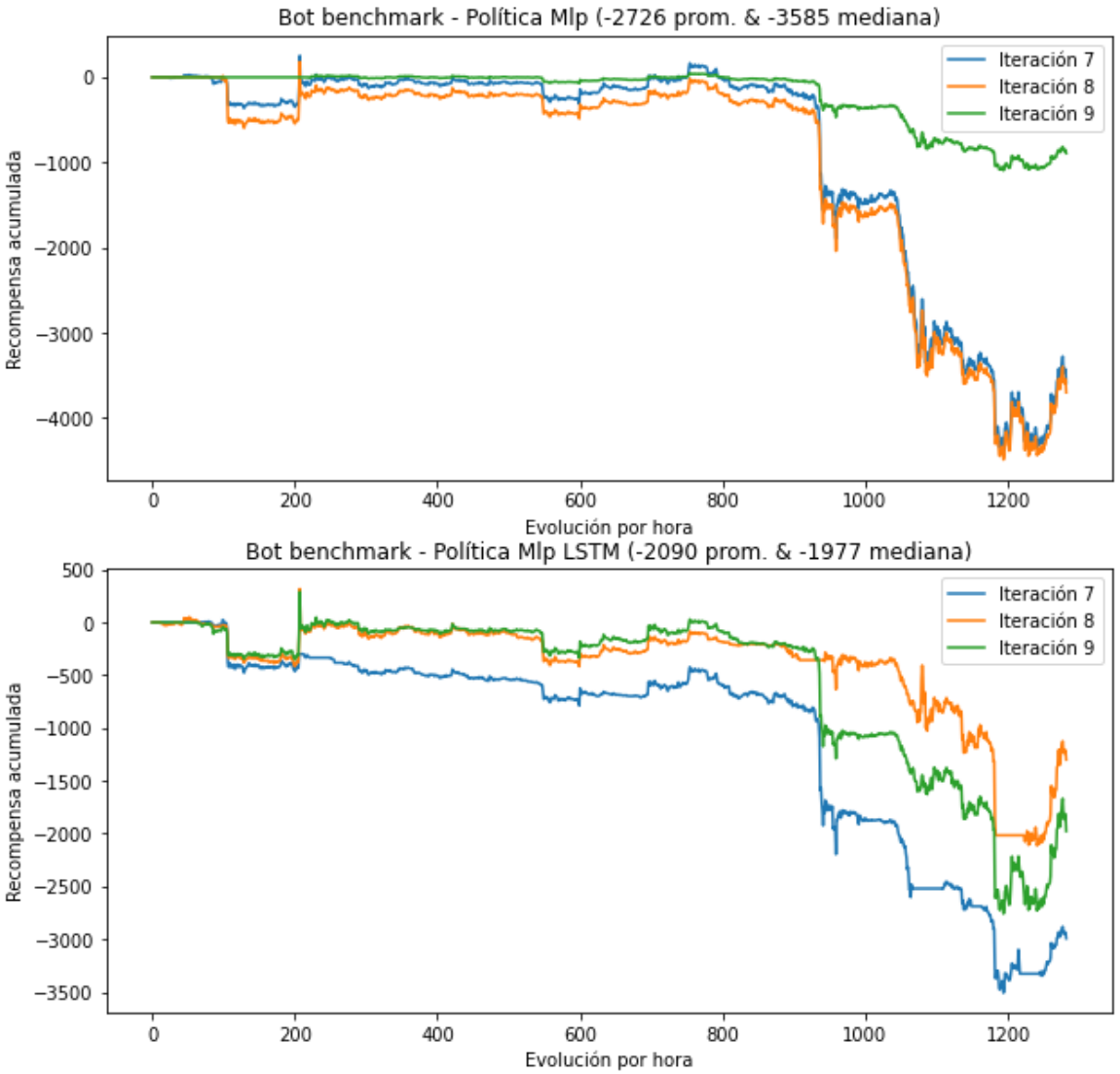
[21] Documentación original: https://stable-baselines.readthedocs.io/en/master/modules/policies.html

Figura 3.5.2.1: Política Mlp vs Política Mlp LSTM



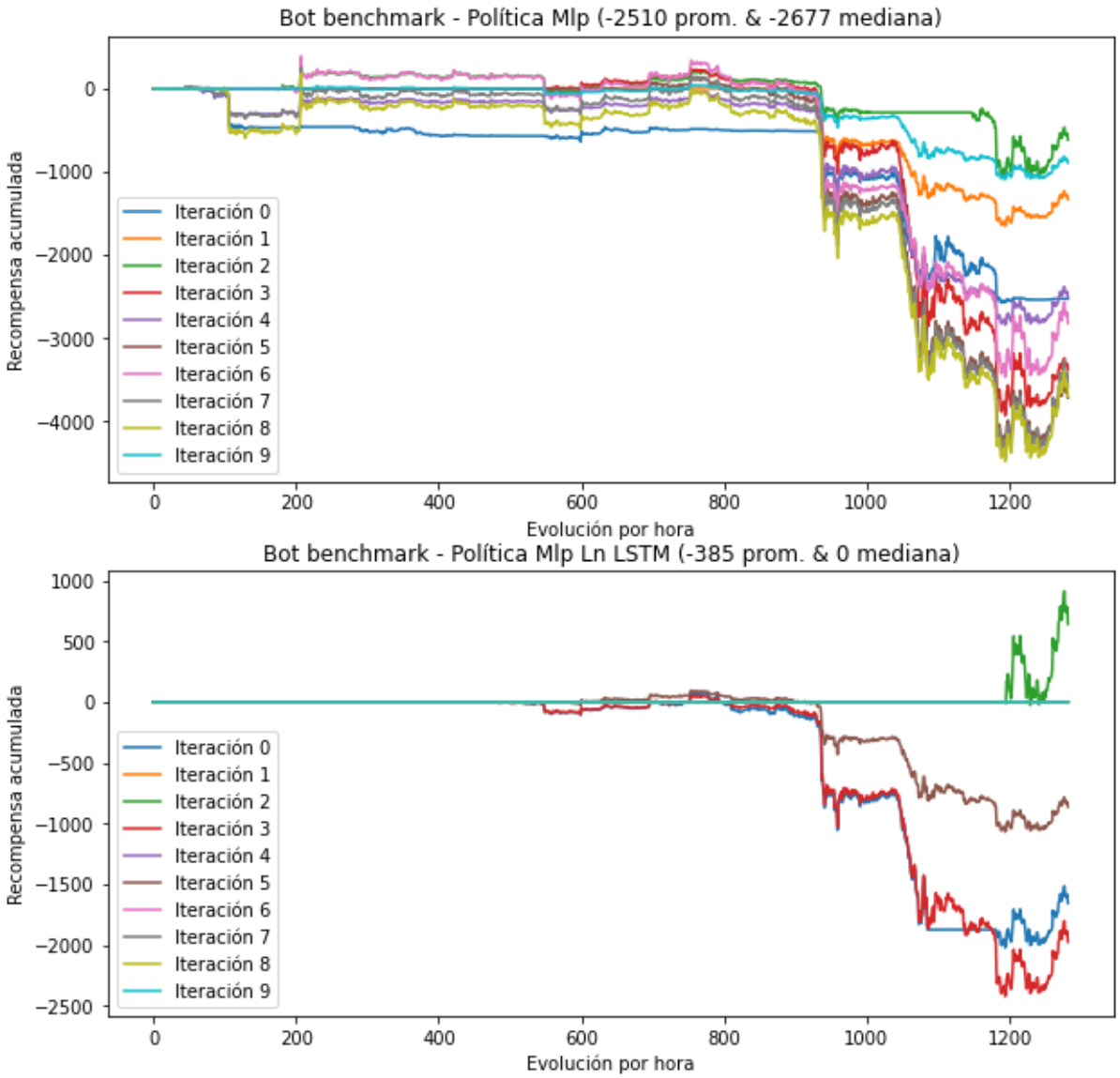
A simple vista, se llega a un escenario similar a como se viene trabajando. (Figura 3.5.2.1). Se aprecia una cierta mejora en términos de recompensa neta promedio y mediana, ya que la nueva política consigue un promedio de -2.499 y mediana de -2.569. Comparando las últimas 3 iteraciones esta leve mejora se puede apreciar más claramente. (Figura 3.5.2.2).

Figura 3.5.2.2: Política Mlp vs Política Mlp LSTM - Mejores 3 iteraciones



Sin embargo se sigue sin conseguir los supuestos de recompensa neta “Creciente positiva” y “Estable”. Se esperaba que con un cambio de política apto al problema en cuestión (LSTM para problemas de serie de tiempo) se obtengan mejores resultados. Una hipótesis sobre lo sucedido, puede ser que al estar trabajando con una política que contiene únicamente 2 capas de Perceptrón multicapa, exija que a la estructura de LSTM le sea costoso converger rápido. Una forma de probar esta teoría es regularizar con alguna técnica como *Drop Out* o *Normalización* aplicada sobre la red de LSTM (Srivastava, Hinton, Krizhevsky, Sutskever, Salakhutdinov, 2014). En este caso, la suerte juega a favor ya que la librería stable-baselines ofrece la política MlpLnLstmPolicy, la cual justamente agrega una capa de normalización a la topología. Se prosigue realizando el cambio de política a MlpLnLstmPolicy, y entrenando diez nuevos bots. De esta manera, el único cambio que sufre el modelo de Deep Reinforcement Learning anterior es agregar una capa de normalización al perceptrón multicapa. A continuación, se presentan los resultados del Bot benchmark y el Bot utilizando MlpLnLstmPolicy. (Figura 3.5.2.3).

Figura 3.5.2.3: Política Mlp vs Política Mlp Ln LSTM



Algunas observaciones que se pueden realizar comparando estos resultados. (Figura 3.5.2.3).

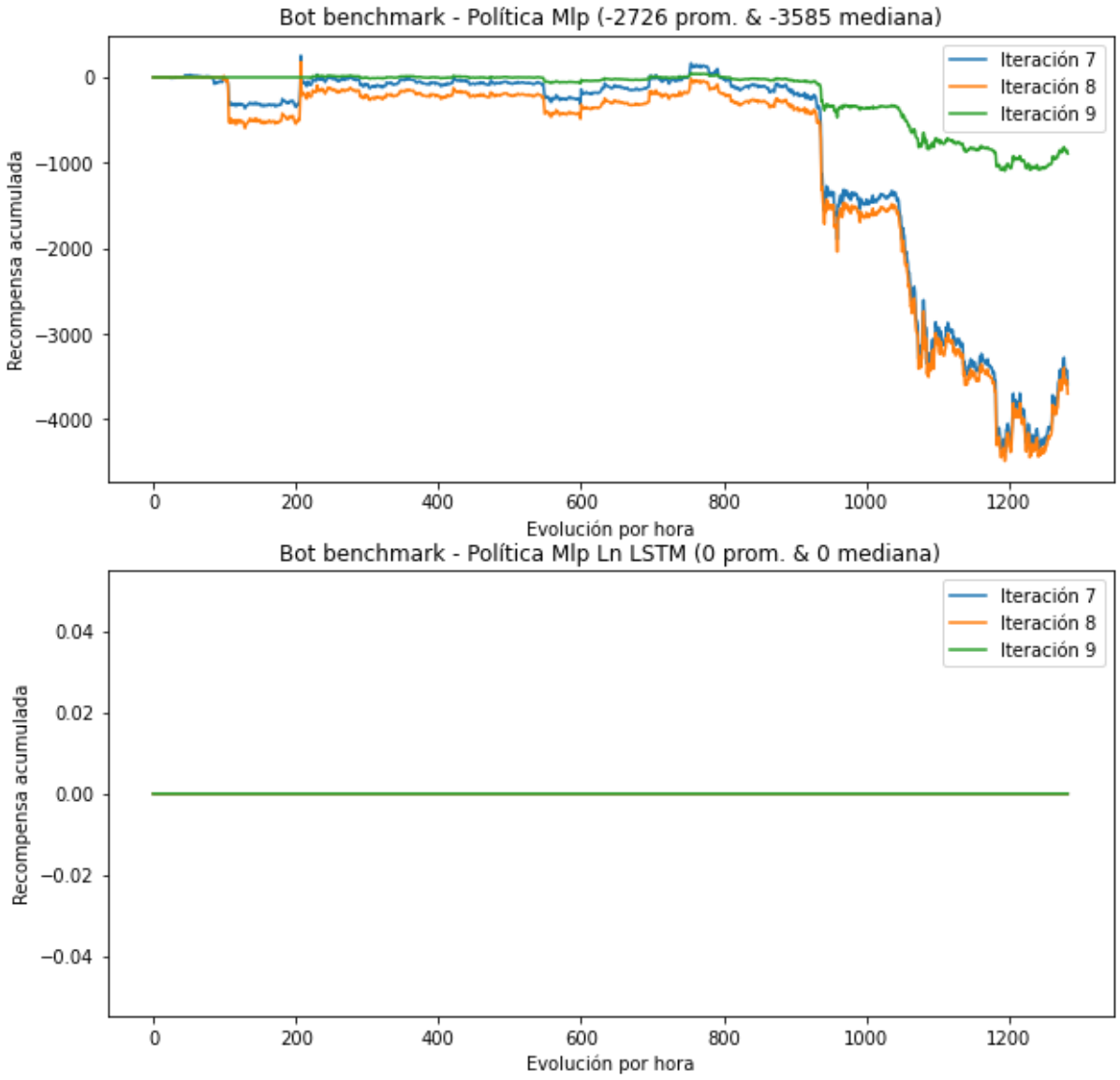
- **El nuevo modelo terminó obteniendo recompensas por arriba del balance inicial en términos de la función de recompensa acumulada:**

Algo que ningún Bot había logrado anteriormente en este trabajo hasta este punto fue generar recompensas netas de niveles llamativamente altos sobre datos de validación. Esto es un dato relevante, ya que como se analizó anteriormente, los datos de validación ocurren durante una caída del precio de BTC con respecto a USD. Como los ambientes comienzan con balances positivos en USD y 0 cantidad de BTC, es realmente un desafío cuidar el balance durante una caída del precio, recibiendo recompensas positivas en dicha situación.

- **El nuevo modelo encontró una estrategia que mantiene el balance de 10.000 usd y la recompensa constante durante todo el periodo:**

Es un llamado de atención a esta estrategia descubierta que el modelo se mantenga con un balance y recompensa constante de principio a fin. Se traduce en que el Bot estaría decidiendo no comprar ni vender BTC, es decir no operar, ya que es la única manera de no ser recompensado o castigado. Se procede a comparar las últimas 3 iteraciones de cada Bot, es decir las que tuvieron mayor espectro de aprendizaje a lo largo del periodo de entrenamiento. (Figura 3.5.2.4)

Figura 3.5.2.4: Política Mlp vs Política Mlp Ln LSTM - Mejores 3 iteraciones



Las últimas 3 iteraciones del Bot con la nueva política terminan obteniendo un balance de recompensa constante, es decir que no encuentra incentivos para arriesgarse a comprar. Se podría comenzar a interpretar que el nuevo Bot, estaría encontrando que la mejor estrategia para invertir es no comprar BTC, lo cual tiene cierto sentido si se piensa en la volatilidad del activo y el precio del mismo durante el período de entrenamiento. Esto no quiere decir que es la mejor estrategia, sino que es el mejor resultado que puede conseguir con la estrategia que estaría aprendiendo el Bot, en base al ambiente y agente con el que se encuentra entrenando. Si bien en este caso utilizando MlpLnLstm se cumple el supuesto de recompensa neta "Estable", se podría decir que es una excepción a la regla, ya que si el Bot no consigue incentivos para arriesgarse a fallar (aprender), no sería válido decir que el Bot tiene una buena performance, por más estable que sea el proceso de aprendizaje. Es por eso que también se definió el supuesto de recompensa neta "Creciente positiva", para asegurar que está aprendiendo y que de alguna manera lo está haciendo correctamente. Teniendo en cuenta lo anterior, al comparar promedio y mediana en términos de recompensa neta, se termina por optar por la política de MlpLnLstmPolicy como la política que mejor se adapta al problema.

3.5.3 Función de recompensa

Por último, se realiza una evaluación sobre cómo el Bot tiene en cuenta el riesgo de ser recompensado o castigado. Se recuerda que en base a dicho riesgo es que luego el modelo puede recibir una recompensa, si es que el resultado de su acción (comprar, vender o mantener) es positivo, o un castigo si el resultado de su acción resulta negativo. Todo esto es computado en la función *"reward"* o función de recompensa*,* definida en el ambiente como una de las funciones que componen al modelo o Bot. Además, se recuerda que, la función matemática "*reward*" llamada en este trabajo como "*Profit*", utilizada para computar la recompensa sobre una observación determinada del Bot benchmark, resulta muy sencilla y con varias oportunidades de mejora (Figura 3.2.1.1). Esta función de riesgo, básicamente premia o recompensa al agente que utiliza el Bot benchmark, cuando el balance neto es mayor al balance neto anterior. En términos matemáticos se está en presencia de una función lineal, que tiene en cuenta como única variable el balance neta en distintos pasos del tiempo, sin contemplar el riesgo asumido de las transacciones anteriores. Esta función no evalúa el riesgo completamente, sino que lo simplifica basándose en el paso inmediatamente anterior. En resumen, es sencilla por demás y tiene varios aspectos de mejora. Se espera que utilizando funciones que calculen el riesgo de manera más minuciosa, el Bot pueda performar mejor a la hora de computar las recompensas generadas.

En el mundo financiero, existen varias alternativas para medir la relación entre el riesgo y el retorno, en base a una inversión a la hora de tomar una decisión. En el marco teórico se presentaron algunas de las más conocidas. Entre ellas, la más comúnmente utilizada es *Sharpe ratio*. Si bien la definición de este fue explicado en el marco teórico, es bueno recordar que cuanto mayor es el *Sharpe Ratio*, mejor es la rentabilidad del fondo en relación a la cantidad de riesgo que se ha tomado en la inversión. Por lo tanto, la recompensa se sigue midiendo de manera positiva.

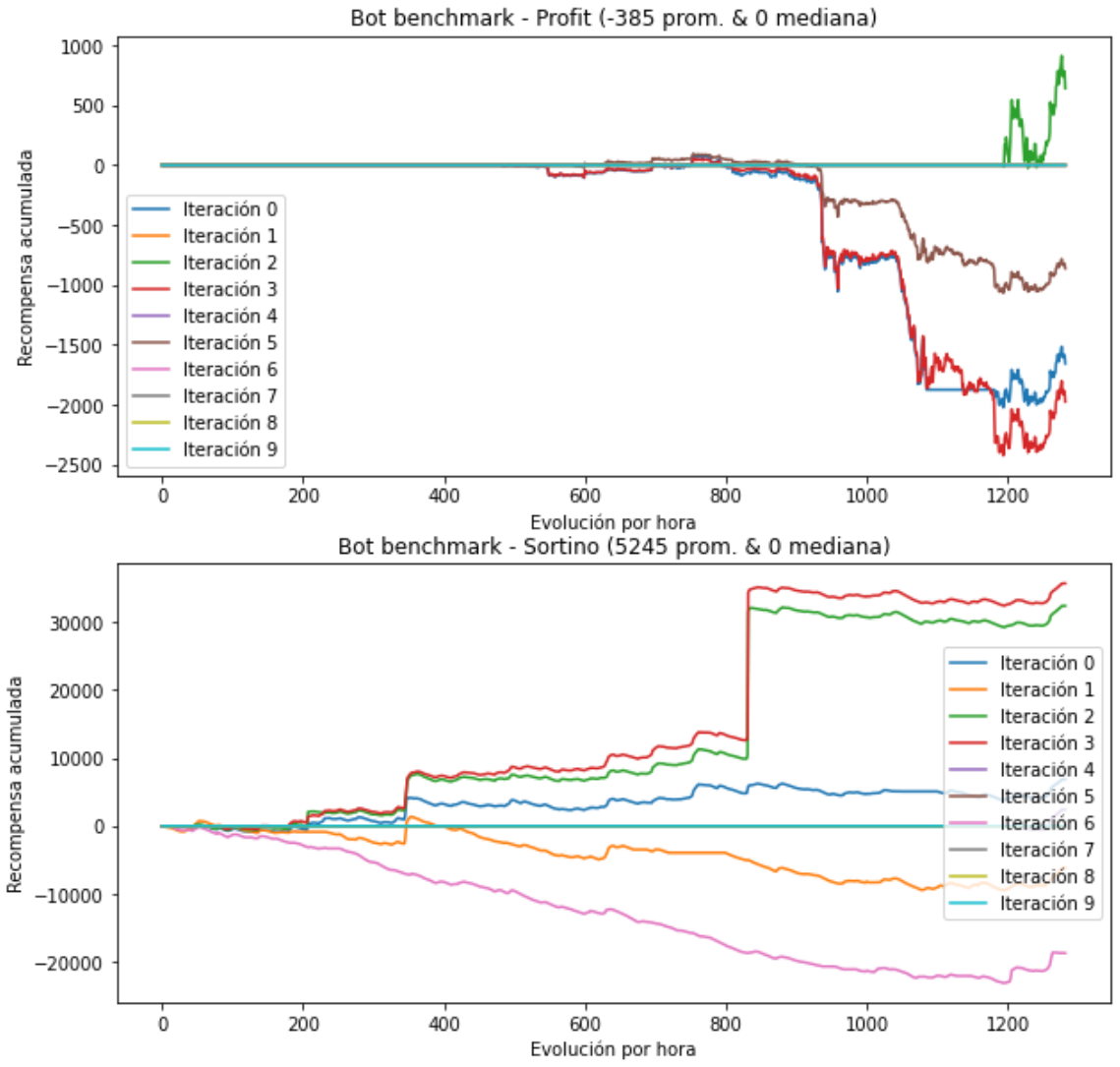
$$Sharpe = \frac{Rp - Rb}{\sigma p} = \frac{Portfolio\ returns - Benchmark\ returns}{Standard\ deviation\ of\ portfolio}$$

El problema que tiene esta fórmula matemática para el trabajo en particular es que penaliza la volatilidad hacia arriba. Esto puede ser perjudicial para la performance del bot, ya que es la volatilidad del Bitcoin la que permite realizar ganancias de dinero con tasas llamativas para el ámbito financiero. Por lo expuesto, se exceptúa realizar una prueba con esta función de riesgo y se pasa directamente a considerar *Sortino ratio.* Este ratio es muy similar a *Sharpe ratio,* salvo que únicamente considera riesgosa la volatilidad hacia abajo, premiando fuertemente la volatilidad hacia arriba.

$$Sortino = \frac{Rp - Rb}{\sigma d} = \frac{Portfolio\ returns - Benchmark\ returns}{Standard\ deviation\ of\ downside}$$

A continuación se observa la comparación del nuevo modelo benchmark, A2C con política MlpLnLstmPolicy y la función de recompensa simple, a la cual se llamó *Profit*, contra el mismo agente pero con una función de recompensa que utilice Sortino ratio. (Figura 3.5.3.1).

Figura 3.5.3.1: F. de recompensa Profit vs Sortino



En el gráfico superior se ven las iteraciones del bot anterior, mejorado con la nueva política, del cual ya se sabía que su comportamiento no generaba recompensa neta relevante. Por otro lado, en el gráfico inferior, se ve el mismo bot utilizando una función de riesgo diferente, *Sortino ratio*. Lo primero que llama fuertemente la atención, son aquellas dos iteraciones que terminan en una recompensa neta por arriba de 30.000. También es posible apreciar que algunas de las iteraciones terminan con recompensas netas negativas, hasta por debajo del balance inicial. Si bien no se nota en el gráfico (quedan por detrás), varias iteraciones volvieron a copiar la estrategia del balance constante.

Se analizan detenidamente los supuestos sobre la recompensa neta para determinar cuándo un modelo performa bien para este problema:

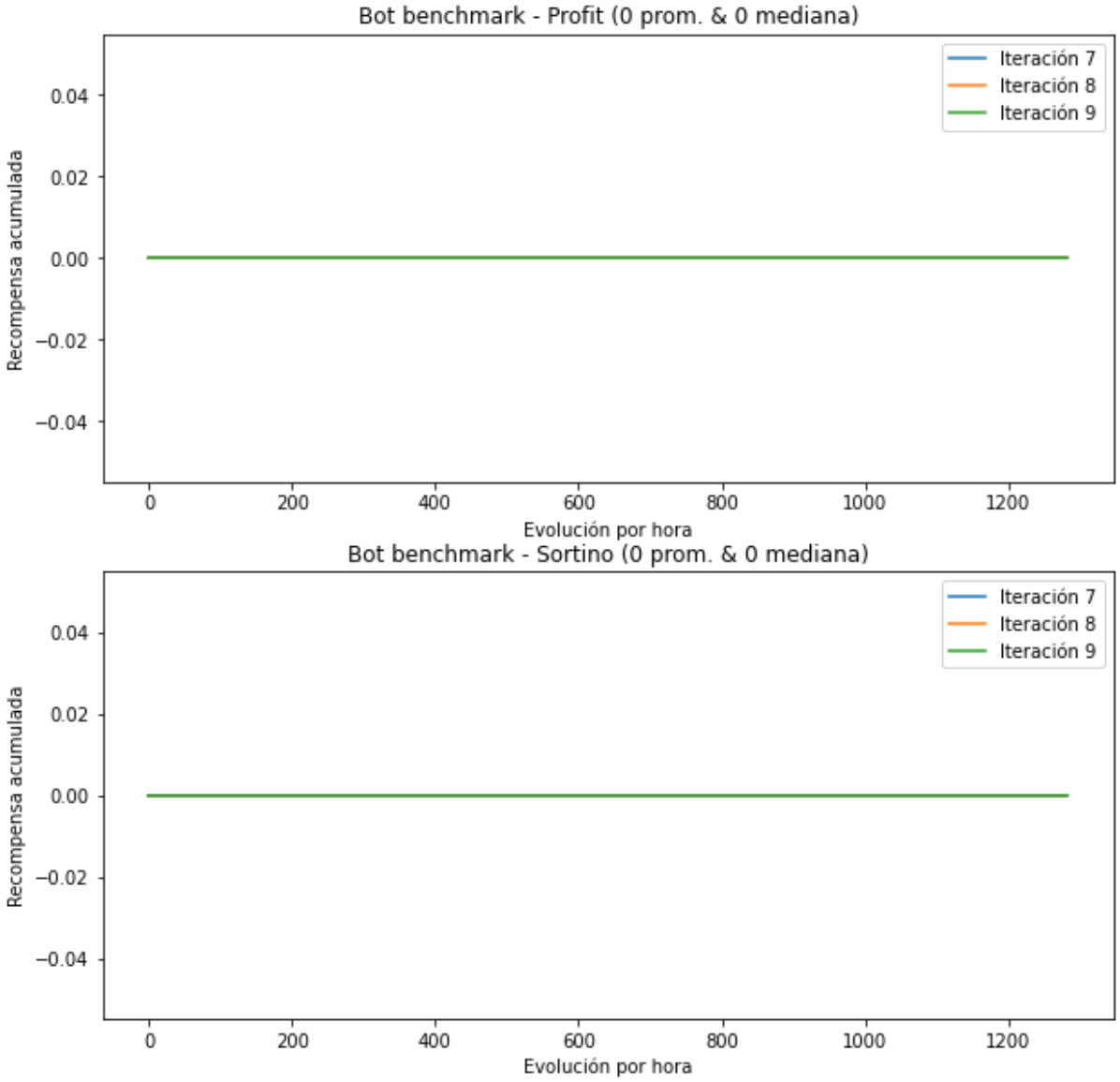
- **Creciente positiva:** Si bien la recompensa neta es creciente positiva en algunas iteraciones, en otras termina siendo negativa. Lo cual no indica que se esté ante un buen modelo, sino ante un modelo que potencialmente en algunas iteraciones esté sobre entrenado (Géron, 2019).

- **Estable:** Con solo mirar el gráfico es posible definir que no se está en presencia de un modelo estable. Consigue recompensas extremadamente altas y en otras iteraciones es penalizado con recompensas negativas, por debajo de 0. Del paso 300 al 400 se puede observar cierto efecto de “catastrophic forgetting”, ya que la curva crece rápidamente, alcanza un máximo y vuelve a decrecer. Justo después del paso 800 también se puede ver problemas de “reward shape”, ya que de un paso al otro, el Bot consigue una recompensa muy exagerada con respecto a los anteriores.

- **Promedio y mediana:** Se obtiene un promedio de 5.245 y mediana de 0. Este supuesto pierde cierto sentido para ser analizado por lo expresado en los dos supuestos anteriores y en el gráfico anterior.

Es interesante comparar las últimas 3 iteraciones de ambos modelos (Figura 3.5.3.2), ya que como se dijo repetidas veces, son las que mayor conocimiento adquirieron en el entrenamiento.

Figura 3.5.3.2: F. de recompensa Profit vs Sortino - Mejores 3 iteraciones



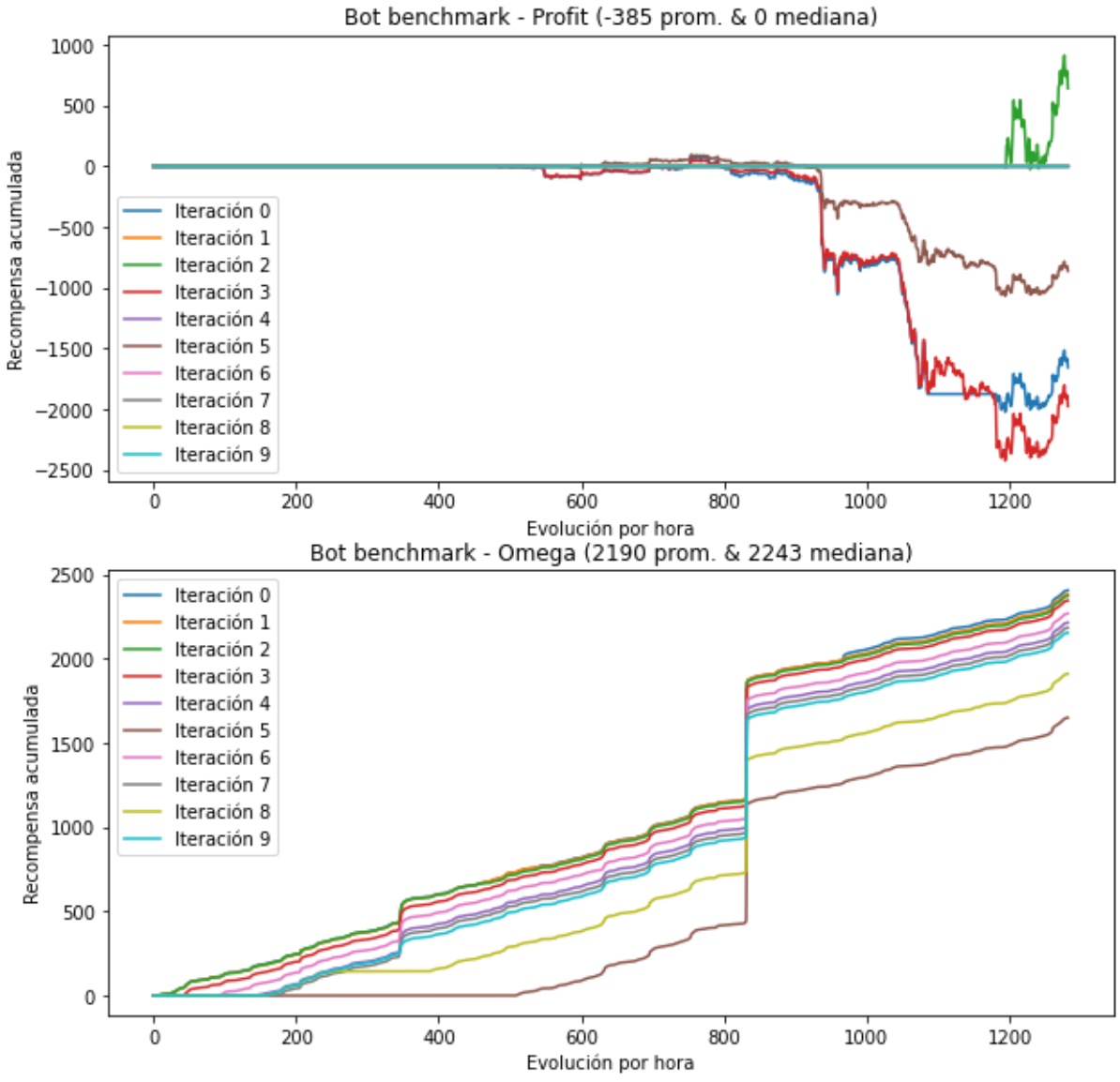
Si bien la función de recompensa que utiliza *Sortino ratio,* queda claro que es demasiado inestable e inadecuada para el problema, ambiente y agente, en las últimas 3 iteraciones convergen al mismo resultado que se ve con la función de recompensa que utiliza *Profit*.

Se busca estudiar una tercera opción de función de riesgo: *Omega ratio.* Se prueba la función de riesgo de retorno *Omega ratio*. De esta función se podría esperar encontrar mejores resultados al medir el riesgo frente al retorno obtenido con *Sortino ratio.* Esto es así, ya que como indica la fórmula de *Omega ratio*, es capaz de tener en cuenta el riesgo completo sobre la distribución del retorno en una sola métrica.

$$Omega = \frac{\int_{Rb}^{b} 1-F(Rp)dx}{\int_{a}^{Rb} F(Rp)dx} = \frac{Upside\ potential}{Downside\ potential}$$

Para llegar a dicha métrica, se calcula la distribución de probabilidad del portfolio, moviéndose sobre o por debajo de un determinado benchmark. Termina computando ambos ratios, *Upside y Downside*[22]. Luego, cuanto mayor sea la probabilidad del ratio total, mayor la probabilidad *Upside*. Por lo tanto, la función de recompensa sigue siendo positiva. A continuación, se muestran los resultados obtenidos al entrenar diez iteraciones del Bot mejorado con la política MlpLnLstmPolicy, utilizando la función de riesgo de retorno *"Omega ratio"* contra el nuevo Bot benchmark.

Figura 3.5.3.3: F. de recompensa Profit vs Omega



[22] Integrales pertenecientes al cálculo de Omega.

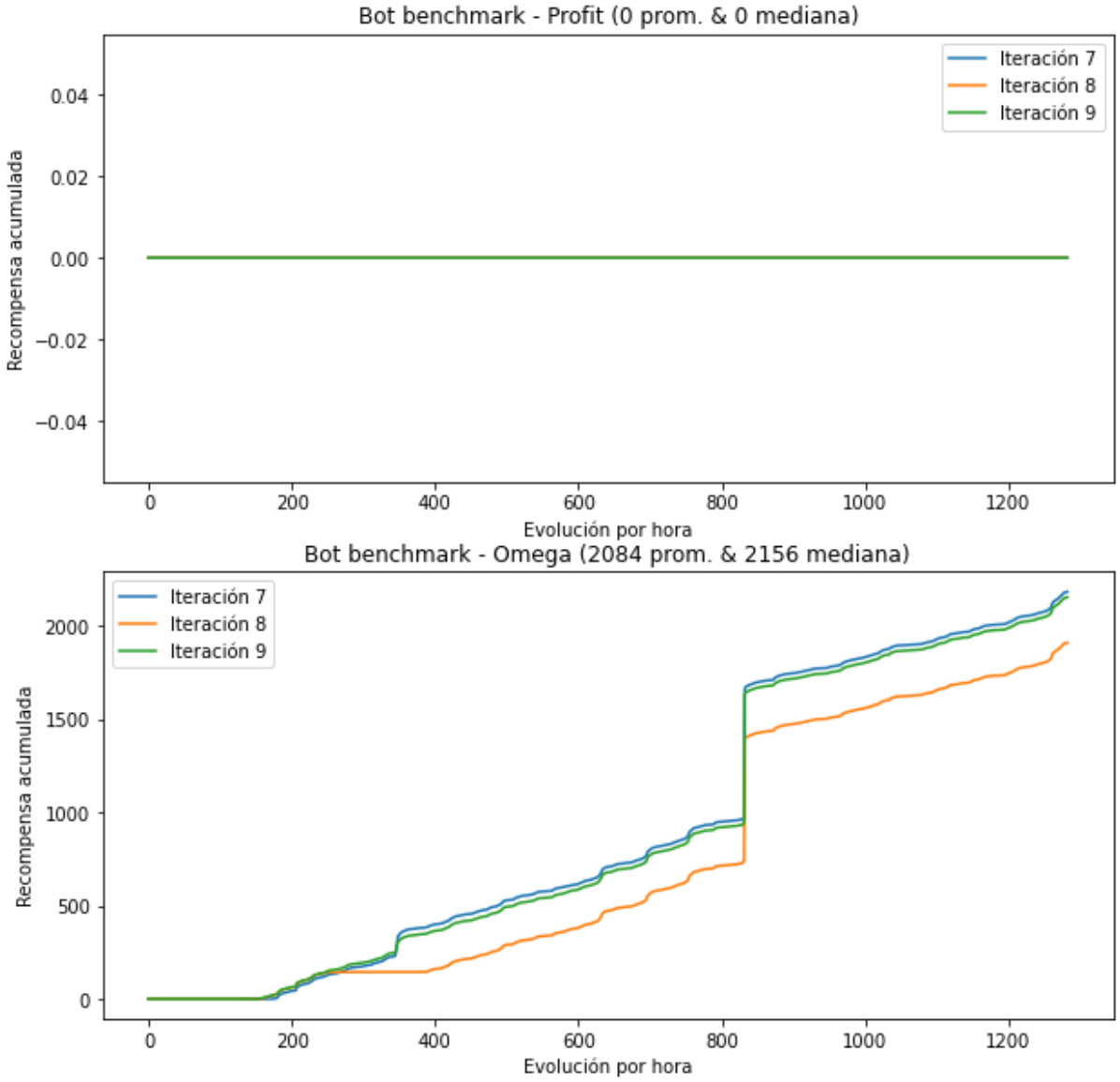
Esta vez, sí se arriba a un comportamiento más cercano al que se estaba buscando obtener, más allá del resultado de la recompensa neta acumulada final. En el gráfico inferior, se puede ver el Bot utilizando una función de riesgo de retorno *"Omega ratio",* y en el superior, nuevamente al Bot utilizando la función de riesgo de retorno *"Profit".* Se analizan los resultados obtenidos y el comportamiento del Bot utilizando *"Omega ratio"* por supuestos nuevamente.

- **Creciente positiva:** Las funciones de recompensa dan una recompensa neta positiva en todas sus iteraciones. Además, al observar la evolución de la recompensa acumulada, se puede ver que se mantiene creciente. Es decir, no obtiene recompensas negativas, si bien se pueden destacar ciertas mesetas llamativas.

- **Estable:** Si bien hay algunos indicios de problemas con "reward shape", que se pueden ver repetidamente entre el paso 300 y 400 e inmediatamente luego del paso 800, y algunas mesetas que podrían decaer causando "catastrophic forgetting", es el primer modelo que no contiene iteraciones significativamente inestables. Además, se puede observar cómo terminan siguiendo un patrón en cada iteración, en la que se va mejorando la recompensa neta total. Sin embargo, hasta en las mejores 3 iteraciones, se ven mesetas que llevan a cuestionar el cumplimiento de este supuesto. (Figura 3.5.3.4).

- **Promedio y mediana:** No solo la recompensa neta total va aumentando paso a paso, sino que también el promedio de la recompensa neta alcanza valores positivos, 2.190, y una mediana similar de 2.243, que también contribuye como indicador de estabilidad entre iteraciones.

Figura 3.5.3.4: F. de recompensa Profit vs Omega - Mejores 3 iteraciones



En base a la definición y supuestos sobre un bot que performe bien para el ambiente y el problema en cuestión, es posible afirmar, que se arribó a un modelo que se aproxima a dicha definición, más que el Bot benchmark original. Por último, se cierra esta sección concluyendo que, en base a lo explorado con los cambios de función matemática realizados en el cálculo de la recompensa, teniendo en cuenta el riesgo de una manera más robusta, y según los resultados obtenidos, se llega a mejoras en el perfomance del Bot relevantes. El nuevo bot ahora puede mantener recompensas “promedio y medianas positivas y similares”, “crecientes positivas” y aproximadamente “estables”, evaluando el riesgo a partir de *Omega ratio*. Por lo tanto, se continúa utilizando la función de recompensa con Omega ratio.

3.5.4 Seleccionando la mejor configuración

Recapitulando, se llegó a la última parte de esta sección. En la misma, se buscó presentar y probar distintas alternativas, contemplando los aspectos más relevantes que hacen al agente y el ambiente en Reinforcement Learning. Además, se dejó atrás el campo de Reinforcement Learning, para pasar al campo del Deep Reinforcement Learning. Por último y no menor, se trabajó sobre la forma de evaluar el riesgo por parte del agente, para tomar decisiones más certeras. Como resultado de la exploración, se arribó a lo que se llamó *"la mejor configuración",* haciendo referencia a la combinación de las distintas características que se tomaron como más adecuadas para el problema que se busca resolver. Por ende, la siguiente tabla resume los resultados obtenidos y lo que se considera *"la mejor configuración".* Modelo número 6 (figura 3.5.4.1). Esta configuración es seleccionada como la mejor sobre el resto[23], por ser la que más se aproxima a los supuestos iniciales, los cuales definen lo que significa un modelo con buena performance para el problema al que se enfrenta este trabajo. El modelo nro. 6 es el único que cumple en todas sus iteraciones con el supuesto de "Creciente Positiva", y que además, presenta recompensas promedio y mediana positivas y similares.

Figura 3.5.4.1: Tabla resumen de la performance los Bot entrenados en la sección 3.5

		Configuración del modelo				Supuestos iniciales			
Nro	**Sección**	**Estrategia**	**Agente**	**Política de predicción**	**F. de recompensa**	**C.P.**	**Est**	**Recomp. neta promedio**	**Recomp. neta mediana**
1	3.4	Benchmark	A2C	MlpPolicy	Profit	NO	NO	-2.510	-2.677
2	3.5.1	RL	PPO2	MlpPolicy	Profit	NO	NO	-4.111	-4.008
3	3.5.2	Deep RL	A2C	MlpLstmPolicy	Profit	NO	NO	-2.499	-2.569
4	3.5.2	Deep RL	A2C	MlpLnLstmPolicy	Profit	NO	NO	-385	0
5	3.5.3	Deep RL	A2C	MlpLnLstmPolicy	Sortino ratio	NO	NO	5.245	0
6	3.5.3	Deep RL	A2C	MlpLnLstmPolicy	Omega ratio	SI	NO	2.190	2.243

[23] Los valores de recompensa neta promedio y mediana calculados con diferentes funciones de recompensa no son directamente comparables. Los bots que utilizan diferentes funciones de recompensa son comparables al considerar todos los supuestos iniciales en conjunto.

3.6 Ingeniería de atributos

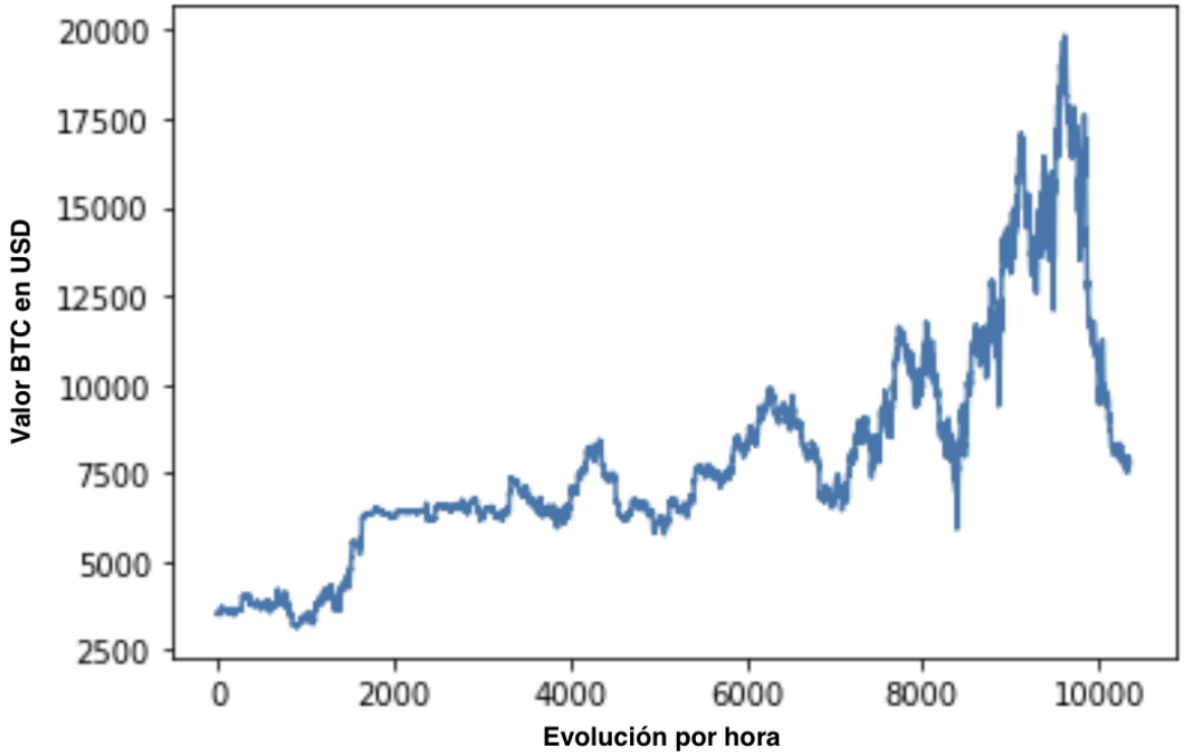
Hasta el momento, se trabajó manteniendo el foco en configurar el Bot de la manera más apta posible, para que pueda defenderse de forma performante, ante el ambiente que se le presenta. Además, se trabajó sobre su capacidad de evaluación, para hacerla lo más realista y semejante posible a lo que podría ser un trader humano. Se puede decir que llegada esta sección, se pasa a la parte más relevante o que busca ser la más relevante de todo el proyecto. Esto es porque en esta sección se trata de brindar mayor información a nivel de análisis estadístico, análisis técnico y por último y más importante en lo que enmarca la hipótesis de este trabajo, análisis fundamental. Dicho esto, en esta sección, no se pierde de vista el objetivo principal: obtener un nuevo Bot que a base de nueva información y procesamiento de la misma, consiga obtener mejores resultados en términos de factibilidad.

En específico, en este capítulo se trabajará sobre los datos y lo que se conoce como *Ingeniería de atributos*. La ingeniería de atributos o feature engineering, se utiliza para revelar características importantes en un conjunto de datos que no son fáciles de identificar en primera instancia. Es por esta razón, que se suele apelar a la experiencia sobre el dominio en el cual se está trabajando, para llevar adelante este tipo de técnicas. Al aplicar feature engineering, se generan nuevas columnas a partir de las variables originales del conjunto de datos con el que se está trabajando. Las características resultantes de feature engineering son posteriormente filtradas, para eliminar aquellas irrelevantes, redundantes o altamente correlacionadas. En otras palabras, se reduce la carga de datos utilizados para entrenar un modelo, con el objetivo de incrementar su performance en general (Géron, 2019). En particular, para el problema que se enmarca en el dominio del trading, se busca crear nuevas variables que aporten valor desde distintos enfoques sobre el mismo.

3.6.1 Análisis estadístico

Cuando se habla de transformaciones clásicas, en el campo de la ciencia de datos o machine learning, se hace referencia a ciertas técnicas y tratamientos que se suelen probar, y que comúnmente pueden llegar a mejorar la performance de los modelos. En este caso, más allá del problema en cuestión, se está trabajando con datos temporales, donde cada registro u observación guarda cierta relación con el anterior y así consecuentemente para cada uno de ellos. Esto último, lleva a analizar los datos con los que se está entrenando los agentes que componen el Bot (Figura 3.6.1.1).

Figura 3.6.1.1: Variable "Close" en datos de entrenamiento con tendencia y estacionalidad



A primera vista, da la impresión de que la serie de tiempo bajo análisis tiene una clara tendencia y estacionalidad. Para poder confirmar esta hipótesis, se realiza un test de hipótesis. Se corre el test Augmented Dickey–Fuller (ADF). La hipótesis nula indica que la serie sería estacionaria. La hipótesis alternativa indicaría que la serie contiene tendencia y estacionalidad (Mushtaq, 2011).

El resultado del test da un P-value igual a 0.375, por lo que se puede interpretar que no se puede descartar la hipótesis nula con un alto nivel de confianza. Por lo tanto, la serie de tiempo contiene tendencia y estacionalidad (figura 3.6.1.1). Una serie de estas características podría afectar los modelos que se suelen utilizar para hacer predicciones sobre series de tiempo (Hanke, Wichern, 2010). Algo que se suele realizar comúnmente para quitar la tendencia sobre una serie de tiempo es diferenciarla. Luego, algo que se podría realizar es aplicar una transformación logarítmica para quitar la estacionalidad, y así aproximarse a una distribución normal. A continuación, se puede observar la serie de tiempo diferenciada (Figura 3.6.1.2), buscando quitar la tendencia. En la figura siguiente a la anterior (Figura 3.6.1.3), se puede observar como resulta de ser diferenciada y con la aplicación de la transformación logarítmica.

Figura 3.6.1.2: Variable "Close" en datos de entrenamiento diferenciados

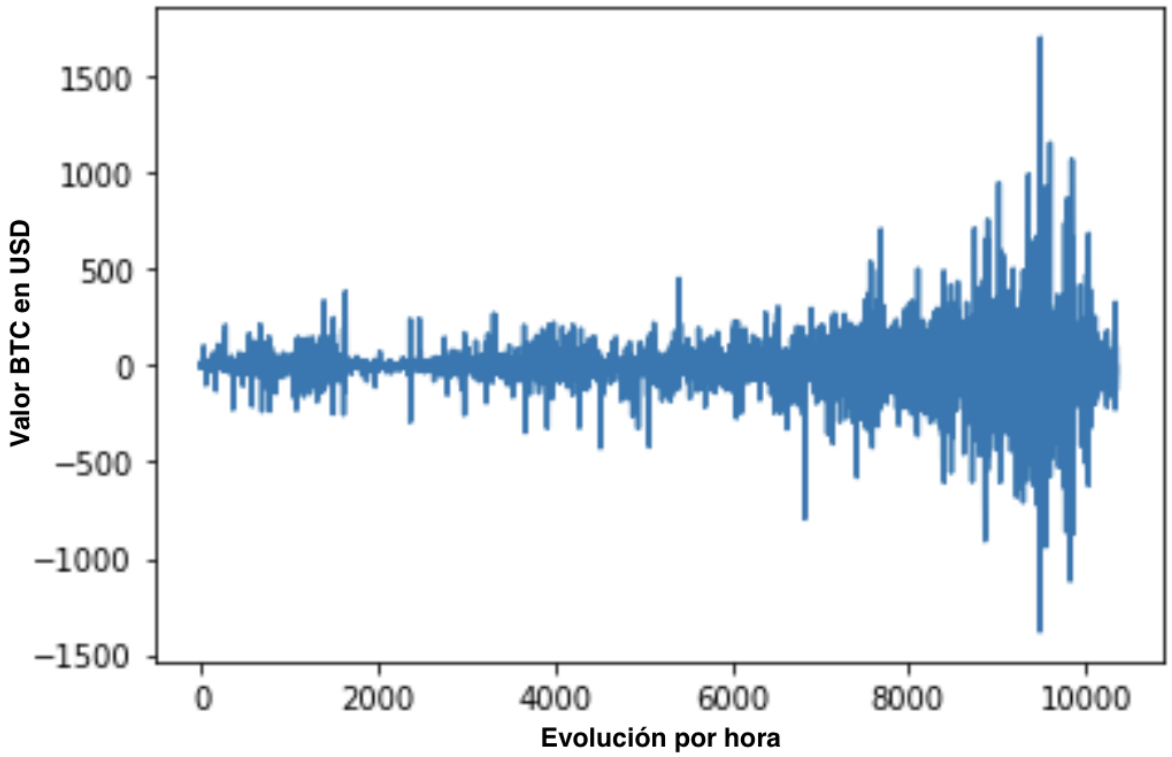
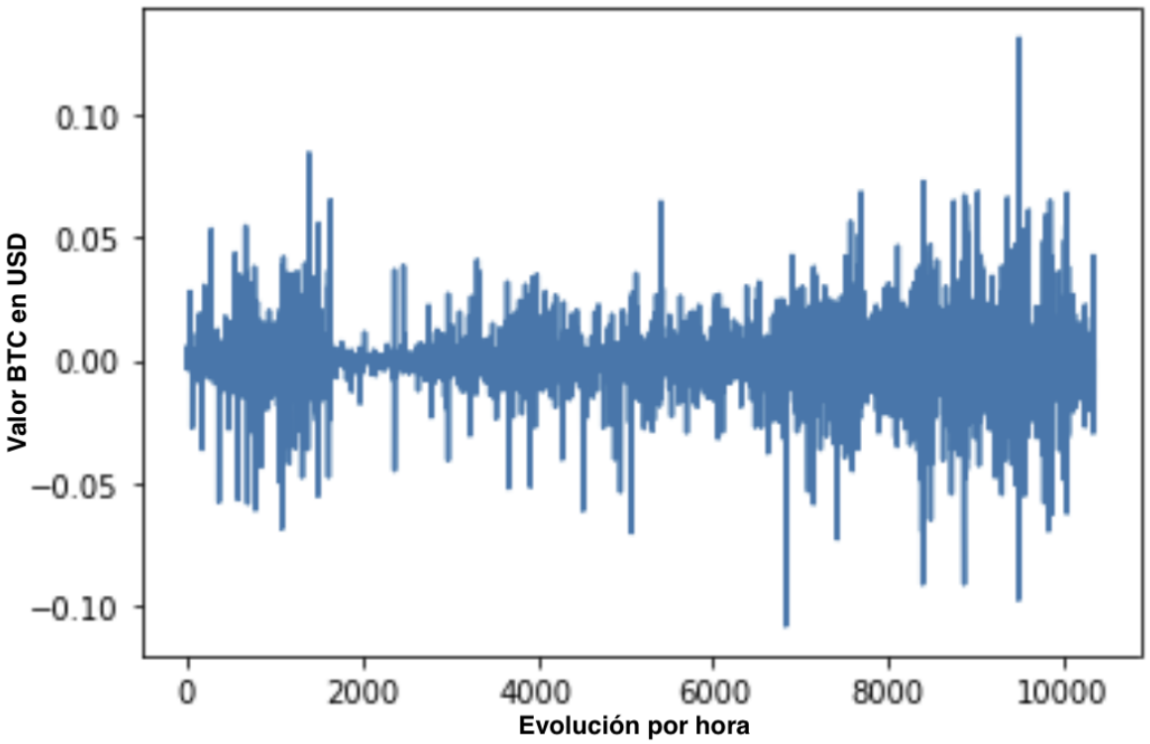
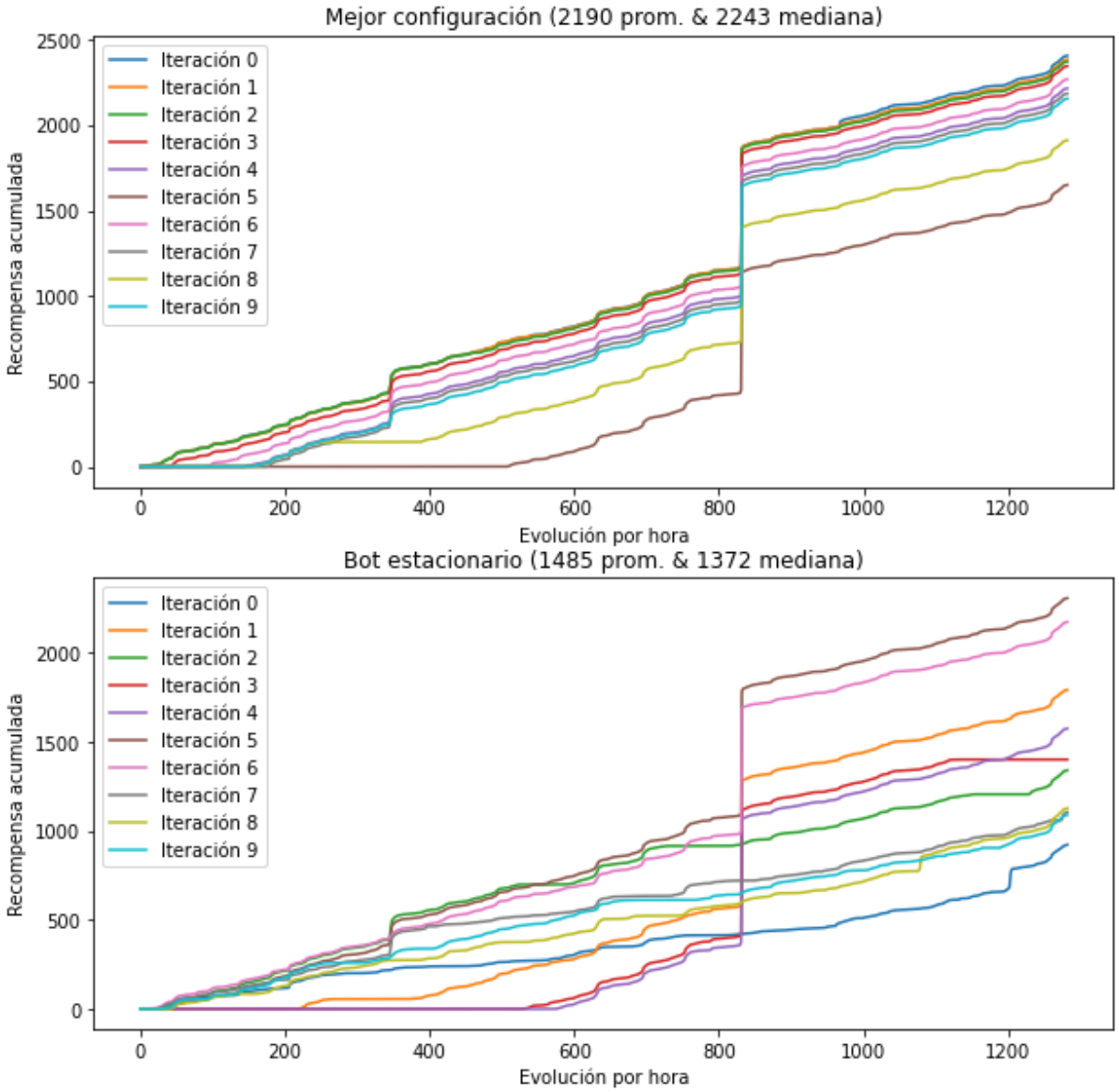


Figura 3.6.1.3: Variable "Close" en datos de entrenamiento estacionarios



Se corre nuevamente el test de Augmented Dickey–Fuller test (ADF) para confirmar que se llega a una serie sin tendencia y estacionalidad, lo cual deja como resultado una serie de tiempo estacionaria. Una serie de tiempo estacionaria, es aquella donde su esperanza y su varianza se mantienen constantes a lo largo del tiempo (Mushtaq, 2011). Con los nuevos resultados del test de hipótesis y un P-value igual a 0, se puede decir que con las transformaciones clásicas aplicadas, se consigue una serie de tiempo estacionaria (figura 3.6.1.3). Con estas transformaciones aplicadas a cada una de las variables en el conjunto de datos con el que se viene trabajando hasta el momento, se entrena un nuevo bot con diez iteraciones y se lo compara con el Bot conseguido en el capítulo anterior, calificado como el más performante o “La mejor configuración” (modelo nro 6).

Figura 3.6.1.4: Mejor configuración vs Bot estacionario

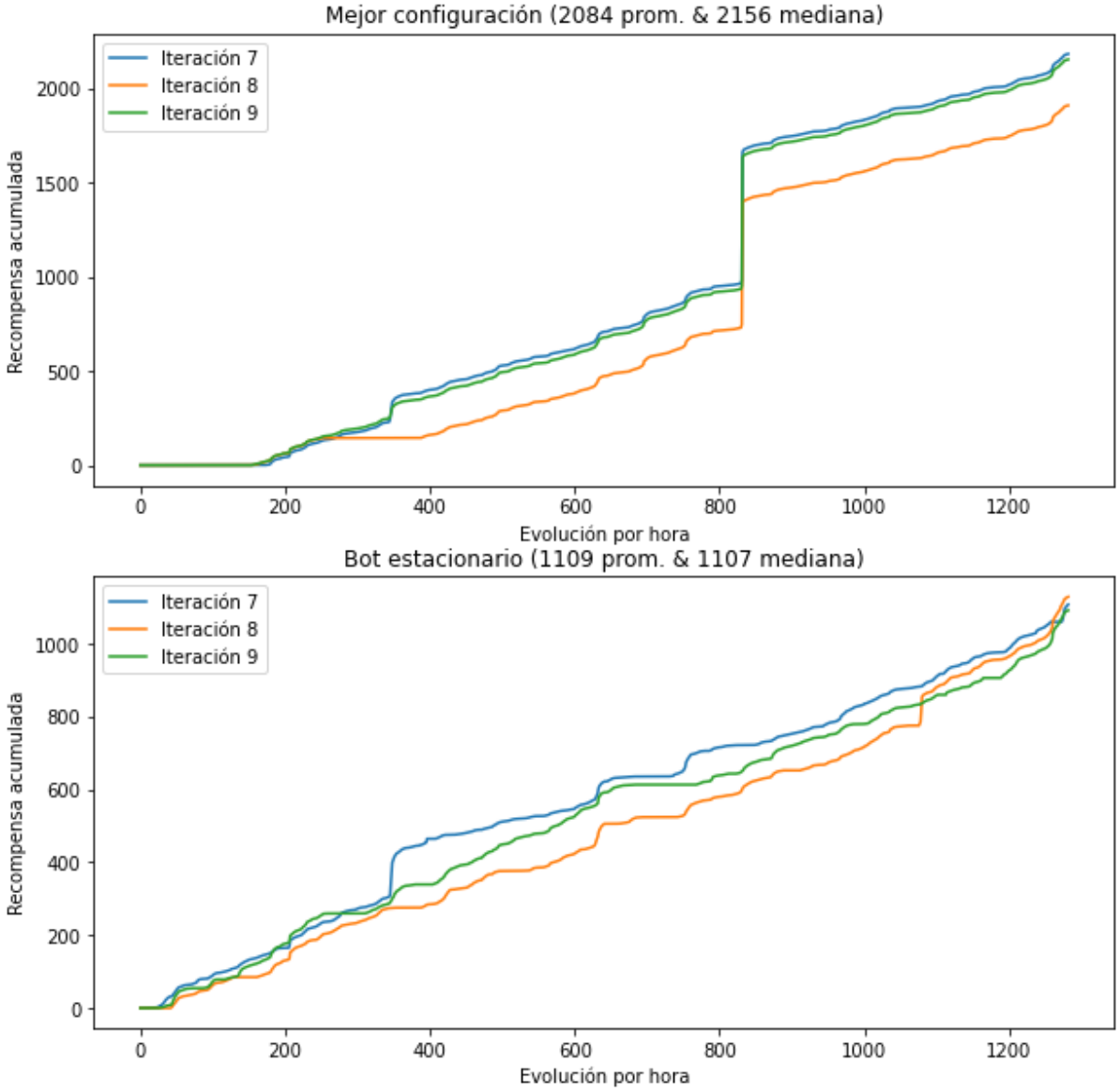


En el gráfico superior se observa el modelo con la mejor configuración y en el gráfico inferior el modelo estacionario (Figura 3.6.1.4). Fácilmente, se puede observar dos cosas:

- Por lo general, de estas transformaciones se espera que sumen estabilidad. En términos de estabilidad, por lo menos comparando con las iteraciones del Bot entrenado sin datos estacionarios, no pareciera ser que el efecto obtenido aporte estabilidad.

- Por otro lado, cuando se observa el promedio (1.485) y la mediana (1.372), si bien son similares, en definitiva son inferiores a los del modelo anterior. A continuación se pasa a observar las 3 mejores iteraciones de cada uno. (Figura 3.6.1.5).

Figura 3.6.1.5: Mejor configuración vs Bot estacionario - Mejores 3 iteraciones



Nuevamente en el gráfico superior se ven las iteraciones de la mejor configuración y en el inferior las iteraciones del bot estacionario (figura 3.6.1.5). En las últimas 3 iteraciones, se observa un Bot que se aproxima fuertemente a los supuestos iniciales, con los que se determinó cuándo un modelo tiene buena performance. Seguidamente, se analizarán en profundidad cada una de ellas:

- **Creciente positiva:** Las funciones de recompensa, dan una recompensa neta positiva en las 3 iteraciones. Además, al observar la evolución de la recompensa acumulada, se puede ver que se mantiene creciente. Es decir, que no obtiene recompensas negativas relevantes. Por lo tanto, este modelo está alcanzando el primer supuesto de performance.

- **Estable:** Si bien se sigue observando un pequeño salto relacionado a problemas con “catastrophic forgetting”, que se pueden ver entre el paso 300 y 400, se nota un suavizado mayor y crecimiento constante a lo largo de toda la curva. Pareciera ser que, trabajar con los datos estacionarios, ayudan fuertemente a corregir los problemas de “reward shape”, ya que no se observan saltos de recompensa relevantes de un paso inmediatamente al otro. Por lo tanto, también se puede decir, que el modelo se aproxima mucho a este supuesto, por lo menos mucho más que cualquiera de los anteriores.

- **Promedio y mediana:** En términos del supuesto sobre promedio y mediana, ambos son positivos y similares al comparar las últimas 3 iteraciones. Por lo que, también se estaría alcanzado la misma. No obstante, en este caso, el modelo performa tanto en promedio de recompensa neta (1.109), como en mediana (1.107) de las últimas 3 iteraciones, casi a la mitad de lo que performa el modelo anterior.

Tomando las observaciones realizadas sobre los tres supuestos, se puede decir, que se consiguió un modelo que se aproxima fuertemente a todos ellos. Esto último, confirma que el modelo debería tener una buena performance en general, según los supuestos establecidos al principio de este trabajo. Por lo anterior, se esperaría que consecuentemente, si este modelo performa sobre datos nunca antes vistos (datos de test) de la misma manera, entonces sea factible al operar.

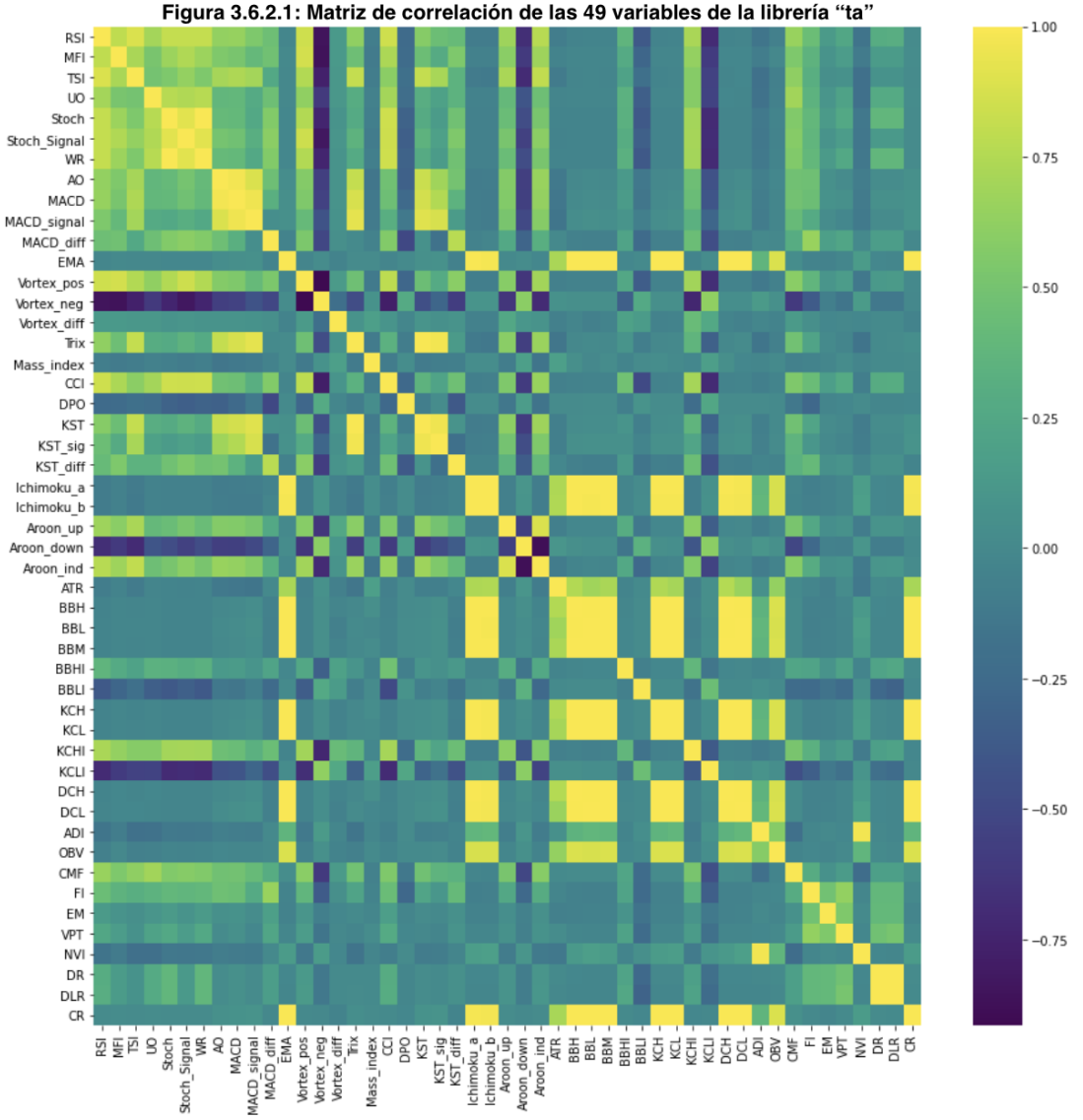
3.6.2 Análisis técnico - ta

Esta sección, se centra en el análisis técnico, el cual es mayormente utilizado dentro del mundo de las criptomonedas, a falta de información intrínseca sobre el mercado general para realizar análisis fundamental. Para realizar análisis técnico, se mencionó que los traders humanos suelen observar gráficos y distinto tipo de métricas, las cuales proveen un panorama general de la volatilidad y cambios del precio de un activo. Todas estas métricas son calculadas a partir de los datos que componen el conjunto de datos principal: Apertura, Cierre, Máximos, Mínimos y Volumen. Específicamente en esta sección, se trabaja con la librería *ta*[24], la cual fue desarrollada con el fin de facilitar el cálculo de todas estas métricas. La librería “ta” contiene funciones, que a partir de las variables mencionadas anteriormente,

[24] https://technical-analysis-library-in-python.readthedocs.io/en/latest/index.html

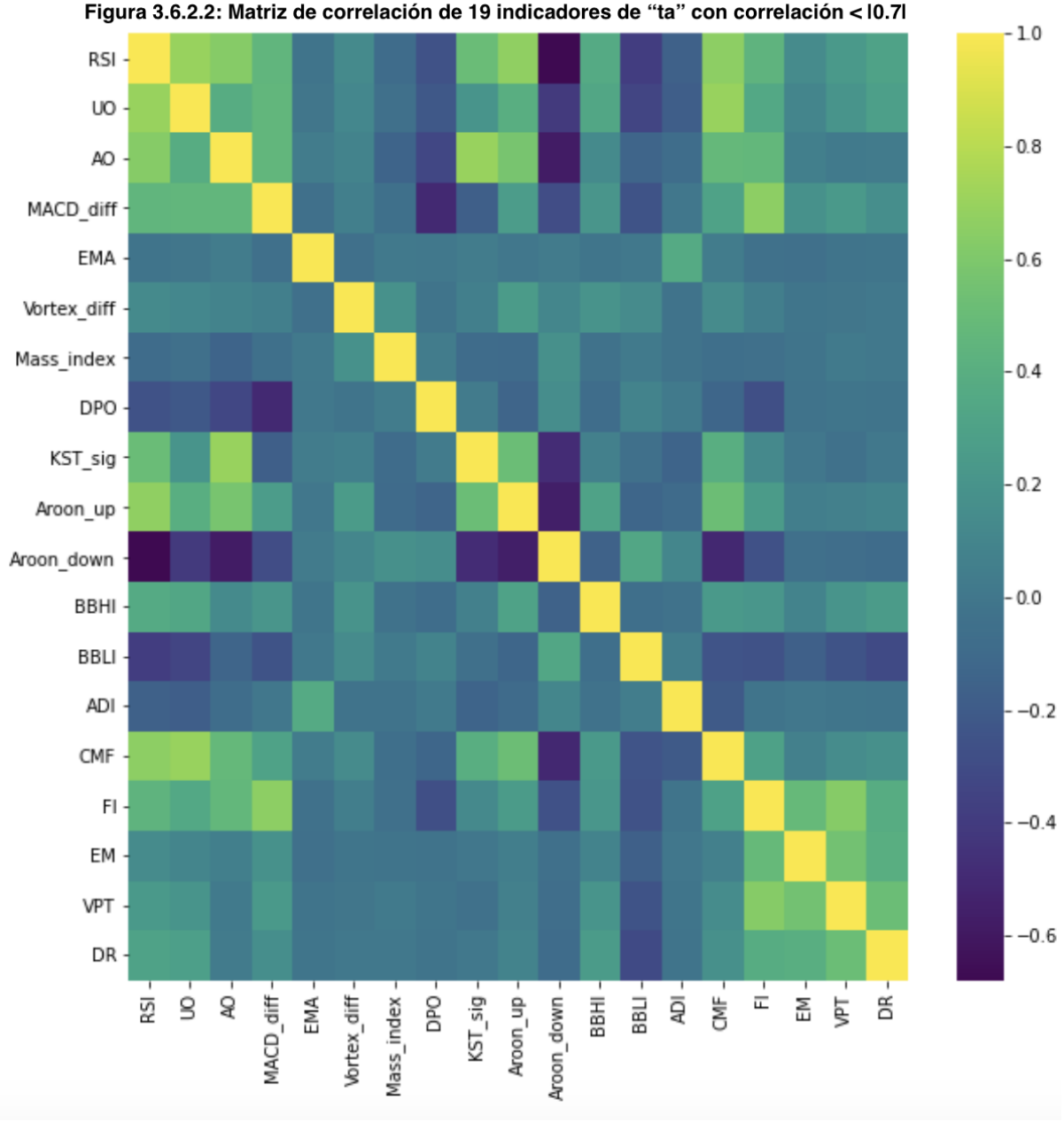
generan nuevos indicadores técnicos. A continuación, se utilizan las principales funciones de la librería *ta*, consiguiendo un total de 49 variables nuevas usando los datos de entrenamiento. Luego, se realiza una matriz de correlaciones, con el objetivo de entender el comportamiento de las mismas con respecto al resto (figura 3.6.2.1).

Figura 3.6.2.1: Matriz de correlación de las 49 variables de la librería "ta"



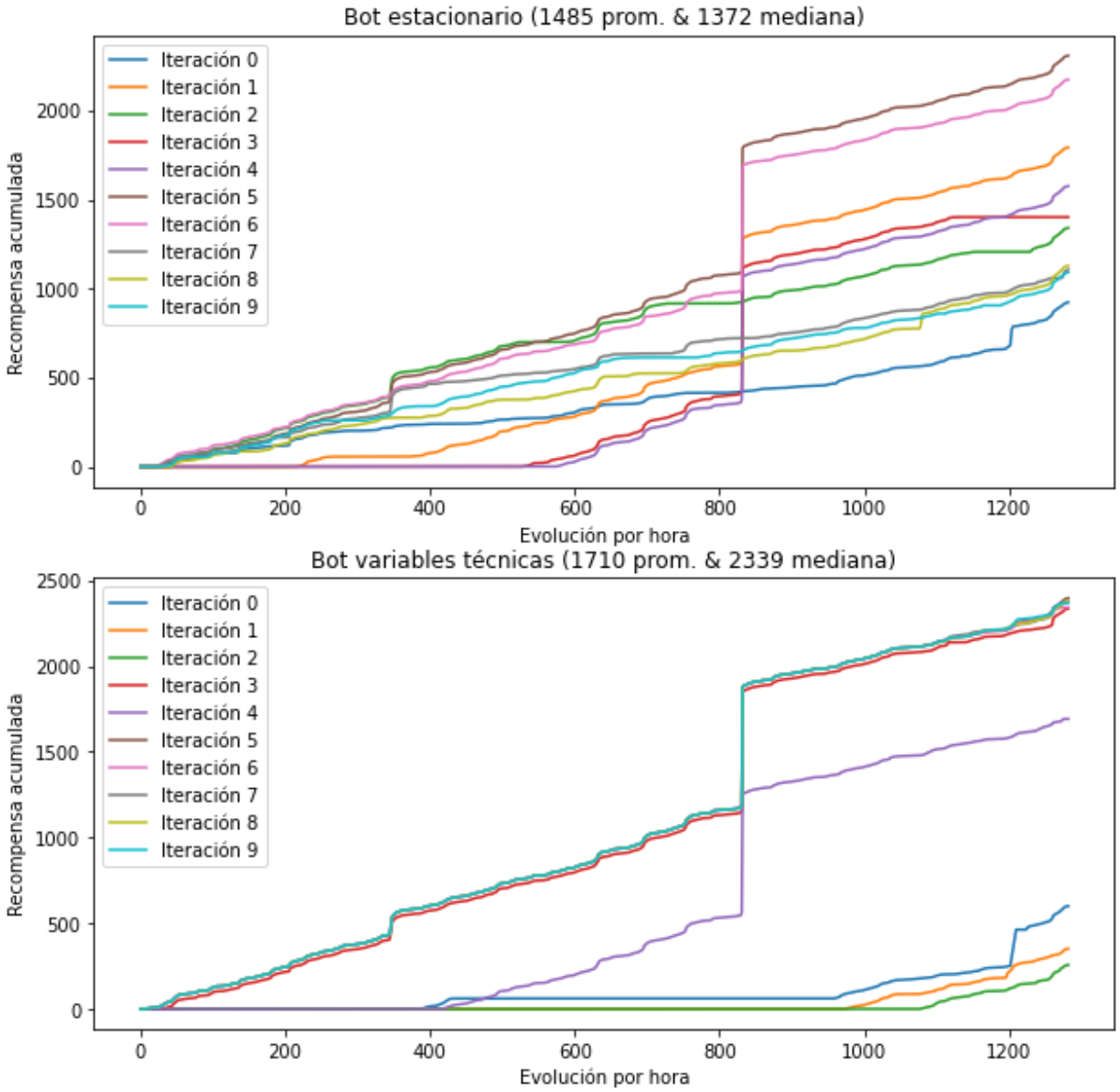
Al observar la matriz de correlación anterior, es posible apreciar que existen una cantidad significativa de variables altamente correlacionadas entre sí, tanto de forma positiva como de forma negativa. Una alternativa, es trabajar con la menor cantidad de variables que aporten la mayor cantidad de información a los modelos. Para ello, se determinó trabajar con un conjunto de variables que tengan un valor absoluto de correlación máximo de 0.7. Esto es, porque a partir de una correlación de |0.7|, se puede decir que las variables están altamente correlacionadas (Ratner, 2009). Como resultado, se obtiene el siguiente conjunto de variables (Anexo I): un total de 19 indicadores, donde la correlación entre cada una de ellas es menor a |0.7|.

Figura 3.6.2.2: Matriz de correlación de 19 indicadores de "ta" con correlación < |0.7|



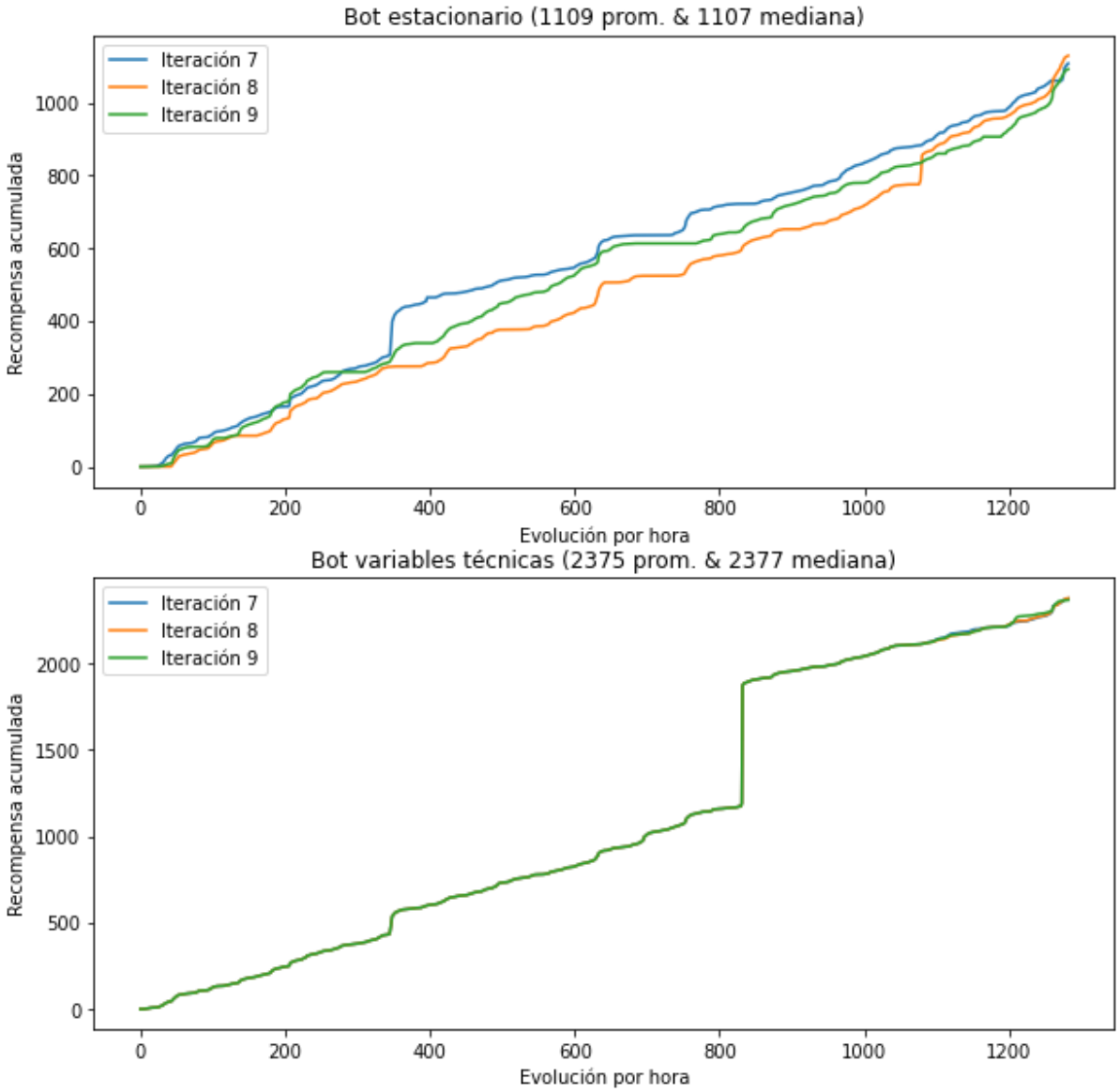
Volviendo a los modelos, se entrenan diez iteraciones del bot con la mejor configuración, utilizando el conjunto de datos con las transformaciones estacionarias ya aplicadas, sumando el conjunto de variables que no están altamente correlacionadas entre sí, 19 en total (Anexo I).

Figura 3.6.2.3: Bot estacionario vs Bot variables técnicas



En el gráfico inferior, se pueden observar las diez iteraciones del modelo que incorpora las 19 variables nuevas de análisis técnico (Figura 3.6.2.3). A simple vista, pareciera ser que se termina retrocediendo más que avanzando en términos de performance, ya que se pueden ver por lo menos 3 iteraciones que vuelven a elegir una estrategia de no comprar BTC hasta el paso 1.000, y de esa manera mantener la estabilidad. Se pierde el supuesto de *estabilidad* nuevamente al cruzar el paso 800. Se presentan a continuación las 3 mejores iteraciones (Figura 3.6.2.4).

Figura 3.6.2.4: Bot estacionario vs Bot variables técnicas - Mejores 3 iteraciones



Se observa un patrón muy similar al modelo base sin datos estacionarios de la sección anterior, con una performance en términos de promedio y mediana también muy similar. Esto se puede deber a que, como las variables que se utilizan en el análisis técnico parten de ser calculadas y creadas de las variables que ya estaban siendo utilizadas en el conjunto de datos, es muy probable que la información que aportan ya esté siendo considerada por la política de deep learning que están usando los agentes durante el entrenamiento. En breves palabras, el conjunto de variables seleccionadas de la librería "ta", no están aportando información significativamente nueva y relevante al modelo.

3.6.3 Análisis fundamental - Análisis de sentimientos

Como se mencionaba al principio, detrás del objetivo principal de este proyecto, que es conseguir entrenar un Bot de Deep Reinforcement Learning que genere la mayor cantidad de ganancia posible, se encuentra identificar la estrategia de como llegar a dicho punto. También, se mencionaba sobre los distintos análisis que utilizaban los traders humanos en su trabajo diario: el análisis técnico y el análisis fundamental. Hasta el momento, se trabajó sobre la configuración del ambiente, de los agentes bajo estudio y sobre el conjunto de datos, agregando variables y transformaciones que aportan cierto valor e información. Todo esto, se enmarca en lo que se conoce como la parte técnica de la labor del trader humano.

En esta sección, se prueba la hipótesis relacionada al aporte de información fundamental. Consiste en extraer datos de las redes sociales, en este caso Twitter, y transformarla de tal manera que sirva para aportar información del tipo fundamental al Bot, para que de esta forma pueda performar aún mejor, en base a datos con un aporte de información diferente. Estas transformaciones hacen referencia al análisis de sentimientos. El análisis de sentimientos, es una técnica proveniente del campo de estudio del Procesamiento de Lenguaje Natural, el cual estudia la sintaxis y semántica del texto con variados objetivos. Para extraer información fundamental de la red social Twitter, será necesario analizar lo que se conoce como tweets, que son el tipo de publicaciones o mensajes que se comparten en la plataforma y que pueden contener texto plano, links a otros sitios, imágenes, emoticones, etc. Los tweets, son analizados con un clasificador de sentimientos. En específico, se utiliza el modelo VADER[25] (Gilbert, 2013), desarrollado para hacer este tipo de análisis particularmente sobre redes sociales.

El proceso consiste en lo siguiente: se descarga un tweet de la plataforma, este pasa por el clasificador de sentimientos VADER, y luego se obtiene como resultado la polaridad de cada sentimiento, seguido por un score final que es el promedio de las polaridades:

- Polaridad Positiva
- Polaridad Negativa
- Polaridad Neutra
- Score compuesto

[25] Este clasificador es el resultado de un trabajo de investigación del MIT y considerado “Open Source”, por lo que es posible encontrarlo en GitHub con su código disponible.

De estas cuatro variables se calcula:

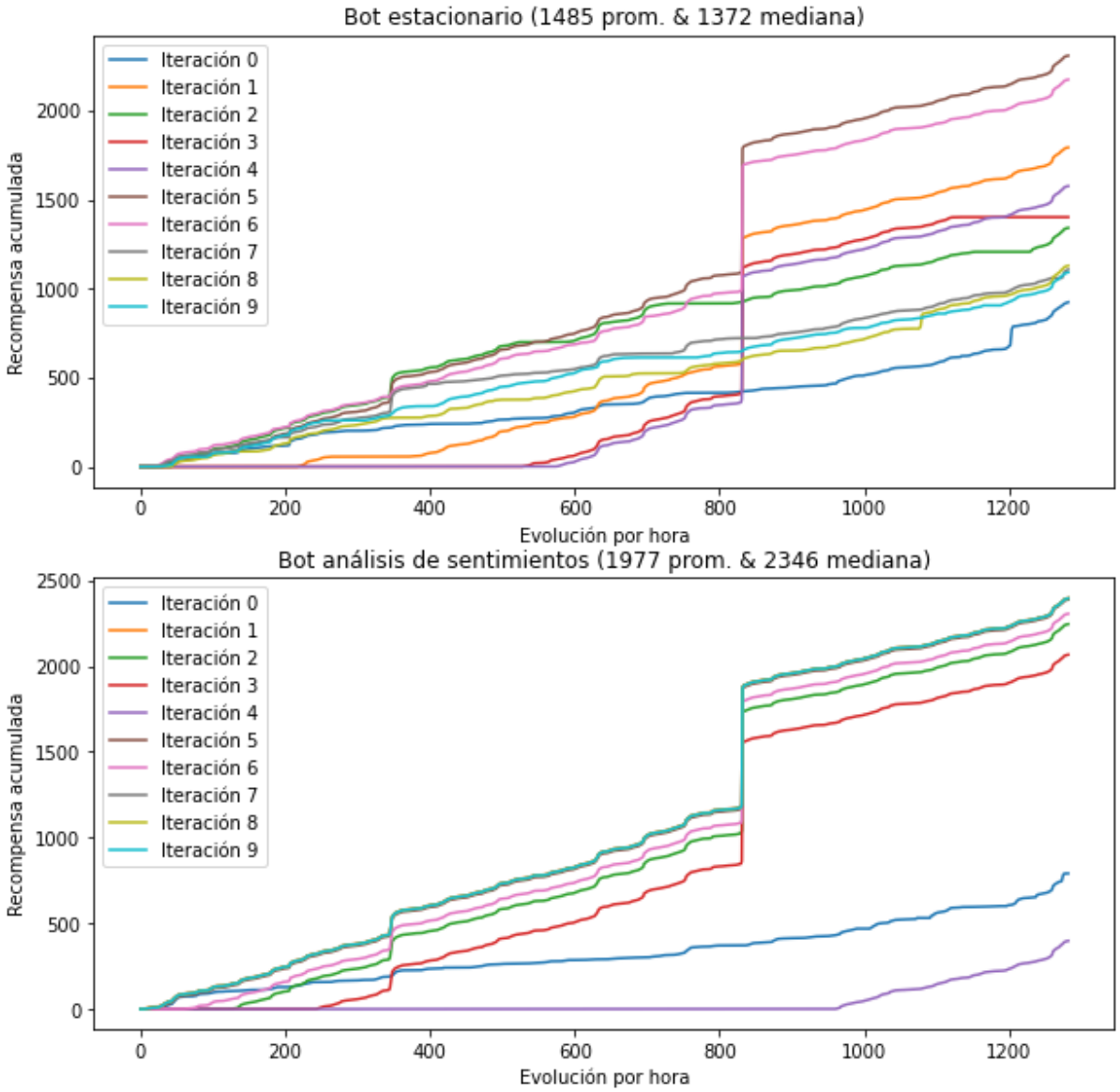
- Cantidad de tweets con sentimiento positivo (mayor polaridad positiva) en una hora determinada.
- Cantidad de tweets con sentimiento negativo (mayor polaridad negativa) en una hora determinada.
- Cantidad de tweets con sentimiento neutro (mayor polaridad neutro) en una hora determinada.
- Polaridad positiva promedio de tweets en una hora determinada.
- Polaridad negativa promedio de tweets en una hora determinada.
- El score compuesto resultante de promediar toda la polaridad en una hora determinada.

Además, se agregan algunas variables adicionales:

- Cantidad de tweets con links de noticias en una hora determinada.
- Cantidad de tweets publicados por bots en una hora determinada.
- Cantidad total de tweets en una hora determinada.

Se analizan todos los tweets que contengan la palabra “Bitcoin” durante el periodo de estudio. Esto suma una cantidad total de 17.7 millones de tweets. Estos se encuentran en idioma inglés. Para simplificar el desarrollo de todo este proceso, se utilizó un conjunto de datos ya procesados y trabajados exactamente de la manera descrita. El mismo fue desarrollado y utilizado en la investigación de Jaime Badiola Ramos (Badiola Ramos, 2019). Se cruza este conjunto de datos por el campo “Date” con el conjunto de datos original con el que se viene trabajando. Consecuentemente, se vuelve a entrenar diez iteraciones del Bot con la mejor configuración y las transformaciones estacionarias, y se suman las variables que aportan la información fundamental. (Figura 3.6.3.1).

Figura 3.6.3.1: Bot estacionario vs Bot análisis de sentimientos



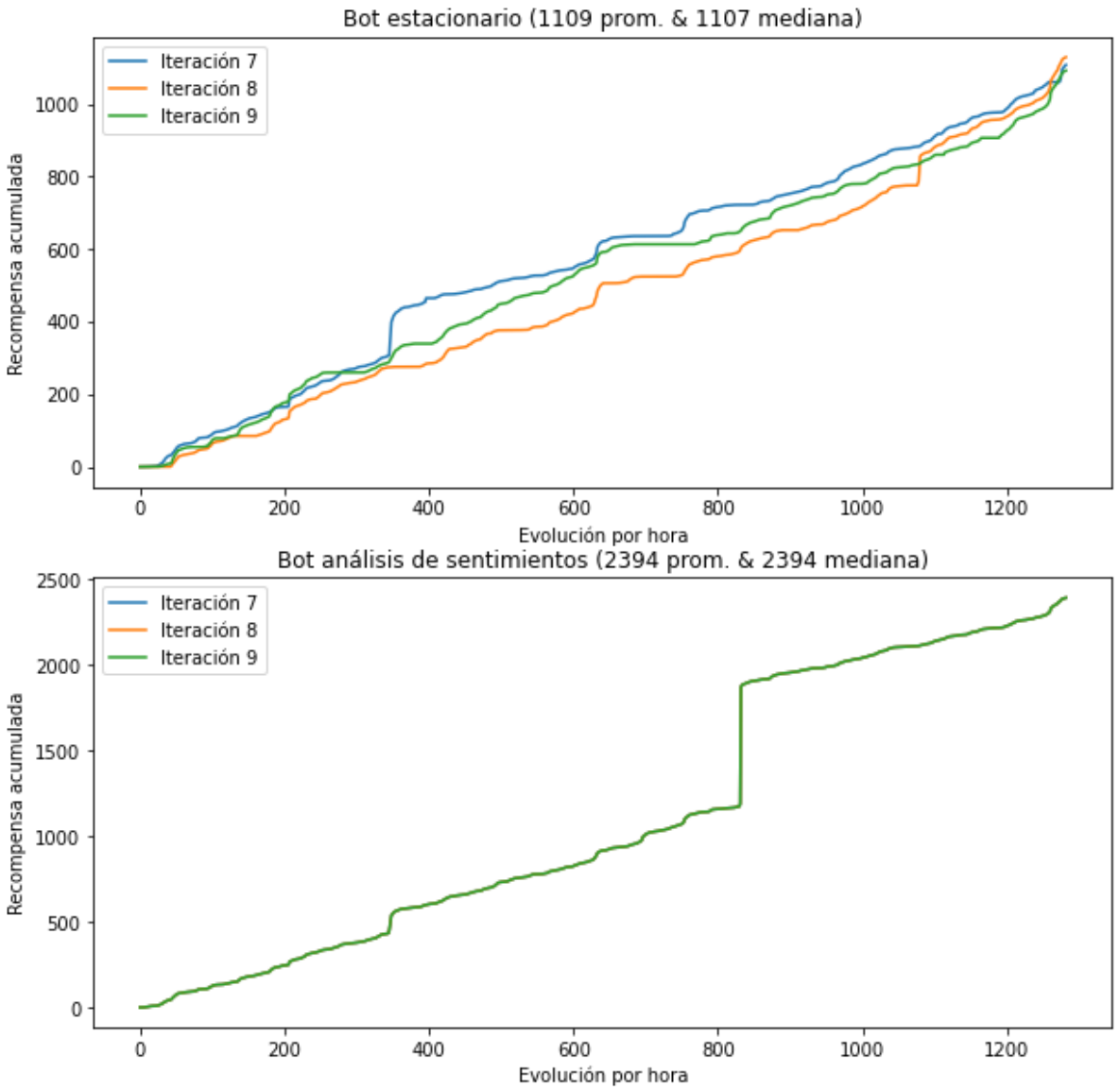
Finalmente, se puede observar lo siguiente:

- Se ven por lo menos 4 iteraciones que superan a la iteración con mayor recompensa neta acumulada del modelo anterior.

- A diferencia de cuando se agregaron las variables de “ta”, se tiene algunas iteraciones más estables, las cuales no son la mayoría. Por lo tanto, se vuelve a perder el supuesto de estabilidad.

A continuación, se muestran las últimas 3 iteraciones de cada modelo, en el cual se puede observar mejor la diferencia entre cada uno de ellos. (Figura 3.6.3.2).

Figura 3.6.3.2: Bot estacionario vs Bot análisis de sentimientos - Mejores 3 iteraciones



Al parecer, las últimas 3 iteraciones del modelo que utiliza las variables que pretenden dar información fundamental, convergen exactamente igual en los datos de validación. Se analizarán los supuestos de performance:

- **Creciente positiva:** La recompensa neta es creciente positiva en todas las iteraciones y no se aprecian mesetas o caídas por castigos de recompensa. Esto indica que se está ante un modelo que pareciera no estar sobre-entrenado en estas últimas iteraciones. Por lo tanto, este supuesto se cumple.

- **Estable:** Al agregar las variables de sentimientos, nuevamente se pierde la estabilidad luego del paso 300 y del paso 800, donde se presentan problemas de *reward shape*. Por lo tanto, este supuesto no se cumple.

- **Promedio y mediana:** Se obtiene un promedio de 2.393 y mediana de 2.393. Se consigue valores mayores en términos de promedio y mediana, de lo que se observaba trabajando con el modelo que contiene “la mejor configuración” y únicamente variables técnicas.

Dadas estas observaciones, añadiendo las variables fundamentales, se podría pensar que se arriba a un modelo que generaliza bien, que presenta cierta inestabilidad y que alcanza buenos promedios y medianas en sus mejores iteraciones con respecto a los resultados obtenidos hasta el momento.

3.7 Evaluación de performance y factibilidad

3.7.1 Pre-selección de modelos por performance

En esta sección se busca recapitular, resumir y comparar la performance de los modelos desarrollados hasta el momento sobre las mejores iteraciones de cada uno en general. Luego, en base a los supuestos establecidos al inicio para medir la performance de los modelos, se selecciona la mejor iteración de aquellos modelos que más se aproximan a cumplirlas. Con estas iteraciones, se mide su performance y factibilidad sobre datos nunca antes vistos (conjunto de datos de test). A continuación, se muestra una tabla que resume y compara los resultados de performance de los modelos desarrollados. Luego, se realizan observaciones sobre cada uno.

Figura 3.7.1.1: Tabla comparativa sobre los modelos preseleccionados

				Supuestos - 3 mejores iteraciones			
Nº	**Sección**	**Configuración del Modelo**	**Ingeniera de atributos**	**C.P**	**Est**	**Recompensa neta promedio**	**Recompensa neta mediana**
1	3.4	RL Bot benchmark	Sin Ing. de atributos	NO	NO	-2.510	-2.677
6	3.5.3	DRL_A2C_MlpLnLstm_Omega	Sin Ing. de atributos	SI	NO	2.190	2.243
7	3.6.1	DRL_A2C_MlpLnLstm_Omega	Datos Estacionarios	SI	SI	1.485	1.372
8	3.6.2	DRL_A2C_MlpLnLstm_Omega	Datos Est. + var ta	SI	NO	1.710	2.339
9	3.6.3	DRL_A2C_MlpLnLstm_Omega	Datos Est. + Análisis de Sentimientos	SI	NO	1.977	2.346

- El modelo nroº 1, es el modelo benchmark básico. Este respeta la configuración original y no cumple con ninguno de los supuestos realizados al comienzo de este trabajo. Es decir, su recompensa neta en datos de validación, no es “Creciente Positiva” ni tampoco es “Estable”. No tiene una recompensa neta promedio y mediana positivas y similares en ninguna de las mejores iteraciones. Por lo tanto, se esperaría que este modelo no performe bien en test, y en consecuencia, no sea factible. La mejor iteración de este modelo es la 9na.

- Al anterior, le sigue el modelo nroº 6, el modelo benchmark con los ajustes de configuración al que se lo llamó “la mejor combinación”. Cumple con el primer supuesto, viola el supuesto de estabilidad, y su promedio y mediana si bien son similares, no son de los más altos obtenidos. La mejor iteración en este caso es la 7ma.

- El modelo nroº 7, es el que mayormente se acerca a cumplir con los supuestos establecidos al comienzo. Es el único que alcanza el supuesto de estabilidad. La desventaja que presenta es que su promedio y mediana no son las más altas. Por lo tanto, se define que es un modelo que performa lo suficientemente bien como para esperar que sea factible. La mejor iteración en este caso es la 7ma.

- Cuando se trabajó sobre el modelo nroº 8, se concluyó que probablemente las nuevas variables no estarían agregando nueva información, sino más bien ruido. Además, no cumple con los supuestos de “Estable”, ni tampoco presenta los mejores promedios y mediana. La mejor iteración en este caso es la 9na.

- Por último, el modelo nroº 9, al cual se le incluyeron las variables de análisis de sentimiento que pretenden aportar información fundamental, es el que presenta el mejor promedio y mediana, ya que son los más elevados y similares simultáneamente. También cumple el supuesto de “Creciente Positivo”, pero presenta ciertos momentos en los que viola el supuesto de estabilidad. Tiene un patrón muy similar al modelo nroº8. La mejor iteración en este caso es la 9na.

Por lo expuesto anteriormente, se pre-seleccionan los siguientes modelos e iteraciones, de los cuales se espera encontrar la mejor performance en datos de testeo, para luego medir su factibilidad y compararlo con el Bot benchmark y las estrategias baseline:

- Modelo nro 1, 9na iteración: Bot benchmark.
- Modelo nro 8, 7ma iteración: Único modelo que consiguió el supuesto de estabilidad.
- Modelo nro 9, 9na iteración: Modelo con variables de análisis de sentimientos y mayor recompensa promedio y mediana.

3.7.2 Medición de performance y factibilidad

En esta sección, se evalúa la mejor iteración de cada uno de los modelos preseleccionados anteriormente, por ser los que más se acercan a cumplir los supuestos iniciales. Primero, se evalúa su performance en datos de testeo nunca antes vistos por los modelos, y se los compara con la performance alcanzada en los datos de validación presentados en secciones pasadas teniendo en cuenta los supuestos de performance. Luego, se realiza un análisis de factibilidad, también basado en las definiciones de factibilidad establecidas al inicio del proyecto sobre los datos de testeo. Por último, se realiza una comparación, teniendo en cuenta todas estas métricas con respecto a los resultados obtenidos, y se realizan las observaciones pertinentes que dan pié a las conclusiones.

3.7.2.1 Modelo nroº1, 9na iteración: Bot Benchmark y estrategias baseline

Para facilitar la comprensión del análisis de factibilidad, se comienza analizando directamente la factibilidad de las estrategias baselines, las cuales carecen de métricas de performance por no ser modelos de Reinforcement Learning. A continuación, se observa cómo resulta la factibilidad sobre los datos de test (Figura 3.7.2.1.1).

Figura 3.7.2.1.1: Estrategias baseline operando en datos de test

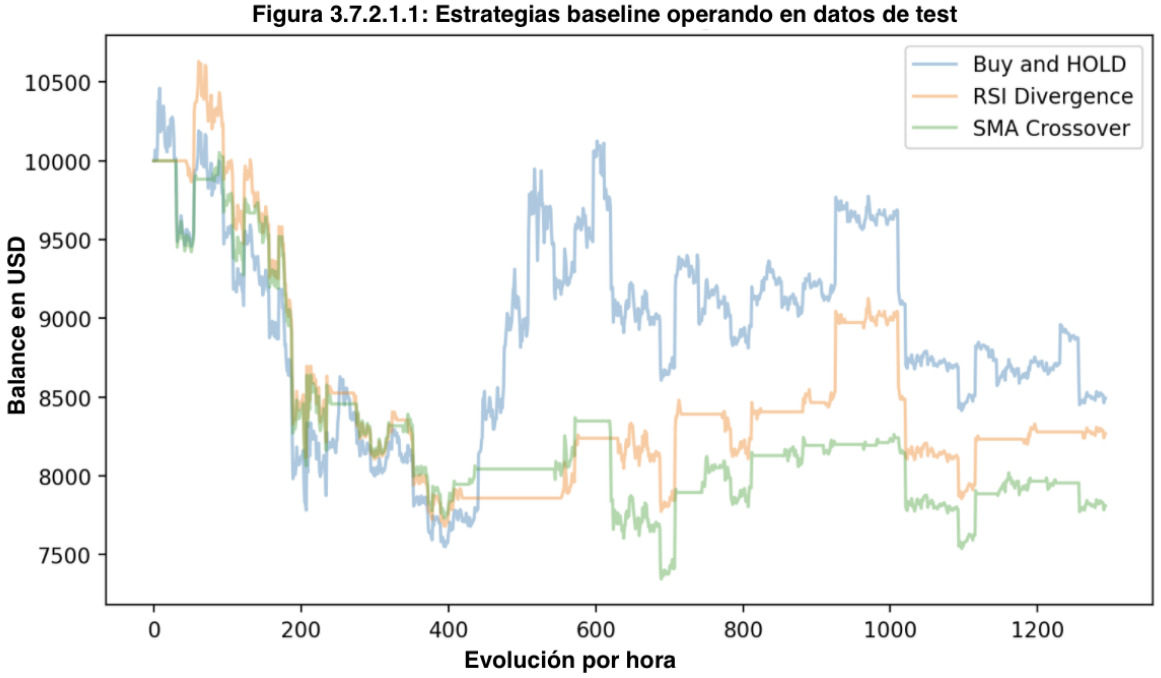
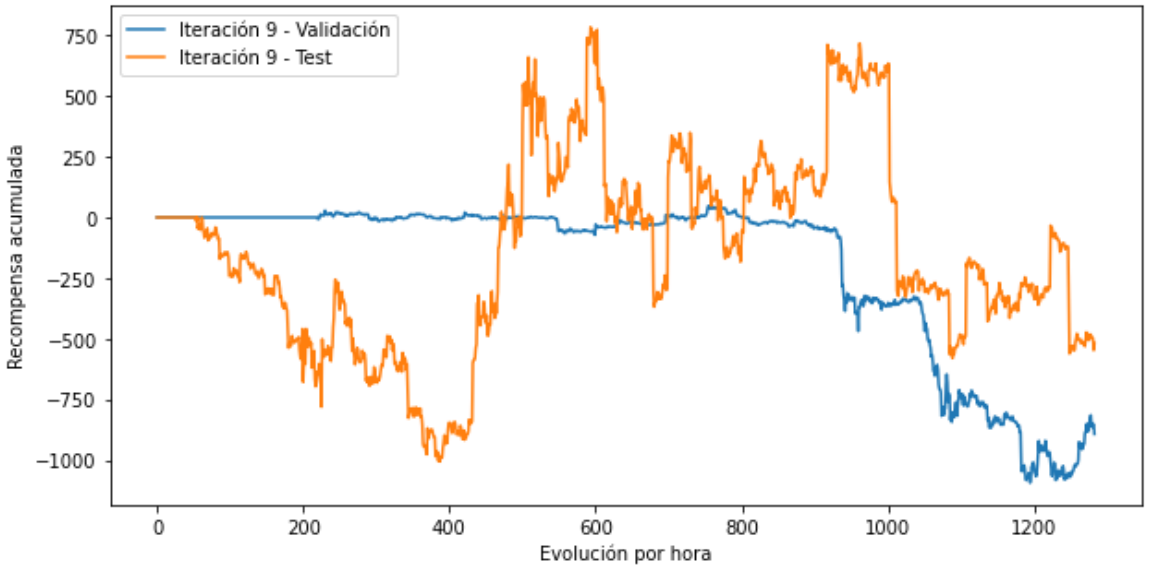


Figura 3.7.2.1.2: Análisis de factibilidad de estrategias baseline

	Análisis de Factibilidad - Datos de test		
Métrica	**Buy & Hold**	**SMA Crossover**	**RSI Divergence**
# Pasos rentables	53	37	87
# Pasos NO rentables	1.240	1.256	1.206
% Pasos rentables	4.10%	2.86%	6.73%
% Volumen de ganancias	0.5%	0.01%	0.77%
Volumen prom. ganancias	132 usd	5 usd	175 usd
Volumen prom. pérdidas	1.125 usd	1.780 usd	1.629 usd
Factibilidad final	-1.073 usd	-1.729 usd	-1.508 usd

En la figura 3.7.2.1.2, se puede apreciar que la factibilidad de Buy & Hold supera la del resto de las estrategias. Sin embargo, ninguna termina siendo factible, ya que su resultado de factibilidad es negativo para los tres casos. Esto es así, principalmente porque el porcentaje de tiempo en el que las estrategias se mantienen rentables, ronda entre 4% y 7%. El resto del tiempo su rentabilidad se mantuvo negativa. Además, el volumen promedio de ganancias, es prácticamente despreciable comparado con el volumen promedio de pérdidas. Estos resultados son los que llevan a obtener una factibilidad final negativa.

Figura 3.7.2.1.3: Bot benchmark. - Validación vs Test - Iteración 9 (-892 prom. & -519 prom.)



Para el Bot benchmark, se procede a comparar la performance obtenida en datos de validación con datos de test. (Figura 3.7.2.1.3). En términos de performance, el modelo performa de una manera muy diferente sobre datos de validación contra datos de test. Al

observar los valores de recompensa promedio, el bot pareciera performar mejor sobre datos nunca antes vistos. Si se presta atención a las formas de las curvas (Figura 3.7.2.1.3), pareciera ser que la recompensa está totalmente alineada a la volatilidad del precio de BTC durante ese periodo. Este comportamiento se podría traducir como: el bot está siendo recompensado o castigado según suba o baje el precio del Bitcoin. En términos más técnicos, podría no estar aprendiendo la estrategia que se le pretendió enseñar, operando a la deriva de la volatilidad del activo. Este resultado era de esperarse, ya que no se cumple ninguno de los supuestos iniciales. Se presenta el Bot benchmark operando en datos de testeo nunca antes vistos (Figura 3.7.2.1.4), en conjunto con el análisis de factibilidad correspondiente. (Figura 3.7.2.1.5).

Figura 3.7.2.1.4: Balance Bot benchmark operando vs precio de BTC en USD - Datos de test

Figura 3.7.2.1.5: Análisis de Factibilidad - Test - Modelo Nº4

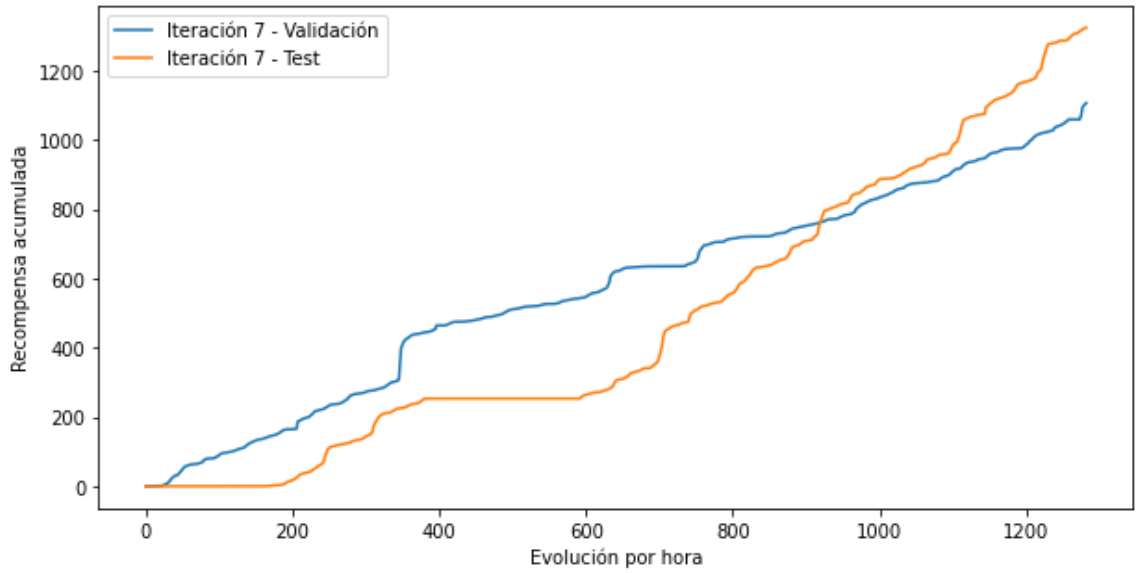
Métrica	Iteración 9
# Pasos rentables	630
# Pasos NO rentables	651
% Pasos rentables	49.18%
% Volumen de ganancias	32.58%
Volumen prom. ganancias	78 usd
Volumen prom. pérdidas	157 usd
Factibilidad final	-41 usd

Se observa en datos nunca antes vistos por el modelo, que los resultados obtenidos terminan dependiendo de cómo varía el precio del BTC. En la figura 3.7.2.1.4, se puede ver que del paso 0 hasta antes del paso 200, la iteración nro 9, se mantiene sin comprar durante la caída del precio del BTC, pareciendo dar indicios de una decisión inteligente, pero luego termina comprando durante la caída. Si bien, casi el 50% del tiempo se mantuvo en pasos rentables, el volumen de usd durante los pasos no rentables en promedio fueron mayores. Se termina con un valor de factibilidad negativo igual a -41 usd.

3.7.2.2 Modelo nroº8, 7ma iteración: Modelo con mayor estabilidad

Ahora se replica el mismo tipo de análisis para el modelo e iteración que, según los supuestos propuestos, fue el que más se acercó a la exigencia de estabilidad. Se compara la performance sobre los datos de validación y test nunca antes vistos por el modelo. (Figura 3.7.2.2.1)

Figura 3.7.2.2.1: Bot estacionario - Validación vs Test - Iteración 7 (1107 prom. & 1325 prom.)



En color azul, se presenta la curva de performance del modelo durante el período de validación, y en naranja durante el período de test. A diferencia del Bot benchmark, pareciera que el comportamiento de la mejor iteración se mantiene en datos nunca antes vistos. Esto estaría dando indicios de que el modelo no estaría sobreentrenado como el Bot benchmark, ya que se comporta de una manera similar ante datos diferentes, por lo que se podría pensar que generaliza bien. Se recuerda que además, este es el modelo que más se acercó a cumplir con los 3 supuestos de performance, por lo que en definitiva es el modelo que se esperaba performe mejor. En otras palabras, que mantenga su comportamiento ante datos nunca antes vistos. Esta hipótesis se estaría confirmando al observar la figura 3.7.2.2.1. Esto último, no garantiza que el Bot genere ganancias de dinero (sea factible), sino que de alguna manera, se puede decir que aprendió a comportarse siguiendo la estrategia que se le enseñó (performance). A continuación, se puede observar cómo opera sobre los datos de test. (Figura 3.7.2.2.2).

Figura 3.7.2.2.2: Balance Bot estable operando vs precio de BTC en USD - Datos de test

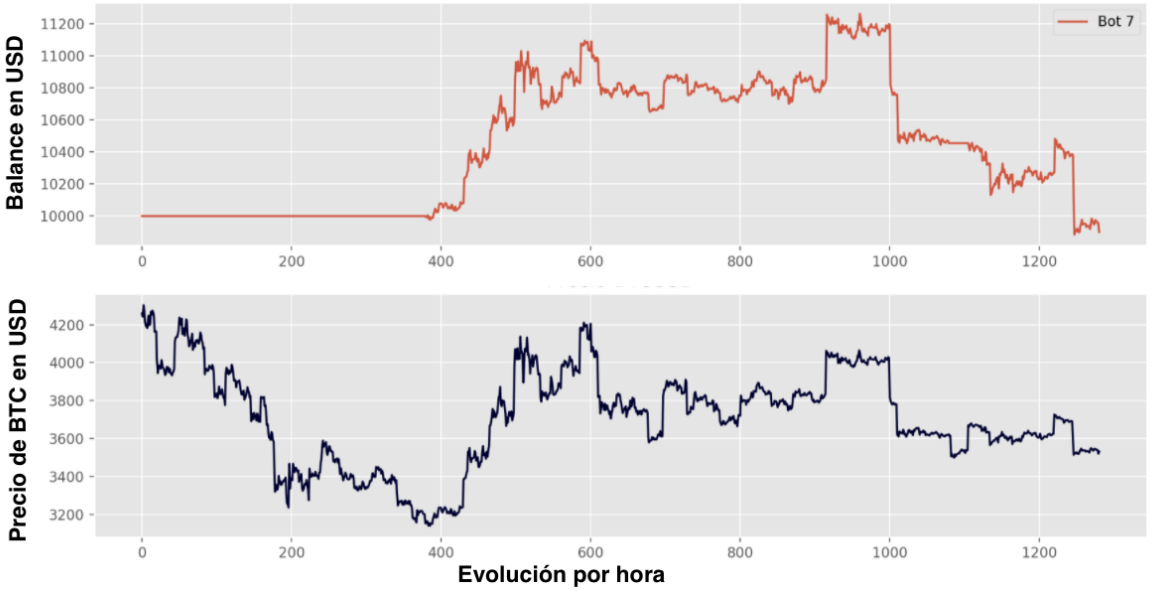


Figura 3.7.2.2.3: Análisis de Factibilidad - Test - Modelo Nº7

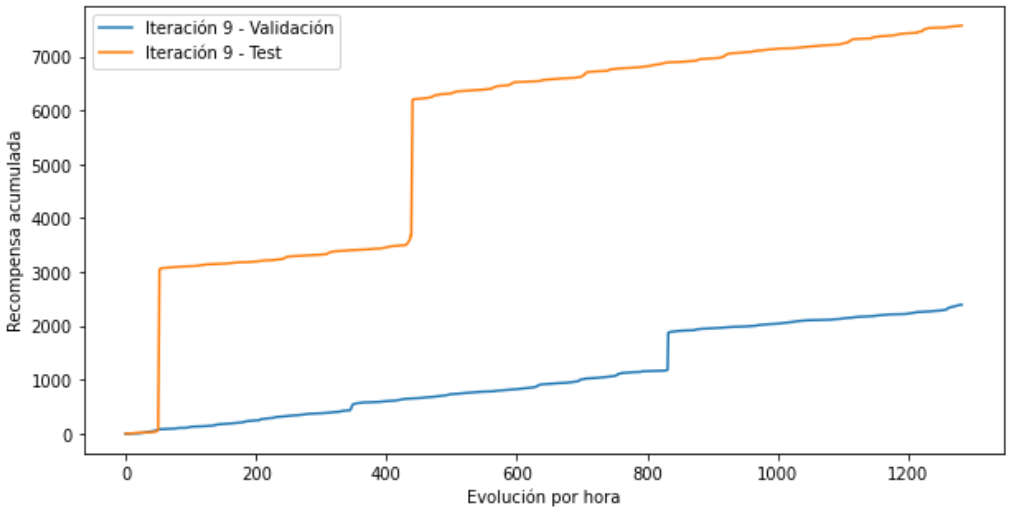
Métrica	Iteración 7
# Pasos rentables	1238
# Pasos NO rentables	43
% Pasos rentables	96.64%
% Volumen de ganancias	99.63%
Volumen prom. ganancias	460 usd
Volumen prom. pérdidas	50 usd
Factibilidad final	443 usd

Comenzando a analizar el comportamiento de este Bot (Figura 3.7.2.2.2), primero se puede observar que a diferencia del modelo Benchmark, el modelo nroº7 espera hasta el paso 400 para comenzar a operar, momento en el que deja de descender el precio del BTC y se dispara hacia arriba. Luego, desde el paso 600 hasta el paso 1.000, mantiene la rentabilidad. A partir del paso 1.000, hay una caída repentina del precio del BTC, la cual pareciera haber tomado por sorpresa al bot, donde se pierde una gran parte de la rentabilidad generada hasta ese momento. Sucede algo similar en la caída final del precio del Bitcoin. La iteración nroº7, se mantuvo rentable durante el 96.64% de los pasos que completan la ventana tiempo de test. Durante ese periodo, el volumen promedio de dólares ganados es 9 veces mayor al volumen promedio de dólares perdidos. Termina con un valor de factibilidad positivo de 443 usd, que en definitiva estaría indicando factibilidad. (Figura 3.7.2.2.3).

3.7.2.3 Modelo nroº9, 9na iteración: Modelo con datos de análisis de sentimientos

Para finalizar este capítulo, se presentan los resultados de performance de validación comparados con datos de test, para el modelo que utiliza datos de análisis de sentimientos.

Figura 3.7.2.3.1: Bot análisis de sentimientos - Validación vs Test - Iteración 9 (2394 prom. & 7575 prom.)



En color azul, se puede ver la novena iteración performando sobre datos de validación, y en naranja, el mismo Bot performando sobre los datos de test. (Figura 3.7.2.3.1). La diferencia entre este modelo con respecto a los dos anteriores, es que si bien cumple estrictamente con el supuesto de “Creciente Positiva”, y con los valores promedio y mediana de recompensa neta más altos, el modelo viola el supuesto de “Estabilidad”. Si bien el patrón de comportamiento es similar, la recompensa se dispara premiando de manera exagerada al Bot. A continuación, se observa cómo opera en el conjunto de datos de test. (Figura 3.7.2.3.2).

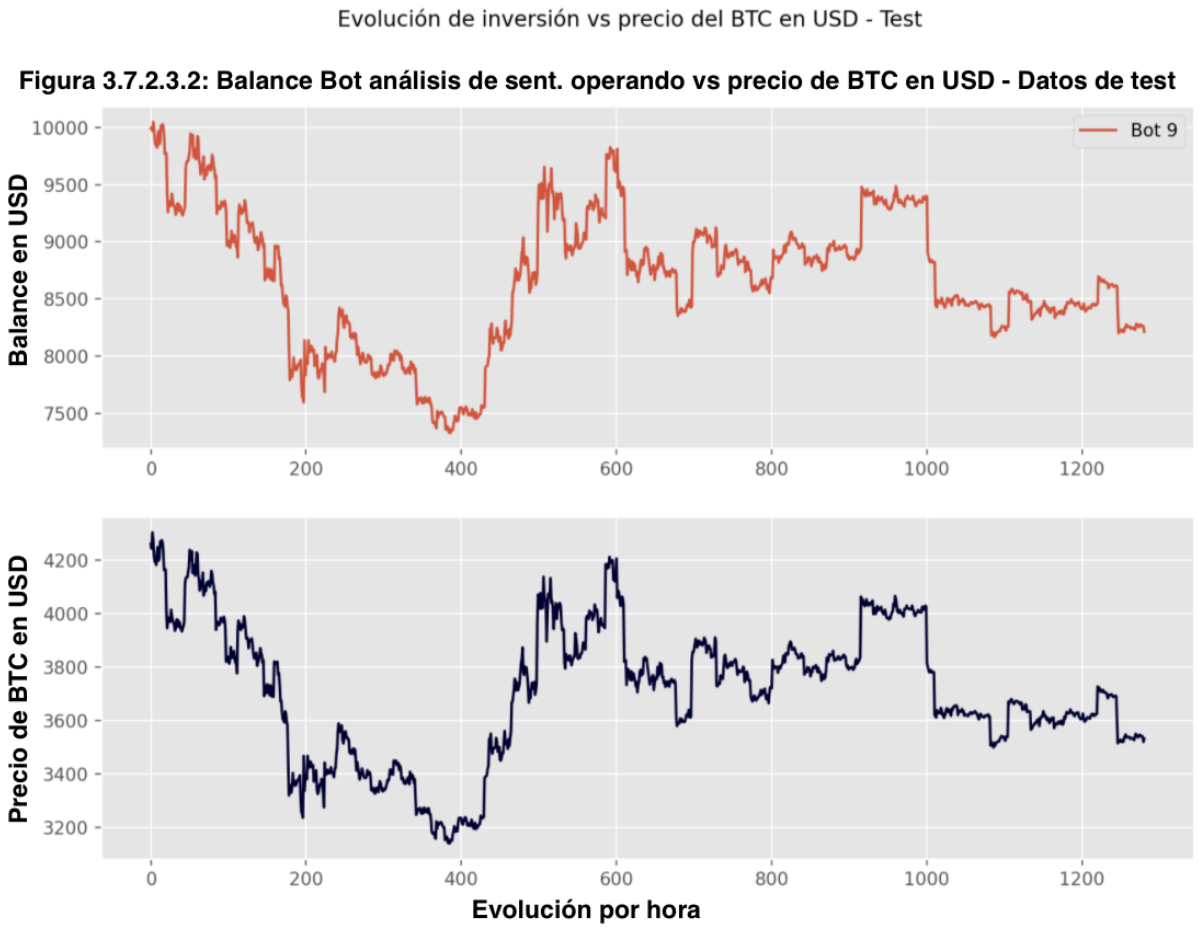


Figura 3.7.2.3.2: Balance Bot análisis de sent. operando vs precio de BTC en USD - Datos de test

Figura 3.7.2.3.3: Análisis de Factibilidad - Test - Modelo Nº9 iteración 9

Métrica	Iteración 9
# Pasos rentables	5
# Pasos NO rentables	1276
% Pasos rentables	0.39%
% Volumen de ganancias	0.01%
Volumen prom. ganancias	21 usd
Volumen prom. pérdidas	1340 usd
Factibilidad final	-1335 usd

Lo que sucede con este Bot al operar (Figura 3.7.2.3.2) es que, es tan alta la recompensa que puede llegar a recibir, que se ve “tentado” a operar con tal de que, en algún momento, reciba dicha recompensa. Como al comienzo, lo único que puede realizar es comprar o mantener, invierte todo el balance en el primer paso y luego se mantiene esperando una gran recompensa repentina. Este es el clásico caso de un problema de “reward shape”, y además, la razón del porqué se planteó el supuesto de “Estabilidad”. Los datos de la tabla sobre análisis de factibilidad, indican prácticamente 0% de pasos rentables y volúmenes de pérdida promedio de dólares elevados, mucho mayores a los volúmenes de ganancia promedio. (Figura 3.7.2.3.3). Al haber incorporado los datos de análisis de sentimientos, el Bot se transformó en un modelo muy inestable. Dado que lo que caracteriza el problema que se está tratando, es la inestabilidad del precio del BTC, es evidente que la combinación de ambos factores no terminan convergiendo en resultados de alta factibilidad. La razón por la cual se genera esta inestabilidad puntualmente es muy difícil de analizar y explicar, ya que este patrón también se vió al agregar las variable de “ta”, y sin embargo, no existen razones evidentes para pensar que guardan alguna relación con las variables de análisis de sentimientos. Aurélien Géron, ex líder del equipo de clasificación de vídeos de Youtube y escritor de: "Hands-on Machine Learning with Scikit-Learn and TensorFlow", expresa en su capítulo sobre reinforcement learning: *“El Reinforcement Learning es notoriamente difícil, en gran parte debido a las inestabilidades del entrenamiento y la enorme sensibilidad [...] Esta es una de las principales razones por las que Reinforcement Learning no se ha adoptado tan ampliamente como el aprendizaje profundo normal ".*

Capítulo 4: Discusión

En este capítulo se procede a resumir los resultados más relevantes obtenidos a lo largo del trabajo y se presentan observaciones concluyentes. Luego se detallan las limitaciones del método y se añade lo que se considera como trabajo futuro por realizarse. Por último, se comenta sobre las aplicaciones prácticas, aportes más significativos y la conclusión final sobre la hipótesis inicial.

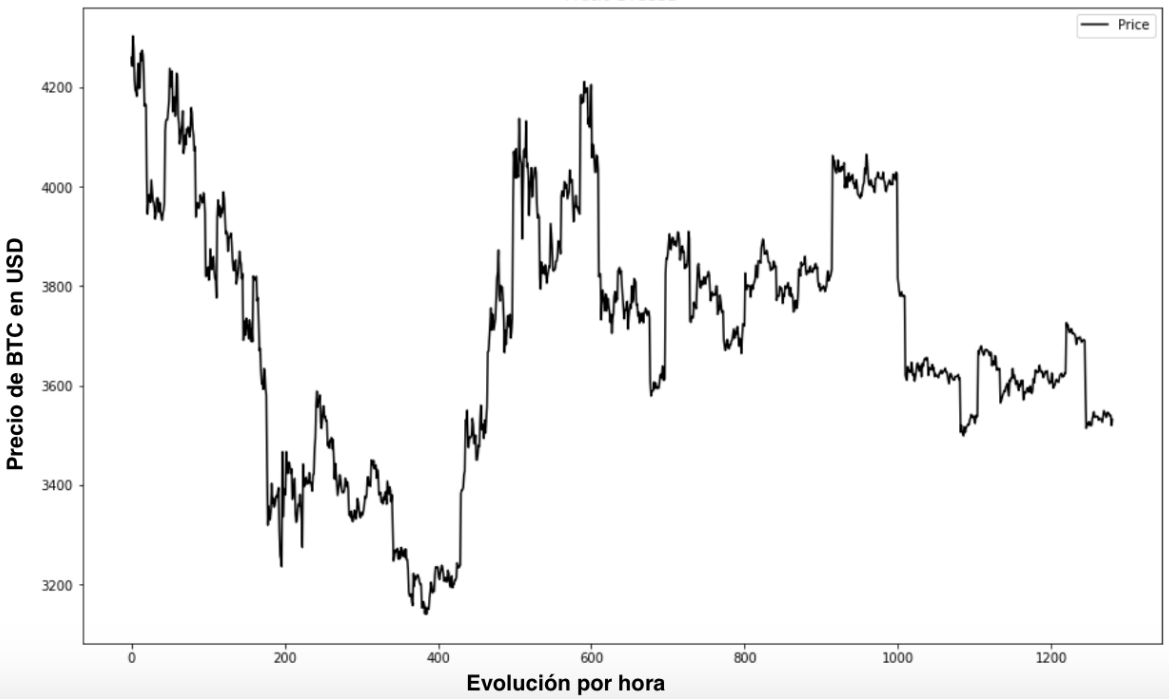
4.1 Resumen de resultados

Recapitulando, a lo largo de este trabajo, se probaron distintas configuraciones de ambientes y agentes para el problema dado. Luego, se trabajó sobre los datos buscando brindar la mayor cantidad de información, para que los modelos entrenados puedan defenderse en un determinado contexto. Se fue interpretando el efecto y comparando el impacto de cada alternativa y cambio que se fue realizando. Por último, se seleccionaron lo que se consideró en términos de performance, los mejores Bots. Se midió la factibilidad de cada uno según la propia definición. Se recuerda que, la perfomance indica que tan bien aprendió un modelo la estrategia que se le enseñó, y la factibilidad indica que tan bien funciona la aplicación de lo aprendido sobre datos nunca antes vistos.

Como resultado final, se obtuvo un único modelo que resultó factible. Este modelo fue el que más se acercó a los supuestos definidos al comienzo, los cuales pretendían garantizar una buena performance. De este resultado, se puede apreciar la importancia de alcanzar el segundo supuesto: “estabilidad”. Esto se convirtió en algo vital para conseguir una buena performance y mantener el mismo comportamiento en datos nunca antes vistos. Si bien agregar las variables de análisis de sentimientos, termina con un resultado de factibilidad muy desfavorable, se entiende que se debe a la falta de estabilidad en la función de recompensa. Como se observa una cierta recompensa promedio y mediana mayor comparando con el resto de los bots, se presume que las variables de análisis de sentimientos algo de información útil, distinta a la que se cuenta con las variables técnicas, podrían estar aportando. Por lo tanto, no se puede descartar la utilidad de las variables de análisis de sentimientos como aporte de información fundamental, diferente a la que aportan las variables técnicas. Lo que sí se concluye, es que estas variables desestabilizan por completo el modelo final. Por lo anterior, también se concluye sobre la importancia de elegir una buena función de recompensa para mantener la estabilidad del modelo.

En la figura 4.1.1, se muestran los resultados presentados al final del capítulo 3 sobre un mismo gráfico, comparando los bots preseleccionados operando sobre el mismo periodo, la variación del precio del BTC con respecto al dólar y las estrategias baseline establecidas al comienzo de este trabajo. Es notable la superioridad del bot estable por sobre el resto.

Figura 4.1.1: Comparación de Bots y estrategias baseline operando vs precio de BTC en USD - Datos de test



4.2 Limitaciones y trabajo futuro

El presente trabajo cuenta con ciertas limitaciones que se considera importante aclarar y comprender, para luego poder dimensionar en términos de esfuerzo, cuánto costaría desarrollar una solución final, totalmente automática y productiva, con potenciales aplicaciones prácticas y reales del método expuesto.

- **Cantidad de datos:**

Si bien la cantidad de datos con los que se trabajó para entrenar, validar y evaluar los modelos se considera suficiente, sería recomendable entrenar los modelos aprovechando más datos históricos con los que se cuenta hoy al día de la fecha. La razón de ver este trabajo limitado al periodo bajo análisis, es por los límites del conjunto de datos ya clasificados, el cual contenía las variables de análisis de sentimientos.

- **Cantidad de iteraciones de entrenamiento:**

Se trabajó con un máximo de diez iteraciones porque se consideró que era suficiente para mostrar los cambios y mejoras de comportamiento entre las pruebas realizadas, manteniendo un tiempo de entrenamiento de máquina razonable para experimentación. No obstante, en reinforcement learning, más iteraciones no siempre garantizan mejores resultados (*catastrophic forgetting*), pero si dan mayor visibilidad de los cambios y aprendizaje del modelo en cuestión (Géron, 2019).

- **Estabilidad de las variables de análisis de sentimientos:**

En este trabajo, se utilizaron variables de análisis de sentimientos ya procesadas, con las cuales en trabajos pasados, se probó cierta relación entre la polaridad de los sentimientos y el precio del Bitcoin (Ramos Badiola, 2019). Lo que no se realizó, fue ejecutar un análisis exploratorio más exhaustivo, para luego aplicar técnicas de ingeniería de atributos sobre las variables de análisis de sentimientos.

- **VADER:**

El conjunto de datos con el que fueron clasificados los tweets para análisis de sentimientos, fueron trabajados con VADER. Si bien este modelo fue desarrollado especialmente para clasificar sentimientos sobre redes sociales, el desarrollo del mismo data del 2014. Con la velocidad de avance de la tecnología, hoy existen técnicas conocidas más robustas que VADER y que están basadas en tecnologías de deep learning, como por ejemplo BERT (Sun, Huang, Qiu, 2019).

- **Descarga de tweets en tiempo real:**

En una posible implementación práctica de la propuesta de este trabajo, en la que se utilice datos de Twitter en tiempo real para que un bot pueda operar, hay que tener en cuenta que la API de Streaming de esta plataforma, se encuentra limitada a compartir una muestra de la totalidad de tweets que se desee descargar, la cual debería ser suficiente. El conjunto de datos de este trabajo fue desarrollado utilizando la librería GetOldtweets, que de manera offline trae la completitud de tweets referidos a un tema del pasado y no en tiempo real.

- **Cortes horarios por registro:**

En este trabajo se utilizó como criterio de corte entre cada registro el comienzo y fin de cada hora del día. Existe la posibilidad de que se pueda utilizar un criterio diferente que aporte mayor valor al método desarrollado. Este criterio podría ser, intentar tomar el comportamiento con mayor granularidad minuto a minuto, o intentar calcular algún diferencial que identifique los momentos en el tiempo con mayor información independientemente del tiempo horario.

En base a las limitaciones mencionadas, se deja como trabajo futuro para la búsqueda del desarrollo de un bot productivo, entrenar un modelo respetando la configuración más estable a la que se arribó en este trabajo, con una mayor cantidad de datos e iteraciones. También se deja abierta la posibilidad de cambiar el criterio de corte entre los registros, ya sea con mayor granularidad horaria o con algún diferencial que indique en qué momento un registro aporta más información que el resto. De tener la intención de sumar variables de análisis de sentimientos, se debería armar un nuevo conjunto de datos a partir de datos extraídos por la API de streaming de Twitter, para que de esta manera se trabaje con una muestra y no con la totalidad de los mismos. Luego, clasificarlos por sentimientos utilizando un modelo más robusto como BERT (Sun, Huang, Qiu, 2019). Por último, aplicar técnicas de ingeniería de atributos para buscar lograr estabilidad, sin perder el aporte de información fundamental.

4.3 Conclusión final y aplicaciones prácticas

Para cumplir con la hipótesis de este trabajo se comienza planteando dos desafíos. Por un lado, desarrollar un bot basado en Deep Reinforcement Learning e ingeniería de atributos, capaz de aprender estrategias, para poder operar sobre un ambiente de trading de Bitcoin y ser lo más rentable posible. El segundo desafío, y a su vez estrategia para alcanzar el primero, fue sumar variables de análisis de sentimientos que cubran el déficit de información fundamental.

Por los resultados obtenidos, se pueden extraer observaciones valiosas y significativamente útiles para trabajar sobre el dominio en cuestión:

- Primero que nada, es importante destacar el aporte de los supuestos iniciales. De los 3, mantener una recompensa creciente positiva, estaría garantizando la generalización de los modelos para distintos periodos de tiempo. Este supuesto, resultó ser muy importante para no caer en un modelo dominado por el contexto y la volatilidad. Se considera que el supuesto más importante, es el supuesto de estabilidad, ya que el mismo permitió encontrar el único modelo factible. Si bien puede sonar evidente, este supuesto también destaca la importancia de intentar mantener un modelo estable sobre un contexto muy volátil e inestable. Al cumplirse los dos supuestos anteriores, el supuesto sobre promedio y mediana, resulta ser muy útil para comparar modelos por performance y tomar decisiones al momento de entrenar los bots.

- Para medir la factibilidad, se tuvo en cuenta dos cuestiones fundamentales. Durante cuánto tiempo se mantiene rentable un bot y qué tan rentable se mantiene. Al medir la factibilidad, se consideró importante abstraerse de la idea de solo observar el balance final, y poner el foco en la evolución de cada operación. La importancia de esto recae en que, la alta volatilidad del Bitcoin, siempre presentará escenarios para operar totalmente distintos. Algunos, más favorables que otros. El secreto, se encuentra en encontrar el modelo que mejor pueda sortear esta variedad de escenarios, manteniéndose lo más rentable posible, buscando “sobrevivir” la mayor cantidad de tiempo como si fuera un juego de Atari.

- Otro punto clave, fue encontrar la manera correcta de premiar al Bot. Si bien la bibliografía de este campo de estudio es insistente en la búsqueda de la correcta función de recompensa, durante la experimentación fue notable la necesidad de encontrar la misma. Al utilizar Omega ratio, se obtuvo un gran aporte de estabilidad, pero se consiguió más valor aún en lo que es la generalización de estos modelos. Esto fue el resultado de utilizar una métrica o ratio, que pueda tener en cuenta la mayor cantidad de información del contexto, para luego definir el tamaño del premio o castigo para el bot.

- Para cerrar esta lista, se considera también muy importante destacar una característica fundamental para trabajar con este tipo de tecnologías aplicadas en contextos extremadamente volátiles. El mayor desafío de este trabajo, se encontró en la búsqueda de estabilidad. Teniendo en cuenta los comentarios de Aurélien Géron compartidos al final del capítulo 3, es evidente la dificultad de hallar modelos estables con Reinforcement Learning. También se vio, la importancia de conseguir estabilidad en estos modelos para obtener resultados factibles. Por lo tanto, se define que, a la hora de modelar con Reinforcement Learning para este tipo de problemas, se debe poner el foco en la búsqueda de estabilidad. Para ello, en este proyecto, lo que más valor aportó en términos de estabilidad, no fue agregar variables técnicas o fundamentales, sino que fue aplicar métodos de ingeniería de atributos.

Para concluir, se considera que se cumplió parcialmente la hipótesis planteada al comienzo de este trabajo. Se demostró que se puede realizar análisis técnico sobre Bitcoin de una manera mayormente automática, basándose en un método que utiliza algoritmos de Deep Reinforcement Learning e ingeniería de atributos. No se pudo probar el aporte de valor de las variables de análisis de sentimientos en búsqueda de cubrir el déficit de información fundamental, pero sí se consiguieron mínimos indicios que dejan las puertas abiertas para seguir investigando sobre este enfoque.

En términos prácticos, para que el bot resultante pueda operar realmente, requiere del desarrollo de algún tipo de sistema sencillo que extraiga los datos de entrada en tiempo real y se los entregue al bot. Este sistema, debe tomar la salida del modelo, tanto para operar vía API en algún exchange que lo permita, o exponer la información como soporte a la toma de decisiones sobre cómo operar hora a hora a través de algún reporte o tablero. De esta manera, se obtiene una herramienta que realiza análisis técnico totalmente automatizada, para realizar operaciones sobre Bitcoin, sin la necesidad del expertise ni la intervención del trader humano.

Bibliografía

- Baggini A. 2017. Análisis técnico y fundamental del valor de la acción de la empresa Petrolera del Conosur S.A. (PSUR) para el año 2016. Universidad Nacional de Rosario. https://rephip.unr.edu.ar/bitstream/handle/2133/11517/TESIS.pdf?sequence=3&isAllowed=y
- Ramos Badiola J. 2019. ¿Podemos comerciar Bitcoin usando análisis de sentimiento sobre Twitter? Universidad Pontificia de Comillas. https://drive.google.com/file/d/1we5kywWQqQRYX0Hy8edsKKN7PdaaFuzY/view
- King A. 2019. Creating Bitcoin trading bots don't lose money. https://towardsdatascience.com/creating-bitcoin-trading-bots-that-dont-lose-money-2e7165fb0b29
- Farell R. 2015. An Analysis of the Cryptocurrency Industry. Wharton Research Scholars. https://repository.upenn.edu/cgi/viewcontent.cgi?article=1133&context=wharton_research_scholars
- Swan M. 2015. Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc.
- Yli-Huumo J, Ko D, Choi S, Park S, Smolander K. 2016. Where Is Current Research on Blockchain Technology?—A Systematic Review. West Virginia University. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0163477
- Housley R. 2004. Public Key Infrastructure (PKI). John Wiley & Sons, Inc. https://onlinelibrary.wiley.com/doi/abs/10.1002/047148296X.tie149
- James G, Witten D, Hastie T, Tibshirani R. 2013. An Introduction to Statistical Learning With Applications in R. Springer.
- Hornik K, Stinchcombe M, White H. 1989. Multilayer feedforward networks are universal approximators. Elsevier Ltd. https://www.sciencedirect.com/science/article/abs/pii/0893608089900208?via%3Dihub
- Géron A. 2019. Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. 2nd Edition. O'Reilly Media, Inc.
- Hochreiter S, Schmidhuber J. 1997. Long Short-Term Memory. Neural Computation. MIT Press Direct. https://direct.mit.edu/neco/article/9/8/1735/6109/Long-Short-Term-Memory
- Sutton R, Barto A. 1998. Reinforcement Learning: An Introduction. MIT Press Direct.

- Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M. 2013. Playing Atari with Deep Reinforcement Learning. ArXiv. https://arxiv.org/abs/1312.5602
- Mnih V, Puigdomènech Badia A, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K. 2016. Asynchronous Methods for Deep Reinforcement Learning. ArXiv. https://arxiv.org/abs/1602.01783
- Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. 2017. Proximal Policy Optimization Algorithms. ArXiv. https://arxiv.org/abs/1707.06347
- Schulman J, Levine S, Moritz P, Jordan M, Abbeel P. 2015. Trust Region Policy Optimization. ArXiv. https://arxiv.org/abs/1502.05477
- Graham B, Dood D. 2009. Security Analysis. 6ta Edición.
- Murphy, J. 2007. Análisis técnico de los mercados financieros.
- Elder, A. 2004. Vivir del trading.
- Attia J. 2019. Evaluating the Effectiveness of Common Trading Models. Brooklyn Technical High School. ArXiv. https://arxiv.org/pdf/1907.10407.pdf
- Cory, M. 2019. Diverge definitions and uses. Investopedia. https://www.investopedia.com/terms/d/divergence.asp
- Brown, C. 1999. Technical Analysis for the Trading Professional. McGraw-Hill Education.
- Sharpe, W. 1966. Mutual Fund Performance. The Journal of Business. http://www.stat.ucla.edu/~nchristo/statistics_c183_c283/sharpe__mutual_fund_performance.pdf
- Sortino F, Price L. (1994). Performance measurement in a downside risk framework. The Journal of Investing Fall 1994, 3.
- Benhamou E, Guez B, Paris N. 2019. Omega and Sharpe ratio. ArXiv. https://arxiv.org/abs/1911.10254
- Denecke K. 2008. Using SentiWordNet for Multilingual Sentiment Analysis. https://www.researchgate.net/profile/Kerstin-Denecke/publication/4330825_Using_SentiWordNet_for_multilingual_sentiment_analysis/links/00463526769e2e3c11000000/Using-SentiWordNet-for-multilingual-sentiment-analysis.pdf
- Hutto C, Gilbert E. 2014. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. ICWSM. https://www.scinapse.io/papers/2099813784#fullText
- Webempresa. 2018. ¿Qué es twitter? ¿Cómo funciona? ¿Cómo puedo usarlo para mi organización?. https://www.webempresa.com/blog/que-es-twitter-como-funciona-2.html

- Poole D, Mackworth A. 2017. Artificial Intelligence: Foundations of Computational Agents. 2nd Edition. Cambridge University Press. https://www.cambridge.org/ar/academic/subjects/computer-science/artificial-intelligence-and-natural-language-processing/artificial-intelligence-foundations-computational-agents-2nd-edition?format=HB&isbn=9781107195394
- BonsaiAI Microsoft team. 2017. Deep Reinforcement Learning Models: Tips & Tricks for Writing Reward Functions. Medium BonsaiAI blog. https://medium.com/@BonsaiAI/deep-reinforcement-learning-models-tips-tricks-for-writing-reward-functions-a84fe525e8e0
- Ratner B. 2009. The correlation coefficient: Its values range between +1/−1, or do they?. Journal of Targeting, Measurement and Analysis for Marketing. Springer. https://link.springer.com/article/10.1057%2Fjt.2009.5
- Sun C, Huang L, Qiu X. 2019. Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence. ArXiv. https://arxiv.org/pdf/1903.09588.pdf
- Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Universidad de Toronto. https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf
- Mushtaq R. 2011. Testing time series data for stationarity. Université Paris I Panthéon-Sorbonne. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1911068
- John E. Hanke, Dean W. Wichern. 2010. Pronósticos en los negocios. 9na edición. Pearson. https://drive.google.com/file/d/1tUp-0Zi_rkRygnPlSloLjPKMSssnMdDz/view
- Li, Y. 2018. Deep Reinforcement Learning: an overview. ArXiv. https://arxiv.org/abs/1810.06339

Anexo I

A continuación se presentan las 19 variables utilizadas en la sección 3.6.2, provenientes de la librería *ta.* La tabla cuenta con las variables y una breve descripción.

Variable	Nombre	Breve descripción
RSI	Relative Strength Index	Compara la magnitud de las ganancias y pérdidas recientes durante un período de tiempo específico para medir la velocidad y el cambio de los movimientos de precios de un valor.
UO	Ultimate Oscillator	Un oscilador de impulso diseñado para capturar el impulso en tres marcos de tiempo diferentes.
AO	Awesome Oscillator	Es un indicador que se utiliza para medir el impulso del mercado. AO calcula la diferencia de un promedio móvil simple de 34 periodos y de 5 períodos. AO se utiliza generalmente para afirmar tendencias o anticipar posibles reversiones.
MACD_diff	Moving Average Convergence Divergence	Es un indicador de impulso que sigue la tendencia y muestra la relación entre dos promedios móviles de precios.
EMA	Exponential Moving Average	Es un tipo de media móvil ponderada (WMA) que da más ponderación o importancia a los datos de precios recientes. Este indicador se utiliza diferenciado.
Vortex_diff	Vortex Indicator	Consiste en dos osciladores que capturan el movimiento de tendencia positiva y negativa. Una señal alcista se activa cuando el indicador de tendencia positiva cruza por encima del indicador de tendencia negativa o un nivel clave. Este indicador se utiliza diferenciado.
DPO	Detrended Price Oscillator	Es un indicador diseñado para eliminar la tendencia del precio y facilitar la identificación de ciclos.
KST_sig	Know Sure Thing Oscillator	Es una señal útil para identificar las coyunturas principales del ciclo del mercado de valores. Su fórmula se encuentra mayormente ponderada por los períodos de tiempo más largos y dominantes, con el fin de reflejar mejor los cambios primarios del ciclo del mercado de valores.
Aroon_up	Aroon indicator (tendencia al alza)	Identificar cuándo es probable que las tendencias cambien de dirección.
Aroon_down	Aroon indicator (tendencia a la baja)	Identificar cuándo es probable que las tendencias cambien de dirección.
BBHI	Bollinger band high indicator	Son bandas de volatilidad situadas por encima y por debajo de una media móvil. La volatilidad se basa en la desviación estándar, que cambia a medida que aumenta y disminuye la volatilidad. Las bandas se ensanchan automáticamente cuando aumenta la volatilidad.
BBLI	Bollinger band low indicator	Son bandas de volatilidad situadas por encima y por debajo de una media móvil. La volatilidad se basa en la desviación estándar, que cambia a medida que aumenta y disminuye la volatilidad. Las bandas se contraen cuando la volatilidad disminuye.

ADI	Accumulation/Distribution Index	Es un indicador basado en el volumen diseñado para medir el flujo acumulado de dinero que entra y sale de un valor.
CMF	Chaikin Money Flow	Mide la cantidad de volumen de flujo de dinero durante un período específico.
FI	Force Index	Ilustra cuán fuerte es la presión real de compra o venta. Los valores positivos altos significan que hay una fuerte tendencia alcista y los valores bajos significan una fuerte tendencia a la baja.
EM	Ease of movement	Relaciona el cambio de precio de un activo con su volumen y es particularmente útil para evaluar la fuerza de una tendencia.
VPT	Volume-price trend	Se basa en un volumen acumulado que suma o resta un múltiplo del cambio porcentual en la tendencia del precio de las acciones y el volumen actual, según los movimientos ascendentes o descendentes de la inversión.
DR	Daily return	Mide el cambio en el precio de una acción como un porcentaje del precio de cierre del día anterior