Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Decisiones de inversión automatizadas en python

Document: Memoria del trabajo final de grado

Alumne: Joan Prieto Montes

Tutor: Ignacio Martín Campos.
Departament: Informática, Matemática aplicada y estadística.
Àrea: Lenguajes y sistemes infromáticos.

Convocatòria (mes/any): Junio 2020

Universidad de Girona

Escuela Politécnica Superior

Decisiones de inversión automatizadas en python

Proyecto Fin de Carrera

Autor:
Joan Prieto Montes

Junio 2020

Índice

1. Introducción, motivaciones, propósito y objetivos del proyecto

1.1. Dónde situamos el proyecto

El proyecto está orientado al campo de las finanzas. Las finanzas corresponden a un área de la economía que estudia la obtención y gestión del dinero, es decir, administra los recursos financieros. En este caso, nos centraremos en la inversión (comprar y vender) de estos recursos. Las *Decisiones de inversión automatizadas en Python* actuarán sobre un producto que pertenece a la empresa *iBroker* (iBroker Global Markets SV, SA). IBroker es una empresa española que se dedica a actuar como un *broker*[1] online. A través de esta compañía, se puede invertir en la gama más completa de mercados y productos derivados, con la transparencia y facilidad de un inversor profesional.

Trabajaremos sobre un producto denominado **Tradingmotion** (tradingmotion.com). Se trata de un marketplace[2] de sistemas automáticos, que es una plataforma que contiene un conjunto de algoritmos de inversión automática que los clientes pueden contratar a cambio de una licencia mensual.

Desde 2002, TradingMotion se ha dedicado a ofrecer *sistemas de trading*[3] de distintos desarrolladores. Después de años, la expansión del número de sistemas, y creación de mejores herramientas para su análisis, están ayudando a expandir su disposición a más inversores a través de acuerdos con brokers líderes de todo el mundo.

Los sistemas automáticos pueden interactuar sobre diferentes instrumentos. Operar un sistema de trading sobre acciones no es lo mismo que sobre futuros, forex u opciones. Los sistemas sobre los que vamos a operar, actúan sobre **futuros**[4]. Sobre este campo los sistemas de trading explotan al máximo las características de compra/venta.

¿Porque sobre futuros?

En la contratación de futuros, las operaciones suelen ser más rápidas que en el caso de las acciones, y en muchos mercados hay liquidez alta y comisiones bajas; es decir, lo ideal.

[1] El término broker hace referencia a la figura de un intermediario entre compradores y vendedores.

[2] Define el sitio en Internet donde se llevan a cabo interacciones comerciales entre diferentes empresas.

[3] Un sistema de trading es un conjunto de reglas que especifican cómo operar en uno o en varios activos financieros con el objetivo de generar una rentabilidad positiva.

[4] El mercado de futuros consiste en la negociación de contratos de compraventa de determinados bienes a una fecha futura mediante un acuerdo de precio, cantidad y vencimiento en la actualidad.

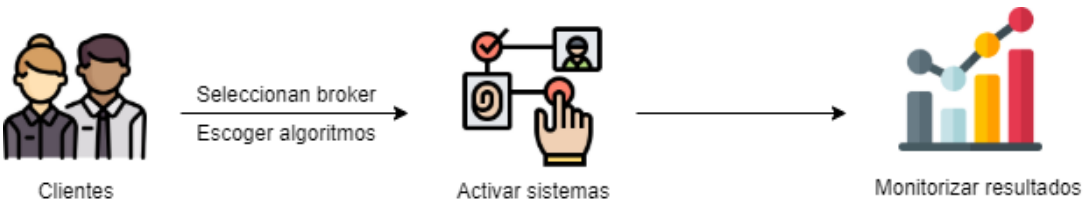


Figura 1: Esquema sobre el funcionamiento de tradingmotion

1.1.1. Finalidad

La finalidad del proyecto es puramente de investigación. Con el análisis como herramienta principal, y junto con unas hipótesis acertadas, se busca encontrar un conjunto de sistemas automáticos interesantes que cumplan nuestro objetivo.
El objetivo es escoger un conjunto de **sistemas automáticos** que pertenecen a la plataforma de *TradingMotion*, los cuales **no son irregulares** y **obtienen beneficios** en el mercado de futuros a lo largo del tiempo. Sabremos que algoritmos debemos seleccionar ya que a través de un amplio análisis, descubriremos que *algoritmos de predicción* son los mejores.

1.2. Motivaciones y propósitos

A lo largo de la carrera he podido tocar muchos temas de la informática. A partir de aquí aprendí que me gustaba más y que menos. Me di cuenta que las tareas de gestión de datos me llamaban mucho la atención, todo lo relacionado a la analítica y predicción a través de herramientas informáticas. Posteriormente encontré que eso tiene un nombre, se llama ciencia de datos o *datascience*.
Durante el primer cuatrimestre de cuarto de carrera le comenté a Víctor, (mi profesor de *Conceptos avanzados de ingeniería del software*) mi propósito de desarrollar el TFG en este campo. Entonces fue ahí cuando me ofreció la oportunidad de realizar este proyecto en la empresa donde ejerce de CTO (iBroker).

El hecho de realizar este proyecto es una gran oportunidad para introducirme en el campo del *datascience* a través de un caso práctico.

Mi motivación es llegar a ver en este proyecto, que la aplicación de técnicas de análisis y predicción tiene mucha importancia en casos reales. Que no son sólo conceptos teóricos, sino que tienen gran repercusión en los proyectos a día de hoy.
En cuanto a mi propósito, es adquirir un seguido de conocimientos tanto en el campo del *datascience*, como en el campo de las finanzas, y a través de estos, crear un proyecto competente que pueda ayudar a cumplir el objetivo.

1.3. ¿Qué vamos a obtener de Tradingmotion?

Los datos que vamos a recibir son los rendimientos de los sistemas que pertenecen a la plataforma Tradingmotion. Los rendimientos representan los beneficios, pérdidas, porcentaje de sesiones ganas o perdidas, ratio de acierto de los sistemas...
Con objetivo, de aprovechar los resultados de las simulaciones de los algoritmos que permanecen activos en la plataforma, para encontrar aquellos sistemas que obtienen grandes beneficios.

1.4. Problema

El no **explotar** los datos de los que disponemos, supone un problema para las empresas, ya que se deja de aprovechar unos datos, a partir de los cuales se pueden obtener beneficios o nuevas ventajas para la propia empresa.

1.4.1. ¿Porque hay que explotar los datos?

- El hecho de tener una plataforma destinada a la compra-venta de acciones, genera una gran cantidad de datos. Por lo tanto, qué mejor manera hay de aprovechar estos datos, que analizando los resultados para decidir qué algoritmos son los que rinden mejor e invertir en ellos.
- Según *the software alliance*: "Hoy, el 90 % de los líderes de negocios citan a los datos como uno de los recursos clave y un factor distintivo fundamental para los negocios, a la par de recursos básicos como las tierras, la mano de obra y el capital".
- La permanencia y análisis de los datos no supone un GRAN coste. A partir de un amplio abanico de herramientas *open source*[5] (python,R...), se facilita mucho el proceso de análisis.
- Los costes de almacenamiento de datos han bajado mucho estos últimos años. Según la publicación de Hagel III, John et al. en Exponential Technologies to Exponential Innovation (2013):

  DISMINUCIÓN DE LOS COSTES DE ALMACENAMIENTO: 38 % AL AÑO

  Por lo tanto, no supone tanto coste mantener los datos con el propósito de aprovecharlos.
- En el sector financiero se analizan los datos. La prueba de ello, está en el estudio que ha elaborado *EY, FrontQuery y Teradata*, acerca del empleo de *Big Data* en el sector

[5] Es un término que se utiliza para denominar a cierto tipo de software que se distribuye mediante una licencia que le permite al usuario final, si tiene los conocimientos necesarios, utilizar el código fuente del programa para estudiarlo, modificarlo y realizar mejoras en él.

financiero español. Dicho informe corrobora que cerca del 30 % de las empresas que componen el sector financiero en territorio español, está aprovechando las oportunidades que ofrece el Big Data.

En conclusión, recopilar y analizar los datos son tareas básicas para aquellas empresas que quieren seguir progresando. Por lo tanto, no analizar los datos supone una pérdida de futuros beneficios.

1.5. Objetivo

Automatizar el proceso de análisis de los datos generados por los rendimientos y movimientos de los sistemas activos. Y con ello, aplicar un seguido de reglas de predicción para filtrar que sistemas de trading son los "buenos".
Finalmente ejecutar este proceso automático cada cierto tiempo, y así decidir que sistemas escogeremos para simular y sobre qué producto[6] hay que invertir.

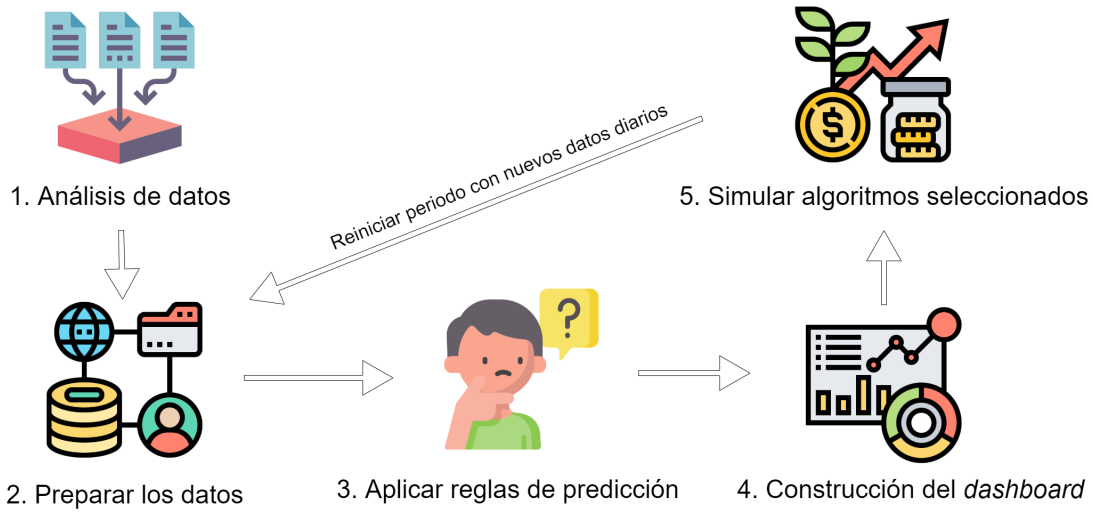


Figura 2: Procesos representativos del proyecto

[6]El producto hace referencia al mercado donde el sistema de trading está operando (por ejemplo: el petrólio, nasdaq, ibex...).

2. Estudio de viabilidad

El estudio de viabilidad de un proyecto es más importante que la planificación. Para poder concluirlo es imprescindible llevar a cabo una investigación completa. Este proyecto depende de la empresa, así que enfocaré si lo planeado es viable para la entidad.

2.1. Viabilidad económica

Asociamos viabilidad económica a analizar los beneficios y costes financieros asociados al desarrollo del proyecto. En este caso el coste del proyecto sería las horas que yo he gastado para completar todo el proyecto, asociando estas horas al precio euro/hora que cobraría un *data-scientist junior* + los costes fijos de luz, equipo...

$$399 \text{ horas} * 25\text{euro}/h = 9975 \text{ euros}$$

En el trading automático se mueve mucho dinero, así que 10k€ es lo que podría conseguir cartera de sistemas en una buena semana. Muy rápido de amortizar económicamente.

2.2. Viabilidad operacional

Es cierto que el mercado financiero siempre tiene una variable *aleatoria*, que no se puede predecir. Pero, tras analizar el comportamiento de los sistemas, podemos afirmar que se puede proponer una solución en la que se obtengan beneficios tanto a corto como a largo plazo.
Evidentemente, siempre depende de si los sistemas ganan o pierden. Pero si se forma una buena hipótesis, obtendremos una buena cartera de sistemas.

2.3. Viabilidad técnica

La complejidad técnica no es elevada, debido a que no requiere de un *hardware* específico, y en cuanto al software, es *opensource* y de fácil uso.

2.4. Viabilidad legal y contractual

Los datos que se utilizan para analizar los sistemas son generados por la propia empresa. Además es el propio *TradingMotion* que ha creado una **API** y facilita los datos a aquellos

clientes que los piden. Evidentemente, estudiar tanto el mercado como los rendimientos es algo legal. También añadir que es un proyecto de uso privado.

2.5. Viabilidad política

El proyecto está alineado estratégicamente con los propósitos de la empresa, no solo porque estamos aprovechando los datos que generamos, sino porque así también conocen los mejores sistemas que operan en su plataforma (*TradingMotion*).

3. Metodología

Para la gestión del proyecto he aplicado parte de la metodología del **PMBOK**, esta es un estándar para la gestión de los proyectos. La guía PMBOK es una guia creada por el Project Management Institute (o PMI), que establece una base de **buenas prácticas** relacionada con la gestión y administración de proyectos a partir de unas técnicas y herramientas.

El PMBOK está basado en *procesos*, lo que significa que diversifica todas las tareas del proyecto en varios "paquetes de trabajo". Este enfoque es muy similar, al mismo usado en otros estándares de gestión, por ejemplo *ISO 9000*. Los proceso interactúan entre ellos y se representan a lo largo de un eje cronológico.
En la sección 4 de planificación, se demuestra el uso de una metodología estándar, incluyendo grandes herramientas como el uso de diagramas de Gantt...

Para la parte del desarrollo del proyecto, he enfocado la metodología a la estructura de las tareas (análisi, preparar, aplicar reglas de predicción...). Debido a que la estructura es secuencial, he ido desplegando las tareas en ese orden establecido, siguiendo la planificación temporal del diagrama de Gantt.

Enfocando a la gestión de productividad, me he ayudado de aplicaciones com *Todoist*. Esta herramienta es un gestor de tareas multiplataforma que permite organizar las tareas por proyectos y añadir temporalidad a las actividades.

4. Planificación

Para seguir una correcta planificación y estructura del proyecto, he decidido separar todas las tareas en paquetes de trabajo, donde cada paquete incluye unas subtareas, estas tareas se relacionan a través de la matriz de dependencias (1). Cada tarea está programada para un periodo de tiempo concreto.

4.1. Paquetes de trabajo

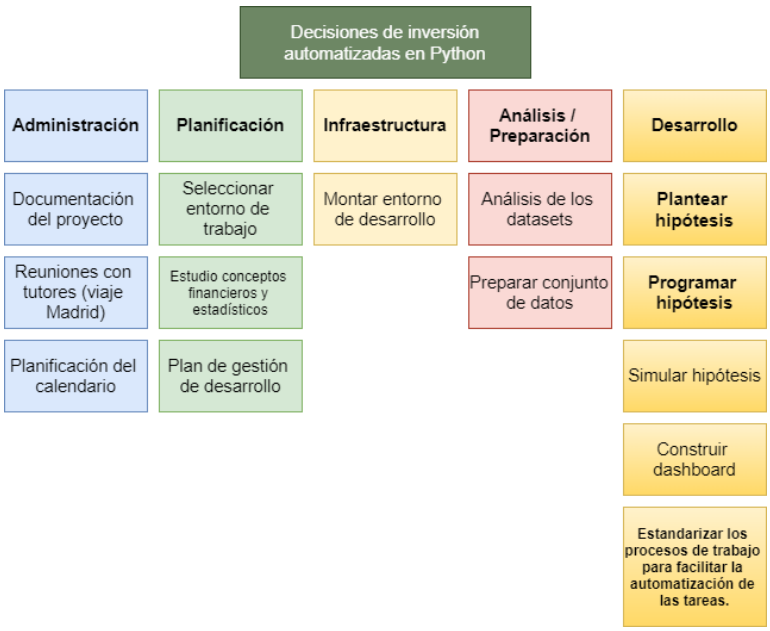


Figura 3: División de paquetes de trabajo

Temporalización:

- 1 dia = 3 o 4 horas de trabajo en el proyecto
- 1 mes = 25 días

Estandarizar los procesos de trabajo para facilitar la automatización de las tareas. Temporalización: A lo largo de proyecto (3 días).
Tareas: ▪ 1. Adaptar el proceso a automático a partir de variables y loops. Resultado: Tener el script automatizado.

Documentación del proyecto
Temporalización: A lo largo de proyecto (1 mes).

Tareas:
- 1.Introducción
- 2.Documentar el desarrollo.
- 3.Extraer conclusiones.

Resultado: Poseer un documento que explique de manera concisa el proceso completo del proyecto.

Reuniones con tutor y co-tutor
Temporalización: A lo largo de proyecto (3 días).

Tareas:
- 1.Concretar alcance del proyecto.
- 2.Controlar las tareas.
- 3.Poner en contexto el trabajo.

Resultado: Haber concretado los pasos que seguir durante el proyectos, junto con la puesta en situación de todo el contexto de TradingMotion.

Planificación del calendario
Temporalización: 3h.

Tareas:
- 1.Previsión de la duración de cada tarea.

Resultado: Una temporalización de las tareas realizadas duranete el proyecto.

Escoger de entorno de trabajo y familiarizarse con él
Temporalización: 8 dias.

Tareas:
- 1.Análisis de entornos
- 2.Formación de herramientas de trabajo.

Resultado: Facilitar el desarrollo escogiendo un entorno ideal y aprender a utilizarlo.

Estudio conceptos financieros y estadísticos
Temporalización: 12 dias.

Tareas:
- 1.Formación básica del funcionamiento del mercado financiero.
- 2.Repaso de conceptos estadísticos

Resultado: Tener más conocimientos orientados al campo donde se sitúa el proyecto..

Plan de gestión del desarrollo
Temporalización: 3h.

Tareas:
- 1.Dividir las tareas en varios pasos.
- 2.Ordenar según cronología

Resultado: Tener planificado todos los pasos de las etapa de desarrollo (programación).

Montar entorno de desarrollo y uso de nuevas librerías
Temporalización: 5 dias.

Tareas:
- 1. Instalar virtualenv
- 2. Crear entorno con librerías seleccionadas.
- 3. Comprobar funcionamiento

Resultado: Tener listo el entorno donde se llevarán a cabo las tareas de desarrollo de scripts.

Análisis de los datasets
Temporalización: 5 días.

Tareas:
- 1.Generar gráficos y tablas
- 2.Qué datos tenemos y cuáles no.

Resultado: Una amplía descripción de que datos estamos manejando a partir de librerías de análsis de sistemas.

Preparar conjunto de datos
Temporalización: 2 días.

Tareas:
- 1. Decidir qué columnas usaremos
- 2. Adaptar a su posterior uso

Resultado: Tener solo aquellos datos que serán utilizados en los posteriores procesos.

Plantear hipótesis
Temporalización: A lo largo de proyecto. (7 días)

Tareas:

- 1.Generar hipótesis.
- 2.Explicar sobre que se basa la hipótesis.

Resultado: Conjunto de reglas para analizar y simular sobre los sistemas de TradingMotion.

Programar hipótesis
Temporalización: A lo largo de proyecto. (15 días)

Tareas:

- 1.Todo el proceso de creación de reglas de predicción basadas en las hipótesis.

Resultado: Un conjunto de scripts que se encargan de filtrar aquellos sistemas que no son irregulares y obtienen beneficios.

Simular hipótesis
Temporalización: A lo largo de proyecto. (10 días)

Tareas:

- 1. Comparar resultados (rendimiento) de las hipótesis planteadas.

Resultado: Ránking de clasificación del rendimiento de cada hipótesis en un periodo de tiempo concreto.

Construir dashboard
Temporalización: 5 días.

Tareas:

- 1.Decidir gráficos representativos.
- 2.Decidir tablas representativas.
- 3.Implementar la generación del dashboard.

Resultado: Tener una fuente de información visual y directa para consultar los resultados de simulación.

	Analizar los datasets	Preparación de los conjuntos de datos	Desarrollar hipótesis	Construir una clasificación de los sistemas	Construcción del portafolio	Simulación de algoritmos seleccionados
Documentación del proyecto	X	X	X	X	X	X
Reuniones con tutores (viaje Madrid)	X	X		X	X	X
Planificación del calendario	X	X	X	X	X	X
Seleccionar entorno de trabajo	X	X	X	X	X	
Estudio conceptos financieros y estadísticos	X	X	X	X	X	
Plan de gestión de desarrollo	X	X	X	X	X	X
Montar entorno de desarrollo	X					
Análisis de los datasets	X					
Preparar conjunto de datos		X				
Plantear hipótesis			X			
Programar hipótesis			X			
Construir dashboard				X	X	
Simular hipótesis						X
Automatización de las tareas.	X	X	X	X	X	X

Cuadro 1: La Matriz de dependencias permite relacionar la dependencia de las tareas, ya que no se puede empezar una tarea sin tener todo listo.

4.2. Gestión del tiempo

A través del recuento de horas planificadas para cada paquete de trabajo, obtenemos que:

$$99 \text{ días} * 4h = 396 \text{ horas}$$

El cálculo total es de 396 horas de trabajo aproximadamente.

4.2.1. Diagrama de Gantt

El Diagrama de Gantt lleva utilizándose durante mucho tiempo y es una de las metodologías más famosas enfocadas a la gestión de proyectos. Está basado en un pequeño eje de coordenadas que mide la duración de las tareas a lo largo del proyecto.

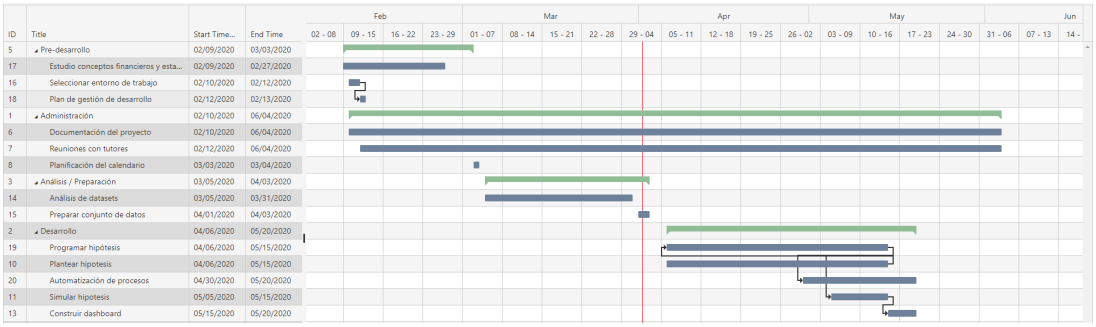


Figura 4: Diagrama de Gantt.

5. Marco de trabajo y conceptos previos

5.1. ¿Qué es el trading automático?

El *trading automático* representa el comercio algorítmico que utiliza a los sistemas de trading para crear ordenes de compra y venta de forma automática en los mercados financieros.

¿A que se debe la popularidad de este comercio automático?
Muchas de las personas que se quieren involucrar en el mercado financiero no tienen muchos conocimientos sobre cuando hay que invertir, esto lo soluciona el *trading automático*, ya que lo hace por tí. También porque son muy accesibles, ya que solo se necesita un ordenador con conexión a internet y para algunos sistemas, no es necesaria una gran inversión.

5.2. ¿Qué son los mercados de futuros?

Los mercados financieros se refieren a cualquier mercado donde se produce la negociación de valores, inclutendo el mercado de valores, el mercado de bonos, el mercado de divisas y el mercado de derivados...
Un tipo de mercado financiero, es el mercado de futuros, que como he explicado en la introducción, consiste en la negociación de contratos de compraventa de determinados bienes a una fecha futura mediante un acuerdo de precio, cantidad y vencimiento en la actualidad. En este caso los sistemas de TradingMotion actuan sobre futuros.

5.3. ¿Qué son los sistemas automáticos?

Como define iBroker, uns sistema automático es:

> Un programa informático suscrito al market data de un instrumento financiero, que opera de forma autónoma con el objetivo de obtener un rendimiento económico en cuentas de gestión discrecional de cartera orientadas al inversor retail.

Es decir, un sistema automático es un algoritmo, que sigue unas reglas de predicción para decidir cuándo hay que comprar, vender o no actuar sobre las acciones del mercado financiero.

5.3.1. ¿Cómo funcionan?

Los sistemas son pequeños programas que reciben como *input* la actividad del mercado financiero (market data). El market data se basa en un flujo de operaciones de compra/venta en el que el precio que da el valor a la acción es el último que se genera.

Este flujo de precios suele resumirse en gráficos de velas, estos representan de manera muy visual cómo varía el valor del de mercado.

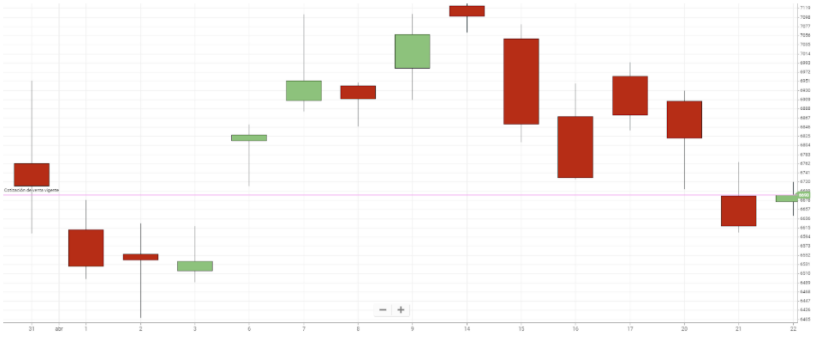
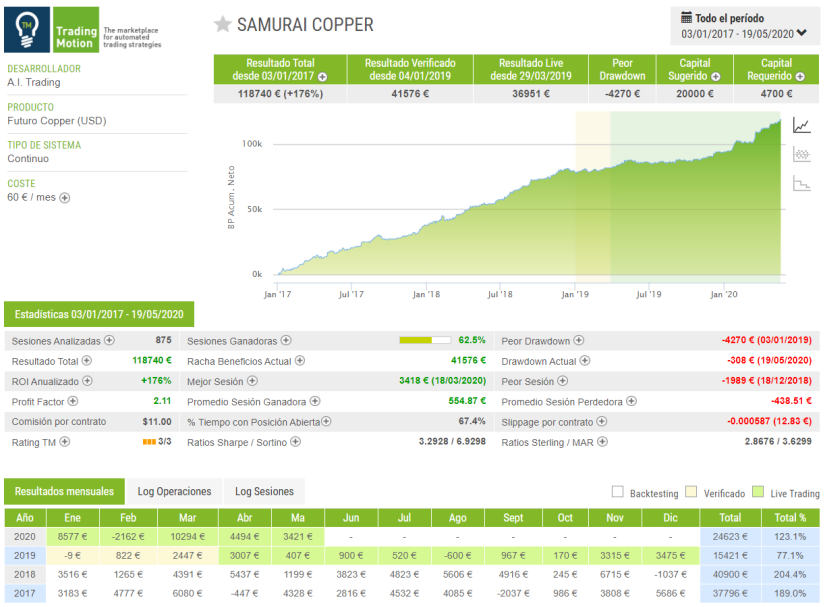


Figura 5: Gráfico de velas

Tras leer los datos de mercado, el sistema aplica las reglas de predicción que tiene programadas, y emite un movimiento (o no). A través de la plataforma con la que operamos el sistema, podemos ver los movimientos y los rendimientos del algoritmo.



Trading Motion — The marketplace for automated trading strategies

DESARROLLADOR: A.I. Trading
PRODUCTO: Futuro Copper (USD)
TIPO DE SISTEMA: Continuo
COSTE: 60 € / mes

SAMURAI COPPER

Todo el período: 03/01/2017 - 19/05/2020

Resultado Total desde 03/01/2017	Resultado Verificado desde 04/01/2019	Resultado Live desde 29/03/2019	Peor Drawdown	Capital Sugerido	Capital Requerido
118740 € (+176%)	41576 €	36951 €	-4270 €	20000 €	4700 €

Estadísticas 03/01/2017 - 19/05/2020

Sesiones Analizadas	875	Sesiones Ganadoras	62.5%	Peor Drawdown	-4270 € (03/01/2019)
Resultado Total	118740 €	Racha Beneficios Actual	41576 €	Drawdown Actual	-308 € (19/05/2020)
ROI Anualizado	+176%	Mejor Sesión	3418 € (18/03/2020)	Peor Sesión	-1989 € (18/12/2018)
Profit Factor	2.11	Promedio Sesión Ganadora	554.87 €	Promedio Sesión Perdedora	-438.51 €
Comisión por contrato	$11.00	% Tiempo con Posición Abierta	67.4%	Slippage por contrato	-0.000587 (12.83 €)
Rating TM	3/3	Ratios Sharpe / Sortino	3.2928 / 6.9298	Ratios Sterling / MAR	2.8676 / 3.6299

Resultados mensuales | Log Operaciones | Log Sesiones

Backtesting · Verificado · Live Trading

Año	Ene	Feb	Mar	Abr	Ma	Jun	Jul	Ago	Sept	Oct	Nov	Dic	Total	Total %
2020	8577 €	-2162 €	10294 €	4494 €	3421 €	-	-	-	-	-	-	-	24623 €	123.1%
2019	-9 €	822 €	2447 €	3007 €	407 €	900 €	520 €	-600 €	967 €	170 €	3315 €	3475 €	15421 €	77.1%
2018	3516 €	1265 €	4391 €	5437 €	1199 €	3823 €	4823 €	5606 €	4916 €	245 €	6715 €	-1037 €	40900 €	204.4%
2017	3183 €	4777 €	6080 €	-447 €	4328 €	2816 €	4532 €	4085 €	-2037 €	986 €	3808 €	5686 €	37796 €	189.0%

Figura 6: Ficha técnica de un sistema de la plataforma

5.4. ¿Qué es una cartera de sistemas?

Cuando seleccionas un conjunto de sistemas de trading para utilizarlos de forma simultanea, permitiendo al usuario actuar sobre varios productos a la vez.
Al diversificar una inversión con una cartera bien configurada de varios sistemas y productos distintos, conseguimos disminuir el riesgo y conseguir una rentabilidad más estable en el tiempo.

5.4.1. Factores a tener en cuenta al generar una cartera de sistemas

En este apartado explicaré los factores hay que tener en cuenta al generar una cartera de sistemas.
Lo bueno de generar una cartera de sistemas a través del script que hemos desarrollado, es que el resultado puede variar durante el tiempo. ¿A qué me refiero con esto? Si un sistema deja de ganar, a través del filtro se tendrá en cuenta que este sistema ya no es apto para seguir en la cartera. Y seleccionará un nuevo candidato.

Dentro de una cartera tenemos un conjunto de sistemas que operan sobre diferentes productos, como por ejemplo el *Nasdaq, ibex, Mini-Russell CME...* De cara a elegir una cartera de sistemas, siempre será preferible **diversificar** los productos sobre los que se invierte, y sobre todo diversificar en sistemas. Siempre es mejor escoger sistemas combinados de diferentes productos, para alcanzar el mayor grado de **descorrelación** posible en nuestra cartera.
Porque el mayor inconveniente de seleccionar varios sistemas que actúan sobre un mismo producto, es que pueden estar muy correlacionados, por lo tanto, al estar muy vinculado, si un sistema pierde, seguramente lo harán los demás.

La inversión en sistemas automáticos de trading puede ser dinámica, podemos alternar los sistemas de nuestra cartera de manera dinámica, aplicando criterios objetivos establecidos.
También hay quien piensa que una vez seleccionado el sistema, se debe mantener para siempre, ya que a largo plazo debería ofrecer beneficios. Eso ya queda a manos del usuario, quien acompañado con las estadísticas de rendimiento de los sistemas, decide sustituirlos o mantenerlos.

6. Requisitos

Como el uso de esta aplicación será interno, dividiremos los procesos de trabajo en requisitos. Los procesos los llevaremos a cabo por niveles, según su importancia y order.

6.1. Requisitos de desarrollo

1. Analizar los datasets (Exploratory Data Analysis): partimos de 3 datasets con información sobre las características, rendimientos y movimientos de los sistemas de trading. Mostrar gráficos relevantes que ayuden a entender cómo se comportan los sistemas a lo largo del tiempo.
   Prioridad 1.

2. Preparación de los conjuntos de datos para su posterior uso.
   Prioridad 1.

3. Desarrollar hipótesis para construir un ranking de los sistemas.
   Prioridad 1.

4. Construir una clasificación de los sistemas (algoritmos) ordenados de más a menos según sus posibilidades de acierto en el siguiente período. El ranking se basará en las hipótesis que planteamos en el paso anterior.
   Prioridad 1.

5. Construcción del portafolio (dashboard) sobre el ranking de sistemas, seleccionar qué algoritmos operaran durante el siguiente periodo según varios criterios (capital disponible, diversificación de productos, diversificación de desarrolladores, etc.,).
   Prioridad 2.

6. Simulación de los sistemas seleccionados durante un periodo concreto.
   Prioridad 2.

6.2. Requisitos sobre los datos

El análisis de datos debe ser específico y acotado según un periodo de tiempo concreto. Los períodos pueden ser diarios (ejecución del análisis diaria con los datos del dia anterior), semanales (fusión de todos los datos de la semana) o mensuales (fusión de los datos mensuales).

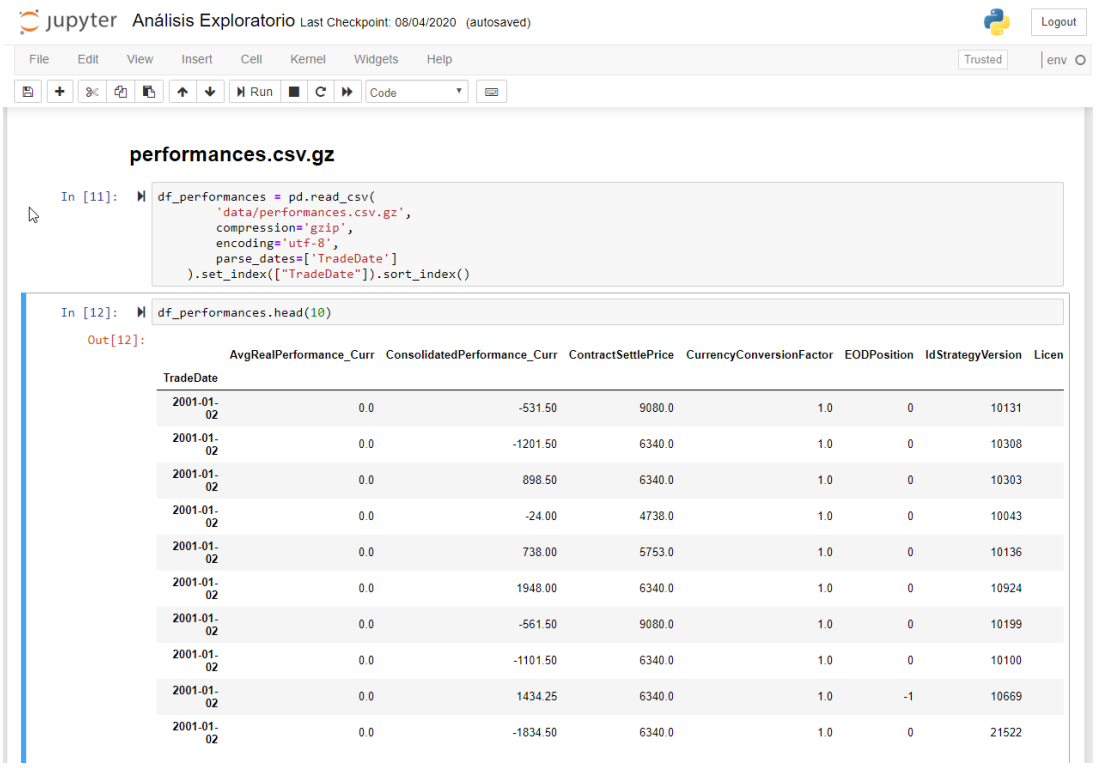
6.3. Requisitos de uso

El uso del proyecto que he desarrollado se basa en ejecutar los scripts que contienen la simulacion de los sistemas, para así visualizar los rankings de rendimientos que situaremos en el *dashboard*. Este *tablero* contiene los indicadores (un seguido de gráficas) que nos describirán como han actuado los sistemas durante la simulación.

7. Entorno y decisiones

He decidido utilizar **Python** para ejecutar todos los procesos del proyecto, ya que es un lenguaje muy versátil, permite hacer muchas cosas con pocas líneas de código y la gran cantidad de bibliotecas, facilita el manejo del lenguaje. Además, la gestión de datos a través de *Python* es tan habitual, que podemos encontrar todo tipo de guías, libros y artículos que sirven de soporte técnico.

Una gran herramienta para crear entornos aislados para *python* es ***Virtualenv***. Este es un entorno virtual que permite instalar paquetes sin interferir en otras versiones ya instaladas en el python del sistema operativo.
La ventaja de este entorno, es que puede mantener varios contenedores virtuales cada uno con una versión de python diferente y otras librerías. Por lo tanto, crear un entorno personalizado con las librerías que nos interesen es la mejor opción para gestionar un proyecto basado en Python. Además, virtualenv permite instalar dentro IDEs (entornos de desarrollo como Django, Jupyter Notebooks...).



TradeDate	AvgRealPerformance_Curr	ConsolidatedPerformance_Curr	ContractSettlePrice	CurrencyConversionFactor	EODPosition	IdStrategyVersion	Licen
2001-01-02	0.0	-531.50	9080.0	1.0	0	10131	
2001-01-02	0.0	-1201.50	6340.0	1.0	0	10308	
2001-01-02	0.0	898.50	6340.0	1.0	0	10303	
2001-01-02	0.0	-24.00	4738.0	1.0	0	10043	
2001-01-02	0.0	738.00	5753.0	1.0	0	10136	
2001-01-02	0.0	1948.00	6340.0	1.0	0	10924	
2001-01-02	0.0	-561.50	9080.0	1.0	0	10199	
2001-01-02	0.0	-1101.50	6340.0	1.0	0	10100	
2001-01-02	0.0	1434.25	6340.0	1.0	-1	10669	
2001-01-02	0.0	-1834.50	6340.0	1.0	0	21522	

Figura 7: Entorno gráfico de Jupyter Notebooks

En mi caso, me he decantado por crear pequeños archivos con Jupyter Notebooks (figura 7). Jupyter Notebook es una aplicación cliente-servidor que permite crear, ejecutar y compartir documentos en Python.
La ventaja de estos notebooks, es que están dedicados al análisis de datos, ya que se puede combinar perfectamente el código con los textos explicativos. Así, se permite ejecutar a tiempo real y en memoria todo tipo de código en python, ecuaciones y una gran variedad de gráficos. Como vemos en la figura 7, tiene una interfaz simple, los *notebooks* proporcionan un excelente punto de partida para que evolucionen muchos proyectos complejos. Un documento *notebook* consiste esencialmente en celdas que se llaman "*input*", y cada celda *input* le corresponde un *output*.

Una librería indispensable para gestionar, analizar y tratar datos en python es Pandas. Pandas es una biblioteca de software que funciona como una extensión de *NumPy*, se usa básicamente para la manipulación y análisis de datos. Es software libre y incluye muchas funciones para datos como lectura, escritura, muestra, transformaciones, mezcla...
No solo utilizaremos pandas, sino que añadiremos varias librerías que nos ayudarán a facilitar todos los pasos de nuestro proyecto:

- **Numpy**: Dedicado al cálculo numérico, seria como añadir un *Matlab* a un entorno de Python.
- **Tqdm**: Añade una barra de progreso a la ejecución de los bucles para medir su progreso.
- **Pyfolio**: Permite analizar rendimientos y riesgos de carteras financieras.
- **Plotly**: Crear gráficos interactivos.
- **Empyrical**: Cálucar riesgo financiero común y métricas de rendimiento.
- **Ipython**: Dedicado a cálculos interactivos.
- **SciPy**: Herramientas y algoritmos matemáticos.
- **Quantstats**: Calcular métricas de *sistemas de trading* a partir del porcentaje de retornos diarios

8. Análisis y diseño de los procesos

En este capítulo, detallaremos las necesidades de los procesos en los que se divide el proyecto, junto con una representación de cada uno.

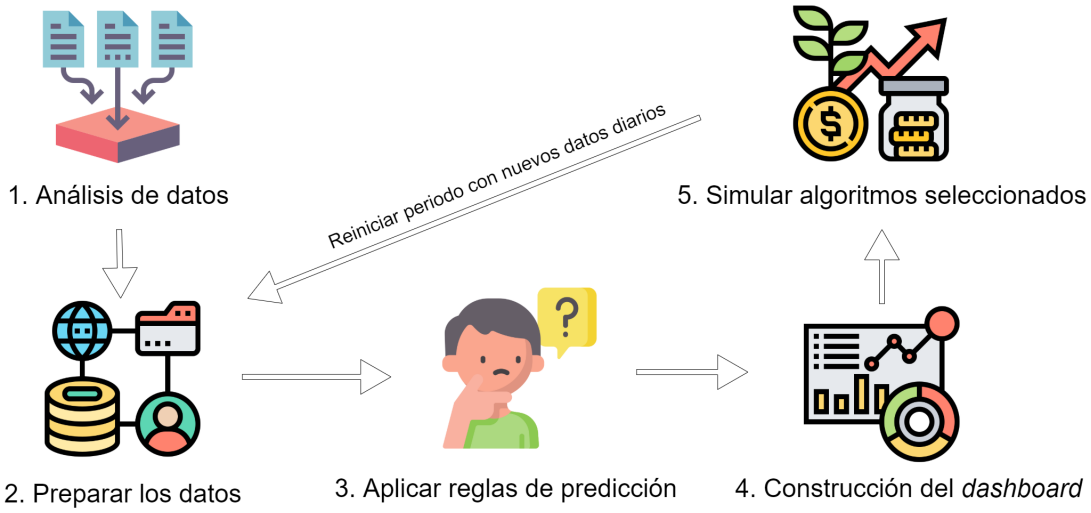


Figura 8: Procesos representativos del proyecto

Como vemos en la figura 8, distribuimos las tareas en 5 procesos. El *pipeline*[7] son dos procesos que se repiten cada cierto periodo:

- **Predicción**: Construir un *ranking de sistemas* ordenados de "más a menos"según su valía.
- **Construcción del portfolio**: Tras generar las predicciones, debemos seleccionar aquellos sistemas que operaran el siguiente periodo según diferentes criterios. Es decir, seleccionar que sistemas formarán parte de nuestra cartera de sistemas.

[7]El "pipeline"hace referencia al volumen de negocio, donde reside la importancia del proyecto.

8.1. Diseño del procesamiento de datos

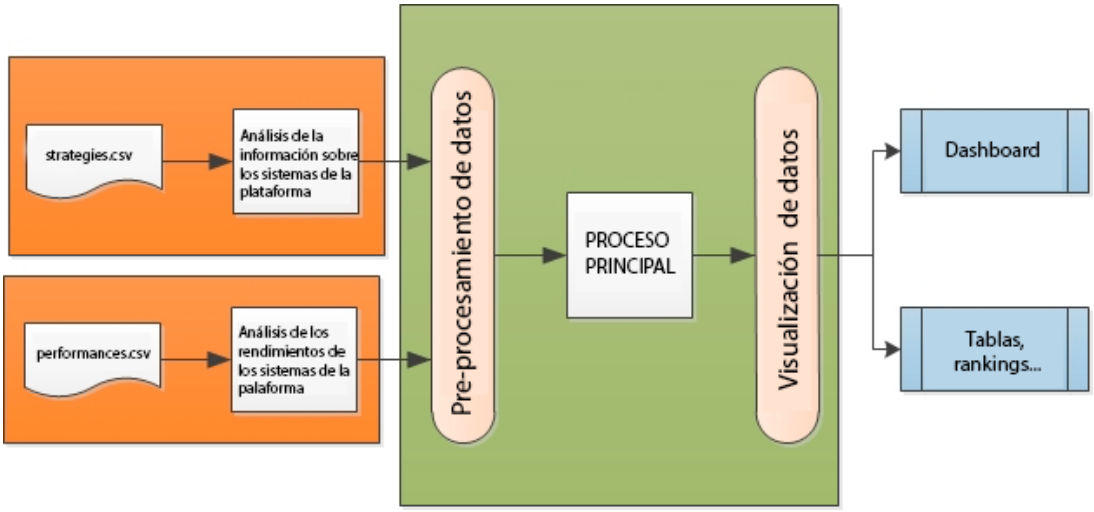


Figura 9: Esquema del procesamiento

Aplicamos uno de los principios del diseño, dividir la entrada y salida en capas. Desde la entrada obtenemos los datos, los analizaremos y pre-procesaremos. En el proceso principal, encapsulamos el conjunto de procesos estandarizados aplicados a programar hipótesis y sus resultados. Finalmente debemos transformar los datos generados a elementos visuales como gráficas (*dashboard*) y tablas

8.2. Análisis de los datos

Los datos los proporciona la propia empresa a los clientes para que puedan consultar todos los rendimientos a lo largo del periodo de simulación y activación del sistema. Se pueden consultar a través del webservice: https://www.tradingmotion.com/api/webservice.asmx

En esta sección, describiremos los 3 datasets que obtenemos en formato CSV (donde los campos son separados por comas).

- **Performances**: *Timeseries*[8] de rendimientos diarios de cada sistema
- **Strategies**: Atributos generales de los 2k+ sistemas del *marketplace tradingmotion*.

[8]Una timeserie es una serie temporal o cronológica. Corresponde a una sucesión de datos medidos en determinados momentos y ordenados cronológicamente.

- **Symbols**: Datos generales sobre los mercados en los que los sistemas pueden actuar. Para simplificar el proceso, **no lo utilizaremos**.

Para este proceso aprovecharemos las facilidades que ofrece *Pandas* para analizar los datos que gestionaremos, y un seguido de librerías que están enfocadas al *trading algorítmico* que muestran una gran cantidad de valores sobre rendimientos.
En concreto utilizaremos ***Pyfolio***. Esta es una biblioteca de *Python* para el análisis de rendimiento y de riesgo de carteras financieras. También permite dividir los periodos en *backtesting* y periodo activo.
Este tipo de librerías se centran en ofrecer un seguido de gráficos que ayudan a describir el rendimiento de forma más visual. A través de estos esquemas y de datos estadísticos obtenemos un análisis completo y rápido.

8.3. Preparar los datos

El pre-procesamiento de datos es un tema muy amplio, porque las técnicas varían dependiendo del tipo de dato que manejamos.

Los diferentes tipos de datos (imágenes, texto, sonidos, videos, archivos csv, etc.) tienen diferentes métodos para el preprocesamiento, pero hay algunos métodos, que son comunes para casi cualquier tipo de datos. Los más importantes de estos métodos son:

- **Carga de datasets**: Como he explicado anteriormente, la carga de los datos es a través de 2 o 3 csv comprimidos en gzip. La importación al entorno de trabajo es a partir de unas funciones que he desarrollado, las cuales se encargan de añadir los datos, sin perder la estructura de estos.
- **Normalización**: Los datos que obtenemos ya están normalizados, la única adaptación en el formato de los datos, ha sido la transformación de las fechas a *datetime* a través de pandas.
- **Tratar con los valores nulos**. Tras el análisis comprobamos que solo hay un campo que tiende a ser nulo, este es *FirstTradeDate* del dataset de strategies. Ya está controlado que si un sistema no ha sido activado nunca por un cliente, su valor será NaT (Not a Time).

La recopilación de datos en el *mundo real* puede contener irregularidades, datos dañados... Estos comprometen la calidad de conjunto de datos. De esta manera, es importante asegurarse que los datos no están incompletos, tienen ruido o son incoherentes.

Como en este caso heredamos los datos directamente de la plataforma de *trading*, todo el proceso tan costoso relacionado con el *data cleaning* no tiene tanto peso en nuestros *datasets*.

Esto es porque los datos que recibimos ya vienen muy limpios, no hay datos erróneos ni vacíos. Los datos recibidos son de calidad, cumplen los requisitos de integridad, consistencia y densidad. Que quiere decir esto, que no hay grandes duplicidades, anomalías y contradicciones sintácticas...

Por lo tanto, en lugar de enfocar la preparación de los datos a la limpieza de estos, he optado por centrarme en todos los procesos previos a facilitar el desarrollo de las hipótesis (la programación).
Una de las tareas, es reducir los *datasets* quitando las columnas que no son necesarias, que no influyen de manera muy directa al rendimiento de los sistemas, por lo tanto las descartamos.

Además es importante reducir la cantidad de datos que estamos tratando, ya que al minimizar esto, se agilizan los procesos de tratamiento y no se ocupa tanta memoria. En este caso, estamos gestionando **903.2+ MB** en memoria para el *dataset* de rendimientos y **346.9+ KB** para el *dataset* de sistemas. Evidentemente, el cómputo de los procesos de estos datos se puede hacer con un ordenador *estándar*. Aunque manejamos 5.505.882 retornos diarios de los sistemas, está muy lejos de considerarse *Big Data*. Ya que, aunque no esté especificado si un conjunto de datos se considera *Big Data* o no, la mayoría de los analistas y profesionales actualmente se refieren a conjuntos de datos que van desde 30-50 *Terabytes* a varios *Petabytes*.
Cabe añadir, que esta cantidad de datos va creciendo diariamente, con la inserción de los retornos diarios del dia anterior.

8.4. Reglas de predicción

La mejor manera de estudiar los rendimientos de los algoritmos, es a través de la aplicación de un método científico orientado a investigar como se comportan los sistemas de inversión y trading.
Esta *idea* es la que desarrolla *David Aronson* en su libro *Evidence Based Technical Analysis*. Él propone utilizar este método hipotético-deductivo para evaluar la eficacia de las señales en un sistema de trading.

La teoría de este método se divide en varias fases:

1. **Observación**
2. **Generar hipótesis**
3. **Predicción**
4. **Verificación**

5. **Conclusión**

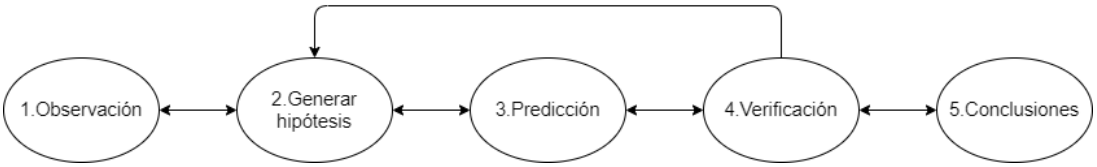


Figura 10: Esquema de las fases del método científico.

Observación. Se trata de adquirir información y detectar unos rasgos específicos o un comportamiento en concreto. La información se adquiere con la ayuda de algún instrumento. Aplicado a los sistemas de trading, sería descubrir una característica descriptiva de las estrategias.

Generar hipótesis. La parte vinculada al razonamiento inductivo (generar una conclusión a través de unas premisas). Basándonos en la observación, formulamos una hipótesis (suposición hecha a partir de unos datos que sirve de base para iniciar una investigación o una argumentación.).

Predicción. Si la hipótesis que hemos formulado es cierta, entonces...
La predicción forma parte del resultado en caso de que la hipótesis sea cierta. Habitualmente, en estadística la hipótesis nula es la suposición de base.
En nuestro caso cogeremos la hipótesis afirmativa como base, ya que solo nos interesa demostrar que la regla de predicción es cierta, para utilizarla en nuestro propio beneficio. Por lo tanto, examinar los datos y en el caso de que exista suficiente evidencia, aceptar la hipótesis afirmativa (o rechazar en caso contrario).

Verificación. A partir del análisis estadístico, junto con la programación de las hipótesis en *Python*, podemos validar la eficacia, la efectividad y eficiencia de estas. Para aceptar las hipótesis, es importante que la rentabilidad sea positiva. Aceptaremos aquella regla de predicción que nos proporciona el resultado que esperamos, un grupo de sistemas con ganancias constantes a lo largo del tiempo.

Conclusión. Hay que corroborar porque la hipótesis es cierta o falta, y extraer una conclusión que nos obligue a reflexionar sobre la deducción y verificación de la hipótesis.

8.4.1. ¿En que consiste la programación de las hipótesis?

La programación de una hipotesis se divide en tres fases:

1. **Cálculo de parámetros**: Cada hipótesis se centra en una o varias cualidades de los sistemas. En el cálculo de parámetros nos encargamos de calcular el valor de las propiedades que decidimos añadir en la hipótesis. Por ejemplo, una de las cualidades que podemos computar es el porcentaje de ganancias respecto a pérdidas. Entonces aplicaremos este cálculo a todos los sistemas y generaremos una clasificación.

2. **Aplicar reglas de filtraje**: Cuando ya tenemos los parámetros cálculados, debemos filtrar aquellos sistemas que creemos que son buenos. Para ello debemos seleccionar un intervalo respecto al parámetro que hemos calculado anteriormente. Por ejemplo, solo filtraremos aquellos sistemas que:

$$55\,\% < \text{porcentaje de ganancias respecto a pérdidas} < 65\,\%.$$

3. **Evaluación de la cartera**: Seleccionamos los 10 sistemas que tienen mejor puntuación respecto al parámetro calculado y filtrado, y los simulamos en los últimos 6 meses. Finalmente, analizamos el resultado final.

8.5. Construcción del *Dashboard*

¿Alguna vez has probado de conducir un coche sin mirar el cuentakilómetros, vigilar el nivel del combustible, la temperatura o cualquier de sus indicadores?
Básicamente conducir un vehículo sin revisar los instrumentos que te informan de su estado es exponerse a un riesgo **muy alto**. Pues es lo mismo que cuando estás aplicando un proceso determinado a tus datos sin revisar el resultado.

Por lo tanto, es vital crear un *dashboard* que facilite a los usuarios interpretar los resultados de manera más rápida. Pero, la importancia de un *dashboard* no solo se basa en ofrecer datos, sino también en el valor que éstos aportan a su posterior toma de decisiones. En este caso, es primordial representar la diferencia de ganancias y pérdidas de los sistemas para poder identificar qué algoritmos merecen la pena.
Para ello, he automatizado que a partir de la generación de una cartera, se muestre un seguido de gráfico analizando el reciente rendimientos de los sistemas que representan la cartera. Así poder evaluar si los sistemas seleccionados son efectivos.

He decidido integrar la librería Plotly con Jupyter Notebooks (el entorno de desarrollo). Plotly es una plataforma orientada a la visualización de datos, que ha implementado librarias para Python , R , JavaScript , Julia y MATLAB.

¿Que caracteriza a Plotly?
Plotly permite crear gráficos interactivos, gestionarlos y modificarlos a tu gusto, consultar el valor en cada punto del gráfico, alejar y acercar la vista del zoom La plataforma se enfoca en

gráficos básicos, pero también de tipo científico, estadístico y financiero.
Por lo tanto, con unas pocas líneas en python podemos generar gráficos bastante representativos para nuestros sistemas.

8.6. Simular algoritmos seleccionados

Una vez implementadas todas las hipótesis, debemos compararlas de alguna forma. Para ello he desarrollado un script que compara los rendimientos de todos las hipótesis en un mismo periodo de tiempo. Es en este paso donde averiguamos si una hipótesis vale la pena mantenerla.

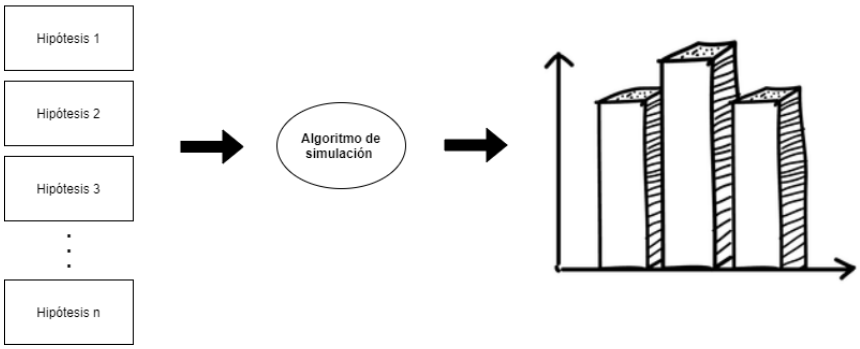


Figura 11: Estructura de los procesos de simulación.

El script desarrollado en un archivo aparte, consta de un seguido de funciones que ejecutan las hipótesis y devuelven un listado de los 10 mejores sistemas de cada hipótesis. A partir de ahí se simulan las carteras de sistemas con los seleccionados por cada hipótesis y se muestra el rendimiento final de cada regla de predicción.

9. Implementación y pruebas

Aquí detallaremos los problemas y soluciones que han aparecido en cada proceso durante su desarrollo.

9.1. Análisis de los datos

Para llevar a cabo el análisis, quise centrarme en encontrar un seguido de gráficas las cuales describan los datos que manejaremos. La dificultad de este proceso reside en decidir que datos son los que queremos destacar y como mostrarlos. Una vez decidido, el problema se centraba en como mostrarlos, para ello aproveché la propia librería de Pandas, la cual permite visualizar los datos a través de diagramas de barras, dispersión, de lineas...

9.2. Preparar los datos

La preparación de los datos se basa en crear pequeñas funciones para evitar la repetición de código e agilizar el proceso de desarrollo.

```
# Develop path: "data/strategies.csv.gz"
def importStrategies(path):
    df_strategies = pd.read_csv(path,
                            compression='gzip',
                            encoding='utf-8',
                            parse_dates=["CurrentPositionTime",
                                         "FirstBackTestedDate",
                                         "FirstTradeDate",
                                         "ReleaseDate",
"StatsLastUpdated"]).set_index("IdStrategyVersion")
    df_strategies.drop(columns=
['AcceptsNewClients','CurrentPositionTime','RecomReqCapitalCurrency',
'systemVersionIdentifier'])

    return df_strategies
```

Figura 12: Importar dataset "Strategies"de los sistemas

El principal problema de este apartado, consistía en decidir que funcionalidades deberían ser transformadas a funciones para re-aprovecharlas. Primeramente implementé aquellas funciones

relacionadas con la entrada y salida del proceso de generación de reglas de predicción, y posteriormente cuando ya programé la primera *hipótesis*, automaticé los procesos claves para utilizarlos en la generación de futuras reglas de predicción.

9.3. Reglas de predicción

Los principales problemas que he tenido para programar las hipótesis, están relacionados con las estructuras de datos. Mediante he programado los algoritmos, he ido aprendiendo cómo aislar, detectar y filtrar determinados conjuntos de datos. Ya que para generar un gráfico, debes introducir los datos a través de una estructura determinada, por lo tanto donde he tenido más problemas ha sido en el proceso de transformar los datos a diversas estructuras.
Otro de los problemas más destacados que he tenido, va relacionado a los parámetros de filtrado que he añadido a las hipótesis. Como he definido anteriormente, la programación de hipótesis consiste en aplicar unos cálculos estadísticos a todos los sistemas y posteriormente filtrarlos en base al resultado del parámetro que queremos filtrar. Por lo tanto, hay que saber en que intervalos de valores filtrar, para ello, hay que analizar el resultado de varios sistemas buenos respecto al parámetro que hemos calculado.

Por ejemplo: Si aplicamos el cálculo de *Profit Factor* (porcentaje de beneficios respecto a las pérdidas) a todos los sistemas, debemos analizar en que rango se situan los 5 sistemas con más profit factor para poder escoger un intervalo que sea bueno.

Profit Factor
2.988829
2.584176
2.988829
2.121234
2.064966

Figura 13: Filtraje por profit factor

Una vez consultado, podemos decidir en ajustar el rango en:

$$1{,}7 < profitfactor < 3.$$

9.4. Construcción del *Dashboard*

Tras un conjunto de pruebas para escoger los gráficos mas representativos, decidí construir cada dashboard por separado. No tuve grandes problemas para generar los dibujos, ya que

Plotly facilita su implementación.

El principal problema fue como fusionar todos estos gráficos en una misma función, así reparovecharla para todas las hipótesis, es decir, como automatizar la generación de dashboard.

Finalmente, añadí todos los gráficos del dashboard en una única función para utilizar los mismos parámetros y facilitar la generación de los gráficos.

```
import plotly.express as px
    df_portfolio_graph = df_performances.loc[top.index]
["ConsolidatedPerformance_Curr"]
df_filter =
df_portfolio_graph.loc[df_portfolio_graph.index.get_level_values(leve
l = 'TradeDate') >= time]


df_filter = df_filter.cumsum()


fig = px.line(df_filter, x=df_filter.index.get_level_values(level =
'TradeDate'), y="ConsolidatedPerformance_Curr",
color=df_filter.index.get_level_values(level = 'IdStrategyVersion'))
    fig.update_layout(
        title="Rendimiento acumulado de cada sistema durante el
periodo",
        xaxis_title="Fecha",
        yaxis_title="Rendimiento €",
        legend_title_text='Sistemas'
    )
fig.show()
```

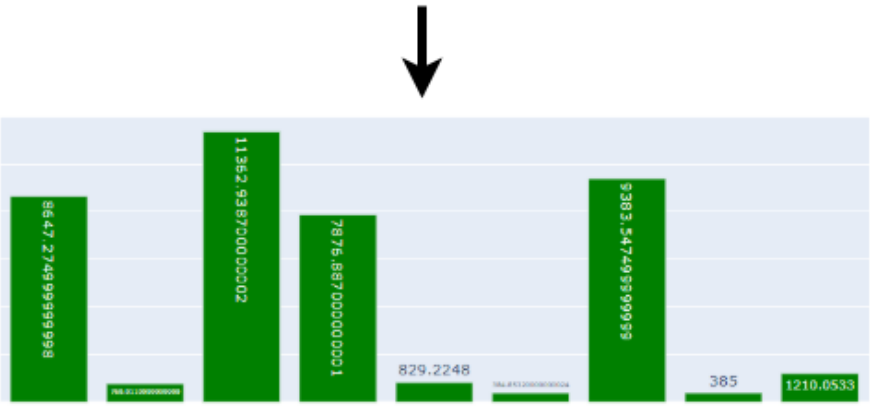


Figura 14: Transformación de código a Dashboard

9.5. Simular algoritmos seleccionados

No he tenido grandes problemas porque el proceso de simulación es similar al que he desarrollado para programar las reglas de predicción. Para ello, he ejecutado cada hipótesis

para extraer los "top 10"sistemas de cada una y simularlos.

```
top10newest = newestHipotesis()

top10Ttest = ttestHipotesis()

top10ProfitFactor = ProfitFactorHipotesis()

top10lowvaolatility = lowVolatilityHipotesis()

top10antiquity = antiquityHipotesis()
```

Figura 15: Top10 sistemas de cada hipótesis.

La función que simula todas las hipótesis para compararlas,la he llamado *PerformanceTest*, esta recibe por parámetro el rango de tiempo que se especifique, entonces calcula los sistemas ideales para cada hipótesis y muestra por pantalla los ID. También realiza la simulación de cada hipótesis durante este periodo y a través de un gráfico y una tabla podemos ver el rendimiento de cada regla de predicción.

```
time = (datetime.datetime.now() - datetime.timedelta(days=6*30))
performanceTest(time)
```

Figura 16: Ejemplo de como ejecutar la simulación

He realizado varias pruebas las cuales podemos visualizar en el apartado de *Implantación y resultados* (10.5).

¿Con cuanto dinero hay que simular la cartera de sistemas de cada hipótesis?
Esta fue una de las dudas que me surgieron cuando quise simular las hipotesis. Entonces lo hablé con Víctor, y él me pasó la fórmula que utilizar para generar un valor equilibrado que se acomode a la cartera, ya que cada sistema tiene un *Capital Recomendado* diferente, así que hay que adecuar uno a la cartera. La fórmula se basa en:

$$3,6 * \text{ máximo drawdown}$$

Y posteriormente, aproximar 5k por encima.

10. Implantación y resultados

En esta sección, comentaremos los resultados finales y su implantación.

10.1. Análisis de los datos

Los resultados finales tras los procesos de análisis de datos son un conjunto de gráficos y estructuras que describen el tipo de datos que estamos tratando.
A través de esta investigación conoceremos aquellos conceptos que representan a los sistemas de TradingMotion y también aprenderemos como consultar el rendimiento de los sistemas de forma manual y a través de librerías preparadas.

10.1.1. Análisis sobre los sistemas

Estudio rápido sobre los datos generados (desde el 02/01/2001 hasta el 22/02/2019) de la tabla *Strategies*.

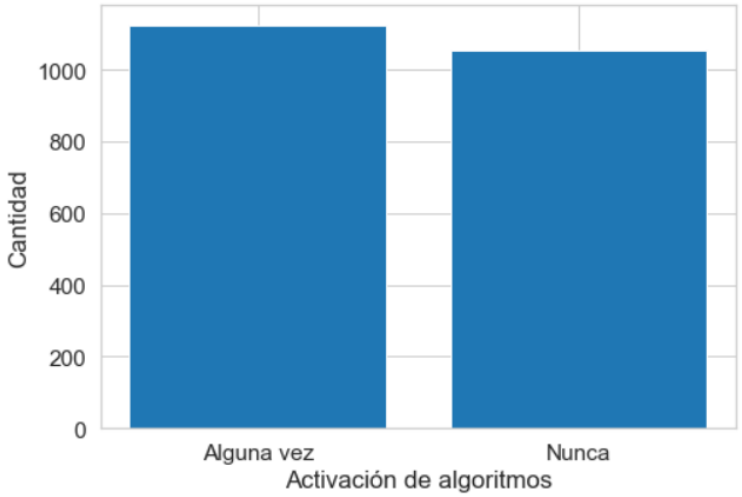


Figura 17: Gráfico de barras de la activación de sistemas

De los 2179 sistemas, 1124 han sido activados por algún cliente en algún momento y 1055 nunca se han operado en el mercado financiero.

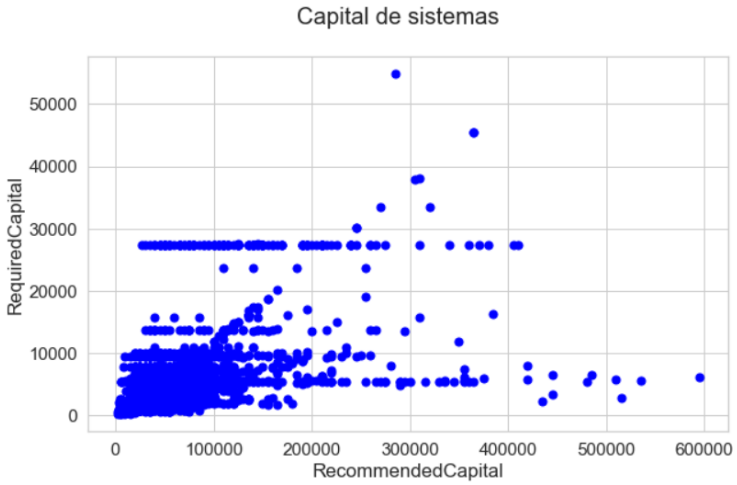


Figura 18: Capital Recomendado VS Requerido de uso de sistemas

La mayoría de sistemas se sitúan entre 0 y 10.000 /$ en capital necesario, en cambio 0 y 100.000 /$ en capital recomendado.

	Desviación estándar	Media	Max
Capital Recomendado	66401	66485	595000
Capital Requerido	6351	6133	54900

Cuadro 2: Capital Recomendado VS Requerido de uso de sistemas

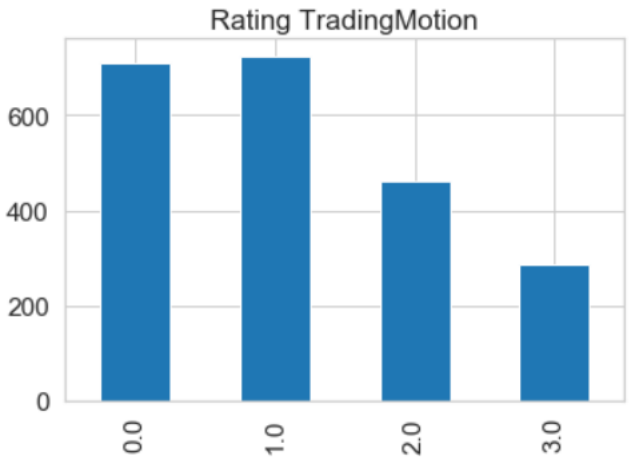


Figura 19: Clasificación de sistemas según su *Rating TM*

Los sistemas de Tradingmotion se puntúan según ciertos criterios, donde el 3 representa la mejor puntuación y el 0 la peor. En cuanto a *rating*, solo 285 sistemas son considerados de puntuación 3, 462 de puntuación 2, 724 de puntuación 1 y 709 de 0.

Figura 20: Cantidad de sistemas publicados a lo largo del tiempo

La activación de sistemas empezó en el 2004, ha ido progresando a un buen ritmo, tanto que los últimos 3 años se publican entre 400-500 sistemas creados por varios desarrolladores.

10.1.2. Análisis sobre rendimientos de todos los sistemas

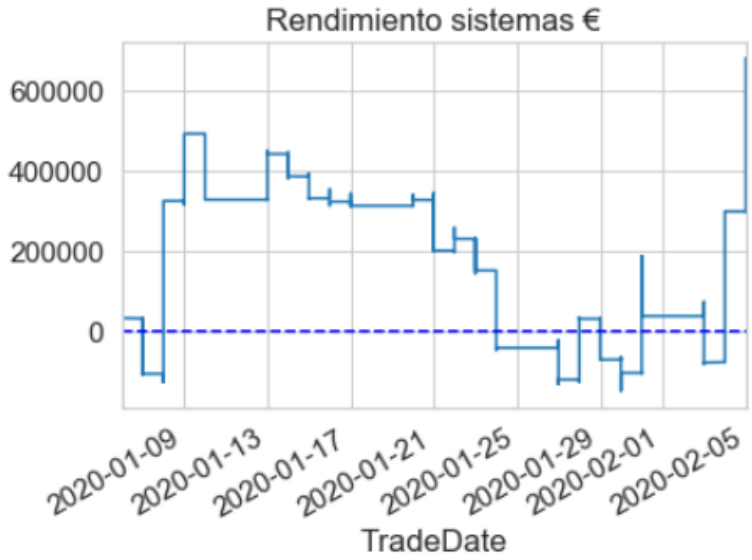


Figura 21: Rendimiento acumulado de todos los sistemas de TM

El gráfico de la figura 21, representa el rendimiento acumulado de todos los sistemas activos de la plataforma durante 1 mes (05-01-2020 hasta 05-02-2020). Con rendimiento acumulado nos referimos a la suma total de beneficios/pérdidas de los sistemas. A través de este gráfico podemos analizar que el rendimiento total de los sistemas no es estable, ya que cuando un sistema está ganando, otro puede estar perdiendo mucho más dinero que el otro. También podemos concluir que hay periodos donde la mayoría de sistemas están ganando (hay más beneficio total) y otros donde muchos pierden (se pierde dinero).
Esto es debido a la aleatoriedad y inestabilidad del mercado, justo en el mes de marzo de este año, la volatilidad de los mercados financieros está causando muchos daños. Con desplomes tan importantes como el del petróleo, que llegó a perder un 9 %. Algunos cambios bruscos son detectados por sistemas bien desarrollados, estos transmiten más estabilidad y seguridad para el cliente.

-1	7616
0	29421
1	9726

Figura 22: Rendimiento acumulado de todos los sistemas de TM

En la figura 22 tenemos las posiciones diarias en las que se han quedado todos los sistemas en el cierre de mercado. Durante el mes del gráfico anterior, hemos encontrado 7616 sistemas en posición corta (vendiendo futuros), 29.421 sistemas en posiciones estable (ni compra ni vende) y 9726 estrategias en posición larga (compra de futuros).

10.1.3. Análisis sobre rendimientos de un sistema

He seleccionado el sistema Ibex Plus Kennedy ya lleva una buena racha de beneficios y actúa sobre el Ibex, que es el principal índice bursátil de referencia de la bolsa española.

A continuación vamos a analizar el rendimiento del sistema en los últimos meses.

Figura 23: Rendimiento del Ibex Plus Kennedy

En el gráfico de la figura 23 podemos visualizar el comportamiento del sistema en los últimos 4 años. La línea negra indica el día que se activó el sistema. Todos los datos generados a la izquierda de la fecha de publicación del sistema, pertenecen al periodo de *Backtest*. El periodo

de **backtest**, es una simulación del comportamiento del sistema si hubiera estado activo durante etapas anteriores.

Analizando el gráfico de rendimiento, visualizamos que el rendimiento no es tan bueno como cuando simulamos el backtest, aunque sigue acumulando beneficios a lo largo del periodo. Para ello, vamos a examinar el gráfico de la distribución de retornos de los dos periodos:

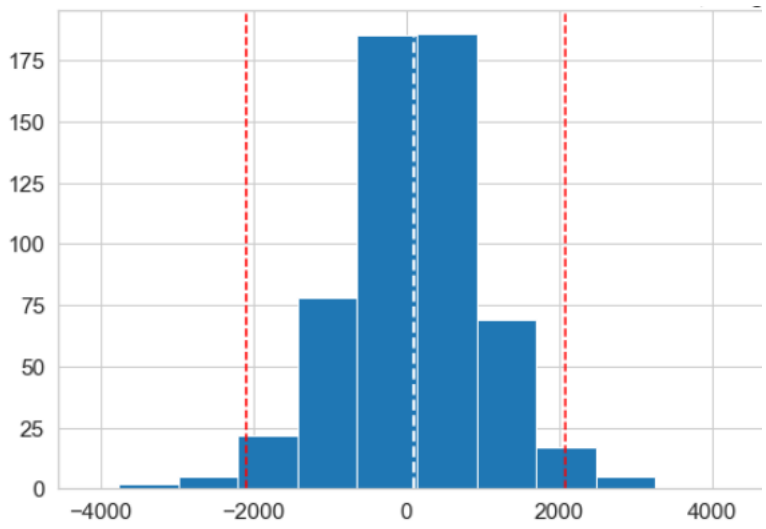


Figura 24: Distribución normal del sistema en el periodo de backtest: MEDIA: 107.38 2sigma: ±2080.28

El gráfico de la figura 24 representa la distribución normal del período de *backtest* de regreso diario. Hemos obtenida que la media de todos los retornos es de 107.38 (de beneficio). Según la regla 68-95-99.7, los valores situados entre 2sigma, pertenecen al 95 % de los retornos *normales* (son aquellos delimitados por las líneas rojas del gráfico). Estos corresponden aquellos valores entre ±2080.28. Por lo tanto si un sistema gana o pierde más de 2080,28, podemos afirmar que su rendimiento está fuera del intervalo de confianza y por lo tanto es un comportamiento fuera de lo esperado.

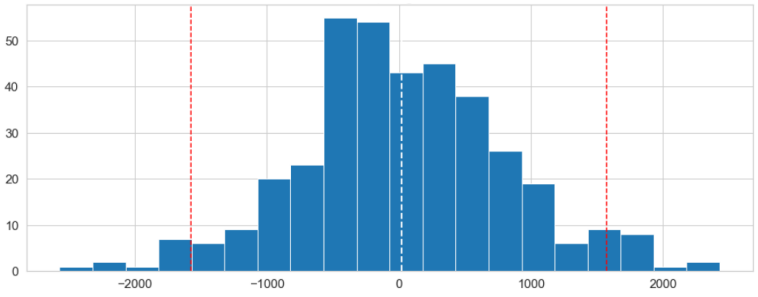


Figura 25: Distribución normal del sistema en el periodo activo: MEDIA: 24.10 2sigma: ±1572.19

En el dibujo de la figura 25, podemos ver que la normal abarca menos rango pero que los resultados son más dispersos. La media de los retornos es de 24,10 y el intervalo de 2sigma se ve reducido a ±1572.19.

Que pasa cuando comparamos el periodo activo con el de backtest?

Si comparamos las normales, distinguimos que la distribución normal del periodo activo ha variado respecto el periodo de *backtest*. La media de los retornos ha disminuido drásticamente a 24 y el rango de valores se ha reducido. Respecto a este cambio, parece que los parámetros del sistema están sobre-optimizados, y por lo tanto podemos afirmar que este sistema que sigue con beneficios no es tan bueno como parecía, ya que la normal ha variado y los resultados están siendo diferentes a los esperados/anteriores.

10.1.3.1 Análisis con pyfolio

Pyfolio es una biblioteca de Python para el análisis de rendimiento y de riesgo de carteras financieras. También permite dividir los periodos en backtesting y periodo activo.

```
import pyfolio as pf


#Obtenemos diarios
df_rtn_capital = df_rtn


#Añadimos el capital recomendado de nuestro sistema para el análsis
df_rtn_capital[0] = df_rtn_capital[0] + df_strategies.loc[21505]
["RecommendedCapital"]
df_rtn_capital_percentage =
df_rtn_capital.cumsum().pct_change().fillna(0)


#Indicamos y transformamos las fechas simulacion del sistema
df_rtn_capital_percentage.index =
pd.to_datetime(df_rtn_capital_percentage.index).tz_localize('Europe/M
adrid')
pf.create_returns_tear_sheet(df_rtn_capital_percentage, \
                             live_start_date=df_strategies.loc[21505]
["ReleaseDate"])
```

Figura 26: Codigo para generar análisis con Pyfolio.

Con este codigo dividimos los periodos en ***In-sample*** (backtest) y ***out-of-sample*** (activo). Al ejecutarlo nos muestran varios ratios estadísticos muy enfocados a cartera financieras como por ejemplo:

Start date			2016-05-30
End date			2020-02-05
In-sample months			27
Out-of-sample months			17
	In-sample	**Out-of-sample**	**All**
Annual return	34.3%	4.5%	21.6%
Cumulative returns	94.8%	6.8%	108.1%
Annual volatility	19.4%	9.5%	16.3%
Sharpe ratio	1.61	0.52	1.28
Calmar ratio	2.40	0.52	1.51
Stability	0.93	0.15	0.90
Max drawdown	-14.3%	-8.8%	-14.3%
Omega ratio	1.38	1.09	1.29
Sortino ratio	2.87	0.76	2.22
Skew	3.91	0.08	4.11
Kurtosis	53.03	0.40	66.80
Tail ratio	1.03	1.12	1.13
Daily value at risk	-2.3%	-1.2%	-2.0%

Figura 27: Tablas estadísticas de los retornos de Pyfolio

- Calmar ratio Medir rentabilidad y riesgo.
- Sharpe ratio Mide el exceso de rendimiento por unidad de riesgo de una inversión. Es decir, mide lo que le compensa al inversor por asumir riesgo en su inversión.
- Tail ratio Proporción entre el percentil 95 y el 5 (absoluto) de la distribución diaria de retornos. Un tail ratio de 0.25, por ejemplo, significa que las pérdidas son cuatro veces más malas que las ganancias.

También nos facilita el *máximo drawdown*, que es la máxima pérdida del sistema en una periodo de inversión.
Como hemos concluido anteriormente, el sistema en el periodo *Out-of-sample* tiene peores resultados que en *In-sample*, ya los indicadores de ratio puntuan que tiene más riesgo la inversión de ahora, que la del periodo de *backtest*. Por ejemplo, el *sharpe ratio* antes era de 1.61 (menos riesgo) y ahora es de 0.52 (más riesgo).

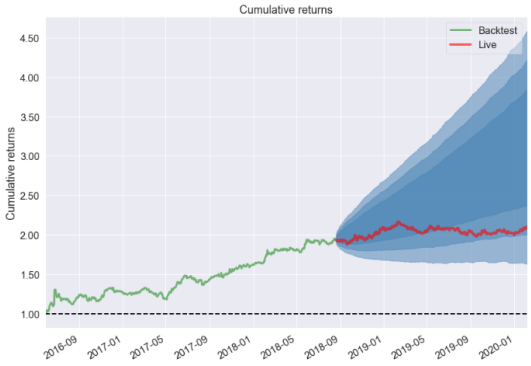


Figura 28: Gráfico predictivo periodo out-of-sample

Otro gráfico muy útil es el de la figura 28 donde hace una previsión de cómo debería comportarse el sistema a partir de la fecha de activación.

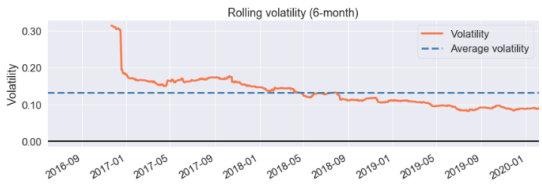


Figura 29: Gráfico de volatilidad

En el caso de la volatilidad, el intervalo de confianza es más pequeño (95 % ś1572), y el riesgo también se ha visto reducido, junto con los beneficios.

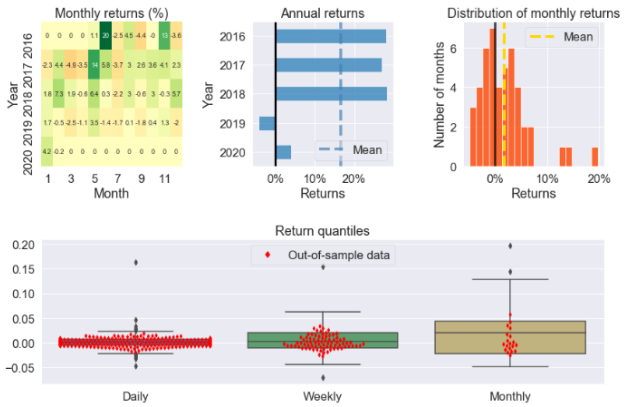


Figura 30: Gráfico de volatilidad

Podemos visualizar el porcentaje de retornos mensuales y anuales, el promedio anual y la distribución porcentual. En el último gráfico podemos apreciar el posicionamiento de los datos *out-of-sample* respecto los *in-sample*.

10.2. Preparar los datos

Los resultados finales del proceso de *preparación de datos* son un seguido de funciones que nos permiten iniciar la programación de reglas de predicción sin tener que volver a programar alguna proceso anterior.

```
# Obtenemos el nombre de un sistema a través de su ID.
def getSystemName(df, idSystem):
    name = df.loc[idSystem]['StrategyVersionName']

    return name
```

Figura 31: Función getSystemName

En la figura 31, tenemos un ejemplo de una función que se encarga de devolver el nombre completo de un sistema a partir de su ID.

```
# Filtramos los "trades" del dataset a partir de una fecha, hasta la
actualidad.
def filterCartera(df, time):
    df_filter = df.loc[df.index.get_level_values(level = 'TradeDate')
>= time]

    return df_filter
```

Figura 32: Filtraje por profit factor

En la figura 32, la función filtra todos los retornos a partir del parámetro de tiempo hasta el dia de hoy. Por lo tanto si decidimos que el parámetro es el 5/2/2020, devolverá todos los rendimientos de sistemas entre el 5 de febrero de 2020 y hoy.

En coclusión hemos generado un seguido de funciones que agilizan el proceso de *desarrollo*.

10.3. Reglas de predicción

En esta sección aplicaremos el método científico de David Aronson a las hipótesis planteadas.

- **Clasificación de sistemas según la diferencia de la distribución *out-sample* comparada con la *in-sample* aplicando la prueba de T de Student, + filtro de análisis de riesgos (Sharpe ratio).**

  Observación
  Comparando la similitud de las distribuciones del rendimiento de los sistemas en los períodos *out-sample* (periodo donde el sistema permanece activo) y *in-sample* (todo el periodo de Backtesting), podemos saber si el sistema es más o menos bueno.

  Generar hipótesis
  Conseguir semejanza en las dos distribuciones (evidentemente tienen que ser distribuciones con beneficios), para poder afirmar que es fiable el *backtest* del sistema sistema, ya que entonces este actuará según lo esperado. A la hipótesis, le añadimos un pequeño filtro de *Sharpe Ratio* para medir el análisis de riesgos.

  Predicción
  Es importante que los dos periodos sigan distribuciones parecidas, ya que el periodo de backtest tiene que servir como base predicción para el rendimiento del sistema en activo. Gracias a esta hipótesis, podemos detectar la *sobreoptimización* en el desarrollo de sistemas de trading.

La **sobreoptimización**, en inglés *curve-fitting*, se produce cuando un algoritmo de trading está tan ajustado a las condiciones del los datos históricos, que funciona muy bien en el *backtest*, pero que durante el periodo en activo fracasa. Esto es debido a que los parámetros del algoritmo de trading están tan optimizados al *backtest*, que luego no funcionan bien.

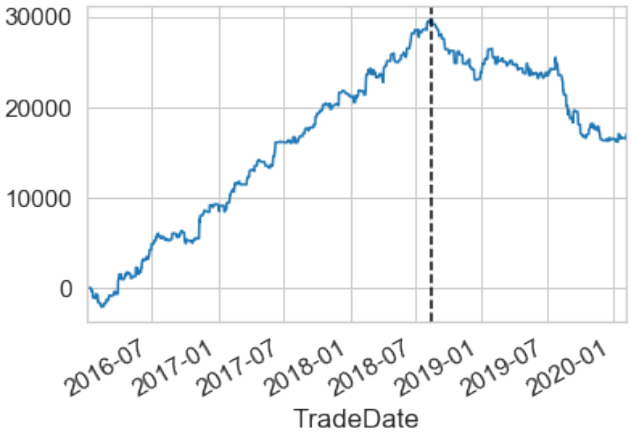


Figura 33: Ejemplo sistema sobreoptimizado, NEWTON ST 707 RUSSELL.

Con el gráfico de la figura 33 apreciamos un claro ejemplo de sobreoptimización, donde a partir de la activación el sistema empieza a perder. Aunque estadísticamente, el sistema ha ganado el 53.9 % de las sesiones, el sistema es malo porque está perdiendo mucho dinero.

Para afirmar la similitud de las distribuciones, aplicamos la Prueba del T de Student. La prueba del T de Student es una prueba estadística que se utiliza para determinar si hay una diferencia significativa entre las medias de dos grupos. En este caso, para comparar las dos distribuciones.

El t-test lo aplicaremos con una librería estadística (SciPy), que nos devolverá dos valores que utilizaremos para filtrar los sistemas que nos interesen.

- El **t-value**, T es simplemente la diferencia calculada que representa el error estándar (en unidades), cuánto más bajo el valor, más probable que las distribuciones sean iguales.
- El **p-value**, es la confiabilidad del test. La probabilidad de que los resultados de los datos de su muestra sean casualidad. Por ejemplo un p-value del 5 %, es 0.05. Esto indica que los datos no ocurrieron por aleatoriedad.

Verificación

Tras ejecutar la simulación de los sistemas escogidos por la hipotesis, obtenemos un rendimiento positivo.

Figura 34: Retornos acumulados de los sistemas de la hipótesis.

En la figura 34, analizamos las ganancias y pérdidas durante el intervalo de tiempo de la simulación. Al principio nuestra cartera perdió dinero, pero a partir del 1 de enero de 2020, empezamos a obtener rentabilidad y a generar beneficios constantes.

IdStrategyVersion	StrategyName	ProductName	Performance	Profit Factor	Sharpe Ratio
22293	SKN mas E-mini Nasdaq 5 min DRAGON	E-mini Nasdaq	-3215.0457	0.407873	-5.174862
22150	SKN MAS BUND 60 min NORMAL	BUND	-592.9324	0.988166	-0.071168
21443	NEWTON ST 707 SP MID CAP	S&P MidCap	2121.5783	1.465649	1.118817
22183	Crude Oil Abel Firme Step Stop-T.Profit Contin...	Crude Oil	4871.4284	1.365189	1.684908
21227	SAMURAI SP	E-mini S&P	430.0673	1.050376	0.264140
21704	MANDALA GASOLINE	RBOB Gasoline	-33.1953	1.008924	0.020228
21341	SwingGribor _ ICE Sugar #11 SB11	ICE Sugar #11	1509.8340	1.542571	2.648949
21393	NEWTON ST 707 NASDAQ	E-mini Nasdaq	1835.6911	2.398176	1.766062
21492	Mini Nasdaq Profit Kennedy 4	E-mini Nasdaq	12050.5850	1.838441	2.996899
21381	CIRUS RZ SUGAR BLACK	ICE Sugar #11	1406.4684	1.468818	2.349933

Figura 35: Resultados de los sistemas de la hipótesis.

La hipótesis ha seleccionado los 10 sistemas que tenian las distribuciones más parecidas. En el campo de *performance* de la figura 35, podemos ver la cantidad de dinero que generado o perdido cada sistema de la cartera. Los sistemas estan bastante diversificados respecto a los productos (excepto que hay 3 que actúan sobre el *E-mini Nasdaq*), áyoria

tiene rentabilidad positiva y ha ganado más de 1000.

Conclusión
Tras la aplicación de la hipótesis podemos concluir que la similitud de las distribuciones es un factor clave para afirmar que poseemos un buen sistema, ya que de esta manera te aseguras que se ha *testeado* durante el desarrollo y está progresando adecuadamente en activo.

- **Ranking de sistemas ordenados por profit-factor, Rating TM y Sharpe Ratio, con un mínimo de 1 año funcionando en activo.**

Observación
El profit-factor (relación entre ganancias y pérdidas) es un buen indicador de aquellos sistemas que han generado beneficios (>1) o pérdidas (<1). Si a esto, le añadimos otros filtros como el Sharpe Ratio y el Rating TM (rating calculado por *TradingMotion*), podemos encontrar una buena cartera de sistemas.

Generar hipótesis
Combinar los tres filtros para generar un pequeño conjunto de sistemas, y ordenar el resultado por profit-factor.

Predicción
El profit-factor es un elemento clave para medir el rendimiento, con solo este factor podemos filtrar muchos sistemas que han ganado a lo largo de toda la simulación. Además añadimos el Rating TM y el Sharpe Ratio para asegurarnos que el sistema seleccionado tiene futuro. Ya que podemos tener un sistema que gane mucho, pero que se expone a un riesgo excesivo, y en activo no esté generando los beneficios esperados. El *Rating TM* se calcula en base al peor *drawdown*, resultado y cantidad de tiempo que lleva.
Es evidente que los 3 filtros pueden estar un poco correlacionados, ya que si el sistema tiene un buen profit-factor, es probable que también tenga un buen Rating TM Pero cada filtro se centra más en una propiedad, por lo tanto si unimos los tres, conseguimos una combinación interesante.

Importante filtrar solo los sistemas que llevan 1 año en activo, ya que hay varios sistemas que operando solo en el backtest han llegado a conseguir profit-factor super elevados, pero que en activo han decaído drásticamente.

Verificación

Para verificar, hemos filtrado los mejores sistemas respecto a los parámetros anteriormente mencinados y hemos simulado una cartera:

Figura 36: Retornos acumulados de los sistemas de la hipótesis.

En la figura 36, podemos ver que hemos obtenido beneficios. El rendimiento ha sido un poco irregular, ya que a partir de noviembre de 2019 hemos perdido dinero.

IdStrategyVersion	ProductName	Performance	Profit Factor	Sharpe Ratio
21635	Silver	5012.8941	1.211429	0.804981
21811	Mini-Dax	647.9838	1.105666	0.390390
21608	FTSE MIB	353.7725	1.054332	0.147364
21251	Gold	-962.8450	0.924050	-0.234188
21262	Silver	15964.9023	1.381529	1.838507
12418	E-mini S&P	8046.1767	1.453236	1.664550
21978	Lean Hogs	5114.9309	1.220438	0.738307
21287	RBOB Gasoline	-3016.5739	0.802506	-0.765813
21334	Lean Hogs	793.5853	1.012555	0.066978
21859	S&P MidCap	11283.0191	1.726471	1.690159

Figura 37: Resultados de los sistemas de la hipótesis.

Hemos filtrado 10 sistemas, la mayoría ha ganado () excepto dos, los cuales han perdido un total de 4000 entre los dos. Cabe destacar el sistema 21262, que ha ganado 16k durante la simulación. Considero que el rendimiento de algunos sistemas que se han filtrado es intermedio, ya que durante un periodo de 6-7 meses, muchos no han ganado

mas de 1000 euros.

Conclusión
Tras los resultados, concluimos que aplicando 3 filtros con unos parámetros razonables, obtenemos un conjunto robusto para obtener beneficios a **largo plazo**. Ya que son sistemas que funcionan muy bien y en activo siguen con beneficios (evidentemente menores que durante el backtest), pero que a la larga son fiables.

- **Sistemas con volatilidad baja, un buen profit factor y un filtro Sortino Ratio.**

Observación
La volatilidad es una medida de la amplitud de los movimientos de un determinado instrumento financiero. A mayor volatilidad, mayor amplitud de los movimientos. A mayor amplitud de movimientos, mayor incertidumbre y riesgo potencial. Pero si enfocamos la volatilidad a rendimientos, contra menos volatilidad más difícil obtener grandes variaciones en los rendimientos (valores atípicos en beneficios o pérdidas).

Generar hipótesis
Filtrar aquellos sistemas que tienen un índice de volatilidad anual bajo, que tienen un porcentaje de beneficios alto y un ratio Sortino elevado. Así asegurarnos que manejamos sistemas que obtienen beneficios sin exponerse a grandes riesgos (en cuanto a pérdidas).

Predicción
La volatilidad representa variación, por lo tanto con menos variación menor exposición al riesgo.

El profit-factor es el factor clave para filtrar aquellos sistemas que realmente están obteniendo beneficios, ya que podemos tener sistemas con una volatilidad baja, però con pérdidas constantes.

Añadir un filtro de Sortino Ratio es importante ya que también mide la volatilidad junto con el rendimiento. La relación de Sortino utiliza el mismo cálculo que la relación de Sharpe Ratio pero con una diferencia importante. El índice de Sortino solo tiene en cuenta la volatilidad a la baja de los rendimientos negativos y no considera la volatilidad al alza de los retornos positivos. Esto crea una medida mejor ajustada de riesgo más precisa del rendimiento comercial.

Esta hipótesis se me ha ocurrido analizando un sistema que actualmente está ganando bastante. He analizado sus resultados, y he consultado que tenía un alto profit-factor y Sortino Ratio. Además consultando la gráfica de rendimientos, la amplitud de los resultados era leve, por lo tanto la volatilidad es menor.

Figura 38: Ejemplo sistema base volatilidad baja

Verificación

En apenas 2-3 meses el rendimiento de la cartera de sistemas con volatilidad baja es espléndido.

Figura 39: Retornos acumulados de los sistemas de la hipótesis.

En la figura 39, vemos que al principio es un poco irregular, pero posteriormente adquiere constancia en los beneficios durante el resto del periodo.

IdStrategyVersion	StrategyName	ProductName	Performance	Profit Factor	Sharpe Ratio
17519	Joseja 2 mini nasdaq	E-mini Nasdaq	10192.2781	2.328121	1.735605
12373	LCH Howard S	Heating Oil	9193.9514	2.455542	2.656522
12296	Atom Dax Dynamic	DAX	21175.0399	4.001052	2.626986
17521	Joseja 2 Mini Dow	E-mini DowJones	5020.9745	5.416027	2.588758

Figura 40: Resultados de los sistemas de la hipótesis.

La hipótesis solo filtra 5 sistemas, estos han tenido un sharpe ratio elevado, por lo tanto significa que son mas seguros (han tenido menos riesgo).

Conclusión
Desafortunadamente en el mundo de la inversión para obtener más rentabilidad hay que aceptar más volatilidad y riesgo. Cuanto más alta es la desviación estándar, mas alta es la volatilidad y el riesgo. Pero en este caso, como queremos mantener una cartera de sistemas segura, que a la larga genera beneficios sin arriesgar excesivamente, esta hipótesis es ideal.

- **Sistemas antiguos (+ de 2 años en activo) con grandes beneficios a lo largo del tiempo, filtrados por Sharpe Ratio, máximo drawdown y TMRating.**

Observación
Es muy bueno que un sistema aporte estadísticas de al menos de 10 años, y si incorpora el año 2008 en su *backtesting* mejor (ese año hubo una crisis) y es conveniente someter al sistema a situaciones extremas del mercado para ver su comportamiento. Por lo tanto los sistemas que llevan un tiempo funcionando bien, son candidatos a ser incluidos en nuestra cartera.

Generar hipótesis
He decidido filtrar todos los sistemas que llevan en activo más de 2 años, ya que considero que si en dos años no han dejado de funcionar, es porque utilizan una buena base estadística/matemática. He decidido volver a aplicar el ratio *Sharpe*, ya que es una propiedad muy representativa del rendimiento. A esto, le he añadido un filtro de máximo *drawdown* para evitar algún sistema que sea muy agresivo para las pérdidas, y un parámetro para analizar el *TMRating* de los sistemas seleccionados.

Predicción
Al filtrar sistemas que llevan tiempo en activo, estamos intentando predecir sistemas

regulares, son aquellos que son constantes, con pequeños o grandes beneficios mensuales, pero que no han decaído notablemente a lo largo del tiempo.

Con estos filtros intentamos encontrar sistemas similares al *Shooter Nasdaq*. Se activó hace dos años, y tiene un profit factor de 1.32.

Figura 41: Ejemplo sistema base volatilidad baja

Es bueno ya que hace tiempo que opera en activo y obtiene beneficios. Aunque solo gana el 48 % de las sesiones, no tiene grandes caídas (*drawdown*), y mantiene la acumulación de beneficios respecto las pérdidas.
Cabe añadir que hay buenos sistemas que llevan poco tiempo en activo (están basados en nuevas tecnologías, muy recientes) y no podemos *testear* más atrás en el tiempo, pero la robustez de un sistema hay que evaluarlo en periodos amplios por estadística. A más datos, estadísticas más precisas y mayor robustez.

Verificación
En la simulación de los sistemas *antiguos*, obtenemos grandes beneficios a lo largo del periodo de 5-6 meses.

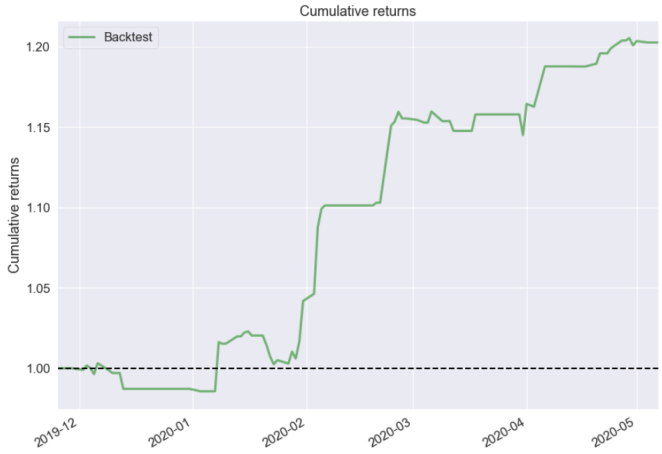


Figura 42: Retornos acumulados de los sistemas de la hipótesis.

En la figura 42, analizamos que la cartera de sistemas no es muy agresiva, hay periodos de tiempo en los que ni se gana ni se pierde, pero suele ser bastante constante en cuanto a no tener grandes pérdidas.

IdStrategyVersion	StrategyName	ProductName	Performance	Profit Factor	Sharpe Ratio
17519	Joseja 2 mini nasdaq	E-mini Nasdaq	10192.2781	2.328121	1.735605
12373	LCH Howard S	Heating Oil	9193.9514	2.455542	2.656522
12296	Atom Dax Dynamic	DAX	21175.0399	4.001052	2.626986
17521	Joseja 2 Mini Dow	E-mini DowJones	5020.9745	5.416027	2.588758

Figura 43: Resultados de los sistemas de la hipótesis.

Para este periodo, la hipótesis solo ha seleccionado 4 sistemas (figura 43), eso significa que el filtro que he añadido es bastante restrictivo. En este caso beneficia mucho, porque todos han ganado cantidades elevadas durante este intervalo de tiempo.

Conclusión

Tras obtener los resultados, la selección de sistemas *antiguos*, junto con unos filtros determinados, obtenemos una exitosa cartera de sistemas. Para esta hipótesis es importante tener en cuenta que el filtro de sistemas es mas *duro*, es decir que se asegura que los sistemas que escoge tienen un alto *profit factor* que ha mantenido durante más de 2 años.

- **Sistemas con mayor profit-factor, con mas de 3 meses en activo y un filtro de drawdown (Sistemas nuevos).**

Observación

Este es un campo que también evoluciona, surgen nuevas tecnologías y teorías, es un sector que se re-inventa. Por lo tanto, es importante apostar por los sistemas nuevos, aquellos que han actuado correctamente durante el *backtest* y que su participación en el periodo activo ha sido positiva.

Generar hipótesis

He considerado que a partir de 3 meses en activo, si no ha tenido grandes pérdidas, es probable que siga la tendencia de *backtest*. A esto le he añadido un filtro de *máximo drawdown*, ya que muchos de los sistemas cuando se activan, empiezan a perder debido a la sobreoptimización.

Predicción

Al limitar el filtraje a todos los sistemas que llevan mas de 3 meses en activo, estamos permitiendo analizar aquellos sistemas que tal vez llevan entre 3 y 12 meses con muy buena racha y que probablemente actuaran bien a largo plazo.

Figura 44: Ejemplo sistema que solo lleva un año operando.

En la figura 44, hace un año que el sistema está funcionando en activo y ha obtenido beneficios sin tener un gran *drawdown*.

Verificación

Después de simular el algoritmo de los sistemas "nuevos", debemos verificar la salida de

la ejecución de la regla de predicción:

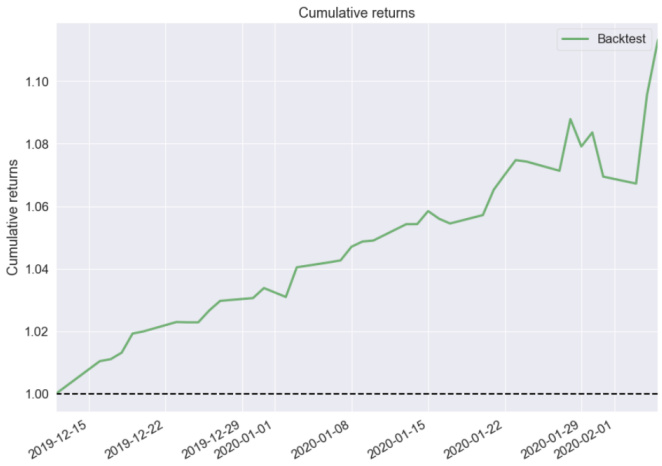


Figura 45: Retornos acumulados de los sistemas de la hipótesis.

En la figura 45, analizamos que en cuestión de 2-3 meses la cartera de sistemas ha obtenido beneficios de forma constante, prácticamente de forma lineal, cosa que tiene mucho valor, ya que no tener un gran *drawdown* determina seguridad.

IdStrategyVersion	StrategyName	ProductName	Performance	Profit Factor	Sharpe Ratio
22090	SKN Selene Crude oil 5 min	Crude Oil	11583.7985	2.899613	5.625643
22041	Mini S&P Ancorma stop-take profit intraday	E-mini S&P	252.4308	1.341960	0.921318
21778	SAMURAI COPPER	Copper	7144.1286	3.078994	5.502113
22054	ROBUST S6 _ E.mini S&P ES	E-mini S&P	5406.8245	1.694517	2.581048
22168	SKN OLLB miniSP 60 min STAIRWAY TO HEAVEN	E-mini S&P	1415.7592	1.666276	1.088276
22055	ROBUST L7 _ E.mini S&P ES	E-mini S&P	2105.5752	1.382792	1.271208
17654	Google Trends Monthly ES	E-mini S&P	8295.8994	1.689587	2.983561
21352	SwingGriBor _ FTSE MIB IFS	FTSE MIB	2045.0000	1.241929	1.213306
21697	Liouville EuroStoxx	EuroStoxx	1902.7426	1.567465	2.471552
17519	Joseja 2 mini nasdaq	E-mini Nasdaq	5161.4116	4.319451	2.652926

Figura 46: Resultados de los sistemas de la hipótesis.

En este caso tenemos 10 sistemas, los cuales han ganado todos durante este periodo. Podemos ver que la mitad actúan sobre el E-mini S&P, cosa que habría que evitar debido a la posible correlación entre ellos. Los rendimientos se situan entre 250 y 11500, prácticamente podemos afirmar que la hipótesis es correcta ya que el resultado es muy bueno.

Conclusión

Vale la pena apostar por los nuevos sistemas con mayor *profit*, aunque no tengan un amplio *backtest*, si tienen una buena base matemática pueden resultar ser muy exitosos.

10.4. Construcción del *Dashboard*

Para la construcción del dashboard, he diseñado 4 gráficas, cada una representa un parámetro diferente del resultado:

- ***Time Series* del rendimiento acumulado de la cartera de sistemas.**

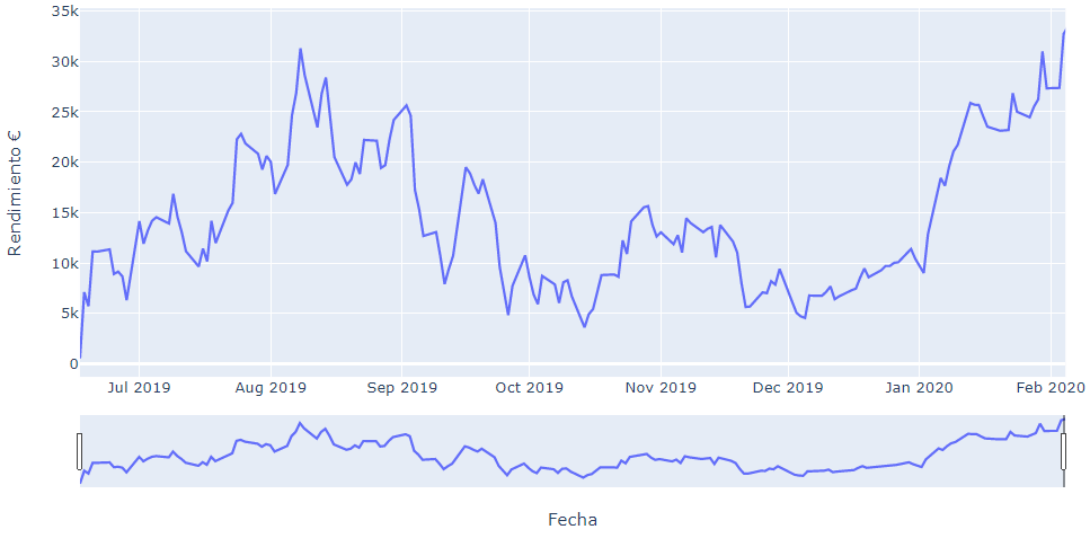


Figura 47: Timeseries del rendimiento acumulado.

En este plano, representamos las suma de los beneficios/pérdidas de todos los sistemas que mantenemos en la cartera durante un periodo de tiempo concreto. Así podemos consultar en qué periodos hemos incrementado los beneficios o hemos tenido más pérdidas. En las *timeseries*, Plotly permite ajustar el rango que queremos visualizar a partir de la barra situada justo debajo del gráfico (figura 47).

- **Diagrama de dispersión de los retornos de rendimientos diarios de la cartera.**

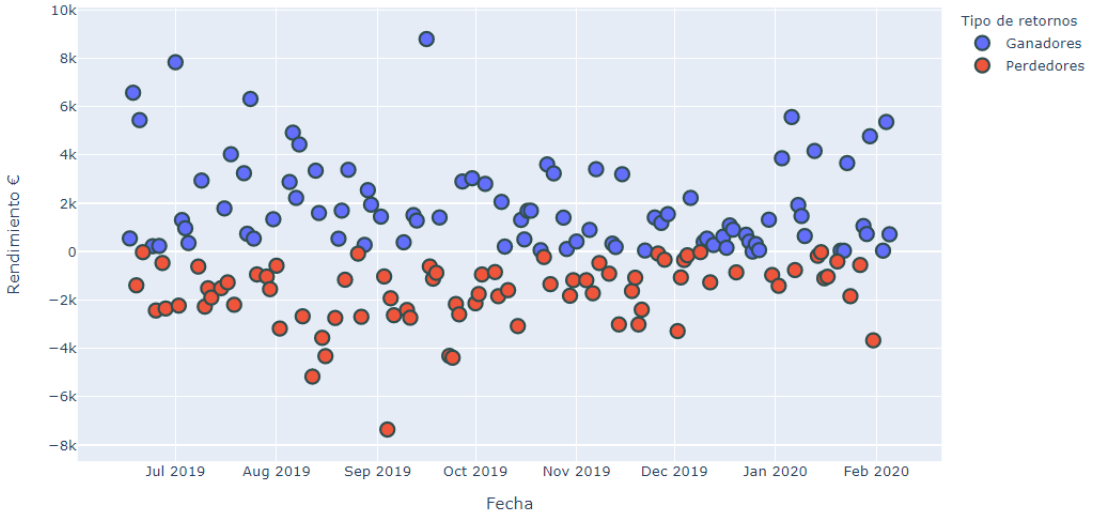


Figura 48: Diagrama de dispersión de retornos diarios de la cartera.

A través de este diagrama(48) podemos analizar el rendimiento diario de nuestra cartera. Cada punto representa a un dia, los puntos azules son los días en los que se han obtenido beneficios, y los rojos representan los días con pérdidas. Así podemos consultar que días hemos obtenido más beneficios o cuando se ha perdido dinero.

- **Gráfico de líneas del rendimiento acumulado de cada sistema**

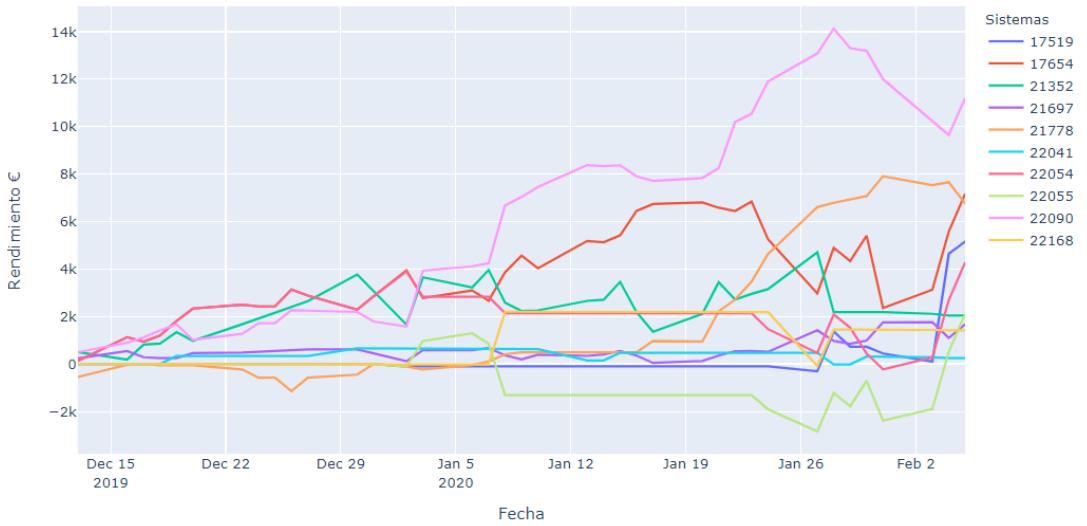


Figura 49: Ranking de *performance* de las hipótesis

En esta representación(49) dividimos el rendimiento de cada sistema para visualizar su comportamiento a lo largo del periodo. Es una buena manera para controlar la correlación entre sistemas, ya que podemos visualizar como se han comportado cada sistema en un día concreto.

- **Diagrama de barras sobre beneficio de cada sistema**

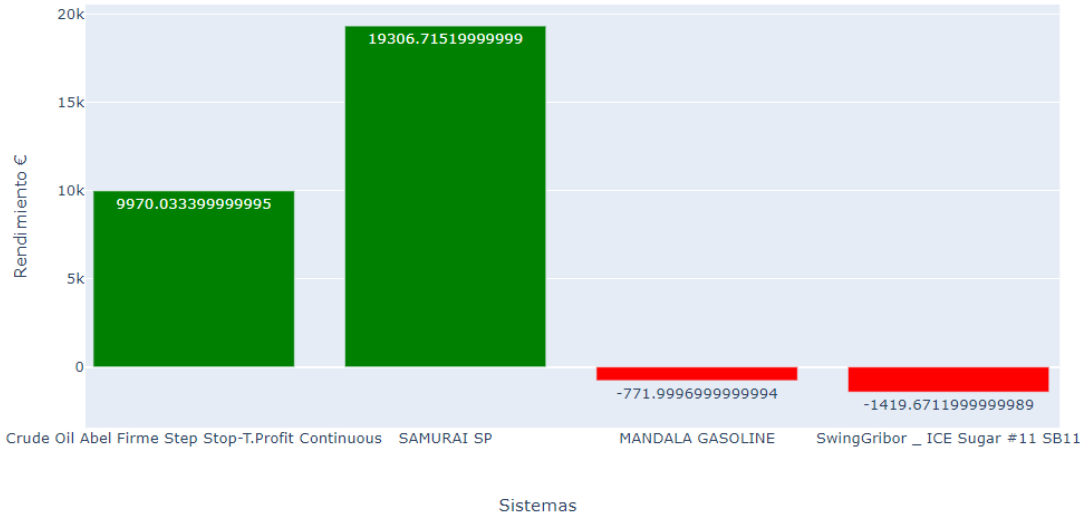


Figura 50: Diagrama de barras sobre *performances*

Con este dibujo, describimos el resultado final de cada sistema durante el periodo que hemos analizado. Aquellos sistemas en rojo son los que han tenido pérdidas, y los verdes los que han ganado dinero.

10.5. Simular algoritmos seleccionados

Una vez implementadas todas las hipótesis, debemos compararlas de alguna forma. Para ello he desarrollado un *script* que compara los rendimientos de todos las hipótesis en un mismo periodo de tiempo y devuelve un listado con los *IDs* de los sistemas escogidos por cada hipótesis.

Hemos ejectuado el scripts con la fecha entre 15/12/19 y el 15/05/20,
y este es el resultado:

- Sistemas más Nuevos: [22090, 22041, 21778, 22054, 22168, 22055, 17654, 21352, 21697, 17519]
- Comparación de distribuciones: [22150, 17919, 21361, 22293, 21031, 22183, 22266, 21443, 21704, 21341]
- Sistemas antiguos: [17654, 17519, 21147, 21218, 17653, 12373, 11088, 12296, 17521, 21212]

- Volatilidad baja: [21692, 22054, 21778, 21874, 22090]
- Profit Factor: [22103, 21828, 21766, 21635, 21733, 21702, 21251, 21608, 21734, 21978]

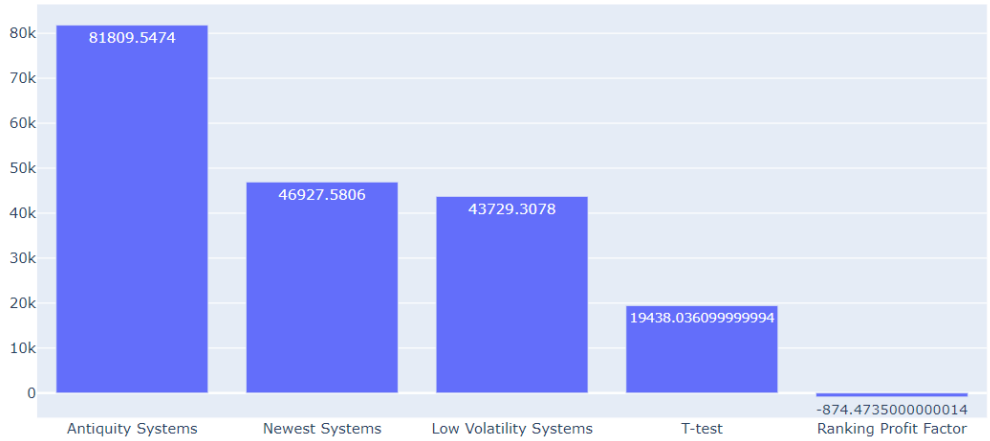


Figura 51: Ranking de *performance* de las hipótesis

	Hipotesis	Rendimiento(€)
Antiquity Systems	Sistemas antiguos	81809.5474
Newest Systems	Sistemas mas Nuevos	46927.5806
Low Volatility Systems	Volatilidad baja	43729.3078
T-test	Comparacion de distribuciones	19438.0361
Ranking Profit Factor	Profit Factor	-874.4735

Figura 52: Tabla de *performance* de las hipótesis

En este caso la hipótesis de los sistemas que llevan activos más de 2 años, es la que ha generado más dinero, y la hipótesis del ranking por *profit factor* la que menos. Esto depende mucho del tramo sobre el cual simulamos.

Si ampliamos el rango de tiempo para la simulación de los rendimientos obtenemos resultados diferentes. En este ejemplo la simulación se sitúa entre 15/05/19 y el 15/05/20.

Figura 53: Ranking de *performance* de las hipótesis

	Hipotesis	Rendimiento(€)
T-test	Comparacion de distribuciones	239405.1803
Newest Systems	Sistemas mas Nuevos	201352.9235
Antiquity Systems	Sistemas antiguos	183456.8259
Low Volatility Systems	Volatilidad baja	94055.2669
Ranking Profit Factor	Profit Factor	-3448.0111

Figura 54: Tabla de *performance* de las hipótesis

El resultado ha cambiado drásticamente para la hipótesis comparativa de distribuciones, analizando el periodo anual, detectamos que a finales del 2019, los sistemas seleccionados por la hipótesis del t-test consiguieron grandes beneficios.

Tras comprobar que las hipótesis han resultado exitosas y obtenemos beneficios, se probará estas cartera de sistemas de forma interna para ver como actuarían en vivo.

11. Conclusiones

Tras simular las hipótesis planteadas, he concluido que el diseño y la implementación de los procesos del proyecto se ha llevado a cabo de forma acertada.

Es la simplicidad del trabajo la que permite que el resultado haya sido muy positivo, ya que al basarnos en el análisis de los sistemas, junto a un buen filtraje de estos mediante hipótesis y algoritmos que no son muy complejos, permitimos obtener un resultado claro y muy fácil de interpretar a través de las tablas y gráficos proporcionados por el *dashboard*.

En conclusión, tanto el entorno seleccionado como los resultados finales, han cumplido con las motivaciones, propósitos y objetivos planteados al inicio del proyecto. La gestión, analítica y predicción de datos orientados a decisiones de inversión, supone una gran ventaja de cara a seleccionar los sistemas automáticos sobre los que invertir.

12. Futuro del proyecto

La cantidad de posibles hipótesis o técnicas de análisis estadísticas es muy amplia, cuanto más aprendes sobre trading algorítmico más ideas o deducciones te surgen. Por lo tanto si ampliamos el periodo de investigación, generaríamos más hipótesis

Durante este trabajo me he centrado mucho en la analítica descriptiva, ya que a partir de esta he podido elegir los sistemas ideales para la cartera. Pero en un futuro sería ideal añadir conceptos de *machine learning* (data mining) para la predicción de sistemas. Aplicar técnicas de ML para encontrar patrones de rendimiento o correlaciones durante periodos concretos. Ya que el aprendizaje automático ofrece la capacidad de realizar predicciones con precisión sobre situaciones que ya se han dado anteriormente.

13. Bibliografía

Referencias

[1] David Aronson, *Evidence-Based Technical Analysis: Applying the Scientific Method and Statistical Inference to Trading Signals*, Wiley Trading Book 274.

[2] Barry Johnson, *Algorithmic Trading and DMA: An introduction to direct access trading strategies.*

[3] Vic Patel, Forex Training Group, *5 Statistics For Analyzing Your Trading Performance.* Recuperado de: https://www.countingpips.com/2018/07/5-statistics-for-analyzing-your-trading-performance/.

[4] Cory Mitchell, *7 Statistics for Analyzing Your Trading System.* Recuperado de: https://vantagepointtrading.com/7-metrics-to- analyze-your-trading-system/.

[5] Oscar Cagigas, Onda4, *¿Por qué fallan los sistemas de Trading?.* Recuperado de: https://formandotraders.com/sistemas-de-trading/.

[6] Jean Folger, *Interpreting a Strategy Performance Report.* Recuperado de: https://www.investopedia.com/articles/fundamental-analysis/10/strategy-performance-reports.asp.

[7] CNMV - Comisión nacional del mercado de valores, *Sistemas de trading automatizados (STA).* Recuperado de: https://www.cnmv.es/DocPortalInv/OtrosPDF/ES-SistemasTrading.pdf

[8] *Empyrical API Reference.*
Recuperado de: http://quantopian.github.io/empyrical/appendix.html

[9] Plotly *Python Open Source Graphing Library.*
Recuperado de: https://plotly.com/python/

[10] Admiral Markets, *Everything You Need To Know About Automated Trading.* Recuperado de: https://admiralmarkets.com/education/articles/automated-trading/what-is-automated-trading

[11] Ivestopedia - Will Kenton, *Financial Markets.* Recuperado de: https://www.investopedia.com/terms/f/financial-market.asp

[12] Rankia - María Granel, *£Qué diferencias existen entre el mercado de opciones y futuros?.* Recuperado de: https://www.rankia.cl/blog/analisis-ipsa/3909492-que-diferencias-existen-mercado-opciones-futuros

[13] TODOBOLSA, *Carteras de sistemas.*
Recuperado de: http://www.todobolsa.com/formacion/carteras_de_sistemas

[14] IBROKER, *Anatomía de los Sistemas Automáticos.*
Recuperado de: https://blog.ibroker.es/anatomia-de-los-sistemas-automaticos/

[15] HRISTO PIYANKOV, *Can machine learning algorithms/models predict the stock prices? If yes, which are the best machine learning algorithm/models to predict the stock prices?.* Recuperado de: https://www.quora.com/Can-machine-learning-algorithms-models-predict-the-stock-prices-If-yes-which-are-the-best-machine-learning-algorithm-models-to-predict-the-stock-prices

14. Anexos

14.1. Creación de mi propio sistema

TradingMotion tiene una aplicación que permite crear tu propio sistema llamada **TradingMotionSDK**, la cual es un conjunto de bibliotecas y herramientas que permiten a los desarrolladores crear estrategias profesionales de trading algorítmico de una manera sencilla. La herramienta está desarrollada en Microsoft .NET, por lo que las estrategias se pueden desarrollar en C# y Visual Basic .NET, y está totalmente integrado con el entorno Visual Studio a través de un conjunto de plantillas listas para empezar.

En mi caso decidí crear un sistema para competir en la décima edición de *ROBOTRADER*.
Robotrader es una competición donde solo pueden participar estudiantes, y trata sobre desarrollar un sistema de trading y simularlo en el mercado durante un periodo de dos meses. Durante este periodo, se genera una clasificación en base a los beneficios que obtienen los sistemas que participan y al final se atribuye un premio.
El concurso busca incentivar a los estudiantes a programar algoritmos para los sistemas de trading, ofreciendo esta experiencia como puente para explorar el sector financiero y las oportunidades que este ofrece.

Figura 55: Logotipo de robotrader.Se puede consultar mas información en https://blogs.upm.es/robotrader/

Para mi participación, creé el sistema a partir de la SDK de TradingMotion. Lo desarrollé en C#, gracias a las plantillas y a algunos ejemplos que me facilitaron los trabajadores de TradingMotion, pude crear e optimizar un algoritmo con un buen rendimiento en el IBEX35. Para este, utilicé algunas librerías basadas en técnicas estadísticas que facilitan mucho el estudio de la situación del mercado, e así desarrollar unos parámetros que regulan cuándo hay que invertir y cuando no.

Una vez desarrollado el algoritmo, hay que seleccionar un mercado donde invertir y simular cómo habría actuado nuestro sistema en el mercado con los datos históricos (tambien llamado **backtest**) que tenemos.

Para el backtest tenemos dos opciones:

- Simular con la ventana de comandos (Figura 56).

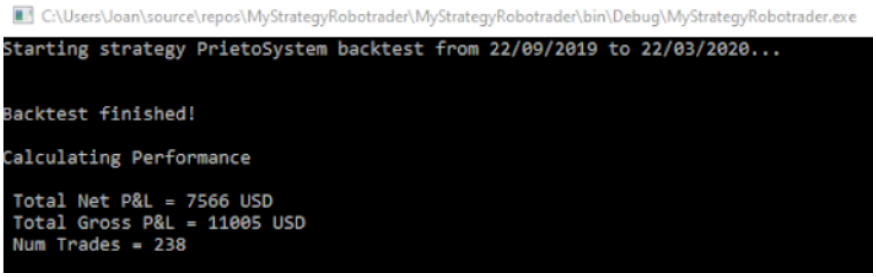


Figura 56: Ejemplo de simulación por consola.

- Simular con la TradingMotionSDK Toolkit. Esta incluye todo tipo de gráficos, valores estadísticos de rendimiento, todos los datos de sesiones, diagramas de dispersión, resultados mensuales (Figura 57)

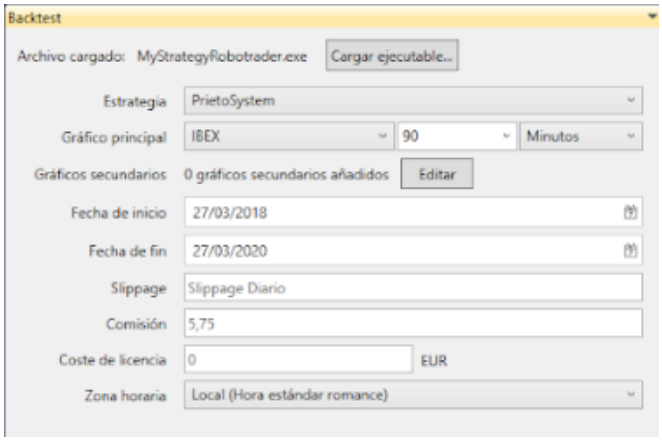


Figura 57: Seleccionamos la estrategia y el periodo de backtest en el que queremos simular el sistema

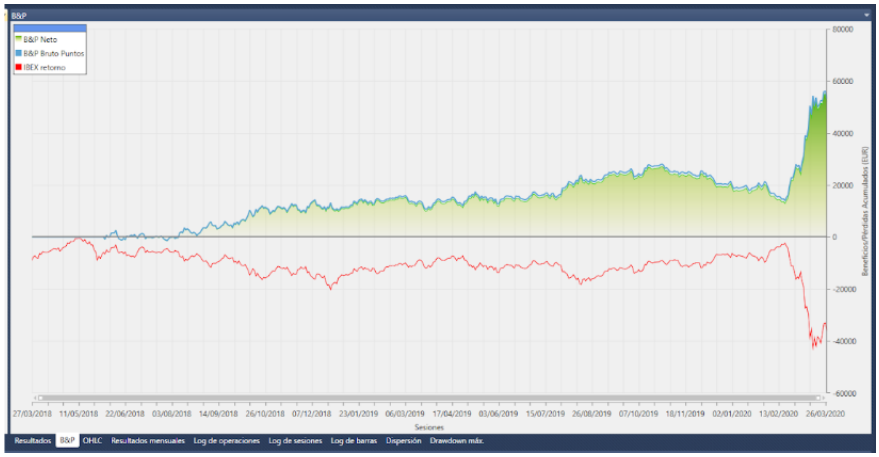


Figura 58: Analizamos los resultados: Gráfica de beneficios/pérdidas del sistema.

Como he mencionado anteriormente, los parámetros del algoritmo seleccionado son la base para decidir cuándo invertir. Un ejemplo de parámetro que he utilizado, es una variable Ticks Stop-Loss, para que a partir de un valor de pérdida, venda la acción automáticamente. Pero, estos parámetros se tienen que regular según un valor, encontrar el valor que nos genere más beneficio. Para ello, el Toolkit tiene una herramienta para optimizar el sistema con los parámetros que generen mejor rendimiento.
La herramienta utiliza un algoritmo de fuerza bruta, donde seleccionas un rango para cada parámetro y el programa ejecuta el sistema con todas las combinaciones posibles en el rango que has decidido para los parámetros. Así, obtener la combinación óptima, por lo tanto, la que mejor rendimiento puede generar nuestro sistema.

Finalmente envié mi sistema a la plataforma para poder ver los resultados en tiempo real y que los demás puedan acceder a él.
Se puede consultar mi sistema a través de https://www.tradingmotion.com/explore/System/PerformanceSheet?Id=22775&PrivateKey=7E7F3B8B-866D-4C7D-A0A1-05D73E859CBC&lang=es