Análisis de estrategia de *copytrading* en ETHEREUM mediante ML

Jaime Fábregas Fernández, *Tarlogic Security*.

***Abstract*—Este documento proporciona una visión sobre la estrategia de trading *copytrading* posible dentro del ecosistema de la blockchain de Ethereum. Este ecosistema ha experimentado una explosión en estos últimos años en términos de crecimiento y provisión de aplicaciones y servicios, entre los que se encuentran, de manera significativa, servicios financieros. Estos son los denominados DeFi (Decentralized Finances). Entre estos servicios se encuentran las casas de intercambio de divisas descentralizadas, que permiten la compra-venta de criptomonedas. Estos servicios se realizan mediante la interacción con *smart-contracts*. La incorporación de *smart-contracts*, programas ejecutables dentro de la blockchain, ha sido la gran aportación de Ethereum al mundo de las criptomonedas. Uniswap es uno de los *smart-contracts* más usados y populares para el intercambio de divisas (trading de ahora en adelante). Por ello es el *smart-contract* que se ha usado para los análisis.**

Este documento expondrá la metodología usada, el diseño de los experimentos para contrastar las hipótesis realizadas y, finalmente, el resultado de estos experimentos.

Los experimentos se han estructurado en torno al uso de técnicas de *Machine Learning* que permitan la predicción y/o clasificación de los mejores intercambios a realizar, medidos en términos de beneficio neto.

Los resultados finales arrojan serias dudas acerca de la viabilidad de esta técnica en un entorno profundamente aleatorio como es el de trading. A pesar de haber obtenido débiles relaciones positivas que han ayudado a hacer estimaciones más precisas, dada la hostilidad del entorno y las altas probabilidades de incurrir en pérdidas, hacen que estas relaciones positivas no sean suficientes para superar esta tendencia bajista y poder obtener beneficios.

I. Introducción

La blockchain de Ethereum ha permitido la creación de un ecosistema de aplicaciones descentralizadas que pueden reemplazar o replicar a entidades financieras del mundo real. Este es el caso de bancos, juegos, pero sobre todo productos financieros como casas de intercambio de divisas. Nos centraremos en éstas últimas.

A. Copy trading

Esta técnica se basa en la identificación de grupos de traders exitosos. Una vez identificados, se procede a copiar sus transacciones para obtener rendimientos similares.

La hipótesis de partida es que aquellos individuos con transacciones exitosas tienen más probabilidades de tener éxito en sus transacciones futuras. Esto puede ser bien porque tengan acceso a mejores fuentes de información o porque hayan desarrollado técnicas, métodos o incluso intuiciones efectivas. En el desarrollo de este análisis se estudiará la eficacia de la estrategia "copy trading".

II. Conjunto de datos

Para el análisis de esta técnica se han procesado todas las transacciones en la blockchain de Ethereum de Uniswap V2 y V3. Estas transacciones comenzaron en Mayo de 2020 y se han procesado hasta Marzo de 2022, incluido. En total, 43 millones de transacciones.

Cada transacción tiene información sobre:

- Wallet origen (se asume individuo/trader/bot)
- Tokens de venta y compra
- Cantidad intercambiada de cada token
- Bloque y fecha de la transacción
- DEX usado (V2 o V3)

III. Preparación de los datos

Antes de realizar cualquier análisis es necesario preparar los datos que se van a utilizar. Se divide en dos pasos: obtención y filtrado.

A. Obtención de datos

Se procesan todas las transacciones agrupándolas por wallet. Se obtiene así un listado de todas las wallets que han realizado transacciones y se agregan los datos por cada una de ellas. Adicionalmente se agrupan por meses, de forma que la misma wallet sólo puede aparecer una vez por mes.

Cada fila representa una transacción de compra realizada, donde se ha incorporado información agregada sobre la wallet originaria, datos de *pricing analytics* del token objeto de la compra y el resultado final de la transacción medido en porcentaje de beneficio.

Las columnas que se incluyen en el listado son las siguientes:

- Wallet origen
- Tasa de éxito de sus transacciones pasadas
- Tasa de actividad por meses

Este análisis ha sido realizado por Tarlogic Security, una de las empresas de ciberseguridad y ciberinteligencia de mayor crecimiento en Europa. Es también la responsable de tecnpología de análisis de información y del software de análisis de comunicaciones Acrylic Wifi.

- Beneficio medio obtenido agrupado por token
- Beneficio mediano de todas sus transacciones
- Monto de ETH mediano usado en las transacciones
- Vida del token sobre el que se transacciona (en escala logarítmica de horas)
- Varianza del precio del token en las últimas 24H
- Varianza del precio del token en la última semana
- Varianza del precio del token en el último mes
- Varianza del volumen del token en las últimas 24H
- Vairanza del volumen del token en la última semana
- Varianza del volumen del token en el último mes
- Beneficio de la cuarta última transacción
- Beneficio de la tercera última transacción
- Beneficio de la penúltima transacción
- Beneficio de la última transacción
- Monto en ETH de la transacción actual
- Fecha
- **ResponseBenefit**: Beneficio esperado
- **ResponseMaxBenefit**: Beneficio máximo

El listado contiene una entrada por cada transacción realizada, junto con los datos agregados de la wallet originaria y el resultado (beneficio) de ella.

En este caso se definen dos variables respuesta, que se definen a continuación:

- **ResponseBenefit**: Beneficio que se obtiene de esta transacción por la wallet en el momento de venta, o bien, si no ocurre una venta en los siguientes 30 días, se considera que se vende igualmente a los 30 días como máximo.

- **ResponseMaxBenefit**: Es el beneficio máximo que se podría haber obtenido de esta transacción vendiendo en el momento de mayor precio de los siguientes 30 días. El precio al que se vende es al del percentil 95. De esta forma se puede considerar que esta medida de respuesta es "ligeramente conservadora".

  Eligiendo el percentil 95 se evitan valores aberrantes y disparos y caídas de precios abruptas que introducirían importantes imprecisiones en el conjunto de datos. Usando percentiles se consigue una medida más *robusta*.

B. Filtrado de datos

La siguiente etapa en la preparación de los datos es el filtrado. Nuevamente, para proceder a un análisis estadístico es importante eliminar valores atípicos que puedan comprometer los análisis.

Para reducir el ruido de los datos se decide comenzar por filtrar aquellas wallets que tienen menos de 5 transacciones. Esto, intuitivamente, es importante para eliminar aquellas wallets que apenas tienen histórico de transacciones y que por tanto no tendría sentido copiar sus operaciones, puesto que no se dispone de suficientes datos para valorar si son exitosas o no.

Se reduce así la población de estudio a aquellas wallets que han interactuado con DEX al menos mínimamente, de acuerdo al umbral definido. Limitar los datos al subgrupo poblacional de interés permite a los algoritmos de ML afinar el resultado, acotando el espacio de búsqueda.

El número total de transacciones inicial tras el primer filtro es de: 96,783.

Seguidamente, se establecen los valores máximos y mínimos para cada variable, eliminando aquellas filas que no entran dentro de los rangos definidos:

TABLA I
FILTROS APLICADOS

Filtro	Eliminados	Porcentaje
Eth transacción actual < 50	963	0.99%
Beficios pasados < 15	252	0.26%
Eth mediano < 40	268	0.27%
-100 < Varianzas < 100	991	1.02%
ResponseMaxBenefit < 50	71	0.07%
	Total	**2.54%**

IV. División del dataset en test y training

Se divide el dataset en training y test, necesario para cualquier análisis con ML involucrado.

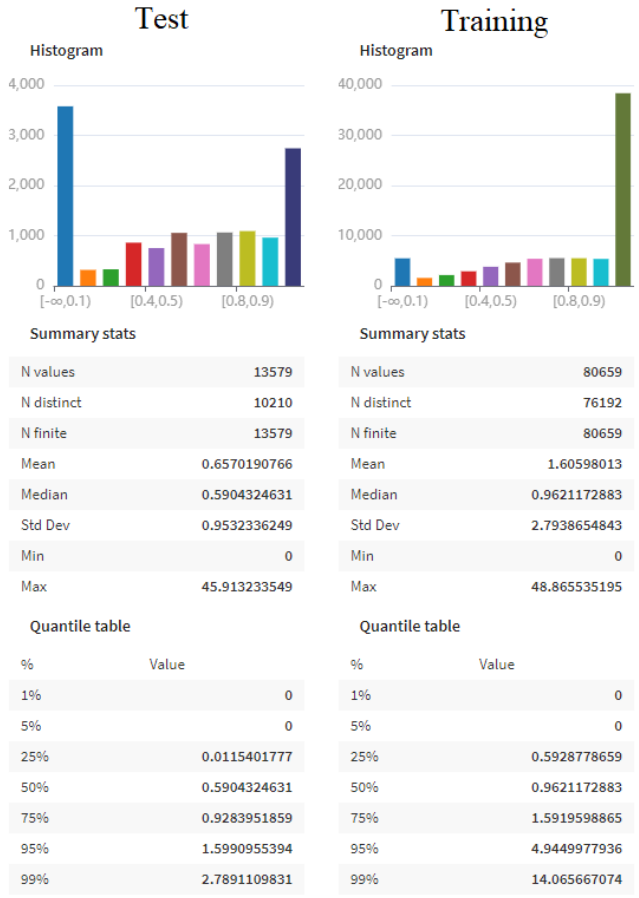

Fig. 1. Análisis de predictora **ResponseMax** en *test* y *training*

La naturaleza de los datos es fuertemente *estacional*, por lo que la división de los datos se realiza en base a una ordenación por fecha. Los datos de training corresponderán al 85% de los datos más antiguos, y el de test al 15% más reciente.

Esta estacionalidad se manifiesta en diferencias significativas de los parámetros estadísticos, tal y como muestra la **figura 1**.

Es importante recalcar las diferencias entre los dos conjuntos de datos, en especial, la proporción tan dispar en las transacciones que tienen un beneficio de 0 entre los dos conjuntos de datos (significando una pérdida del 100%).

La **figura 2** muestra la gran diferencia entre transacciones con beneficio y sin beneficio entre los datasets de test y training.

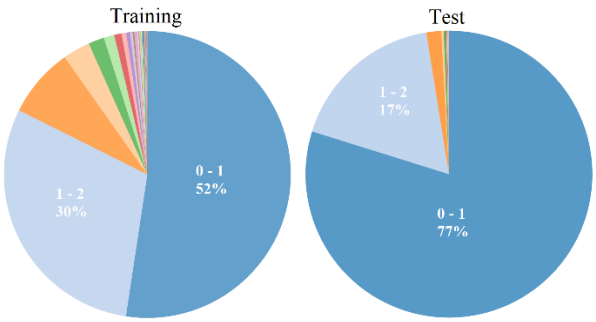

Fig. 2. Gráfica de queso de predictora **ResponseMaxBenefit**

A. Implicaciones en los resultados de ML

Estos datos ponen de manifiesto una volatilidad muy importante, fuertemente estacional. Esto supone un reto aún mayor a la hora de obtener predicciones con ML. Casi la mitad de las transacciones en el dataset de training tienen beneficio, mientras que menos del 20% lo tienen en el de test.

Esto implica que una mejora suficiente como para obtener beneficio en el conjunto de training puede no obtenerlos en el conjunto de test. La mejora necesaria ha de ser *muy significativa*, para que produzca un beneficio en ambos conjuntos.

B. Consideración sobre el cálculo de beneficio

El cálculo del beneficio esperado en los conjuntos de datos no es trivial, puesto que no sólo depende de **ResponseMaxBenefit**, sino también del *target* que se defina, el cual es arbitrario. Si no se llega al *target*, no se vende, y por tanto, el beneficio final dependerá del precio del token al final de los 30 días, momento en el que se realizaría la venta forzosamente. En este caso el beneficio caería al mostrado por la variable **ResponseBenefit**.

IV. Análisis de los datos

Tras realizar la eliminación de los valores atípicos se realiza un análisis de los datos para detectar anomalías. Se realizan dos tipos de análisis: univariante y multivariante.

A.Análisis univariante

Se realiza un análisis univariante de cada variable, incluyendo predictores y variables de respuesta. Se incluye a continuación la distribución de todas las variables por orden de izquierda a derecha y de arriba abajo. Se excluye la variable fecha por ser linealmente creciente y no tener valores atípicos.

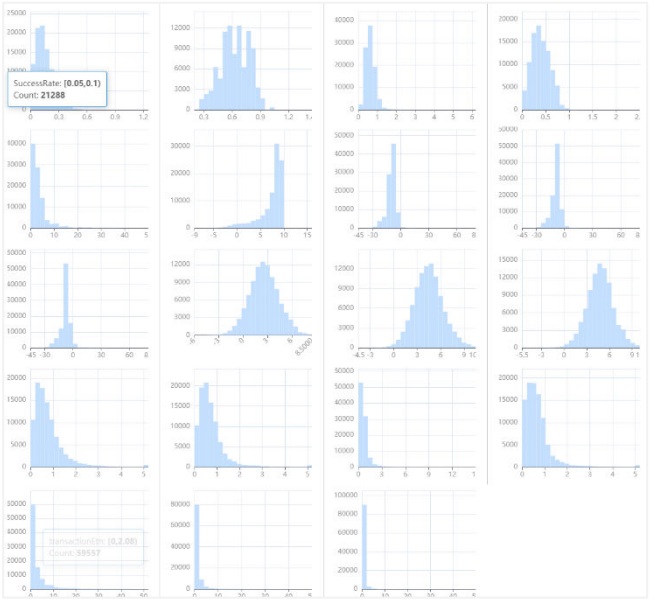

Fig 3. Distribución de las 19 variables tras aplicar filtros

Las distribuciones más extremas corresponden a las dos variables de respuesta (abajo a la derecha). Se decide no acotar más estas dos distribuciones por estar ya limitadas a un beneficio máximo de 50 veces la inversión inicial.

Aunque este valor sea poco frecuente, es perfectamente posible que ocurra en el plazo de un mes, especialmente entre tokens de escasa longevidad, sin ser necesariamente este un valor atípico. Más bien al contrario, nos gustaría encontrar alguna relación entre estos beneficios esperados y los predictores.

B. Análisis multivariante. Correlación

Para realizar un primer análisis de las relaciones que pueda haber entre variables predictoras se realiza un heatmap de correlación. El coeficiente elegido es el de Spearman, en favor del de Pearson, por ser mejor descubriendo relaciones monótonas no lineales entre variables.

Este heatmap también ayudará a descubrir problemas de multicolinealidad en caso de haberlos. Adicionalmente se compararán las variables predictoras con las de respuesta para descubrir posibles relaciones fuertes entre ellas que puedan ser de gran interés.

Fig. 4. Heatmap de correlación Spearman

Se observa que las dos variables de respuesta (últimas dos de la tabla) no muestran una correlación fuerte o moderada con ninguna otra variable (máximo: 0.27).

Adicionalmente se observa una fuerte correlación entre los grupos de variables que miden la varianza del volumen y del precio respectivamente (máximo: 0.96). Esto tiene sentido puesto que cambios en las últimas 24 horas afectan necesariamente a los cambios de la última semana y del último mes.

Puesto que estos efectos están limitados a estos dos grupos de variables y se considera interesante tener variables representando un mínimo de 3 espacios temporales, esto es: día, semana y mes, se decide mantener estas variables.

Adicionalmente, también se observa una correlación (0.76) entre el beneficio por token y la mediana de beneficios con la tasa de éxito. Esto también entra dentro de lo esperado, puesto que mayor tasa de éxito aumenta por lo general el porcentaje de beneficio mediano.

C. Análisis multivariante. PCA.

Para valorar la posibilidad de reducir el número de variables predictoras, se aplica un algoritmo de reducción de dimensionalidad PCA (*Principal Components Analisys*). El interés es descubrir si la mayor parte de la varianza puede ser explicada con un número reducido de variables.

Adicionalmente, este método también tiene la capacidad de solventar los posibles casos de multicolinealidad presentes en el conjunto de los datos.

El resultado del PCA sobre las 17 variables predictoras (se excluyen las variables de respuesta) es el siguiente:

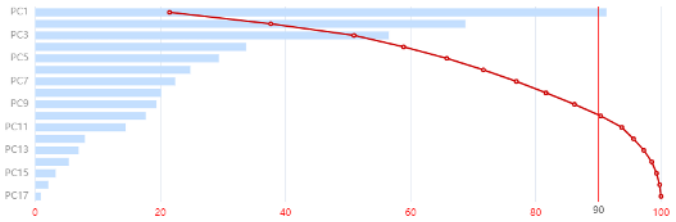

Fig. 5. Ratio de varianza explicada

Se necesitan las primeras 10 componentes para explicar al menos el 90% de la varianza, tal y como se aprecia en **figura 5**.

Atendiendo al heatmap del PCA, como se observa en la **figura 6**, podemos obtener una mayor comprensión de lo que se está capturando en las distintas componentes.

Sabiendo que las 10 primeras componentes explican el 90% de la varianza, podemos centrar la atención en las últimas. Concretamente, las últimas 4 componentes (PC14 – PC17) están concentradas justamente en las variables que miden la varianza de Precio y Volumen, teniendo valores nulos para todo el resto de variables predictoras.

Esto tiene sentido, puesto que ya se había detectado una fuerte correlación entre ellas previo al PCA, por tanto, las relaciones entre ellas proporciona muy poca información, y como consecuencia, puede explicar de manera muy escasa la varianza remanente.

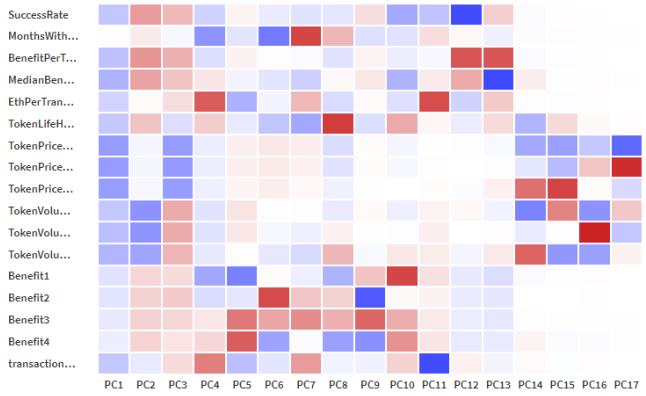

Fig. 6. Heatmap de correlación PCA

En resumen, el PCA consigue reducir el número de variables predictoras de 17 a 10 con un umbral del 90% de varianza explicada. Esta reducción viene en su mayoría causada por la reducción de los pesos en las variables de varianza de precio y volumen (últimas 4 componentes). Este factor ya se tuvo en cuenta en el análisis previo de correlación y se concluyó que debían mantenerse.

En conclusión, el PCA no aporta una ventaja significativa a la hora de tratar los datos que pueda suponer o causar una diferencia en los resultados de algoritmos de ML, puesto que la reducción en dimensionalidad es muy moderada, e introduce, en cambio, una alta complejidad a la hora de interpretar cada una de las componentes.

Por todo ello, no se usarán las componentes de PCA en los algoritmos de ML que se usarán con posterioridad.

Adicionalmente, comentar que también se realizaron pruebas usando el mismo planteamiento diseñado en este proyecto con los resultados obtenidos de la PCA, en lugar de usar los datos directamente sin procesar. Este planteamiento no ofreció ninguna ventaja apreciable.

V. Clustering de datos

El siguiente paso es someter los datos a técnicas de clustering. El objetivo es identificar aquellos algoritmos de aprendizaje no supervisado que sean capaces de extraer conocimiento de los datos en base a su agrupamiento en clústeres.

En este caso, no se pretenden extraer un cierto número de clases conocidas de entre los datos, sino permitir al algoritmo agrupar los datos para posteriormente comparar los distintos clústeres en base a la distribución de las variables respuesta.

El método usado para el scoring del clúster es el *coeficiente de silueta*, usando una métrica de distancia euclídea. Éste mide cómo de separados se encuentran los distintos clústeres entre sí. Sin embargo, a pesar de ser ésta una medida importante, en nuestro caso particular, estamos interesados en medir cómo de eficientemente los clústers separan las transacciones de alto benificio de las de bajo.

La intuición detrás de esto, es que deben existir grupos que tengan un mejor o peor rendimiento basados en las variables predictoras. Es importante hacer notar que para el clustering no se usan las variables respuesta, puesto que esto invalidaría los

grupos generados, por tener acceso a información futura (beneficio que obtendrá) de cada transacción.

De esta forma se valora si la información extraída por el algoritmo de clustering puede ser de valor para los algoritmos de machine learning a aplicar con posterioridad y su posible capacidad para ayudar a extraer información, categorizándola primero.

Se han valorado cinco de los algoritmos de clustering más populares. Se incluyen los resultados en la tabla II.

TABLA II
ALGORITMOS DE CLUSTERING APLICADOS

Algoritmo	N Clusters	Scoring
K Means	4	0.17
Gaussian Mixture	5	0.11
Mini-batch KMeans	3	0.17
Interactive clustering	5	0.19
DBScan	1	0

Se han probado distintas combinaciones de parámetros para cada uno de ellos. El campo Scoring refleja la puntuación más alta obtenida en cada tipo.

Todos arrojan resultados muy similares. Se encuentran clústeres idénticos en todos ellos, con tendencias claras a valorar el mismo conjunto de variables, a pesar de diferir en número.

Estudiamos a continuación cómo afecta la agrupación a la composición de las variables respuesta. Se observa el mismo patrón independientemente del algoritmo elegido y de los parámetros configurados. Se incluye a continuación el resultado de *KMeans* con 4 clústeres y uno adicional de outliers.

Se realiza un plot de mosaico de los clústers con la variable de respuesta apilada en cada barra en intervalos de una unidad.

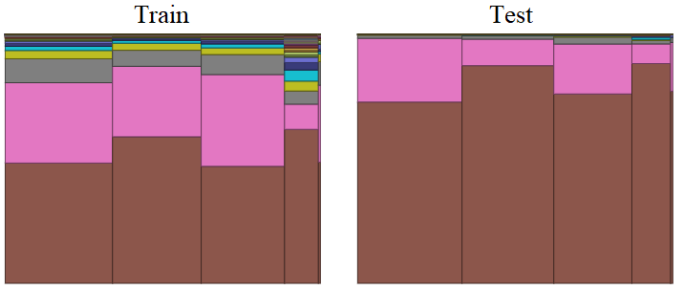

Fig. 7. Plot de mosaico de clústers con variable respuesta

En la **figura 7** se muestra el resultado de aplicar *KMeans* sobre los datos de training y posteriormente sobre los datos de test. Al aplicarlo en el conjunto de datos de test no se ha re-entrenado el algoritmo. Se han usado los mismos centroides obtenidos durante el entrenamiento.

Cada barra representa un clúster. Dentro de de cada barra se apilan los valores de la variable respuesta que muestra el beneficio (**ResponseMaxBenefit**) en intervalos de una unidad. De esta forma, la región marrón representa la cantidad de elementos con un beneficio en el intervalo [0,1) (0 representa una pérdida del 100% y 1 la permanencia de la cantidad inicial, sin beneficio). La región de color rosa representa los elementos con un valor de respuesta en el intervalo [1,2), y así, sucesivamente.

La diferente proporción de la región de pérdidas marrón es debida a un efecto de estacionalidad, puesto que los datos de test se obtienen teniendo en cuenta la fecha de los elementos y contiene los elementos más recientes. En ese subgrupo poblacional la proporción de transacciones con pérdidas aumentó considerablemente.

Se aprecia en la **figura 7** que los clústers son capaces de separar grupos con una mayor y menor proporción de transacciones con beneficio (regiones por encima de la barra marrón). Esto ocurre de manera consistente en los conjuntos de datos de train y test.

Debido a esto, se decide incluir información del clúster al que pertenece cada elemento del conjunto de datos para que sea tratado por los distintos algoritmos de ML.

VI. Machine Learning. Clasificación.

El primer planteamiento es el de someter los datos a una predicción por clasificación de dos clases: beneficio o pérdida.

Para ello se prepara el conjunto de datos y la variable de respuesta pasa a tener únicamente dos valores, representando las dos clases: 0 o 1.

A. Filtrado de datos por clúster

Se usarán dos conjuntos de datos de training (*datasets*):

- **Completo** con la información del clúster incluída.
- **Filtrando** por clúster 0. (Aumenta la proporción de transacciones con beneficio, a costa de reducir el número total de registros a 21.331)

La intuición detrás de esta decisión es la de comparar la precisión de los algoritmos cuando tienen información de clúster o cuando tienen un conjunto de datos más favorable hacia transacciones positivas gracias al filtrado de uno de los clústeres.

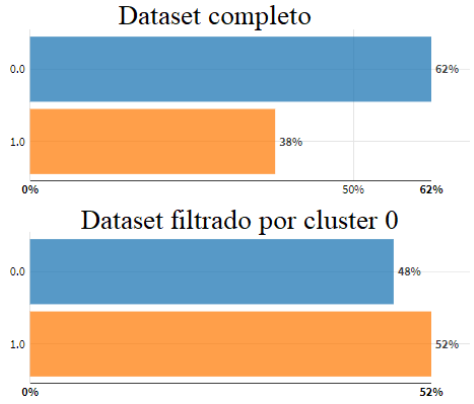

Fig. 8. Proporción de las clases en los dos datasets de training

Como se observa en la **figura 8**, la aplicación del cluster sobre los datos resulta en un aumento significativo de los elementos positivos (clase 1).

B. Métricas.

Los algoritmos de ML se han parametrizado para optimizar la métrica de ***accuracy (precisión)***. Esto es así porque *el mayor interés es reducir la cantidad de falsos positivos*.

Una vez en un entorno de producción, sólo se ejecutarán transacciones con predicción positiva, mientras que ninguna transacción con una predicción negativa se copiará (usando la estrategia de *copytrading*).

Los falsos negativos no provocan una pérdida financiera en esta estrategia, pero sí existe en el caso de falsos positivos, puesto que se copia una transacción que va a producir una pérdida. Es por ello que el interés principal de la predicción es reducir los falsos positivos.

Únicamente una vez que la proporción de falsos positivos sea suficientemente baja, se puede empezar a trabajar en reducir los falsos negativos con el fin de aumentar el número total de transacciones copiadas, y por tanto, aumentar el beneficio en términos absolutos.

C. Validacion cruzada. Cross-validation

Para la obtención de una evaluación acertada de la precisión de los algoritmos se usa el bien conocido método de *cross-validation K-fold*. En este método se subdivide el conjunto total de los datos de training en un número arbitrario (comúnmente 5, en nuestro caso), y se asignan 4 porciones de esos datos a training y 1 a testing, realizando las 5 combinaciones posibles.

En nuestro caso particular, puesto que es necesario mantener el orden por fecha de los datos, se realiza de una manera similar, tal y como se muestra gráficamente en la **figura 9**.

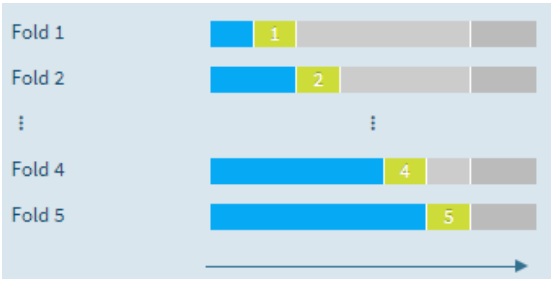

Fig 9. Validación cruzada con datos ordenados por fecha

D. Preparación de los datos de test

Dentro de las datos de training se vuelve a separar en un conjunto de test y training. En total tendremos dos conjuntos de tests, uno global y otro que se usará para medir el *accuracy* de los distintos algoritmos de ML. Al usar varios algoritmos de ML sobre el mismo conjunto de test, crece la probabilidad de que al menos uno de ellos proporcione un resultado positivo por azar. Por ello se reserva un segundo conjunto de test para probar únicamente aquel algoritmo ML que se seleccione como el mejor, y haya arrojado un resultado positivo. Esto es debido a la naturaleza del presente análisis en el que es crítico reducir las pérdidas.

En la preparación de los datos de test se sigue teniendo en cuenta la fecha, de forma que los datos estarán ordenados cronológicamente. Todos los elementos del conjunto de test tendrán una fecha superior a cualquier elemento del conjunto de training.

E. Elección de los algoritmos de ML

Primeramente se eligen cuatro algoritmos ML que usen un planteamiento lo más diferente posible entre ellos a la hora de encontrar una solución. Éstos son:

- **Logistic Regressio**n. Uno de los más conocidos y sencillos. Se elige para obtener información sobre cuál sería la línea base de la que partir de este proceso.

- **Random Forest**. Es un algoritmo que supone un cambio en el paradigma de búsqueda de la solución a través de árboles de decisión. Es decir, tiene una naturaleza distinta a la regresión logística, y este punto es lo que nos interesa evaluar.

- **SVM**. No podía faltar SVM que ha explotado en popularidad en los últimos años. Es un método que se basa en doblar curvas hiperespaciales (manifold) para encontrar una clasificación. Nuevamente en este caso interesa que su naturaleza sea completamente distinta de los anteriores.

- **Neural network.** En este caso se elige una red neural perceptrón multicapa clásica, con tres capas, *tanh* como función de activación y ADAM como optimizador. No se pretende desarrollar una arquitectura optimizada para encontrar un patrón en particular, puesto que en este momento todavía no hemos desarrollado ninguna hipótesis al respecto. Es por ello que se elige una arquitectura clásica para medir su rendimiento.

F. Resultados

Los resultados se incluyen en la tabla III, a continuación.

TABLA III
SCORING ML CLASIFICACIÓN POR DATASETS

Algoritmo	Completo	Filtrado
Logistic Regression	0.423	0.495
Random Forest	0.469	0.502
SVM	0.433	0.492
Neural Network	0.429	0.492

En la tabla III se muestra el F1 scoring. Los resultados son poco mejores que un clasificador aleatorio. Adicionalmente, es necesario remarcar que las medidas de *accuracy* no difieren significativamente.

Es importante estudiar la gráfica de decisión para descubrir el comportamiento del algoritmo en base al umbral establecido para distinguir clases.

Un umbral clásico es de 0.5, sin embargo, es común ajustar estos umbrales para aumentar el *accuracy* de forma que permitamos al algoritmo opinar únicamente sobre aquellos casos en los que esté “muy seguro” del resultado.

Se incluye a continuación la gráfica de decisión (decision chart) del algoritmo Random Forest para el dataset Completo, donde se aprecia que no hay diferencias significativas en la variación del umbral, únicamente en casos extremos donde los casos se reducen a uno o ninguno.

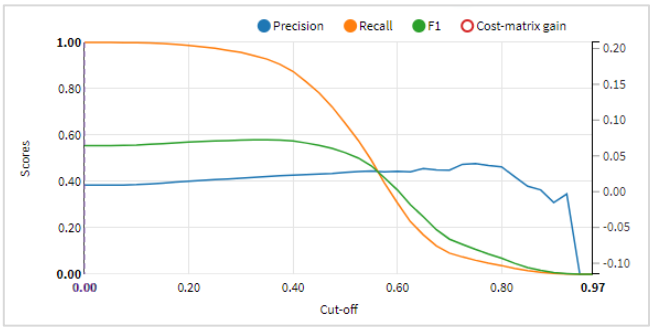

Fig. 10. Gráfica de decisión de Random Forest en *Completo*

En la **figura 10** se puede observar que el *accuracy* apenas varía al aumentar el umbral (línea azul). Este efecto se produce de manera análoga en el resto de algoritmos y datasets. No se incluyen todos por conveniencia.

Al respecto de los resultados por dataset:

- **Completo**: se aprecia una *ligera mejora* del 0.38 de casos positivos (**figura 8**) a 0.469 en el mejor de los casos. Se incluye a continuación la gráfica de Lift (**figura 11**) donde se puede apreciar la escasa ganancia correspondiente al algoritmo de Random Forest.
  El área bajo la curva (AUC) tanto por cantidad como por forma dista mucho de la forma deseada (línea naranja en el mejor de los casos). La ganancia acumulada (línea azul) tiene una pendiente muy similar a la del modelo aleatorio, y especialmente en los valores iniciales del eje x, que es donde más interesa, indicando un rendimiento casi nulo.

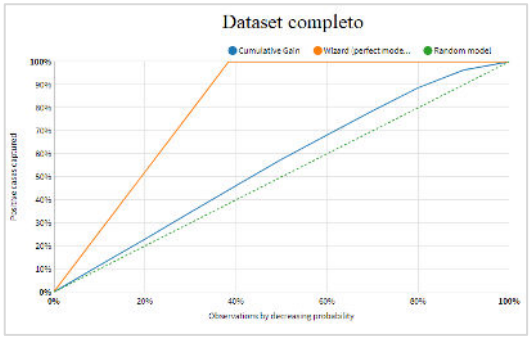

Fig. 11. Lift chart de Random Forest en *Completo*

- **Filtrado:** observamos que los algoritmos tienen una precisión similar a la de un clasificador aleatorio que tenga en cuenta la media de casos positivos de la población: 0.52 (**figura 8**). Es decir, *no hay mejora apreciable*.

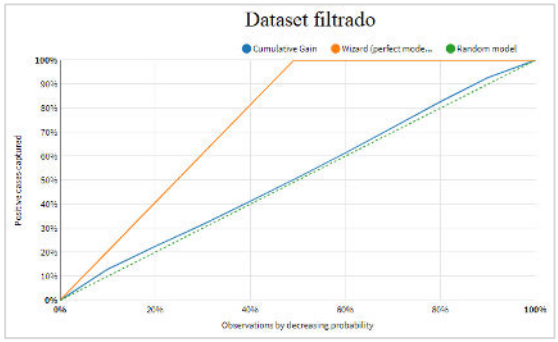

Fig.12. Lift chart de Random Forest en *Filtrado*

Es notable cómo las dos lift charts de Random Forest, **figura 11** y **figura 12** (muy similares al resto de algoritmos ML, se elige Random Forest como ejemplo representativo) apenas ofrecen mejora, y más concretamente, la correspondiente al Dataset Filtrado, apenas se diferencia de la línea diagonal base correspondiente a un modelo aleatorio.

G. Conclusión

Los algoritmos ML aplicados no logran aumentar la tasa de acierto en el caso de los datos ya filtrados por clúster y no llegan a la tasa de acierto que se puede obtener mediante clustering cuando no se filtran.

Podemos concluir por tanto que:

1. El clustering (no supervisado) es más eficaz que algoritmos supervisados de ML a la hora de separar casos positivos de negativos.

2. Los algoritmos supervisados de ML no son capaces de encontrar ninguna relación entre los datos que el clustering no haya encontrado ya.

Al respecto de la mejora máxima esperada es de **0.502**. Esta precisión es insuficiente para obtener beneficio, puesto que la mayor parte de las transacciones con pérdida, lo hacen en su totalidad, y las transacciones con beneficio únicamente en una proporción pequeña pasan del doble.
Es decir, se necesita una precisión mucho mayor para llegar a la zona de beneficio.

Se concluye que este planteamiento no es viable en una estrategia de *copytrading*.

VII. Machine Learning. Predicción

Se define un nuevo planteamiento donde se usarán los algoritmos no para clasificar sino para realizar la predicción del beneficio esperado.
De esta forma se incorpora al algoritmo de ML más información, puesto que se trata ahora de una variable continua en lugar de una binaria, por lo que, intuitivamente, los algoritmos pueden conocer no sólo si estaban equivocados o acertados, sino *por cuánto*.

A. Métricas

Existe una problemática importante a la hora de elegir una métrica en algoritmos ML de predicción en proyectos de este tipo. Esto se debe a que una métrica clásica como puede ser RSME puede proporcionar una idea de lo alejada que la predicción suele estar del valor real, pero no aporta información sobre si la predicción fue correcta o no.

En nuestro caso, nos interesa conocer si, primeramente, una transacción tendrá beneficio o no. Es decir, si la predicción es mayor o menor que uno. Un valor de RSME no aporta información relevante puesto que el algoritmo puede estar acertando beneficios o pérdidas de las transacciones, pero no en la cantidad adecuada y viceversa.

Por ello las dos métricas que se implementan son:

- **Correlación Spearman**. Se elige Spearman en favor de Pearson por su mejor reflejo de correlaciones monótonas no lineales. Además, también representa bien relaciones lineales.
  La intuición detrás de esta elección, es que si existe una correlación entre la predicción y la variable de respuesta se puede jugar con umbrales para aumentar la proporción de verdaderos positivos (*accuracy*) hasta que se alcance un beneficio total.

- **Beneficio real esperado**. Ésta es una métrica personalizada para este caso. Se calcula el beneficio real esperado usando los datos conocidos de respuesta. Éstos son los pasos:
  1. Si la predicción es menor que uno, se ignora
  2. Si la predicción es mayor que uno, se establece el target como la predicción.
  3. Si se alcanza el target, entonces el beneficio es igual al target.
  4. Si no se alcanza el target, entonces el beneficio es igual al *ResponseBenefit*, que es el beneficio tras copiar la venta o realizar la venta después de 30 días si ésta no se ha producido.
  5. El resultado total se divide pro el número de transacciones procesadas (aquellas con predicción mayor y menos que uno).

- **Beneficio real esperado con target fijo**. Es la misma métrica que la anterior pero se establece un target fijo en lugar de dinámico en base a la predicción.

B. Validación cruzada. Cross-validation

Esta sección es análoga a la misma sección en el apartado VI.

C. Preparación de los datos de test

Esta sección es análoga a la misma sección en el apartado VI.

D. Elección de los algoritmos de ML

Para la métrica "Beneficio real esperado con target fijo", se eligen únicamente los dos algoritmos ML que se consideran más representativos de todos ellos: Random Forest y Neural network.

El motivo es para no extender en exceso las pruebas, puesto que se repetirán para distintos valores de target. Únicamente en caso de obtener resultados cercanos a la unidad (sin pérdida) se contempla la posibilidad de probar el resto de algoritmos y comenzar un proceso de refinamiento de hiperparámetros.

Para las otras dos métricas se usarán una amplia gama de algoritmos ML, como última opción antes de descartar la estrategia objeto de estudio. Éstos son:

- Random Forest
- Gradient Boosted Trees
- Lasso
- Light GBM
- XGBoost
- Decision Tree
- SVM
- SGD
- KNN
- Extra tres
- Lasso-Lars

E. Resultados

Se incluyen primeramente los resultados con las dos primeras métricas a continuación, en la tabla IV.

Es importante hacer notar que los dos scoring son completamente diferentes en naturaleza, sin embargo, ambos deberían tender hacia uno.

En el caso de *Spearman*, valores cercanos a 0.6, que en cambio son comunes en el scoring *Real*, implicarían generalmente beneficios superiores a 1. Se comenta esto para resaltar la distinta naturaleza de los scoring. El que mejor refleja el beneficio obtenido en última instancia es el *Real*, ya que directamente mide el beneficio, mientras que en el caso de *Spearman*, hay que medirlo en una etapa posterior.

TABLA IV
SCORING ML CLASIFICACIÓN POR MÉTRICAS

Algoritmo	Spearman	Real
Random Forest	0.025	0.590
Gradient Boosted Trees	-0.065	0.576
Lasso	-0.143	0.632
Light GBM	-0.008	0.638
XGBoost	-0.069	0.607
Decision Tree	0.024	0.625
SVM	0.199	0.386
SGD	-0.106	0.607
KNN	0.050	0.468
Extra trees	-0.051	0.617
Lasso-Lars	-0.146	0.613

A. Spearman

Comenzando por la métrica de Spearman, se aprecia en la tabla IV resultados muy cercanos al cero, indicando que no se

ha podido establecer una relación satisfactoria entre variable respuesta y predicción, ni siquiera moderada.

La puntuación más alta corresponde al SVM con 0.199, sin embargo, tras someter estos datos a prueba con el segundo conjunto de test (más reciente), la correlación de Spearman vuelve a estar muy cercana al cero.

Se evalúan dos algoritmos (SVM y Random Forest) con el segundo conjunto de test. De esta forma se supone una evaluación muy acertada eliminando el sesgo de elección del mejor algoritmo. Puesto que estamos repitiendo el experimento con 11 algoritmos diferentes, se podría considerar un tipo de *p-hacking* al aumentar la posibilidad de un falso positivo. Para eliminar este efecto se comprueba con un conjunto de test distinto y más reciente.

Puesto que el scoring Spearman hace referencia a una correlación, implica que valores crecientes de la predicción deberían relacionarse con valores crecientes de la variable respuesta. Sin embargo, los valores de la predicción no ofrecen información necesariamente sobre el éxito o pérdida de la transacción, puesto que son una medida relativa. Recalcar nuevamente que esto es debido a que la métrica usada ha sido una correlación que no informa al algoritmo de ML sobre cuál debería de ser el valor real de beneficio, únicamente intenta mantener la relación creciente entre predicciones y respuestas.

Por ello, a la hora de medir el *accuracy* real en el conjunto de test, no se establecen umbrales sobre la variable de predicción. En su lugar, se ordenan las predicciones y se usan percentiles para medir su precisión.

TABLA V
SCORING ML CLASIFICACIÓN POR SPEARMAN

Percentil	Beneficio
99.7	0.57
99	0.46
90	0.19

Se aprecian en la tabla V resultados negativos, con pérdidas muy significativas. Esta opción queda descartada.

B. Real

En la medición del beneficio real, se observa que muchas de ellas rondan el 0.6 de beneficio.

En el conjunto de test usado, es posible obtener este resultado simplemente aceptando todas las transacciones y estableciendo un target de 1.2.

Es por ello que, nuevamente, los algoritmos muestran no ser capaces de encontrar patrones que superen en rendimiento al proporcionado por la aplicación aleatoria en base a una media de la transacción.

Esto se comprueba al someterlos a diferentes conjuntos de test y comprobando que el rendimiento varía en la misma proporción que lo hacen los datos y la cantidad de transacciones positivas en ellos.

Por tanto, dado lo alejado del scoring de la zona de beneficio y los indicios de que los algoritmos no están siendo capaces de capturar información relevante del conjunto de datos, esta opción también queda descartada.

C. Real con target

En este caso se limitan los algoritmos a dos, con el fin de reducir el número total de pruebas, y se estudia su comportamiento con distintos umbrales de target (umbral al que se realiza la venta).

TABLA VI
SCORING ML CLASIFICACIÓN POR MÉTRICA CON TARGET FIJO

Target	Random Forest	Neural Network
1.2	0.634	0.611
1.8	0.674	0.649
2.2	0.687	0.661
2.6	0.693	0.668
3	0.702	0.676
4	0.708	0.683
6	0.707	0.687

El resultado obtenido es muy similar al de la métrica *Real*. En este caso sin embargo, el mejor de los casos logra superar la barrera de 0.7. Sin embargo, hay que tener en cuenta nuevamente que estos resultados son prácticamente idénticos a ejecutar todas las transacciones estableciendo un target alto. Por tanto tiene relación con la naturaleza de los datos más que con la capacidad del algoritmo de capturar información de ellos.

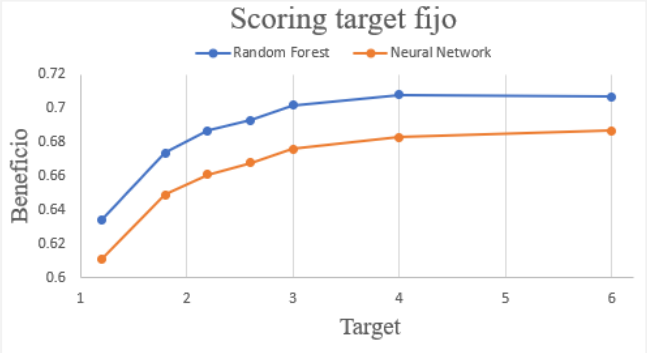

Fig. 12. Beneficio por target fijo

De manera análoga, al probarlo con el conjunto de test más reciente, este beneficio se desploma a 0.3, debido a la distinta proporción de transacciones positivas.

Por tanto, esta opción también queda descartada.

VIII. Conclusiones

La estrategia de *copytrading* no es viable. Con los métodos ML conocidos hasta el momento y usando las técnicas estudiadas no es posible obtener beneficio con esta técnica por un amplio margen.