COMPOSICION DE CARTERA RÉPLICA DEL ÍNDICE BURSÁTIL ALEMÁN DAX ULIZANDO TÉCNICAS PERTENECIENTES AL MACHINE LEARNING: COMPARATIVA ENTRE LASSO, RANDOM FOREST U AUTOENCODER

Andrés Albelda Martínez

Trabajo de investigación 21/010

Master en Banca y Finanzas Cuantitativas

Tutores: Dr. Manuel Ángel Dominguez

Universidad Complutense de Madrid

Universidad del País Vasco

Universidad de Valencia

Universidad de Castilla-La Mancha

www.finanzascuantitativas.com

COMPOSICIÓN DE CARTERA RÉPLICA DEL ÍNDICE BURSÁTIL ALEMÁN DAX UTILIZANDO TÉCNICAS PERTENECIENTES AL MACHINE LEARNING: COMPARATIVA ENTRE LASSO, RANDOM FOREST Y AUTOENCODER

Andrés Albelda Martínez

Master en Banca y Finanzas Cuantitativas

Director: Manuel Ángel Domínguez Toribio

Universidad Complutense de Madrid
Universidad del País Vasco
Universidad de Valencia
Universidad de Castilla-La Mancha

Resumen

El objetivo principal del trabajo es abordar el problema de la selección de los componentes de una cartera que replique un índice bursátil, como es el DAX. Para ello, se utilizarán tres técnicas diferentes pertenecientes al Machine Learning, siendo estas la regularización LASSO, Random Forest y las redes neuronales, más en concreto un tipo de estas conocido como Autoencoders.

Índice

1. Introducción **4**

2. Réplica de índices bursátiles **4**

3. El DAX como referencia. **6**

4. Machine Learning **7**
4.1. Concepto de aprendizaje, etapas y problemas asociados. 7
4.2. Tipos de aprendizajes. 9
4.3. Conjuntos de entrenamiento, test y validación. Validación cruzada. 9
4.4. Entrenamiento. 11
4.4.1. Descenso por gradiente 11

5. Inteligencia Artificial para el diseño de la cartera de réplica **13**
5.1. Regularización. 13
5.1.1. Regularización LASSO o L1 14
5.2. Random Forest 16
5.2.1. Árbol de decisión 16
5.2.2. Calibración del árbol 18
5.2.3. Pasar del árbol al bosque 18
5.2.4. Importancia de los componentes 19
5.3. Redes neuronales artificiales 19
5.3.1. La neurona 19
5.3.2. La red neuronal 20
5.3.3. Tipos de redes según su tipología 21
5.3.4. Autoencoder 22

5.3.5. Selección de componentes del índice utilizando Autoencoder 23

6. Revisión de la literatura. 24

7. Aplicación de la metodología propuesta 25
7.1. Obtención de los datos 26
7.1.1. Tratamiento de los datos. 26
7.2. Regularización LASSO 27
7.3. Random Forest 30
7.4. Autoencoder 33
7.4.1. Estructura con una capa intermedia 33
7.4.2. Estructura con una capa intermedia y regularización Lasso en la capa intermedia 35
7.4.3. Autoencoder con tres capas intermedias 36
7.4.4. Selección de la estructura del autoencoder. 38

8. Comparativa de los diferentes modelos 38

9. Comparativa entre los activos con más información común y los que menos para el Autoencoder 41

10. Posibles lineas de investigación futuras. 42

11. Conclusiones. 43

1. Introducción

El DAX es uno de los principales índices bursátiles a nivel europeo y consecuentemente existe un gran interés por él. En el presente documento buscaremos replicarlo utilizando un número reducido de activos mediante diferentes técnicas relacionadas con la inteligencia artificial, debido a que en los mercados existe una gran cantidad de información que no podríamos procesar mediante otros métodos convencionales. De esta forma, podremos seleccionar los elementos de una cartera réplica contemplando relaciones que no veríamos de otra forma.

Para llevar a cabo esta tarea queremos hacer hincapié en las redes neuronales artificiales, ya que estas permiten captar con bastante facilidad relaciones no lineales. Además, existen antecedentes de sus aplicaciones financieras en temas tan diversos como: la medición del riesgo [11] [12], el mercado eléctrico español [13], el análisis de índices [15], etc. También encontramos su utilización para el tema que vamos a tratar, como es la selección de componentes para replicar el índice [14] [16], encontrando la base de la metodología aplicada mediante el autoencoder en [17] [18]. Esta se basa en el modelo CAPM, con la diferencia de que existen múltiples factores que se combinan de forma no lineal para dar lugar a los rendimientos de los activos.

Además de las redes, también utilizaremos con esta finalidad tanto la regularización LASSO (least absolute shrinkage and selection operator) como Random Forest, que son técnicas utilizadas con frecuencia para llevar a cabo selección de componentes.

2. Réplica de índices bursátiles

Un índice bursátil es un índice de referencia que se forma con un conjunto de valores cotizados en una bolsa de valores. Estos valores pueden estar dotados de diferentes pesos, que pueden depender de características como la capitalización bursátil.

Algunos de los motivos de su importancia son:

- Permiten medir el riesgo y la rentabilidad del mercado.
- Reflejan el sentimiento de mercado.
- Sirven de referencia para medir el desempeño de un gestor de activos.

- Se utilizan para crear carteras que reproduzcan su comportamiento, conocidas como carteras indexadas.

Cuando un fondo toma un índice bursátil como referencia, construye una cartera y compara su desempeño con el del índice, cambiando la composición de la misma de cuando sea conveniente para que el resultado de la comparación alcance el objetivo fijado para el fondo. Así, en función del objetivo que se establezca, se pueden distinguir dos modelos de gestión de cartera:

- La gestión pasiva busca replicar un índice, de forma que la cartera réplica obtenga unos rendimientos similares. Un ejemplo de uso de este tipo de práctica se puede ver en los fondos indexados.

- La gestión activa buscaría batir esta referencia, conformando una cartera que proporcionaría rendimientos mayores.

El problema de ambos modelos es que requieren cambios en la composición, y cada vez que esto sucede se deben pagar comisiones, tanto mayores cuanto mayor es el cambio en la composición del fondo. Esto dificulta la consecución del objetivo. Aunque los fondos pueden optar, por ejemplo, por reproducir el índice con la totalidad de los componentes, estos pueden estar compuestos por un número elevado de empresas (SP500), dando lugar a altas comisiones con cambios en los pesos o la propia composición. Por tanto, los fondos tratan de construir carteras basadas en una cantidad reducida de activos para así ahorrar comisiones. Concretamente, los fondos de gestión pasiva tratan de encontrar un número reducido de activos que resumen/gobiernan los movimientos del índice. Esta reducción de componentes se justifica en las siguientes características:

- . El peso de algunos títulos en el índice puede ser poco relevante. Evitar estos títulos puede no tener un impacto grande en la réplica y nos ahorraría comisiones por mantener activos en la cartera. Además un menor número de activos también permite reducir comisiones debidas a las transacciones.

- . Algunos títulos puedes presentar correlación positiva elevada entre ellos, por lo que un subconjunto de estos podría representar correctamente el comportamiento de todos.

3. El DAX como referencia.

Para este estudio se ha escogido el DAX [29] como índice de referencia. Esto se debe principalmente a su importancia para la economía europea. Además, el objetivo escogido es replicar un índice en nivel, que no reinvierte matemáticamente los dividendos, y sin tenerlos en cuenta, la rentabilidad de este índice sigue siendo bastante elevada.

El DAX es el índice bursátil que recoge las 30 compañías más grandes de Alemania, en términos de volumen de negociación y capitalización bursátil, que cotizan en la Bolsa de Fráncfort. Este índice lo podemos encontrar tanto en rendimientos como en precios, pero solo el primero reinvierte matemáticamente los rendimientos.

Se implementó por primera vez el 1 de julio de 1988 con un valor inicial de 1.000 puntos. Los precios para el cálculo del DAX los encontramos en XETRA [28], que es la plataforma electrónica de negociación de la Bolsa de Alemania. Esta plataforma calculaba el valor del índice cada 15 segundos hasta el 31 de diciembre de 2005 año. A partir del 1 de enero de 2006 se actualiza cada minuto, tomando los precios de XETRA.

Para profundizar un poco más, una compañía califica para la incorporación al DAX en función de los siguientes dos criterios:

- . Rotación de la cartera de pedidos en XETRA y el trading floor de Fráncfort (en los doce meses anteriores)
- . Capitalización de mercado free float (acciones en circulación - acciones restringidas) en una fecha específica (último día de negociación de un mes): esta cifra se calcula utilizando el precio promedio ponderado por volumen (VWAP) de los últimos 20 días de negociación.

Deutsche Börse publica los datos sobre estos dos criterios en sus "rankings de índices de acciones" mensuales, que sirven de base para la toma de decisiones en las revisiones de índices.

La composición original del DAX fue: Allianz, BASF, Bayer, Bayerische Hypotheken- und Wechselbank, Bayerische Vereinsbank, BMW, Commerzbank, Continental, Daimler, Degussa, Deutsche Bank, Deutsche Babcock, Deutsche Lufthansa, Dresdner Bank, Feldmühle Nobel, Henkel, Hoechst, Karstadt, Kaufhof, Linde, MAN, Mannesmann, Nixdorf Computer AG, RWE, Salzgitter AG, Schering, Siemens, Thyssen, Veba, Viag y Volkswagen.

Actualmente consta de los componentes siguientes: Adidas, Allianz, BASF, Bayer, BMW ST, Continental AG, Covestro, Daimler, Delivery Hero, Deutsche Bank AG, Deutsche Boerse, Deutsche Post, Deutsche Telekom AG, Deutsche Wohnen, E.ON SE, Fresenius Medical Care, Fresenius SE, Heidelbergcement, Henkel VZO, Infineon, Linde PLC, Merck, MTU Aero, Munich Re, RWE AG ST, SAP, Siemens AG, Siemens Energy AG, Volkswagen VZO y Vonovia. Conforman alrededor del 75 % de la capitalización del mercado alemán.

4. Machine Learning

Se llama machine learning [9] a un conjunto de métodos de análisis de datos cuyo objetivo es la construcción automática de modelos de relación entre variables con diferentes utilidades. Se trata de una vertiente de la inteligencia artificial que se basa en que los sistemas pueden reconocer patrones, aprender de datos y tomar decisiones sin la intervención de la mano humana, excepto por la definición previa de unas reglas básicas de aprendizaje a partir de la observación de la realidad y sus resultados.

Gracias a la capacidad de aprender y reconocer patrones, haciendo uso herramientas pertenecientes a este campo, podemos seleccionar el subconjunto de carteras que utilizar para replicar el índice de una forma óptima, que podría ser difícil de conseguir mediante otros métodos convencionales, en los que el observador introduce su particular percepción de la realidad en el modelo que define.

4.1. Concepto de aprendizaje, etapas y problemas asociados.

Anteriormente hemos hablado de la existencia de modelos que aprenden, pero sin especificar a qué nos referíamos. En términos generales, se define una función criterio, que llamaremos función de coste (FC(W)) y el aprendizaje consiste en definir reglas que automáticamente permitan el cambio de valor de los parámetros (W) que afectan a la función, típicamente buscando minimizarla. Cabe remarcar que, aunque en los métodos convencionales también minimizamos funciones objetivo, estas no cuestionan la forma del propio modelo. El aprendizaje de un sistema de Inteligencia Artificial frecuentemente tiene lugar en diferentes etapas o fases. La fase inicial, se entrena al método.

Esta afirmación en la jerga de la Inteligencia artificial, consiste en seleccionar la forma de aprendizaje, lo que involucra habitualmente la selección de la función de coste y la elección de ciertos valores llamados metaparámetros que determinan la forma básica en que se tratan los datos, la definición de las reglas de aprendizaje y la flexibilidad del algoritmo para adaptarse a los datos entre otras cosas. La segunda etapa, el entrenamiento, es el proceso por el que se permite al modelo aplicar las reglas de aprendizaje utilizando información muestral. Concretamente, cuando entrenamos un modelo para que aprenda, lo que buscamos es que, a partir de la información de entrada, descubra las diferentes tendencias, relaciones, etc, para así realizar una predicción que nos proporcionará como salida. Pero los datos de entrada están compuestos tanto por información relevante para ello, la señal, como por otra que no lo es, el ruido. Al entrenar el modelo en un conjunto de datos concreto, corremos el riesgo de que el modelo no capture correctamente la señal debido al ruido, siendo incapaz de captar al completo la información verdaderamente relevante. De esta forma estaríamos frente a un modelo incapaz de generalizar, dado que para una muestra diferente, con diferente ruido, no hay garantía de que sea capaz de proporcionar resultados acertados [7], [10], [3].

Se define como overfitting cuando nuestro modelo no ha conseguido aislar correctamente la señal, utilizando algunas de las características del ruido para obtener el resultado. Por ejemplo, clasificando variedades de flores en función de algunas características, como longitud del tallo, color de los pétalos y anchura de estos, se da el caso accidentalmente, en los datos que se utilizan para el aprendizaje, que todas las flores de uno de los tipos presentan tallo largo, sin ser esto necesariamente así (ruido), podemos tener problemas a la hora de generalizar, descartando algunas por tener tallo corto aunque pertenezcan al grupo.

Por otro lado, se define como underfitting cuando al intentar aislar la señal, se descarta parte de ésta, omitiendo información relevante. Continuando con el ejemplo de las flores, si entrenamos con una cantidad demasiado pequeña de observaciones, pueden dejar de captarse patrones relevantes para la clasificación, como podría ser el tamaño de los pétalos. Nuevamente, podemos tener problemas al generalizar, dado que el modelo ignorará características que a priori son importantes para realizar la predicción.

Debido a la gran flexibilidad que presentan los modelos pertenecientes al machine learning, si todo se lleva a cabo con el mismo conjunto de datos, es fácil que la elección de los hiperparámetros y parámetros quede contaminada,

incorporando el modelo tanto la señal como el ruido. Para evitarlo, la muestra se separa en conjuntos de train, test y validación.

4.2. Tipos de aprendizajes.

Encontramos dos tipos diferentes de aprendizajes en función de que conjuntos de datos utilizamos para realizar el entrenamiento. Estos son:

- **Aprendizaje supervisado**. Se conoce el valor de la(s) variable(s) cuyo comportamiento se desea reproducir en diferentes situaciones así como el de otras variables que se cree que influyen en dicho valor en cada una de las situaciones. A la primera la llamamos salidas o outputs en la jerga de Inteligencia Artificial, y a las segundas se conocen como entradas o inputs. El aprendizaje consiste en tratar de descubrir la forma en la que se relacionan las variables para sacar provecho de ello. Esto puede consistir en dar un diagnóstico de la situación del output cuando se conocen los inputs (por ejemplo en diagnostico médico), elegir los inputs para hacer más verosímil un cierto valor del output (por ejemplo en resultados de tratamientos o como en el caso que nos interesa, la construcción de una cartera de réplica en finanzas), etc. En la función de coste encontraremos algún tipo de métrica que refleje la discrepancia entre los valores obtenidos y los previamente conocidos, dado que buscamos reproducirlos.

- **Aprendizaje no supervisado**. En este caso no se utiliza el valor de las salidas, por tanto la función de pérdidas que se utilice no dependerá de estas.

Para la elaboración del presente apartado se han utilizado las referencias [6], [9], [2].

4.3. Conjuntos de entrenamiento, test y validación. Validación cruzada.

La submuestra utilizada para entrenar un modelo se conoce como conjunto de entrenamiento y es el resultado de dividir en dos el conjunto de información que tenemos. La otra parte recibe el nombre de conjunto test [6], [10].

El conjunto train o de entrenamiento, como su nombre indica, se utiliza para llevar a cabo la elección de los parámetros del modelo. Es habitual, sobretodo cuando tratamos con redes neuronales, que de este conjunto se desprenda un tercero conocido como conjunto de validación. Este nos permite comparar el modelo para diferentes valores de algunos de sus hiperparámetros o establecer un criterio de parada durante el entrenamiento, comparando el rendimiento entre conjuntos.

El conjunto test sirve tanto para comprobar si el modelo funciona correctamente, sin problemas de generalización, como para comparar diferentes posibles estructuras o modelos.

Para comprobar la generalización, se obtienen las predicciones que realiza sobre el conjunto test, de los cuales conocemos los resultados, y utilizando la misma métrica en ambos conjuntos (error cuadrático medio, error absoluto medio, etc) vemos si el resultado es similar. Si lo es, podemos concluir que nuestro modelo está generalizando de forma correcta. Si no lo es, esto nos indica que el modelo no ha sido entrenado correctamente, lo que puede indicar overfitting (estas comprobaciones las hemos llevado a cabo para cada uno de los métodos). Si queremos comparar diferentes estructuras, lo que conviene es ver, con la misma métrica, para cuál de ellas obtenemos un mejor resultado.

En algunas situaciones, podemos querer evaluar los resultados de un análisis estadístico (en nuestro caso, en la sección 7.2, calcularemos el error de LASSO, para ver su variación con los metaparámetros), pero estos pueden depender de la partición train-test que se realice. Para evitarlo, podemos llevar a cabo la técnica utilizada como validación cruzada [6], [7], [4], [3], [2]. Existen diferentes formas de implementarla, pero vamos a presentar el caso particular que utilizaremos más adelante conocido como K-folds con X splits. Esta consiste en:

- Dividimos el conjunto test en X subconjuntos con la misma cantidad de datos.
- Iteramos X veces. En cada una de estas iteraciones, apartamos uno de los subconjuntos, entrenamos el modelo con la información restante y llevamos a cabo el análisis estadístico. El conjunto no utilizado para el entrenamiento es el conjunto de validación.
- El resultado del análisis será el promedio de las X obtenidas mediante las diferentes iteraciones.

4.4. Entrenamiento.

Para entrenar muchos modelos pertenecientes al Machine Learning, lo que buscamos es minimizar la función de coste, que define el criterio de similitud entre los outputs que queremos obtener y los resultados que arroja el modelo para aproximarlos. Esta función puede ser algo tan simple como la suma del cuadrado de las diferencias entre salidas obtenidas y los valores que conocemos para dichos valores, si estamos tratando un caso de aprendizaje supervisado.

Centrándonos ya en el proceso propiamente dicho, en primer lugar se inicia el modelo con pesos (o parámetros) arbitrarios. A continuación se obtiene la salida que proporciona esta y se evalúa la función de coste. Entonces tiene lugar un proceso iterativo consistente en, mediante la modificación de los diferentes pesos de los enlaces, llevar la función de pérdidas lo más próxima a cero como sea posible, para así obtener el valor óptimo para los diferentes pesos.

Hay una variedad de algoritmos para llevar a cabo esta selección de parámetros. En este trabajo trataremos el descenso por gradiente y algunas de sus variantes, que son los más habituales en aplicaciones. En las siguientes secciones los introduciremos brevemente buscando una mejor comprensión de los resultados obtenidos, utilizando como referencia para su redacción [10], [9], [1], [3], [4].

4.4.1. Descenso por gradiente

Recuérdese que el valor de la función de costes (FC) depende de los parámetros del modelo. Dado que el gradiente de la función señala la dirección de máxima variación de FC, el método propone, modificar el valor de los parámetros en la dirección opuesta a la marcada por dicho gradiente para conseguir otros que proporcionen un mejor valor de FC. La magnitud de la modificación, también llamada ratio de aprendizaje, es un metaparámetro (aquellos parámetros del módelo que no se ven afectados por el proceso de optimización, siendo ejemplo de esto, el número de capas, las neuronas por capa, etc) del modelo cuyo valor se determina exógenamente al entrenamiento de la red. Una vez definido el nuevo valor de los parámetros, evalúa FC y su gradiente, y se propone un nuevo valor para los parámetros con el mismo criterio del gradiente y magnitud de desplazamiento. El proceso de repite todas las veces que sea necesario hasta cumplir un criterio de convergencia,

por ejemplo, que la mejora alcanzada en las nuevas iteraciones, medida en términos del valor de FC, sea insignificante. El tamaño crítico para considerar si el cambio es insignificante también es un metaparámetro del método.

Formalmente, podemos definir las ecuaciones que definen este proceso iterativo de determinación del valor de los parámetros como:

$$W^{i+1} = W^i - \alpha \frac{\partial FC}{\partial W^i} \tag{1}$$

donde FC es la función de coste, W el vector de pesos, α el ratio de aprendizaje, W^i los pesos iniciales de la iteración y W^{i+1} los que obtenemos al final de esta.

Cabe destacar la importancia de elegir un buen ratio de aprendizaje, dado que elegimos uno muy pequeño, tardará mucho en llegar al mínimo, pero si es demasiado grande, nos podremos encontrar con que los valores de la función de coste irán dando vueltas alrededor del mínimo que esperamos encontrar, ya que dará saltos demasiado grandes como para acercarse.

Este problema dista de ser sencillo de resolver, dado que el número de parámetros es muy elevado, y consecuentemente también lo es el espacio en el que llevar a cabo la optimización. Además, debemos tener en cuenta el batch size, que es el número de observaciones que se tienen en cuenta cada vez que el algoritmo recorre los parámetros del modelo.

Algunas variantes de este método son:

- **Descenso por gradiente estocástico**. Debido a la gran cantidad de pesos y observaciones que se tienen en cuenta, el cálculo de la derivada parcial es, como poco, bastante costoso a nivel computacional. Para evitar esto, se introduce un factor estocástico, consistente en elegir solo una de las observaciones consideradas en el batch size.

- **Adaptive Gradient Algorithm (AdaGrad)**. En lugar de considerar un ratio de aprendizaje constante constante para todos los pesos, se mantiene un factor específico para cada uno de ellos. Dado que calcular estos factores de cero resulta inviable, AdaGrad toma el valor inicial del ratio, lo escala y adapta para cada dimensión, teniendo en cuenta el gradiente acumulado en cada iteración.

- **Adadelta**. Variación de AdaGrad que, en lugar de considerar el gradiente acumulado desde el inicio solo tiene en cuenta las últimas a iteraciones.

Como podemos ver, estos métodos son costosos a nivel computacional. Este es el principal motivo por el que, en caso de ser posible, se utilizan alternativas, aún siendo correcto. Es el caso que encontraremos con LASSO, donde lo utilizamos para explicar su funcionamiento debido a la facilidad que nos proporciona para ilustrarlo, aunque la optimización se haya realizado mediante otro método más eficiente.

5. Inteligencia Artificial para el diseño de la cartera de réplica

A continuación presentamos brevemente las características principales de las técnicas de Inteligencia Artificial que hemos utilizado en este trabajo para abordar el problema que hemos planteado.

5.1. Regularización.

La regularización [3], [2] consiste en la introducción de un término en la función de costes que penalice la complejidad del modelo. Puede ser un método en sí mismo o incorporarse a otros con el objeto de prevenir el overfitting.

Así por ejemplo, si se dispone de información proporcionada por muchas variables, la penalización puede consistir en un término que aumente el valor de la función de costes si utiliza la información de una cantidad grande de ellas. En concreto, en el ámbito de aprendizaje supervisado, dado un output Y, y un modelo lineal para su ajuste por mínimos cuadrados utilizando el valor de inputs, X, la regularización podría consistir en plantear como función de costes a

$$FC = (Y - XW)^2 + \beta P(W) \tag{2}$$

donde P, la penalización, es función de los parámetros del modelo. Por otro lado, β es el factor que determina la magnitud de la penalización y es otro metaparámetro del método. Aquí podemos ver que la función de coste es fácilmente interpretable como un Lagrangiano, donde el primer término es la función sin tener en cuenta ninguna restricción y el segundo el producido por esta. Esto nos ayudará a entender como funciona el método.

Como hemos visto, la regularización persigue encontrar un compromiso entre ajuste y simplicidad del modelo.

5.1.1. Regularización LASSO o L1

En la regularización L1 o LASSO (least absolute shrinkage and selection operator) la penalización aplicada a los coeficientes del modelo es la media de sus valores absolutos, es decir,

$$P = \sum_{i=1}^{N} |W_i| \tag{3}$$

donde N es el número de coeficientes.

Para ver que diferencias existen al incorporar esta penalización frente a no hacerlo, aplicamos el descenso por gradiente, dado que nuestro objetivo es minimizar la función de coste. Recordemos que la expresión que sigue la podemos encontrar en la ecuación 1. En ella podemos ver que necesitamos la derivada de la función de coste, pero antes de ello vemos la del valor absoluto, que responde a

$$\frac{\partial |W_i|}{\partial W_j} = \begin{cases} sgn(W_i) \ si \ i = j \\ 0 \ si \ i \neq j \end{cases} \tag{4}$$

donde sgn es la conocida función signo. Con esto, el resultado de aplicar el descenso por gradiente una vez sería

$$W_i^{l+1} = W_i^l - \alpha(\frac{dFC_i^l}{dW_i^l} + \beta \frac{d|W_i|}{dW_i}) \tag{5}$$

excepto cuando el parámetro en cuestión es nulo, dado que en esta situación, se deja de actualizar. En la ecuación anterior podemos distinguir dos casos:

$$W_i^{l+1} = \begin{cases} (W_i^l - \alpha \frac{dFC_i^l}{dW_i^l}) - \alpha\beta \ si \ w_i > 0 \\ (W_i - \alpha \frac{dFC_i^l}{dW_i^l}) + \alpha\beta \ si \ w_i < 0 \end{cases} \tag{6}$$

donde el término común sería el resultado de aplicar el método a la función de coste original. El otro término empuja los pesos en la siguiente iteración $W^{\hat{\imath}+1}$ hacia 0. Así, si para fijar ideas se considera que la derivada parcial es positiva, determina una reducción del valor del parámetro que finalmente se ve acentuada por la penalización.En cambio, si la derivada fuera negativa, la primera parte del algoritmo aumentaría el valor del parámetro pero si el valor de la derivada no es suficientemente grande, la ganancia en términos de FC no compensa y el término de penalización del algoritmo del gradiente

reduce en todo caso el valor del parámetro. Esto lo que hace es que empuja los pesos hacia el 0, haciendo que de esta forma las variables menos relevantes desaparezcan del ajuste final. Gracias a esto, la regularización LASSO nos puede ayudar a reducir el número de variables a utilizar en la regresión, dado que puede arrastrar a 0 varios coeficientes correspondientes a las acciones de algunas de las empresas del DAX (las que en principio serían menos relevantes, ya que permiten un buen ajuste), llevándonos así a una selección. Cabe remarcar que, en caso de imponer la positividad de los parámetros, desaparecería la segunda opción de la ecuación 6.

Una forma alternativa de entender el efecto de la regulación LASSO es a partir de la interpretación de la función objetivo penalizada en términos de un Lagrangiano correspondiente a una optimización restringida, de forma que para cada penalización β hay un problema restringido equivalente. Con esto en mente, podemos ver en la figura 1 el motivo detrás de que se produzca la selección. En esta encontramos representado el problema de optimización para el caso concreto de regresión LASSO con dos coeficientes, donde el cuadrado es la zona restringida (dos valores absolutos) y la elipse las curvas de nivel de la función de coste no restringida. Dado que la solución del problema se halla en una intersección, si el espacio restringido es lo suficientemente pequeño, es plausible esperar que puede tener lugar en una de las esquinas del cuadrado, donde alguno de los parámetros es 0. Si en lugar de dos parámetros tenemos más, puede anularse más de uno a la vez. Cabe remarcar que un mayor hiperparámetro β lleva a un menor cuadrado, y por tanto, es más fácil que la intersección tenga lugar en una esquina en lugar de en una área donde los diferentes parámetros tomen valores no nulos.

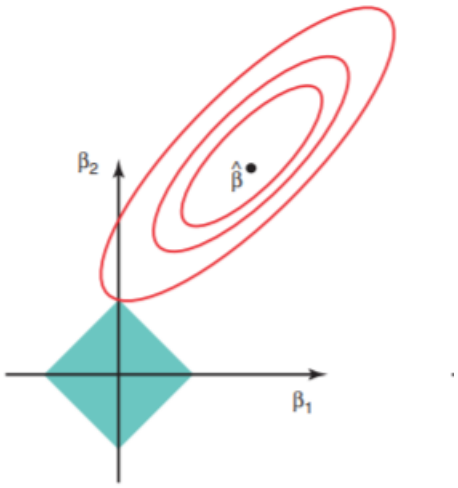


Figura 1: Representación de una regresión LASSO con dos parámetros extraída de la referencia [3].

Para finalizar, debemos tener en cuenta que el valor óptimo de β, el metaparámetro de la regresión, dependerá de dónde se pretenda llegar con la regresión. Dado que en el presente trabajo se busca reducir el número de componentes utilizados para reproducir un índice, este valor deberá ser lo suficientemente grande como para cumplir con dicho objetivo, dejando solo los activos con los que contruir nuestra cartera réplica.

5.2. Random Forest

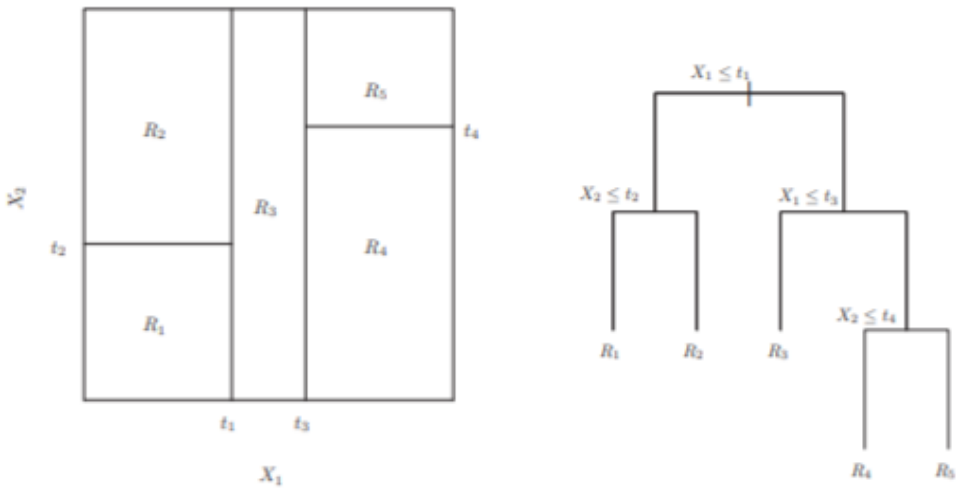
Para hablar de Random Forest, primero necesitamos conocer en qué consiste un árbol de decisión, dado que es una combinación de estos. Para la redacción de los siguientes apartados se han utilizado las referencias [3], [2], [6].

5.2.1. Árbol de decisión

Los árboles de decisión son una de aprendizaje supervisado más utilizadas. El modelo consta de una serie de nodos, en los cuales se toman decisiones lógicas de forma secuencial. Por ejemplo, si quisiéramos conocer si a alguien le deberíamos conceder un crédito, podríamos ver si en primer lugar obtiene más o menos de X ingresos. En caso afirmativo podríamos querer saber si

lleva más de Y años en su trabajo, o si tiene algún aval. Un árbol de decisión no es más que una acumulación de este tipo de decisiones, donde empezamos en el nodo conocido como raíz, pasamos por los nodos intermedios y nuestro resultado es proporcionado por un nodo terminal, que serían las hojas del árbol.

Al final, los árboles de decisión lo que hacen es atribuir un valor constante al output dentro de regiones que suelen ser rectángulos definidos en el espacio de los inputs. Por ejemplo, cuando se define un valor crítico (threshold) en una variable (se elige una frontera para tomar una decisión u otra), y se pregunta cual es el mejor predictor para valores superiores al valor crítico y otro para los inferiores, y esto se hace luego con otra variable, lo que se hace es particionar el dominio de los inputs en 4 pedazos y elegir el predictor constante dentro de cada uno de los pedazos. La ventaja principal de estos métodos en la sencillez de interpretación. Esto lo podemos ver en la figura 2. En ella encontramos un caso con dos variables, donde el dominio de la variable X_1 se ha dividido en tres regiones, mientras que el de X_2 se descompone en función del valor de la primera.



(a) Output de una partición binaria recursiva en un caso con dos características.

(b) Árbol correspondiente a la partición.

Figura 2: Ejemplo de árbol de decisión extraído de la referencia [2].

La idea es que a medida que se vaya separando la muestra de entrena-

miento a través de los diferentes nodos, se vayan formando grupos cada vez más homogéneos, hasta que las decisiones no generen grupos más uniformes.

Debido a que el método consiste en fragmentar el espacio formado por las variables de entrada y asignar un valor de salida para cada uno de estos subespacios, nada impide que este método sea capaz de capturar relaciones no lineales.

5.2.2. Calibración del árbol

Conocemos que la finalidad del árbol de decisión es separar en grupos lo más homogéneos posibles. Para ello necesitamos un criterio, que para una regresión numérica puede ser tanto el error cuadrático medio como el error absoluto medio. La mejor división será aquella que al realizarla reduzca en mayor medida este valor. Por ejemplo, si tomamos el RMSE, para cada característica, se determina el valor óptimo para realizar la división como aquel que produzca una mayor reducción del RMSE. El RMSE se calcula como la suma ponderada de los RMSE de cada mitad resultante. Tras ello, se compara la reducción entre las diferentes características (empresa, etc), para seleccionar, nuevamente, aquella que proporciona un mejor resultado. Esto se repetirá hasta que queden grupos homogéneos o se llegue al límite de profundidad del árbol que se escoja.

Al final, lo que hacemos es escoger para cada nodo aquella división que da lugar a una mayor reducción del RMSE.

5.2.3. Pasar del árbol al bosque

Los árboles de decisión son muy sensibles a los datos utilizados para entrenarlo. Una de las formas técnicas que podemos utilizar para reducir la varianza debida a esta sensibilidad es el bagging. Esta consiste en muestrear con reemplazamiento la muestra original y construir un árbol de decisión para esta muestra artificial. Cuando esto se repite un número B de veces se tendrán B árboles de decisión distintos y el bagging propone como ajuste el promedio de todos estos árboles artificiales.

Random Forest consiste en realizar este bagging, pero al entrenar los árboles en cada muestra artificial, solo se utilizan L inputs/variables seleccionadas aleatoriamente entre el conjunto de todas las características disponibles.

Debido a que no en todos los árboles utiliza las mismas características, Random Forest puede determinar que características son más relevantes,

viendo si los árboles donde aparecen obtienen o no buenos resultados.

5.2.4. Importancia de los componentes

El criterio que tendremos en cuenta es el de la importancia de Gini, cuyo nombre se debe a que en los árboles utilizados para clasificación, en lugar del RMSE, pueden utilizar el índice Gini, que es una medida de la homogeneidad, además de la base de la ordenación.

Este método consiste en, para cada división y cualquiera de los árboles del modelo, calcular la reducción del RMSE (o MAE). Con esto, para cada atributo, se obtiene la reducción promedio que tiene lugar cuando se realiza una división a partir de este, siendo más importante cuanto mayor sea esta reducción. De esta forma Random Forest nos permite conocer los componentes más importantes, permitiéndonos así seleccionar un subconjunto de ellos para realizar la réplica.

5.3. Redes neuronales artificiales

5.3.1. La neurona

Consideremos una muestra del vector de k+1 variables aleatorias $(Y, X_1, X_2, ..., X_k)$. Supongamos que se dispone de una muestra de tamaño n de dicho vector. Una neurona es una función de R^k en R que a cada $X_1, X_2, ..., X_k$ le atribuye un valor $N(W_0 + W_1X_1 + ... + W_kX_k)$, donde N es una función conocida como de activación que es generalmente no lineal. Los diferentes W_i son los pesos del método y se eligen de forma que el resultado de la transformación sea lo más similar posible a todos los valores muestrales de Y. Si el criterio de similitud es el de sumar los cuadrados de las discrepancias de todos los datos en la muestra, la neurona es simplemente un modelo lineal generalizado. Pero al contrario de este modelo, la neurona no busca ajustar, sino predecir un valor Y externo a la muestra, utilizando como variables de control los valores X que aparecen junto a él. Podemos encontrar una representación en la figura 3. Para la redacción de los siguientes apartados, se han utilizado las referencias [7], [10], [4], [3], [1].

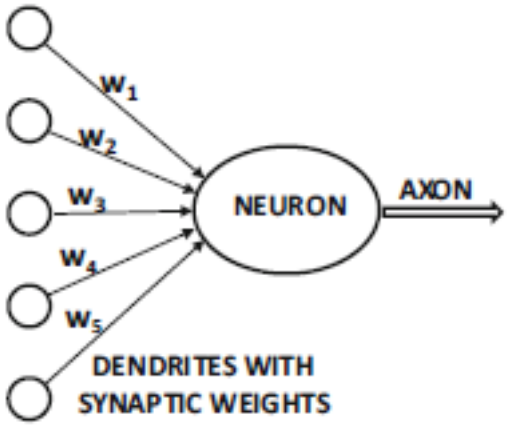


Figura 3: Representación esquemática de una neurona artificial [1]

Hay una gran variedad de funciones de activación, tanto lineales como no lineales (sigmoidal, tangente hiperbólica, ReLu). Es importante tener en cuenta que para resolver un problema no lineal necesitaremos una función de activación no lineal. Esto es debido al propio funcionamiento de las neuronas, ya que combinando funciones lineales nunca llegaremos a obtener una función no lineal.

5.3.2. La red neuronal

Una red neuronal es un algoritmo que propone predecir Y con el valor que arroja el resultado de alimentar neuronas secuencialmente con el resultado de otras, de forma que los valores transformados por unas se convierten en los inputs de las siguientes. El resultado es una composición de funciones N que permite que la salida sea altamente no lineal si el problema en cuestión así lo requiere. La estructura de la red neuronal, es decir, la forma en la que se relacionan unas neuronas con otras es ad-hoc, aunque hay algunas que por su buen comportamiento aparecen reiteradamente en aplicaciones y cuyas propiedades teóricas se conocen mejor en el apartado 5.3.3.

Es común que las encontremos las neuronas en forma de redes, que se organizan en diferentes capas. La primera, encargada de recibir los inputs del modelo, recibe el nombre de capa de entrada.

Tras la capa de entrada, podemos encontrar una capa intermedia u oculta, cuyas entradas son las salidas provenientes de la capa de anterior. Este

esquema se puede repetir las veces que se crea conveniente para el modelo hasta llegar a la última, la cual proporciona los outputs y recibe el nombre de capa de salida. En la figura 4 podemos ver un ejemplo de la estructura de una red que consta de tres neuronas de entrada, una capa oculta y una única neurona de salida. Es habitual encontrar que la última capa consta de una única salida, dado que son las necesarias para clasificar o realizar una regresión. Al final, la salida es el resultado de componer sucesivamente las diferentes funciones de activación.

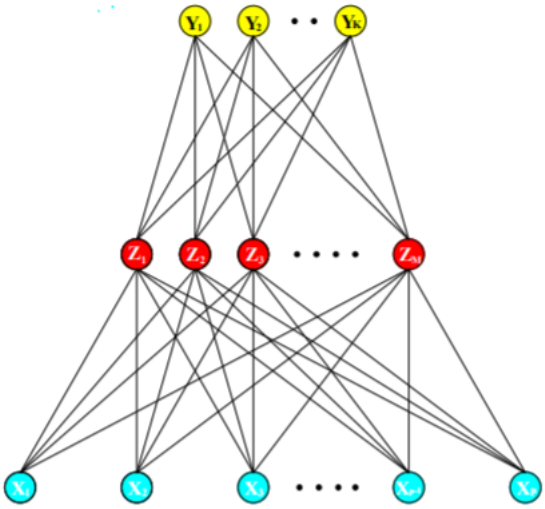


Figura 4: Ejemplo de una red neuronal con una única capa intermedia [3].

A la hora de construir una red, podemos variar el número de capas, el número de neuronas por capa, cómo están conectadas las neuronas y la función de activación que utilizamos en cada una de las capas, siendo la propia estructura un metaparámetro. Esto da lugar a que este tipo de estructuras computacionales presenten una gran flexibilidad, pudiendo utilizarse en problemas de diversa índole.

5.3.3. Tipos de redes según su tipología

Los diferentes tipos de redes podemos encontrar según la estructura que presentan son:

- **Precepton**. Se trata de una red constituida por una sola neurona, teniendo así la expresión más simple de red neuronal que podemos encontrar. Esta recibe una serie de entradas, que son ponderadas y pasan por la función de activación, dando lugar a la salida.También se utiliza este término para referirse a redes prealimentadas (feedforward) de una sola capa.

- **Red prealimentada o feedforward**. Este tipo de redes solo pueden transmitir la información en una dirección.

- **Redes neuronales recurrentes (RNN)**. Las neuronas pueden enviar su información de salida a otras de capas anteriores. Gracias a esta retroalimentación este tipo de redes neuronales pueden desarrollar memoria.

- **Redes neuronales convolucionales(CNN)**. Las neuronas de una capa no están conectadas con todas las capas siguientes, sino con un subconjunto de estas. Este tipo de redes se utilizan para el reconocimiento de imágenes.

5.3.4. Autoencoder

Un autoencoder es una técnica de reducción de la dimensionalidad que utiliza dos redes neuronales feedforward que se acoplan dando lugar a una estructura simétrica. En el conjunto resultante el número de neuronas por cada capa se reduce a medida que nos movemos de uno de los extremos hacia el centro, como podemos ver en la figura 5. La primera red recibe el nombre de encoder y es la encargada de la reducción de la dimensionalidad. La segunda, llamada decoder, trata de recuperar la información de entrada a partir de la reducida, producto del encoder. Los parámetros se eligen de forma que la salida del decoder sea lo más parecida a la entrada del encoder.

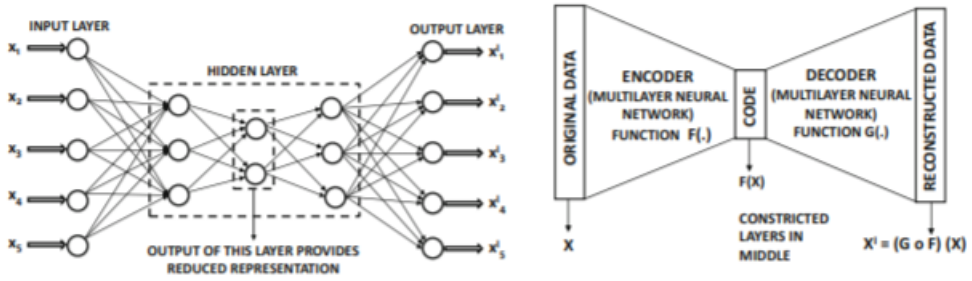


Figura 5: Izquierda: autoencoder con tres capas intermedias. Derecha: esquema con las partes del autoencoder. Referencia [1].

De esta forma se obtiene un método de aprendizaje no supervisado que se plantea la reducción de la dimensionalidad de un conjunto de variables, es decir, si K es el número de neuronas en la capa central, busca los K factores que mejor explican no linealmente el valor de las variables originales. Para ello, define como función de coste la disimilitud entre el valor de las variables inputs y el de los outputs.

Estos K factores vendrían a ser el equivalente a los obtenidos mediante un análisis de componentes principales, pero captando dependencias no lineales.

5.3.5. Selección de componentes del índice utilizando Autoencoder

En nuestro caso, dado que buscamos seleccionar un número de factores de entre los originales, no estamos interesados en la capa intermedia sino en la salida del autoencoder. Estos serán una réplica, más o menos exacta, de las entradas, por lo que viendo cuánto se desvían de la entrada podremos ver si capturan en mayor o menor medida la información de los factores comunes. Es decir, cuanto más se parezca una variable de salida a su correspondiente entrada, será porque los factores que hay detrás, aquellos de la capa intermedia, son capaces de explicarla en mayor medida. Es decir, los factores comunes estarán muy relacionados con dichas variables. Esto nos permite seleccionar que carteras elegir, ya que aquellas que mejor reflejen la información del mercado nos permitirán describirlo con mayor precisión. Esto podríamos entenderlo como una especie de CAPM con varios factores y no lineal, ya que la segunda mitad del autoencoder reconstruye las carteras a partir de los factores de mercado.

6. Revisión de la literatura.

Existe en la literatura financiera diversos trabajos relacionados con la selección de cartera y/o el estudio de los índices bursátiles, utilizando en muchos casos técnicas relacionadas con el machine learning.

Respecto a la selección de cartera, encontramos artículos como el de Alberto Fernández y Sergio Gómez (2005) [14], donde se utilizan redes neuronales con la finalidad de calcular la frontera eficiente y obteniendo una generalización del modelo de Markovitz de media-varianza aplicando diversas restricciones. Además compara el resultado obtenido con este método con otros tres heurísticos, siendo el primero el que proporciona mejores resultados. En 2006, Rafael Moral-Escudero, Rubén Ruiz-Torrubiano y Alberto Suárez [24] también extendieron el problema de selección de la cartera óptima para añadir restricciones. El mismo año, Chi-Ming Lin, Jih-Jeng Huang, Mitsuo Gen y Gwo-Hshiung Tzeng [25] presentan un modelo para selección de cartera dinámica haciendo uso de redes neuronales. Recientemente encontramos metodologías más innovadoras para resolver este tipo de problemas, como presentaron Samuel Fernandez-Lorenzo, Diego Porras y Juan José García-Ripoll en 2020 [19], donde utilizan un modelo híbrido cúantico-clásico con circuitos cuánticos basados en compuertas variacionales para llevar a cabo la selección de componentes, para posteriormente, mediante un algorítmo de poda heurístico, obtener la combinación ponderada de activos sujetos a restricciones de cardinalidad.

Limitándonos ya al campo de los índices bursátiles, encontramos el artículo publicado en 2008 por Rubén Ruiz-Torrubiano y Alberto Suárez [23]. En él se utiliza un algoritmo genético para seleccionar los activos a utilizar y posteriormente la programación cuadrática se encarga de obtener los pesos. El año 2012, Carla Moreno Navarro [27] presentó un criterio de selección de los componentes para replicar el Ibex35 basado en el análisis por componentes principales, donde veía que activos funcionaban de forma independiente de los componentes obtenidos para seleccionar aquellos que proporcionaban información no superpuesta con el resto. Un año después, N. C. P. Edirisinghe publicó un artículo en el que comparaba, para el SP500, la selección de cartera mediante tracking eficiente con la aproximación de media-varianza de Markovitz. En 2016, J. B. Heaton, N. G. Polson y J. H.Witte [21] publicaron un artículo con la finalidad de mostrar el interés de aplicar deep learning, como los autoencoder, en los mercados financieros. Lo ilustran con un ejemplo con el índice IBB, donde seleccionan inversiones con buenas propiedades de

seguimiento. Dos años después, Hongbing Ouyang, Xiaowei Zhang y Hongju Yan [17] presentan, para el Hang Seng Index (HSI) un modelo basado en autoencoder para seleccionar los componentes más relevantes de un índice a la hora de replicarlo. Tanto este artículo como el publicado por Chi Zhang, Shuang Liang, Fei Lyu y Libing Fang en 2020 [18], también para el HSI, consisten en utilizar autoencoders para replicar, a partir de información de menor dimensionalidad (capa intermedia), los componentes del índice. Estos los interpretan como una especie de modelo CAPM multifactorial no lineal, llevando a que los mejores componentes para la réplica son los que se reconstruyen con mayor fidelidad. Esto se debe a que los factores de la capa compacta viene se interpreta como factores de mercado y aquellos que mejor respondan a esta aportarán mayor información. En 2019, Yu Zheng, Bowei Chen, Timothy M. Hospedales y Yongxin Yang [20] abordan el problema de la obtención de la cartera que minimiza el tracking error, pero con un enfoque diferente. Normalmente este problema se resuelve mediante métodos heurísticos, pero ellos reparametrizan el problema y lo resuelven mediante una red neuronal estocástica. El año siguiente, Chia-Cheng Chen, Chun-Hung Chen y Ting-Yin Liu [15] compararon, para el SP500, la capacidad de predicción de tres modelos diferentes pertenecientes al machine learning, siendo estos las redes neuronales artificiales, random forest y máquinas de vectores soporte. Estos conseguían rendimientos superiores al propio índice. En 2021, Reza Bradrania, Davood Pirayesh Neghab y Mojtaba Shafizadeh utilizan una red neuronal para que les indique el mejor criterio para llevar a cabo la selección de cartera, utilizando como input información del mercado.

Existe una amplia literatura, tanto para el problema de selección de cartera como para el tema del machine learning. No solo eso, dado que si nos fijamos en las fechas y el tipo de publicaciones, el interés por aplicar técnicas pertenecientes al mundo del machine learning a las finanzas, concretamente a la selección de carteras y estudio de índices bursátiles, parece ser cada vez mayor.

7. Aplicación de la metodología propuesta

Para la realización de los siguientes apartados se ha utilizado tanto el lenguaje de programación Python como varias de sus bibliotecas.

7.1. Obtención de los datos

Las series utilizadas para el presente trabajo son las correspondientes a la Bolsa de Fráncfort, obtenidas de Yahoo Finance, para el Dax y las empresas Adidas, Allianz, BASF, Bayer, Beiersdorf, BMW, Continental, Covestro, Daimler, Deutsche Bank, Deutsche Börse, Deutsche Lufthansa, Deutsche Post, Deutsche Telekom, E.ON, Fresenius, Fresenius Medical Care, HeidelbergCement, Henkel, Infineon, Linde, Merck, MTU Aero Engines, Münchener Rück, RWE, SAP, Siemens, Volkswagen Group, Vonovia y Wirecard, para el periodo que abarca desde el 2015-10-07 hasta el 2021-04-03. Debido al rango escogido, influido por la necesidad de una cantidad elevada de observaciones, se ha tomado la composición del índice a fecha de marzo de 2020. Esta decisión de debe a que se ha considerado que las empresas incorporadas de forma más reciente no tendrán la relevancia suficiente a lo largo de estos años, aunque podrían añadirse sin complicaciones.

Cabe remarcar que la empresa Wildcard quebró en 2020, así que en caso de ser seleccionada se deberá tener en cuenta.

Añadir que, como el objetivo de replicar un índice es obtener los mismos rendimientos que proporciona, los diferente métodos de selección trabajarán con ellos, así que los obtendremos a partir de los precios.

7.1.1. Tratamiento de los datos.

En primer lugar, lo que necesitamos es tratar las observaciones faltantes en las series temporales, aunque al ser pocas (hablamos de unas 20 como máximo en series de más de 1300 observaciones), el método no tiene un impacto significativo. Para ello, se ha decidido sustituir los espacios vacíos con una copia del valor anterior en la propia serie, dando lugar a rendimientos logarítmicos nulos (dado que el signo de los rendimientos es, en principio, aleatorio, evitamos así introducir datos con un signo determinado). Si bien esta es una opción, también podríamos haber utilizado la media de los valores que encontramos a su alrededor, pero nunca reemplazar los valores con ceros. Esta última opción no solo proporciona valores mucho más alejados de la realidad, sino que trae problemas al calcular rendimientos logarítmicos.

Recuérdese que el objetivo planteado en este trabajo es el de reproducir los valroes/rendimientos del DAX con una cartera de pocos activos.

7.2. Regularización LASSO

Hemos visto con anterioridad en qué consiste la regularización LASSO y como nos puede ayudar a seleccionar los componentes más relevantes para reproducir el DAX. Esto no es tan simple como realizar directamente una regresión en la que se ajuste el DAX a través de sus componentes, dado que es importante utilizar el hiperparámetro β (el factor de escala que acompaña la penalización) adecuado para aquello que buscamos.

En primer lugar, separamos la muestra de la que disponemos en dos subconjuntos: train, donde entrenaremos el modelo, y test, donde lo evaluaremos. Para ello utilizaremos la función train test split de la biblioteca sklearn.

A continuación, buscamos realizar una regresión LASSO para un espectro de parámetros β que comprende desde 10^{-10} hasta 10^2. De esta forma podremos ver como evoluciona el modelo para los diferentes valores. Normalmente realizaríamos una regresión LASSO estándar, pero en este caso utilizaremos una con validación cruzada, ya que de esta forma podemos obtener una estimación más precisa del error. Consiste en, para cada regresión, dividir la muestra de entrenamiento en varias de igual tamaño mediante extracciones aleatorias sin repetición, 10 en nuestro caso. Se entrena el modelo con las observaciones de 9 de estos bloques y se validan los resultados con las observaciones del bloque restante calculando el error que comete el ajuste para ese conjunto de datos. Se repite el proceso 10 veces, cada vez apartando para la validación/test un bloque diferente. El error para este modelo es el promedio de los 10 anteriores, evitando de esta forma que se obtenga un error artificialmente bajo debido a un sobreajuste del modelo. El error utilizado es el RMSE. Además, en las diferentes regresiones del apartado se han impuesto (o no impuesto) las siguientes restricciones:

- **Positividad**.No consideramos ventas en corto.
- **Constante nula**. Buscamos que nuestros rendimientos reproduzcan los del índice, por lo que no queremos una parte de estos que no sean recogidos por los activos constituyentes.
- **No se ha impuesto que la suma de los pesos sea la unidad**. Pretendemos encontrar la cantidad a invertir en cada activo, no los pesos en la cartera.

A continuación generamos un modelo idéntico, pero sin validación cruzada, dado que nos interesa mostrar como cambia el desempeño del modelo

para diferentes valores del hiperparámetro de penalización y para ello no necesitamos el error. Lo utilizamos para obtener la variación de los coeficientes con β, que podemos ver en la figura 6.

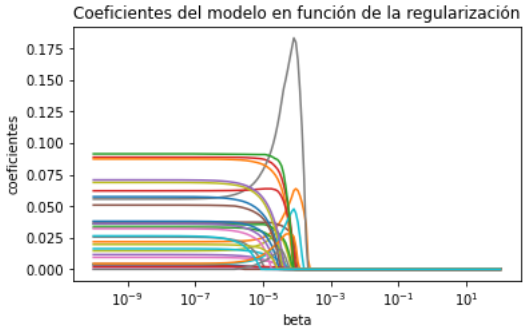


Figura 6: Evolución de los coeficientes de la regresión con el hiperparámetro β.

Como podemos ver, valores pequeños del parámetro de penalización no alteran el modelo, proporcionándonos así el mismo resultado que una regresión lineal al uso. Esto se debe a que el espacio restringido es lo suficientemente amplio como para englobar la solución original. A medida que la penalización crece, coeficientes correspondientes a ciertos activos empiezan a acercarse a 0, mientras que otros cobran mayor importancia. Un poco más allá de 10^{-4}, los valores restantes también se empiezan a acercar a 0. Esto se debe a que la penalización es excesiva y el compromiso entre ajuste y simplicidad se decanta por la simplicidad, proponiendo un modelo trivial.

También podemos ver la evolución del número de predictores incluidos (figura 7), donde se aprecia con mayor claridad el descenso que se produce.

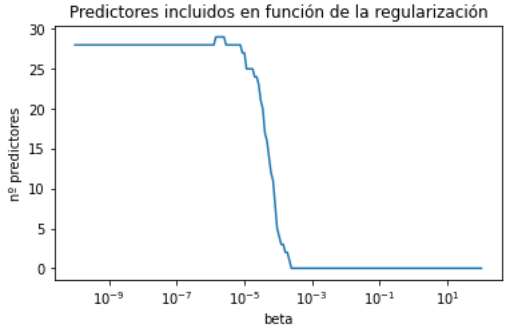


Figura 7: Evolución del número de predictores incluidos con el hiperparámetro β

A continuación, representamos el RMSE obtenido en el modelo con validación cruzada, donde se marca el error mínimo, que sería el del modelo que mejor reproduce los rendimientos del índice y el cuál incluye todos los predictores, y que se desvía una desviación estándar del primero (figura 8).

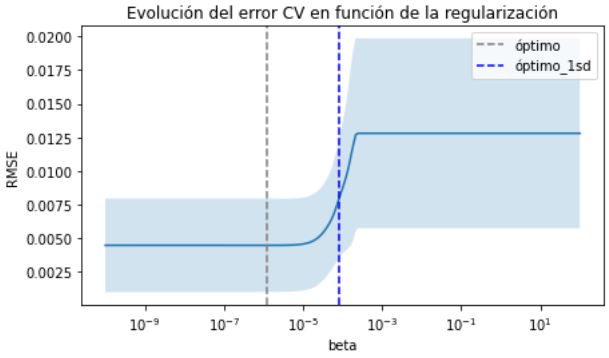


Figura 8: Evolución del RMSE con el hiperparámetro β

A partir de aquí, necesitamos elegir un valor de β con la información que disponemos. Para ello, tomamos aquel valor correspondiente al error que se aleja una desviación estándar del mínimo, dado que buscamos reproducir el índice de una forma bastante acertada, pero con un menor número de activos.

Realizamos una regresión Lasso para el hiperparámetro escogido. En la figura 9 podemos ver representado el valor de los coeficientes. Las empresas con coeficientes no nulos son: Deutsche Bank (DBK), Siemens (SIE), Heidelberg-Cement (HEI), Daimler (DAI), Bayer Aktiengesellschaft (BAYN), Infineon

Technologies (IFX) y MTU Aero Engines (MTX). De esta forma habríamos reducido el número de activos utilizados para replicar el índice de 30 a 6.

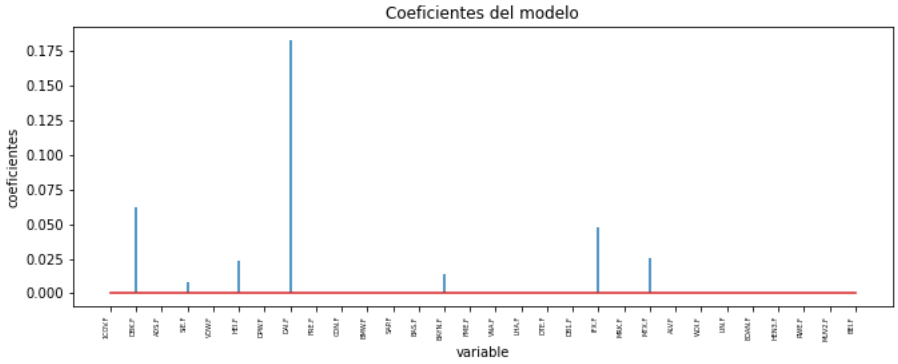


Figura 9: Coeficientes de la regresión.

Por último, representamos los rendimientos del índice obtenidos con nuestro modelo frente a los valores de los que disponíamos originalmente (figura 10). Se puede ver que los valores se encuentran dibujando una especie de linea recta, como cabría esperar.

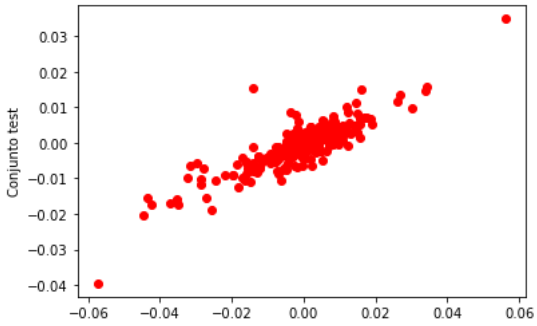


Figura 10: Valores calculados frente a los reales, LASSO

7.3. Random Forest

Se ha visto con anterioridad en que consiste el método de Random Forest y cómo nos puede indicar qué componentes son los más relevantes y así permitiéndonos seleccionarlos para nuestra cartera réplica. Para ello, volvemos

a separar la muestra en conjuntos train y test. Cada vez que hemos realizado esta división inicial, hemos utilizado una semilla para que sea idéntica en todos los modelos, evitando así las diferencias. De esta forma, si existen rendimientos excepcionalmente grandes o pequeños en el conjunto de entrenamiento, será común para todos los modelos.

Una vez hecho esto, debemos decidir los diferentes hiperparámetros del modelo. Para ello, nos hemos fijado en que el error obtenido en el conjunto test sea similar al del conjunto train, ya que esto es lo que nos ayudará a identificar si se generaliza o no de forma correcta. Los valores escogidos para ellos son:

- **Número de árboles** : 100. Otros valores relativamente cercanos también proporcionan valores similares y no tiene sentido complicar el modelo añadiendo muchos más, dado que no proporciona resultados significativamente menores.
- **Criterio** : error cuadrático medio (MSE). El MAE proporciona un resultado es similar. Se utiliza este criterio tanto para elegir que característica se utiliza para llevar a cabo la división como para encontrar el punto donde tendrá lugar.
- **Profundidad máxima de cada árbol** : 50. Si aumentábamos este número de forma notable el modelo empezaba a sobreajustar notoriamente, dado que proporcionaba un error significativamente menor en el conjunto train que en el test.
- **Tamaño de la submuestra utilizada para entrenar cada árbol aplicando bagging** : 250. Si dejábamos este parámetro por defecto utilizaba unas 750 observaciones para realizar cada entrenamiento. Debido a que disponemos de un total de poco menos de 1400 observaciones, esto llevaba a que muchos de los submodelos vieran datos comunes, dando lugar a un sobreajuste.
- **Para el resto de parámetros se han utilizado los valores predeterminados, ya que cambiarlos no proporcionaba un ajuste significativamente mejor**. Estos permiten limitar el número de hojas, el número de observaciones para realizar una bifurcación en el árbol, evitar el bootstrap para así entrenar todos los árboles con el total del conjunto de entrenamiento, etc.

Entrenamos el modelo y representamos los rendimientos del índice obtenidos con nuestro modelo frente a los valores de los que disponíamos originalmente (figura 11). Se puede ver, nuevamente, que los valores se encuentran dibujando una especie de linea recta.

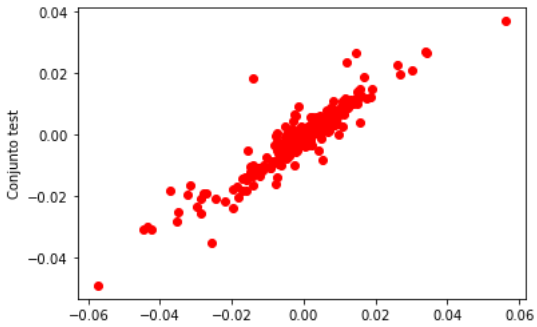


Figura 11: Valores calculados frente a los reales, Random Forest

A continuación obtenemos la importancia de los diferentes componentes (figura 12), siendo las 7 (tomamos el mismo número de activos que en el apartado anterior) empresas más relevantes: Allianz (ALV), Siemens Aktiengesellschaft (SIE), BASF (BAS), Daimler (DAI), Deutsche Post (DPW), SAP y BMW.

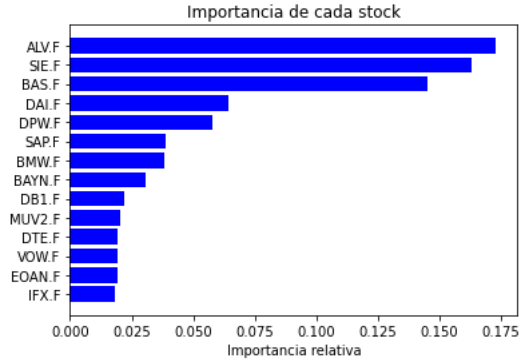


Figura 12: Componentes del DAX por importancia, Random Forest

Por último llevamos a cabo una regresión lineal, sin constante, en la que

buscamos ajustar el índice a partir de los componentes seleccionados. Podemos apreciar que ajusta correctamente en la figura 13.

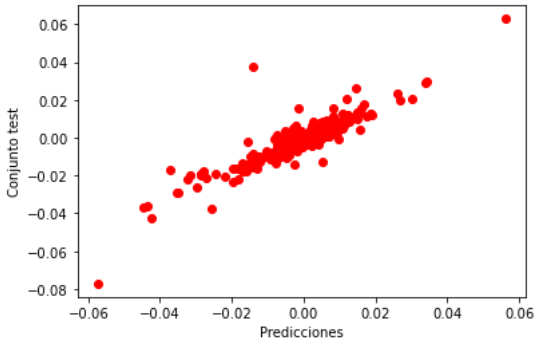


Figura 13: Valores calculados frente a los reales del modelo seleccionado mediante Random Forest

7.4. Autoencoder

Debido a la flexibilidad que presentan las redes neuronales a la hora de construirse, veremos algunas de las posibles estructuras que se pueden utilizar para el autoencoder y adjuntaremos los resultados de una ejecución del código para cada una de ellas. Para construir todas ellas hemos utilizado las funciones de la librería Keras.

7.4.1. Estructura con una capa intermedia

La primera estructura está formada por un total de 3 capas. Entrada y salida del encoder y entrada y salida del decoder, donde la entrada de la segunda es la salida de la primera. Debido a que buscamos replicar los 30 componentes del DAX, para posteriormente ver cuales representan mejor la información común necesitamos 30 neuronas en la primera capa y otras 30 en la última. En la capa intermedia encontramos 15 neuronas. Recordemos que la capa central necesita un menor número de neuronas para que así, el encoder proporcione información con una dimensionalidad menor.

Conociendo la distribución de las neuronas, lo siguiente es la función de activación. En esta hemos probado las más habituales, las cuales encontramos en Keras, y el resultado proporcionado por la función ReLu (ReLu(X)

= max(0,X)) es el mejor (consideramos mejor resultado aquel que da lugar a un menor RMSE en el conjunto test cuando comparamos las series originales con las réplicas proporcionadas por el decoder) de todos, ya que parece que es la que incorpora mejor el componente no lineal en valores alrededor del 0, donde encontramos los rendimientos. Esto coincide con la bibliografía anterior, donde parece ser la opción predilecta de los investigadores. Solo aplicamos la función de activación en la salida del encoder, dejando la del decoder únicamente con una función lineal. Esto se debe a que si lo hacemos también en la capa de salida, está nunca podrá proporcionarnos rendimientos negativos, impidiendo así una buena réplica.

Con la estructura clara, procedemos a entrenar la red. Para ello utilizamos el optimizador Adadelta y utilizaremos como función de coste el RMSE entre las 30 series originales y las 30 reconstruidas a partir de los factores de la capa intermedia. Tanto el MAE como el RMSE proporcionan resultados similares. Además de esto, debemos tener en cuenta dos hiperparámetros importantes: batch size, que es el número de ejemplos que se le pasa al algoritmo en cada iteración durante el aprendizaje, y epochs, es el número de veces que va a pasar cada ejemplo de entrenamiento por la red. Estos los hemos fijado en 10 y 500, respectivamente, ya que son los que nos proporcionan un menor valor de la función de coste al final del proceso de forma rápida.

Tras realizar el entrenamiento y comprobar que no hay motivos para pensar que hay problemas de generalización (comparando errores en train y test), vemos qué activos presentan una reconstrucción más próxima a los originales. Se ha utilizado, para los diferentes subapartados de la sección Autoencoder, como métrica la suma de las diferencias al cuadrado (sin dividir para obtener la media), pero se podría haber utilizado una métrica diferente. Los resultados obtenidos para los 7 activos más similares los encontramos en la tabla 1.

Siemens (SIE)	0.1046
Beiersdorf (BEI)	0.1061
SAP (SAP)	0.1075
Venovia (VNA)	0.1417
Deutsche Borse (DB1)	0.1462
E.ON (EOAN)	0.1546
MERCK (MRK)	0.1689

Tabla 1. 7 componentes con menor RMSE en su reproducción. Autoencoder 1.

Con ellos realizamos una regresión lineal (figura14) de las mismas características que la del apartado de Random Forest. Si representamos los valores del índice frente a los que obtendríamos mediante el ajuste con los 7 componentes, obtendremos nuevamente un resultado similar a los anteriores.

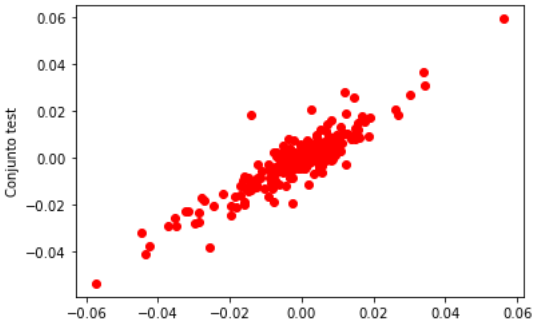


Figura 14: Valores calculados frente a los reales del modelo seleccionado mediante el primer autoencoder

7.4.2. Estructura con una capa intermedia y regularización Lasso en la capa intermedia

La estructura que presentaremos es similar a la del apartado anterior, con las mismas funciones de activación, con la diferencia de que en la capa intermedia tendremos 30 neuronas y sobre las salidas de estas aplicaremos una regularización LASSO (se añade a la función de coste el término de penalización). La finalidad de esto es evitar tener que seleccionar el número de neuronas en la capa intermedia mediante la combinación de esto y la función de activación ReLu, que proporciona con facilidad salidas nulas. De esta forma tenemos una dimensionalidad reducida en la capa intermedia que se escoge sin que debamos elegir el número de factores. El factor β escogido es 0.01, pero se obtenía un resultado similar para valores tanto un orden mayor como menor.

Tras entrenar y descartar indicios de problemas de generalización, vemos nuevamente que activos presentan una reconstrucción más similar a los originales. Los resultados obtenidos para los 7 activos más parecidos los en-

contramos en la tabla 2. En este caso, se puede apreciar que la reproducción de los componentes es más certera.

Beiersdorf (BEI)	0.02207
Deutsche Telekom (DTE)	0.02435
Vonovia (VNA)	0.02666
Henkel (HEN3)	0.02846
Deutsche Borse (DB1)	0.02848
Münchener (MUV2)	0.02866
MERCK (MRK)	0.03034

Tabla 2. 7 componentes con menor RMSE en su reproducción. Autoencoder 2.

Realizamos la regresión lineal (figura 15) y representamos los valores del índice frente a los que obtendríamos mediante el ajuste con los 7 componentes, obteniendo nuevamente un resultado donde los puntos se agrupan en torno a una linea recta.

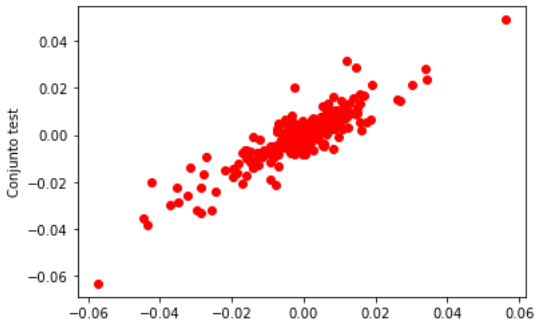


Figura 15: Valores calculados frente a los reales del modelo seleccionado mediante el segundo autoencoder

7.4.3. Autoencoder con tres capas intermedias

La estructura del autoencoder que presentamos en este apartado consta de 3 capas intermedias. El número de neuronas por capa es, en orden, 30, 10, 5, 10 y 30, nuevamente. También se han probado otras combinaciones

sin llegar a obtener resultados mejores. Cada una de las capas intermedias tienen nuevamente una función de activación ReLu.

Tras entrenar y descartar indicios de problemas de generalización, observamos que activos presentan una reconstrucción más fiel a los originales. Los resultados obtenidos para los 7 activos más próximos los encontramos en la tabla 3.

Deutsche Bank (DB1)	0.0036375
Allianz (ALV)	0.006894
Deutsche Telekom (DTE)	0.007156
BASF (BAS)	0.009028
Daimler (DAI)	0.009773
Siemens (SIE)	0.009778
SAP	0.010841

Tabla 3. 7 componentes con menor RMSE en su reproducción. Autoencoder 3.

Realizamos la regresión lineal (figura 16) y representamos los valores del índice frente a los que obtendríamos mediante el ajuste con los 7 componentes, obteniendo nuevamente un resultado similar a los anteriores.

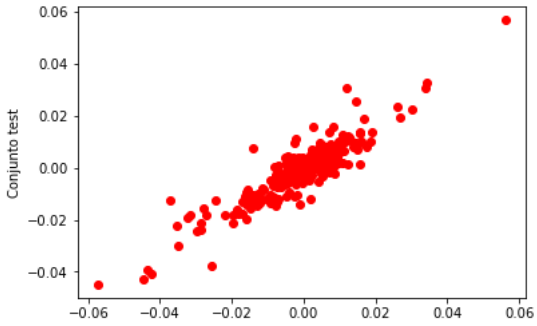


Figura 16: Valores calculados frente a los reales del modelo seleccionado mediante el tercer autoencoder

Como podemos ver, la selección es similar a la llevada a cabo por Random Forest.

7.4.4. Selección de la estructura del autoencoder.

Cada uno de los diferentes autoencoder ha sido entrenado diferentes veces. Esto se ha hecho de forma intencionada, dado que el algoritmo podía presentar inestabilidad. De hecho lo hemos encontrado en los dos primeros modelos, dado que, si bien los resultados eran similares entre las diferentes compilaciones para las carteras que mejor reproducción conseguían, su ordenación podía cambiar de forma significativa, dejando fuera de las siete seleccionadas algunas que antes habían estado incluidas. Debido a esto la variabilidad del RMSE del ajuste lineal presentado al final de los apartados correspondientes ha sido elevada. Este problema no sucede con la tercera red, dado que pocas veces se ha encontrado que se excluya alguna de las que han sido seleccionadas en la ejecución recogida en el apartado anterior. Ahora bien, existe algo en común entre los dos modelos que presentan este problema y es que tienen solo una capa intermedia. Si además tenemos en cuenta que la función de activación escogida es la ReLu, la cuál no hemos podido utilizar en la última de las capas debido a que no permite replicar rendimientos negativos, tenemos que, si bien se forman factores comunes en la capa intermedia mediante combinaciones no lineales, solo se combinan de forma lineal para llegar a los outputs. Esto no sucede en el último de los autoencoders, dado que al presentar capas adicionales, sí que se produce una transformación no lineal de los factores subyacentes para reconstruir los rendimientos de los activos. Esto nos lleva a seleccionar el autoencoder con tres capas intermedias como el mejor de los tres para nuestro objetivo, dado que proporciona resultados estable. Esto nos permite intuir que efectivamente existen relaciones no lineales entre los activos contemplados, ya que incluir transformaciones de este tipo nos proporcionan mejores resultados (también se han entrenado todas todas las redes sin funciones de activación no lineales, proporcionando peores resultados). Por otro lado, si nos fijamos en los errores de reproducción de las empresas seleccionadas, el tercer autoencoder es el que consigue una mejor reproducción, lo que refuerza la idea de que es la mejor elección.

8. Comparativa de los diferentes modelos

En el presente apartado se comparan los resultados obtenidos por los diferentes modelos para ver cuál consigue replicar el índice mejor. Para ello

utilizaremos los componentes que selecciona cada modelo con la finalidad de, mediante una regresión lineal, reproducir el DAX en precios.

Para LASSO, el resultado obtenido lo podemos ver en la figura 17. En esta podemos ver que la réplica produce valores más extremos que el DAX en varias situaciones, tanto con mayores como menores rendimientos.

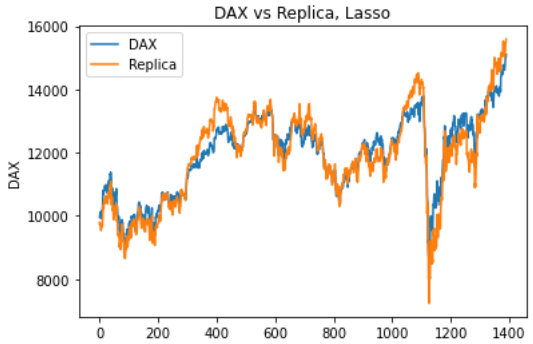


Figura 17: Reproducción del índice haciendo uso de la selección realizada con Lasso.

Para Random Forest, el resultado obtenido lo encontramos en la figura 18. En esta podemos apreciar que el ajuste resulta ser bastante certero, no dejando zonas con grandes discrepancias entre ambas series.

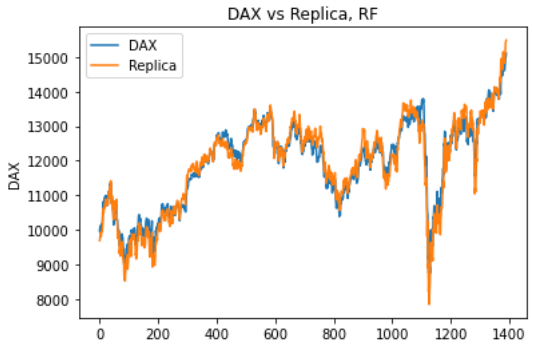


Figura 18: Reproducción del índice haciendo uso de la selección realizada con Random Forest.

En el caso del Autoencoder, donde se ha utilizado el que consta de tres

capas intermedias, podemos ver el ajuste en la figura 19. Nuevamente, encontramos una reproducción bastante fiel, igual al caso del Random Forest. Los resultados para los otros autoencoders se encuentran a mitad camino entre LASSO y Random Forest, siendo, como esperábamos, peores que la estructura seleccionada (debido a la inestabilidad de la selección no se adjunta ningún valor exacto, dado que con cada selección de carteras el resultado difiere).

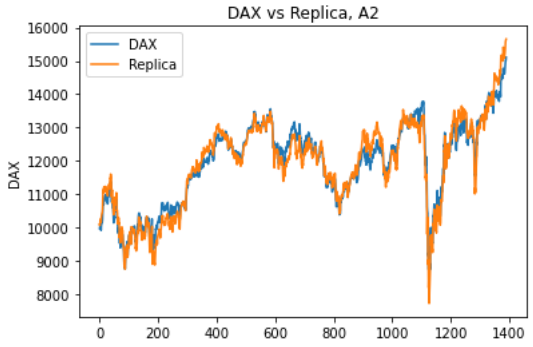


Figura 19: Reproducción del índice haciendo uso de la selección realizada con Autoencoder.

Para cada uno de los modelos, el RMSE obtenido es:

	LASSO	RForest	Autoencoder
RMSE	444.82	249.76	212.68

Tabla 4. RMSE del ajuste del nivel del DAX utilizando la selección de componentes indicada por cada uno de los métodos..

Aquí podemos ver que el modelo que proporciona un peor ajuste, con diferencia, es LASSO, dado que el error asociado es el doble del que encontramos para los otros dos. El siguiente es el Random Forest, seguido del Autoencoder, que proporciona el mejor resultado. Este resultado se repite para las diferentes compilaciones del código, llevándonos a concluir que el autoencoder, si consta de una estructura adecuada, es capaz de proporcionar los mejores resultados. Para encontrar una explicación debemos recurrir a las características de los modelos.

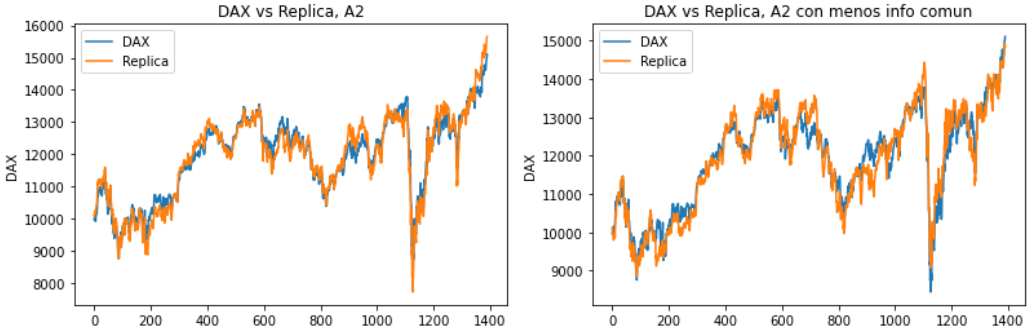
La principal diferencia de LASSO con los otros dos se encuentra en que no es capaz de capturar dependencias no lineales. De esta forma, si no puede captar parte de las relaciones existentes, la selección que llevará a cabo será notablemente peor. Esto también se puede interpretar al revés, llevándonos a que, si LASSO proporciona peores resultados, probablemente se deba a que existan relaciones no lineales entre los diferentes componentes del índice, principalmente en las zonas que peor reproduce, donde encontramos cambios bruscos de tendencia.

Tanto Random Forest como el Autoencoder son capaces de capturar dependencias no lineales, proporcionando ambos modelos resultados relativamente similares (la selección de componentes también es parecida). Sin embargo, el Autoencoder proporciona sistemáticamente resultados mejores. Debemos tener en cuenta que el Bosque Aleatorio no deja de ser un modelo bastante simple que no hace más que asignar valores constantes a diferentes regiones que encontramos en el espacio constituido por los valores de los activos entre los que se lleva a cabo la selección, por tanto, es de esperar que este presente limitaciones.

Con esto, para el caso del DAX, llegamos a que el Autoencoder nos ayuda a llevar a cabo una mejor selección de los componentes más relevantes del índice utilizando teoría económica, dado la red solo reproduce los valores originales y el criterio introducido para elegir las empresas tiene una base financiera.

9. Comparativa entre los activos con más información común y los que menos para el Autoencoder

Anteriormente, al tratar los autoencoders, se ha comentado que los componentes del índice con más información común eran la mejor opción para reproducir el índice, ya que captaban mejor la información de los factores subyacentes. Para ilustrar esto se procede a replicar el índice, tanto con las 7 carteras que mejor se reproducen como para las 7 que peor lo hacen. Podemos encontrar este resultado en la figura 20, donde se aprecia que el primer caso reproduce mejor el DAX que el segundo. Los RMSE son, respectivamente, 212.68 y 326.88.



(a) Más información común. (b) Menos información común.

Figura 20: Réplica del DAX utilizando la información proporcionada por el autoencoder.

Resulta destacable que el RMSE del segundo ajuste proporcione una réplica más fiel que LASSO. Es posible que las carteras que menos información común comparten, al presentar una débil dependencia respecto a los factores del mercado, no tengan una ordenación tan clara. Si dos series se reconstruyen a partir de series que no reflejan bien su evolución, que una réplica sea mejor que la otra no tiene por qué aportar información alguna.

10. Posibles lineas de investigación futuras.

En el presente documento hemos visto como una red neuronal nos puede ayudar a seleccionar seleccionar que componentes del DAX utilizar para replicarlo, utilizando unos pocos de ellos. Ahora bien, ¿que pasos se podrían tomar a continuación? Vemos algunos de ellos:

- Aplicar el método en otros mercados, con otros índices. Si estos presentan relaciones no lineales entre los activos entre los que se realiza la selección, el método debería ser útil.
- Ampliar la lista de activos entre los que llevar a cabo la selección. Debido a que la composición del índice cambia con el tiempo, se ha elegido la composición en una fecha concreta, pero se podría aplicar no solo a las empresas que se contemplan en un momento para elaborar el DAX, sino aquellas que se han utilizado con anterioridad o que se piensen que son relevantes para el mercado.

- El método se ha aplicado utilizando una muestra extensa en el tiempo, pero podemos querer ajustar la cartera con cierta frecuencia, además de que las relaciones entre los activos pueden cambiar con el tiempo. Para ello se tomarían los precios de las acciones los días anteriores (con diferentes observaciones por día para disponer de una mayor muestra en caso de ser necesario) y se aplicaría el método a los componentes del índice (esto debería ser cuando se produzcan cambios importantes en el mercado, como, por ejemplo, cambios en la composición del índice o variaciones significativas en los pesos). Al disponer de pocos activos, cabe esperar que la reestructuración de la cartera conlleve un coste menor que aquella con todos los activos.

11. Conclusiones.

En el presente documento, hemos utilizado tres metodologías diferentes que repliquen el comportamiento de un índice bursátil. Para el caso concreto del DAX y para el periodo del estudio, el autoencoder con 3 capas intermedias se ha revelado como el más fiable, seguido Random Forest. Tras este, los siguientes eran los autoencoders con una sola capa intermedia, finalizando con LASSO. Esto nos ha llevado a pensar que en el DAX existe un número notable de relaciones no lineales entre los diferentes componentes, ya que los mejores resultados los proporcionan dos métodos que capturan no linealidades con facilidad, mientras que los autoencoders de 3 capas totales no reconstruyen los rendimientos mediante transformaciones no lineales y LASSO no es capaz de tenerlas en cuenta en su selección. En caso de que otros índices bursátiles presenten también relaciones no lineales entre sus componentes, Random Forest, pero sobretodo un autoencoder con varias capas intermedias pueden ser metodologías útiles, así que podría ser conveniente aplicarlas en caso de que sospechemos que puede haberlas.

Si nos remitimos a los componentes seleccionados por cada uno de los métodos, se observa que los dos métodos que mejor funcionan proporcionan resultados similares mientras que los resultados de los otros varían notablemente entre sí o al volver a correr el código (en el caso de los autoencoder). Esto probablemente se deba a que existe una alta correlación (lineal) entre algunos de los activos que componen el índice, dando lugar a que pequeños cambios en el criterio lleven a una selección notablemente diferente, si la metodología no capta correctamente no linealidades. Esto podría confirmarse

con un pequeño estudio de simulación.

También hemos comprobado que una réplica realizada con los activos que mayor información comparten, basándonos en el autoencoder, proporcionan una mejor réplica que aquellas que menos comparten. Es importante llevar a cabo esta comprobación, dado que esta es la base de la selección. Aquellos componentes que son reproducidos con mayor fidelidad a partir de los factores que encontramos en la capa intermedia, los cuales reflejan la información del mercado, son los que mejor reproducen el índice.

Referencias

[1] Charu C. Aggarwal *Neural Networks and Deep Learning*. IBM T. J. Watson Research Center International Business Machines Yorktown Heights, NY, USA

[2] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani *An Introduction to Statistical Learning*. Springer Texts in Statistics

[3] Trevor Hastie, Robert Tibshirani, Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics, Stanford, California. August 2008

[4] Yves Hilpisch. *Artificial Intelligence in Finance*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopo. October 2020

[5] Yves Hilpisch. *Derivatives Analytics with Python*. John Wiley Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom. 2015

[6] Stefan Jansen. *Hands-On Machine Learning for Algorithmic Trading*. 2018 Packt Publishing.

[7] Hariom Tatsat, Sahil Puri and Brad Lookabaugh. *Machine Learning and Data Science Blueprints for Finance*. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. October 2020.

[8] Jannes Klaas. *Machine Learning for Finance*. 2019 Packt Publishing.

[9] Marcos M. López de Prado. *Machine Learning For Asset Managers*. University Printing House, Cambridge CB2 8BS, United Kingdom. 2020.

[10] Matthew F Dixon, Igor Halperin, Paul Bilokon. *Machine Learning in Finance From Theory to Practice*. Springer Nature Switzerland AG 2020.

[11] Mario Peña, José Orellana. *Red neuronal para clasificación de riesgo en cooperativas de ahorro y crédito*. Grupo de investigación de Ingeniería Industrial/Ciencias Químicas/Ingeniería Industrial, Universidad de Cuenca Ciencias Químicas/Ingeniería Industrial, Universidad de Cuenca

[12] Fredy Ocaris Pérez Ramírez, Horacio Fernández Castaño. *Las redes neuronales y la evaluación del riesgo de crédito*. Revist a Ingenierías Univer sidad de Medellín, 27/04/2007

[13] M. López, S. Valero, C. Senabre, J. Aparicio, A. Gabaldon. *Application of SOM neural networks to short-term load forecasting: The Spanish electricity market case study*. Electric Power Systems Research 91 (2012) 18–27, 23 May 2012

[14] Alberto Fernández, Sergio Gómez. *Portfolio selection using neural networks*. Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, Campus Sescelades, Avinguda dels Països Catalans 26, E-43007 Tarragona, Spain. 15 August 2005

[15] Chia-Cheng Chen, Chun-Hung Chen, Ting-Yin Liu. *Investment Performance of Machine Learning: Analysis of SP 500 Index*. International Journal of Economics and Financial Issues, 2020, 10(1), 59-66.

[16] Reza Bradrania, Davood Pirayesh Neghab, Mojtaba Shafizadeh. *State-dependent stock selection in index tracking: a machine learning approach*. Swiss Society for Financial Market Research 2021.

[17] Hongbing Ouyang, Xiaowei Zhang, Hongju Yan. *Index tracking based on deep neural network*. School of Economics, Huazhong University of Science and Technology, China. 3 October 2018.

[18] Chi Zhang, Shuang Liang, Fei Lyu and Libing Fang. *Stock-Index Tracking Optimization Using Auto-Encoders*. School of Management and Engineering, Nanjing University, Nanjing, China. 17 September 2020.

[19] Samuel Fernandez-Lorenzo, Diego Porras, Juan José García-Ripoll. *Hybrid quantum-classical optimization for financial index tracking*. BBVA

Client Solutions Research Patents, Calle Sauceda 28, 28050 Madrid, Spain. 27 Aug 2020.

[20] Yu Zheng, Bowei Chen, Timothy M. Hospedales, Yongxin Yang. *Index Tracking with Cardinality Constraints: A Stochastic Neural Networks Approach*. Southwestern University of Finance and Economics, ArrayStream Technologies Limited, University of Glasgow, University of Edinburgh. 14 Nov 2019.

[21] J. B. Heaton, N. G. Polson, and J. H.Witte. *Deep learning for finance: deep portfolios*. Applied Stochastic Model in Business and Industry. 7 October 2016.

[22] Ruben Pauwels, Evaggelia Tsiligianni, Nikos Deligiannis. *HCGM-NET: A Deep Unfolding Network For Financial Index Tracking*. ETRO Department, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium, Department of Computer Science and Engineering, University of Ioannina, Greece yimec, Kapeldreef 75, B-3001 Leuven, Belgium. 2021.

[23] Rubén Ruiz-Torrubiano, Alberto Suárez. *A hybrid optimization approach to index tracking*. Springer Science+Business Media, LLC 2008.

[24] Rafael Moral-Escudero, Rubén Ruiz-Torrubiano, and Alberto Suárez, Member IEEE. *Selection of Optimal Investment Portfolios with Cardinality Constraints*. 2006 IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006

[25] Chi-Ming Lin, Jih-Jeng Huang, Mitsuo Gen, Gwo-Hshiung Tzeng. *Recurrent neural network for dynamic portfolio selection*. Applied Mathematics and Computation 175 (2006) 1139–1146

[26] N. C. P. EDIRISINGHE. *Index-tracking optimal portfolio selection*. University of Tennessee, USA. 18 April 2013.

[27] Carla Moreno Navarro. *Composición De Cartera Réplica Para La Predicción Del Índice Bursátil Español IBEX 35*. Facultad de Administración y Dirección de Empresas, Universidad Politécnica de Valencia. Septiembre, 2012.

[28] *XETRA*. https://www.xetra.com/xetra-en/

[29] *Deutsche Börse Group.* https://deutsche-boerse.com/dbg-en/media/deutsche-boerse-spotlights/spotlight/DAX-benchmark-and-barometer-for-the-German-economy-139948