Los test de independencia de dos variables y su aplicación a Redes Neuronales para la predicción de mercados financieros utilizando series temporales

Luis Puyol Lombos, Carlos Maté Jiménez

Departamento de Organización Industrial, Escuela de Ingeniería ICAI, Universidad Pontificia Comillas, Madrid, España

***Abstract*- El uso de redes neuronales puede y de hecho ha traído múltiples beneficios al sector financiero, convirtiéndolo en un área con un gran potencial de investigación. Las pruebas de independencia son esenciales para evaluar si dos o más variables deben usarse simultáneamente en una red neuronal.**

A lo largo del proyecto, se programarán algunas pruebas de independencia y se analizarán múltiples arquitecturas de modelos para encontrar la arquitectura más adecuada para el problema en discusión. Además, también se espera estudiar la creación de nuevas arquitecturas a partir de las ideas de otros modelos.

El proyecto desarrollado es una adaptación del referenciado en [13].

***Index Terms*— Artificial Neural Networks, Long Short -Term Memory (LSTM), Interval Multi-Layer Perceptron (IMLP),**

I. INTRODUCCIÓN

La nueva tecnología de redes neuronales proporcionada por el aprendizaje automático representa un avance importante que ha traído un cambio de paradigma innovador en los últimos años. Una de las aplicaciones más directas es el mercado financiero, especialmente la predicción del valor futuro de acciones y otros productos financieros.

La ventaja de aplicar tecnologías de redes neuronales es hacer objetivos los criterios por los cuales se toma una decisión.

Antes de que las tareas pudieran automatizarse utilizando estas metodologías, todo dependía de la confianza en una o un equipo de personas para tomar según su criterio personal, adquirido por la experiencia o formación.

Otra razón por la que estas tecnologías son cada vez más fáciles de desarrollar es que no dependen de la habilidad para desarrollar un algoritmo lógico para un problema, sino que están fuertemente vinculadas a los datos y la digitalización de muchas áreas, así como el surgimiento de empresas dedicadas a su generación, allí hay un fuerte volumen de información generada y registrada.

Esto puede llevar a muchas oportunidades nuevas para usar los datos de manera apropiada y generar utilidad

El sector financiero no está muy atrás, y empresas como Bloomberg ya han sido vistas desarrollando los datos más recientes para atender sus aplicaciones.

II. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Para el desarrollo del proyecto, la aplicación de metodologías de Machine Learning será necesaria.

Por esta razón, uno de los primeros pasos fue realizar un análisis exploratorio de los diferentes paquetes y características dentro de lenguajes de programación como Python que resultaron útiles para los propósitos del proyecto.

Lo que se ha investigado son modelos de redes neuronales como el perceptrón multicapa, que son fácilmente implementables y económicos en términos de coste computacional.

Python

Dentro de de Python, existe un amplio conjunto de funcionalidades para el uso de redes neuronales dentro de Tensorflow, de Google.

Dentro de este paquete se encuentra una función que se puede utilizar para generar redes MLP.

Para ello, se deberán adaptar los datos y realizar ajustes en su configuración.

III. ESTADO DE LA CUESTIÓN

Para poder realizar la predicción correctamente se han de seguir una serie de pasos.

En primer lugar , recopilar los datos. Sin importar qué tan bueno sea un modelo lógico , pierde su utilidad si no se aplican los datos apropiados .

Debido a la facilidad y sencillez a la hora de utilizar las tecnologías de modelado se está dando cada vez más importancia a este primer paso, antes el talento para diseñar un

algoritmo específico del problema era lo realmente complejo.

El siguiente paso es limpiar los datos , para así eliminar los valores nulos que pueden haber aparecido debido a alguna circunstancia.

En algunos casos también es necesario preparar los datos antes de su entrada en el modelo para generar estadísticas que se sospecha que son de interés particular para el modelo en cuestión.

El último paso antes del entrenamiento es escalar los datos.

La razón es que los modelos están listos para trabajar con datos normalizados y son más eficientes. Este proceso no significa pérdida de información, por lo que no hay inconvenientes, es un procedimiento común.

Después de los datos están en el formato correcto sin ningún NA, se entrena el modelo.

Para ello se seleccionan 3 grupos en los que se subdivide el conjunto de datos.

El conjunto de entrenamiento se usa para probar variantes del modelo de modo que la salida que se obtenga para una entrada se compare con la salida de deseada.

A partir de ahí, se construye una función de error que intentaremos minimizar. Si el error es alto, el modelo se ajustará para que en las salidas, el error sea lo más pequeño posible.

Cuando se completa el paso de entrenamiento, los conjuntos de validación y test entran en juego.

El error del conjunto de prueba se verifica y se valida con el correspondiente a validación. Es de esperar que el error en entrenamiento sea menor, eso es prácticamente inevitable, pero no mucho menor.

Si la fase de validación y test muestra signos de sobreentrenamiento o los resultados generales muestran un error muy alto indicando que el modelo es de buena calidad, será recomendable repensar el paso de entrenamiento , disminuyendo o disminuyendo la complejidad del modelo según proceda.

Con el modelo ya entrenado y validado correctamente está listo para realizar las predicciones que sean necesarias.

En [1], se mostraron los resultados en múltiples metodologías de red neuronal, como pueden ser el perceptrón multicapa (MLP), las redes neuronales estocásticas (SNN) y las redes larga memoria a corto plazo (LSTM).

Las redes neuronales del tipo LSTM se basan en una red neuronal recurrente (RNN). En la figura siguiente se ilustra la estructura de la red a alto nivel.

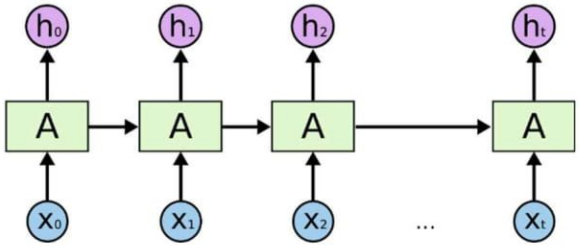

Ilustración 1: Estructura LSTM, tomado de [11]

Este tipo de modelos han mostrado unos muy buenos resultados para la información que esté compuesta por series temporales. Por esta razón parece uno de los candidatos más sólidos para la predicción de los mercados de acciones u otros instrumentos financieros, que se pueden modelar como un modelo de predicción de una serie temporal con facilidad.

Las redes neuronales estocásticas funcionan como una red neuronal convencional, pero con una importante novedad, la introducción de aleatoriedad en las variaciones del modelo durante el proceso de entrenamiento, que contribuye a evitar los mínimos locales.

Estos modelos se han aplicado a la predicción del mercado de valores con unos resultados relativamente buenos en comparación al resto de mercados, se estima un porcentaje de precisión de un 85.37% en predicciones de tendencia, muy alto, de hecho, el segundo más alto de los estudios incluidos dentro del artículo [1]. A continuación, se muestra la gráfica de la predicción en comparación con el valor actual del índice Nikkei.

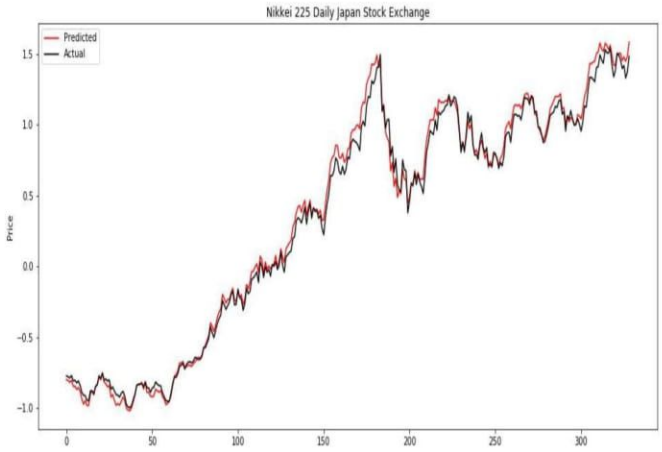

Ilustración 2: Predicción de SNN en comparación con el valor real del Nikkei, tomado de [1]

Una de las ideas que surgieron entorno a estas dos últimas redes fue la de fusionarlas en un único modelo y ver si resultaba más preciso en el artículo [1]. Para ello se creó un modelo compuesto por dos capas ocultas, una SNN y otra LSTM.

El objetivo de la fusión de los modelos fue la de aprovechar las cualidades de cada metodología de modelaje. La memoria de las LSTM, idónea para series temporales como el mercado de valores y la aleatoriedad de las SNN, que ayuda a que se converja hacia un modelo que sea tan preciso como sea posible, alcanzando el mínimo error global en lugar de uno de un

entorno local.

El resultado fue una red con la siguiente estructura:

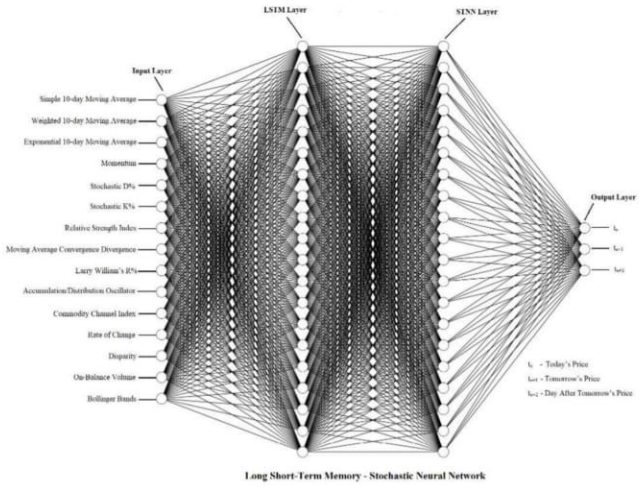

Ilustración 3: Diagrama red LSTM+SNN, tomado de [1]

Cuando se construyó el modelo y se entrenó con datos del Nikkei se llegó a una predicción ligeramente mejor que la del modelo puramente estocástico, de un 86.28% de precisión en la predicción de la tendencia.Una representación más visual se puede ver en la figura siguiente, donde se muestran las predicciones y los valores a predecir del modelo desarrollado.

Ilustración 4: Predicción modelo LSTM+SNN vs datos objetivo, tomado de [1]

El perceptrón multicapa (MLP) consiste en un conjunto de neuronas o perceptrones con estructura de red dentro de los cuales existe una función de activación de forma que produzca una señal no nula ante ciertos estímulos adecuados. Está considerada como una estructura base a partir de la cual se construyen otras variaciones de redes neuronales que puedan adaptarse mejor a cierto tipo de problemas.

Como con el resto de los modelos, también se ha utilizado MLP para la predicción de los mercados. Uno de los ejemplos se da en el artículo [2]. Los resultados en la predicción fueron los que se muestran en las figuras siguientes. En la primera figura se muestran las predicciones en comparación con los valores reales mientras que la segunda figura es una representación del error relativo en porcentaje de las predicciones del modelo en función de cada día.

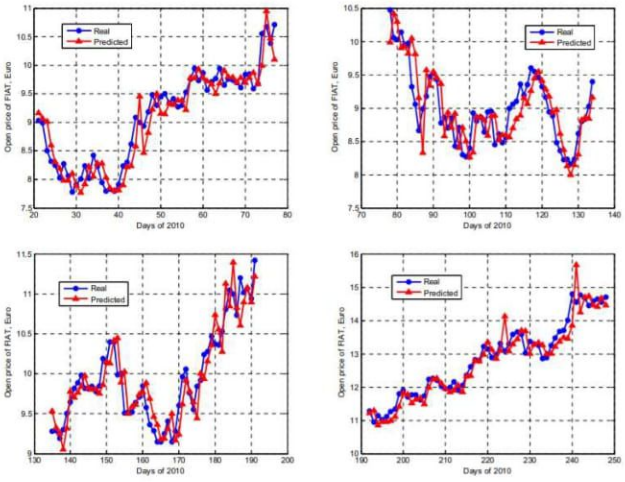

Ilustración 51: Valor real vs Predicción usando MLP, tomado de [2]

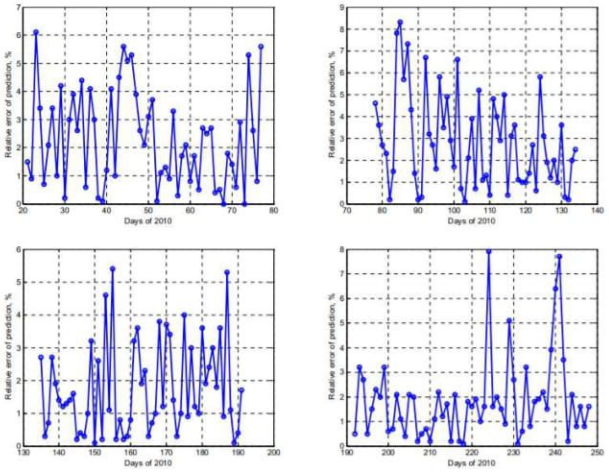

Ilustración 62: Error relativo a las predicciones en MLP, tomado de [2]

Utilizando un enfoque menos visual para valorar el modelo se llega a la conclusión de que un 91% de los valores que se habían predicho presentaban un error relativo menor o igual que el 5% y, si profundizamos un poco más, se observa que el 33% de las predicciones tenían un error de, a lo sumo, un 1%.

Otro de los casos estudiados es el referenciado en [3], donde se estudia el iMLP, que está asociado a los datos de intervalo.

Cabe destacar que los datos del tipo intervalo se dan como un segmento de la recta real, para ello se pueden dar de varias formas, como los extremos del intervalo o su centro y su radio.

Si se calcula el error en el cálculo del centro se podría asociar al error de sesgo que presenta el modelo, mientras el error de radio podría estar relacionado con la precisión y variabilidad de las predicciones. El paso de unas coordenadas a otras es sencillo, usando operaciones aritméticas. Los valores singulares son, por el contrario, un único valor, lo que hace el modelo más sencillo. Un ejemplo para el que usar intervalos podría ser de utilidad en un modelo sería en el caso de que las variables de entrada fuesen mediciones de un sensor que presentase un error de medida.

Una vez se ha hecho la distinción entre esos dos tipos de variables se pueden ver las distintas variaciones de los modelos si alteramos el tipo de variables que se utilizan para las

variables de entrada, que se denominan input, las de salida, llamadas output, y los pesos y sesgos que se utilizan en las capas de dentro del modelo. Si aplicamos esas tres dimensiones la figura resultante es un cubo de 2x2x2, por lo que se construyeron dos tablas de 2x2 para ilustrar las distintas variaciones.

Input | Output	Interval data	Single value data
Interval data	New iMLP	No data
Single value data	Ishibuchi	Classical MLP

Tabla 1: Pesos y sesgos con valores singulares. Elaboración propia

Input | Output	Interval data	Single value data
Interval data	Simoff, Beheshti	Patiño-Escarcina
Single value data	Ishibuchi	Drago and Ridella

Tabla 2: Pesos y sesgos con valores de intervalo. Elaboración propia

En cada celda se muestran los autores que han desarrollado un modelo de esas características y que se han recopilado. La metodología que se propone en este artículo es la correspondiente a la celda New iMLP, que utiliza datos de intervalo tanto a la entrada como a la salida y valores singulares en los pesos y sesgos.

Para desarrollar el modelo se utilizó un MLP con una capa oculta y una única variable de salida.

La estructura de la red sería la siguiente.

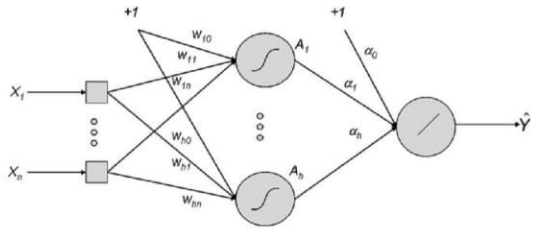

Ilustración 7: Estructura del modelo iMLP, tomado de [3]

Donde hay n variables de entrada y h neuronas ocultas.

Si se calcula el valor de la entrada de cada neurona de la capa oculta (S_j) se obtiene que:

$$S_j = \langle w_{j0} + \sum_{1=1}^{n} w_{ji} x_i^C \,|\, \sum_{i=1}^{n} |w_{ji}|\, x_i^R \rangle$$

La función que se tomó como función de activación fue la tangente hiperbólica, por lo que, tras desarrollar la fórmula se obtuvo el siguiente resultado:

$$A_j = \left\langle \frac{\tanh(S_j^C - S_j^R) + \tanh(S_j^C + S_j^R)}{2} \,\middle|\, \frac{\tanh(S_j^C + S_j^R) - \tanh(S_j^C - S_j^R)}{2} \right\rangle$$

Donde A_j es la salida de la neurona oculta j.

Finalmente, si se agrupan los resultados de las salidas de la capa oculta en la capa de salida se obtiene la salida $\hat{Y}$, la cual se rige según la expresión que se muestra a continuación:

$$S_j = \langle \alpha_0 + \sum_{i=1}^{n} \alpha_i a_i^C \,|\, \sum_{i=1}^{n} |\alpha_i|\, a_i^R \rangle$$

Una de las posibles aplicaciones para este modelo sería la de aproximar una aplicación cuyo conjunto de inicio fuera el n-ésimo producto cartesiano de un conjunto cuyos elementos fueran intervalos y el conjunto imagen con el que se establece la relación tuviese también elementos en intervalo.

En este escenario es necesario definir la función de error con la que se entrenará el modelo.

$$E = \frac{1}{2} \sum_{t=1}^{p} d(Y(t), \hat{Y}(t)) + \lambda \phi(\hat{f})$$

Donde $\lambda\phi(\hat{f})$ es el término regulador de la función estimadora, p es el número de muestras y d es la función para medir la distancia entre las estimaciones y los valores objetivo. En este trabajo se utilizó la distancia euclídea ponderada según el parámetro β entre dos intervalos. Cuanto mayor sea el valor de β, mayor será la importancia de obtener precisión para hallar el centro y menor la del radio.

$$d(A, B) = \beta(a^C - b^C)^2 + (1 - \beta)(a^R - b^R)^2$$

Con todo esto se procedió a aplicar el modelo a un conjunto de datos relacionados con la energía. El objetivo fue predecir el precio de la electricidad a partir de la información de múltiples factores: generación de energía por carbón, petróleo, centrales nucleares, etc. Todos estos factores fueron adaptados al formato de intervalo pasando valores como el máximo y el mínimo diarios o el valor al principio del día y al final del mismo.

El siguiente paso fue entrenar el modelo con los datos y comprobar que todo había funcionado correctamente por medio de la comprobación de la similitud de la precisión en los conjuntos de entrenamiento y validación. Una vez se comprobó esto se compararon los resultados en la precisión con los de un modelo clásico que no utilizaba la metodología sobre la que se había desarrollado el modelo iMLP. Los resultados fueron sorprendentemente positivos y se pueden ver en la siguiente tabla.

	Training set MAPE		Validation set MAPE	
	Centre (%)	Radius (%)	Centre (%)	Radius (%)
Naive model	15.85	33.82	17.86	33.44
iMLP	8.86	25.83	11.38	24.54

Tabla 3: Precisión del modelo iMLP frente al simple. Tomada de [3]

A. Justificación

Posibles formas de ampliar los trabajos anteriores

Sobre las distintas variaciones de los modelos con datos en intervalo la única que no ha sido estudiada es la que tiene entradas en forma de intervalo y salidas, pesos y sesgos como valores singulares. Este podría ser una rama que valga la pena explorar.

El primer paso sería ver si es posible y tiene sentido aplicarlo al problema de la predicción de mercados. El carácter de los datos de los pesos no resultaría en principio ningún problema al estar asociado a la lógica interna del modelo. Las variables de entrada podrían adaptarse fácilmente al formato de

intervalo y, de hecho, tendría sentido puesto que cada registro en los datos financieros corresponde generalmente a un intervalo (gráficos en vela). Los datos de salida, por el contrario, serían singulares, por lo que la salida podría perfectamente adaptarse. Sin embargo, es probable que el modelo se beneficie más de una salida en forma de intervalo que aporte algún tipo de noción puntual del error del modelo aparte de estadísticos como el MSE.

Si se decidiese continuar por esta vía se podrían ver los distintos modelos de las referencias del artículo y ver cómo realizan las transformaciones dentro del modelo para pasar de variables de intervalo a variables singulares.

Otra posible ampliación del iMLP podría ser la de incrementar las salidas del modelo para conseguir información del propio intervalo. Se podría decir que la predicción del modelo es un intervalo de confianza, pero se desconoce el porcentaje de confianza. Este podría ser la otra variable de salida o, alternativamente, una de entrada. Para poder si quiera plantear si es posible, sería necesario poder calcular el intervalo de confianza de forma empírica, que sería probablemente muy complejo si es que es posible.

También se podría estudiar la introducción de otro tipo de variables de entrada que no sean de intervalo. La primera solución es la de convertirlas a un intervalo con radio 0, pero tal vez podría causar problemas.

Finalmente se podría analizar la posibilidad de combinar las dos primeras ampliaciones y hacer así un modelo que calcule un intervalo de confianza y su porcentaje de confianza como variable singular.

IV. Objetivos

Crear un modelo basado en iMLP adaptado a Python para facilitar trabajos futuros y para que sea más sencilla su aplicación y modificación.

También se buscará establecer una métrica de calidad de un modelo para poder así ir variando los parámetros base de la red y encontrar la mejor configuración para hacer el modelo más preciso.

Crear otros múltiples modelos a partir de distintas metodologías a la hora de construir una red neuronal. Es de esperar que los modelos fuesen desde el más básico hasta otros más complejos como LSTM.

Finalmente se buscará comparar los resultados de todos los modelos y extraer conclusiones sobre cuáles tienen las cualidades más deseables en lo referente al error cuadrático medio y otros parámetros de precisión.

V. METODOLOGÍA

Para cumplir el primer objetivo se estudiarán trabajos anteriores de adaptación del código iMLP a lenguajes como Matlab y se usarán como base para ver qué errores evitar y qué pasos seguir.

Con respecto a los modelos adicionales a crear, se elaborará una lista y se irá uno por uno buscando cuáles son los parámetros idóneos para obtener la mayor precisión posible. Cabe destacar que se utilizará la metodología de ventanas de tiempo sobre los modelos. La aplicación de ventanas temporales al trabajo con redes neuronales ha mostrado resultados prometedores, como se muestra en [4].

VI. IMPLEMENTACIÓN DEL IMLP

El primer paso para llevar a cabo este objetivo es desarrollar un código básico a través del cual se pueda ejecutar el script en C del iMLP. Es importante remarcar que el código sirve como interfaz entre Python y el iMLP programado en C, por lo que ninguna de las partes lógicas de la red está programada en Python. Esto mejora la velocidad de procesamiento a la vez que permite el uso de todas las facilidades de Python con respecto a C.

La forma en que interactúa el programa en C con el de Python es a través de varios ficheros. Estos son el train.txt, test.txt y val.txt para leer los datos y el zcub.net para definir el modelo junto a un archivo de pesos del iMLP.

En cada archivo, cada elemento del conjunto de datos se representa en una fila y las columnas representan las variables, primero las de entrada y, a continuación, las de salida (todas ellas separadas mediante tabulaciones). Cada una de las variables se representa mediante dos columnas, una para el centro y otra para el radio. En el siguiente ejemplo se representan p elementos descritos por m variables de entrada y n variables de salida:

```
in_1c_1 in_1r_1 … in_mc_1 in_mr_1 out_1c_1 out_1r_1 … out_nc_1 out_nr_1
in_1c_2 in_1r_2 … in_mc_2 in_mr_2 out_1c_2 out_1r_2 … out_nc_2 out_nr_2
…  ……  …  …  …  …  …  …
   ……  …
in_1c_p in_1r_p … in_mc_p in_mr_p out_1c_p out_1r_p … out_nc_p out_nr_p
```

A continuación, se muestra un ejemplo de un código que parte de un dataframe con valores de máximo y mínimo ya normalizados de múltiples días anteriores y posteriores y lo adapta al formato exigido por el programa.

```
# Adaptación al formato de intervalo

dfIMLP = pd.DataFrame()
dfIMLP["Dates"] = df["Date"]
# Variables de entrada
```

Para calcular el centro en cada uno de los intervalos (Low, High), se toma el valor medio de los extremos. Una vez se ha obtenido el centro se puede obtener el radio de forma sencilla tomando la diferencia entre el máximo (High) y el centro. Esto se hace sobre los puntos pasados, que constituirán las entradas del modelo.

```
for t in range(delt_t+1):
        dfIMLP['centro_t='+str(t)] = 0.5*(df["High"].shift(t) + df["Low"].shift(t))
        dfIMLP['radio_t='+str(t)] = df["High"].shift(t) - dfIMLP['centro_t='+str(t)]
```

De forma similar se aplican las transformaciones a los intervalos de días posteriores, que serán las salidas a predecir por parte del modelo.

```
# Variables de salida
  for t in range(1,15):
```

```
dfIMLP['objetivoCentro'+str(t)]=dfIMLP['centro_t=0'].shif
t(-t)

dfIMLP['objetivoRadio'+str(t)]=dfIMLP['radio_t=0'].shift(
-t)
```

Se eliminan, a continuación, aquellos datos que contengan valores nulos debido a que los primeros intervalos no tendrán suficientes datos pasados que incluir en su información.

```
# Se eliminan las primeras delt_t filas
dfIMLP = dfIMLP.iloc[delt_t:]
```

Ahora que los datos están adaptados es necesario hacer las particiones correspondientes al entrenamiento, validación y test. En este caso se utilizó como criterio tomar aquellos posteriores y anteriores a ciertas fechas seleccionadas.

```
# Filtrado fecha
  dfIMLP = dfIMLP.loc[dfIMLP['Dates'] <= fechaFinal]

  fechaLimite = datetime.datetime(2021,2,1,0,0)

  dfTrain     =     dfIMLP.loc[dfIMLP['Dates']     <=
fechaLimite]
  dfTest = dfIMLP.loc[dfIMLP['Dates'] > fechaLimite]
```

Finalmente se eliminan los índices dado que no deben estar presentes en las entradas del modelo y se introducen en los ficheros .txt necesarios.

```
  del dfTrain['Dates']
  del dfTest['Dates']

  x = dfTrain.to_string(header=False,
                  index=False,
                  index_names=False).split('\n')
  vals = [' '.join(ele.split()) for ele in x]

  np.savetxt('train.txt',    vals,    delimiter="\n",
fmt="%s")

  x = dfTest.to_string(header=False,
                  index=False,
                  index_names=False).split('\n')
  vals = [' '.join(ele.split()) for ele in x]

  np.savetxt('val.txt',     vals,     delimiter="\n",
fmt="%s")
```

El único fichero que queda por modificar es el .net que definirá el modelo. Este fichero se puede utilizar para dos funciones: entrenamiento y validación.

Para configurarlo en modo entrenamiento será necesario editar el fichero de forma similar al ejemplo que se muestra a continuación.

```
my_file = open("zcub.net", "w")



text_list = ["mp(13,15,1)\n",
             "learn\n",
             "nco(500)\n",
             "wd(0.00)\n",
             "ftrain(train.txt)\n",
             "ftest(test.txt)"]
my_file.writelines(text_list)
my_file.close()
```

La parte más importante está en la definición de text_list, donde cada línea define el modelo.

La primera línea indica la estructura del iMLP, donde el primer número es el número de entradas, el segundo el número de neuronas de la capa oculta y el tercero es el número de salidas. Sólo se pueden modelar iMLPs con una sola capa oculta.

La segunda línea indica el tipo de entrenamiento:

- learn : con esta opción se inicializa aleatoriamente los pesos y se inicia un entrenamiento
- c_learn : con esta opción se carga el fichero de pesos previamente ajustados y se prosigue el entrenamiento

La tercera línea indica el número de ciclos de optimización, siendo razonable establecer un número entre 100 y 500.

La cuarta línea indica el valor del término de regularización que evita que los pesos de la red tomen valores inusualmente altos. Los valores de este término oscilan entre 0 y 0.0010.

La quinta línea indica el nombre del archivo con los datos de entrenamiento.

La sexta línea indica el nombre del archivo con los datos de test. Puede ser el mismo archivo que el usado para el entrenamiento.

Las líneas de comentarios en estos archivos irán precedidas de *.

Si lo que se desea hacer es validar el modelo, el código cambia ligeramente.

```
my_file = open("zcub.net", "w")



text_list = ["mp(13,15,1)\n",
             "ftest(val.txt)"]
my_file.writelines(text_list)
my_file.close
```

En este caso solo hay que indicar la arquitectura y el fichero de validación.

Una vez se han configurado todos archivos necesarios, el último paso es ejecutar el .exe.

```
subprocess.run(["mp2wd_interval.exe", "zcub"])
```

Como ultimo apunte, para que el programa funcione correctamente habrá que tener todos los archivos y el ejecutable dentro de un mismo directorio, así como importar la librería subprocess y cualquier otra que sea necesaria.

VII. ARQUITECTURA DEL SISTEMA

Con el modelo preparado para ser ejecutado desde Python, surge la duda de qué arquitectura utilizar en el desarrollo de los modelos predictivos. La primera estructura por la que se optó fue un modelo donde se tomaba la información de los k días anteriores para realizar predicciones sobre el día siguiente.

Para poder programar el modelo fue necesaria la adaptación de los datos de entrada al formato de intervalo. Esto desembocaba en dos opciones con respecto al intervalo que analizar: el intervalo de apertura a cierre y el del mínimo al máximo.

Se utilizó por el intervalo de mínimo a máximo. La razón por la que se optó por la pérdida de información del crecimiento frente a la pérdida de la representatividad de la información es que la información de un intervalo predicho ya informa del crecimiento en términos probabilísticos.

Con este dilema resuelto y tomada la decisión, se pudieron adaptar los datos al formato de centro y radio requerido por el iMLP.

Si suponemos que se han realizado una serie de análisis a través de los cuales se ha llegado a la conclusión de que es altamente probable que el mercado presente un comportamiento abrupto decreciente o cualquier variación, sería conveniente adecuar la herramienta a esa clase de información. Por esta razón se decidió crear una variable de

entrada nueva que aporte información al respecto, pero que esta, a su vez sea opcional. Los distintos tipos de comportamiento sobre los que se planteó el trabajo fueron los siguientes:

- Comportamiento lateral
- Crecimiento abrupto
- Crecimiento suave
- Decrecimiento suave
- Decrecimiento abrupto

Para introducir esta información se optó por una codificación one hot. Esto es un vector de dimensión igual al número de casos posibles, y hacer que cada componente valga uno o cero en función de si es o no el caso que corresponde

En el artículo [9], se profundiza sobre la idea de descomponer el problema en otros más pequeños. Aplicando esto a los modelos se planteó la posibilidad de entrenar múltiples modelos que luego se juntarían para predecir una serie temporal.

Esta descomposición consistió en particionar los datos en base a ciertas características comunes. Dichas características fueron la hora del día, el día de la semana, y el momento del año.

Para un modelo particionado por horas se desarrollarían 24 modelos, uno para cada hora del día, donde se buscaría hacer la predicción correspondiente a esa misma hora pasado uno o varios días. Se haría lo mismo con las particiones basadas en los días de la semana y los momentos del año.

A la hora de definir el formato de las variables a partir de las cuales hacer la partición se probaron múltiples opciones, entre ellas las dos anteriormente descritas, una con el número y otra con codificación one hot. Adicionalmente también se estudió una codificación en la que se escribía el número en binario, pero esto daba problemas al ser la distancia euclídea variable en algunos casos cuando se incrementaba el valor en una unidad. Esto se solucionó aplicando codificación Gray, donde la distancia siempre es uno.

Otro de los formatos fue el siguiente, el cual se aplicó sobre la característica del momento del año.

$$\mathbf{p} = \left[\sin\left(2\pi\frac{\#(i+\tau)}{366}\right) \quad \cos\left(2\pi\frac{\#(i+\tau)}{366}\right)\right]$$

Una vez se han creado todos estos modelos se juntan y se genera un predictor para todos los instantes.

Una de las ventajas que aporta esta metodología es la simplicidad de los modelos individuales desarrollados, que tienen una complejidad mucho menor.

La siguiente cuestión viene a la hora de decidir cuál debe ser el número adecuado de días anteriores a introducir en las entradas de la red, así como el número de neuronas en la capa oculta. Para ello se utilizó la fuerza bruta.

Se entrenó el modelo para un amplio rango de combinaciones para los valores de los hiperparámetros. Este procedimiento requirió de mucho tiempo y procesamiento, pero se logró optimizar la arquitectura del modelo.

De este procedimiento surge una nueva pregunta. La idea subyacente es la de probar muchos casos y tomar el mejor. Se dispone de un sistema que permite probar muchos casos, pero no de una forma adecuada de tomar el mejor. Una de las formas más comunes de evaluar esta calidad en un modelo es a través del cálculo de estadísticos como el error cuadrático medio, el error absoluto medio, porcentaje de aciertos, etc. Esto son, en definitiva, métricas de la calidad del modelo. El problema está en que la aplicación de cualquiera de los estadísticos mencionados no es posible debido a que los datos de salida tienen estructura de intervalo.

Para esta clase de datos no resulta sencilla la implementación de una métrica de error porque el equivalente a distancia entre intervalos no es sencillo, y si hay métricas, están estarán probablemente lejos de representar el error.

Tras investigar posibles estadísticos para los datos de intervalo, las que se seleccionaron fueron iARV, u de Theill, CR y ER.

$$iARV = \frac{\sum\left(\widehat{Y_L} - Y_L\right)^2 + \sum\left(\widehat{Y_U} - Y_U\right)^2}{\sum\left(\overline{Y_L} - \widehat{Y_{L,t}}\right)^2 + \sum\left(\overline{Y_U} - \widehat{Y_{U,t}}\right)^2}$$

$$U\ de\ Theill = \frac{\sum\left(\widehat{Y_L} - Y_L\right)^2 + \sum\left(\widehat{Y_U} - Y_U\right)^2}{\sum\left(Y_{L,t} - Y_{L,t-1}\right)^2 + \sum\left(Y_{U,t} - Y_{U,t-1}\right)^2}$$

$$CR = \frac{Min\left(\widehat{Y_U}, Y_U\right) - Max\left(\widehat{Y_L}, Y_L\right)}{Y_U - Y_L}$$

$$ER = \frac{Min\left(\widehat{Y_U}, Y_U\right) - Max\left(\widehat{Y_L}, Y_L\right)}{\widehat{Y_U} - \widehat{Y_L}}$$

A continuación, se muestran los resultados obtenidos para una de las combinaciones que surgieron del entrenamiento.

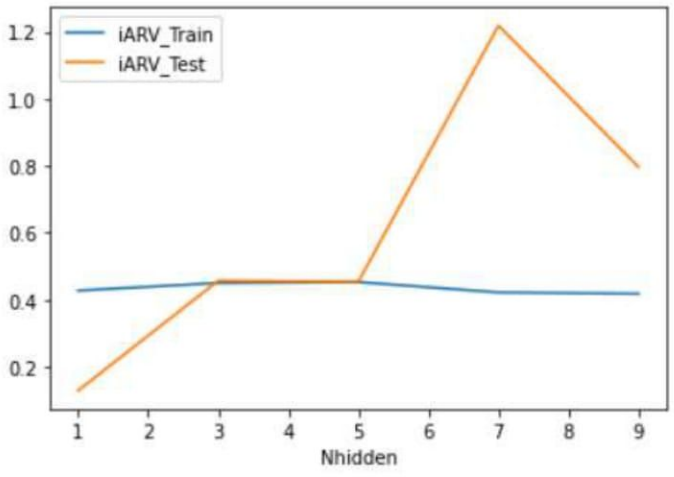

Ilustración 8: Estadístico iARV vs Número de neuronas en entrenamiento y test, de elaboración propia

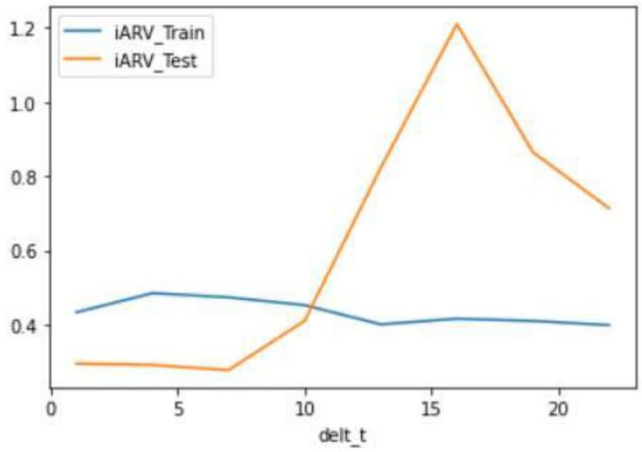

Ilustración 9: Estadístico iARV vs delt_t en entrenamiento y test, de elaboración propia

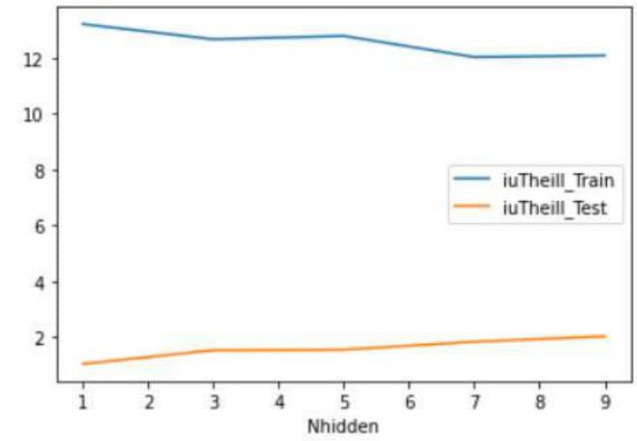

Ilustración 10: Estadístico iu de Theill vs Número de neuronas en entrenamiento y test, de elaboración propia

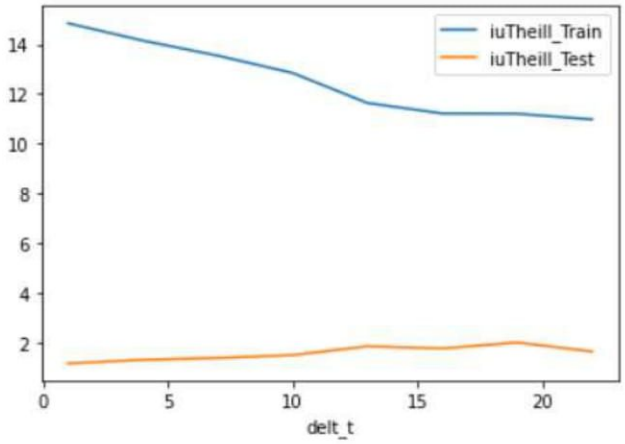

Ilustración 11: Estadístico iu de Theill vs delt_t en entrenamiento y test, de elaboración propia

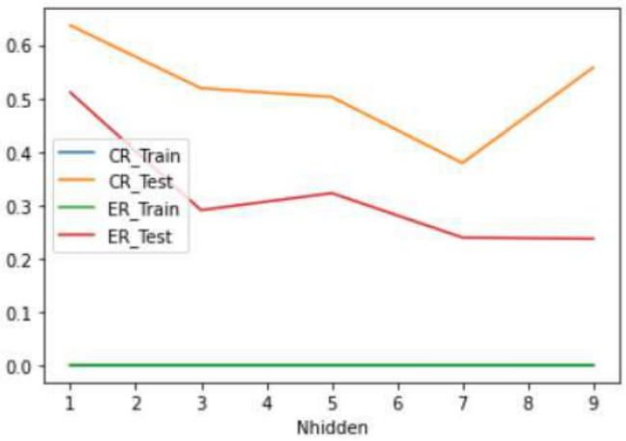

Ilustración 12: Estadísticos CR y ER vs Número de neuronas en entrenamiento y test, de elaboración propia

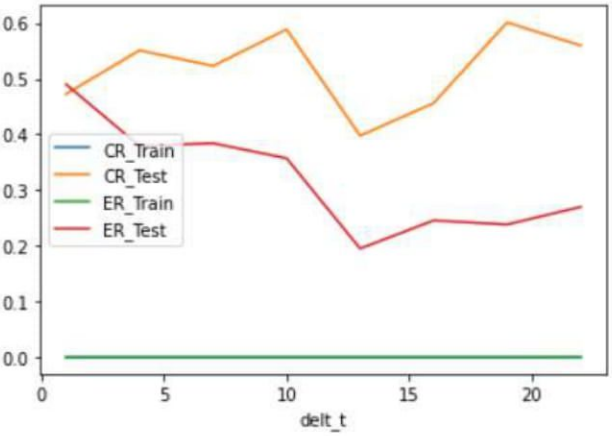

Ilustración 13: Estadísticos CR y ER vs delt_t en entrenamiento y test, de elaboración propia

	delt_t	Nhidden	iARV_Train	iARV_Test	iuTheill_Train	iuTheill_Test	CR_Train	CR_Test	ER_Train	ER_Test
0	1	1	0.130369	0.117266	14.640694	0.996348	0.0	0.631478	0.0	0.602085
1	10	1	0.455296	0.117361	12.998602	1.017461	0.0	0.647967	0.0	0.514040
2	22	1	0.449249	0.117660	12.028546	1.008509	0.0	0.697581	0.0	0.518438
3	7	1	0.474978	0.122635	13.581828	1.041035	0.0	0.631905	0.0	0.507354
4	1	3	0.525260	0.125546	15.048160	1.015504	0.0	0.652994	0.0	0.614999

Tabla 4: Estadísticos de los 5 mejores modelos, de elaboración propia

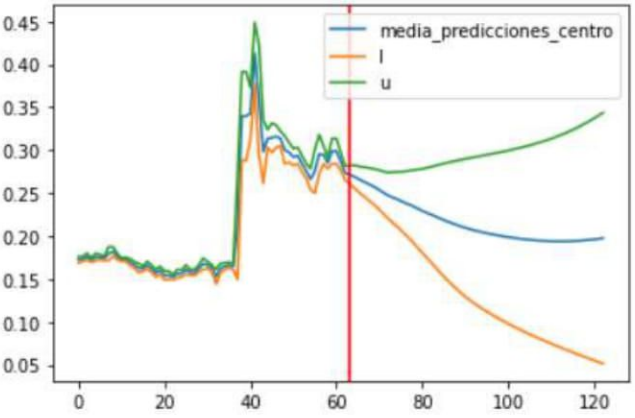

Ilustración 14: Predicciones de los próximos 60 días, de elaboración propia

Estos resultados llaman la atención porque los modelos poseen buenas métricas de iARV, u de Theill, ER y CER en la

teoría, pero, sin embargo, las predicciones muestran valores absurdos, donde el intervalo predicho se ensancha hasta alcanzar prácticamente todos los valores. En las figuras se indica con una línea roja el punto a partir del cual el modelo no tiene datos futuros de los días siguientes. Cabe destacar que la degradación, hasta cierto punto, tiene sentido porque las predicciones son cada vez menos fiables, por lo que el intervalo predicho es cada vez más amplio para así englobar esa variabilidad hasta que contiene a todos los valores posibles.

Tras un análisis se concluyó que la causa estaba en la metodología a la hora de hacer esas predicciones. Como no se tienen los valores reales de los días siguientes, el modelo utiliza sus propias predicciones de esos días como datos de entrada para predecir el día siguiente.

La degradación del modelo es inevitable si se utiliza esta metodología porque la base de todo modelo son los datos con los que se entrena. Esta degradación puede ser tolerable en muchas situaciones si el modelo es suficientemente preciso o si el horizonte de predicción es lo bastante corto.

Sin embargo, dada la naturaleza del problema y su dificultad a la hora de hacer predicciones, el error es considerable y se buscan horizontes de predicción cada vez más amplios. De aquí se concluye que el modelo desarrollado no es el más adecuado si se pretende predecir un número elevado de días.

Ahora que se ha identificado el error en el modelo, es necesario buscar una solución capaz de mitigar o eliminar los efectos del problema. Para ello el objetivo fue eliminar las predicciones de predicciones, pero sin que esto redujese el horizonte.

El cambió que se estudió implementar fue el de modificar la arquitectura de la red para que, en vez de tener una salida (el intervalo del día siguiente), tuviese un número igual al horizonte de predicción, de forma que cada una de las salidas aportasen la información de intervalo correspondiente a un día en concreto del horizonte.

Estos cambios en la arquitectura del modelo suponen varios retos nuevos.

Los cambios en la arquitectura incrementaron considerablemente la complejidad del modelo desde el punto de vista computacional. Los tiempos de ejecución superaron con creces a los anteriores hasta el punto en que la ejecución del proceso de prueba de combinaciones de hiperparámetros para el modelo era de varios días.

Esto supuso nuevos retos a batir, el entrenamiento consecutivo utilizando todas las posibles combinaciones para cada acción ya no es viable para ejecutar sobre un elevado número de acciones debido a las limitaciones de tiempo. El trabajo realizado al probar la arquitectura anteriormente propuesta, aunque no fue el más acertado, sí que puede ayudar a vislumbrar en torno a qué valores rondan los hiperparámetros del modelo.

También quedaba la cuestión de qué horizonte de predicción sería el más adecuado. En una primera versión se aplicó en torno a un mes (29 días).

En esta primera versión los modelos se evaluaron utilizando los estadísticos de la arquitectura anterior por medio de la toma de la primera predicción de todos los intervalos predichos para así poder calcular de forma más sencilla.

Los resultados de todos los modelos generados se muestran a continuación. Las visualizaciones representan todos los valores de los estadísticos agrupados según los hiperparámetros utilizados, que eran la cantidad de días pasados que se consideraban para la introducirlos como entradas dentro del modelo, así como el número de neuronas dentro de la única capa oculta presente en el modelo.

De cada agrupación se toma la media como operación de agregación. En un principio se consideró utilizar el mejor modelo, sin embargo, si se utiliza la media se pueden prevenir una alta representatividad de las observaciones anómalas que afecte a la medida.

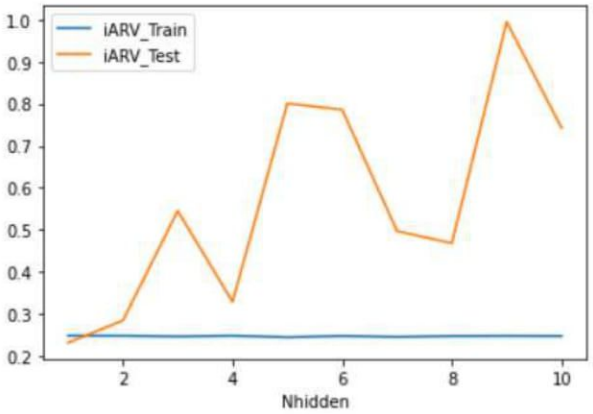

Ilustración 15: Media iARV vs Número de neuronas ocultas, de elaboración propia

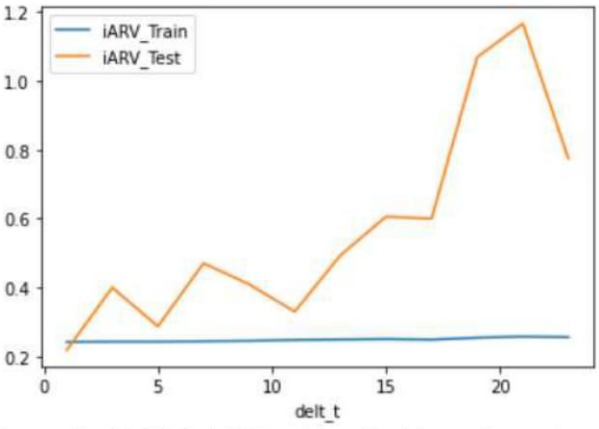

Ilustración 16: Media iARV vs delt_t, de elaboración propia

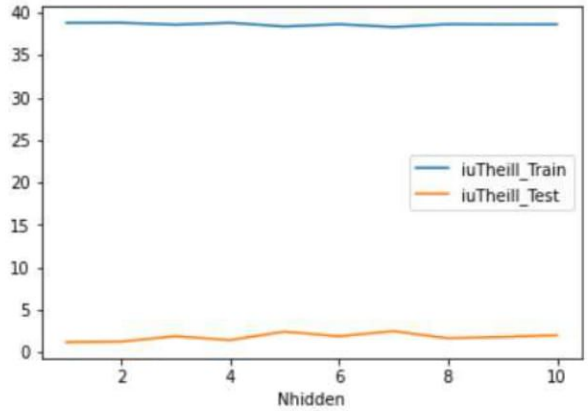

Ilustración 17: Media iu de Theill vs Número de neuronas ocultas, de elaboración propia

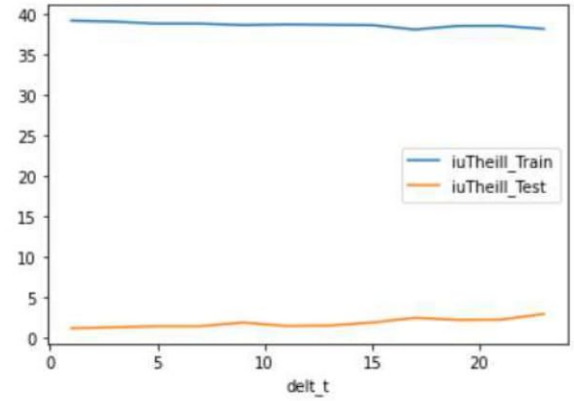

Ilustración 18: Media iu de Theill vs delt_t, de elaboración propia

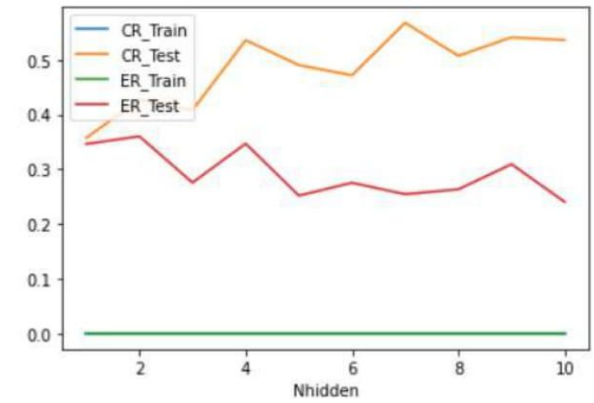

Ilustración 19: Media CR y ER vs Número de neuronas ocultas, de elaboración propia

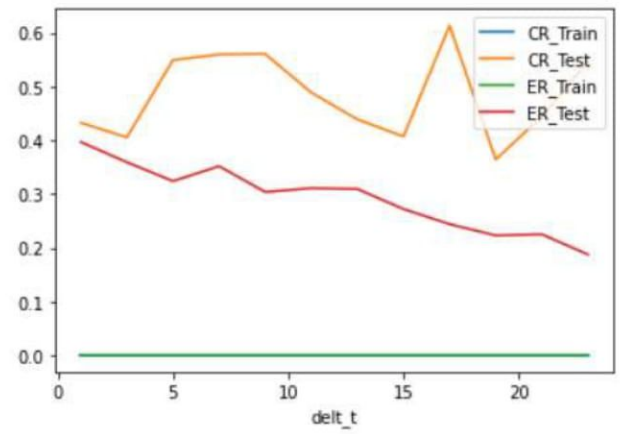

Ilustración 20: Media CR y ER vs delt_t, de elaboración propia

	delt_t	Nhidden	iARV_Train	iARV_Test	iuTheill_Train	iuTheill_Test	CR_Train	CR_Test	ER_Train	ER_Test
6	1.0	7.0	0.241129	0.101933	39.166350	0.990018	0.0	0.610310	0.0	0.496968
8	1.0	9.0	0.241289	0.104807	39.144022	1.014467	0.0	0.585999	0.0	0.492567
3	1.0	4.0	0.241432	0.118754	39.223570	1.000703	0.0	0.534029	0.0	0.489977
32	7.0	3.0	0.243215	0.120688	38.834452	1.076721	0.0	0.564569	0.0	0.461742
31	7.0	2.0	0.243209	0.123016	38.824355	1.024127	0.0	0.499836	0.0	0.474264

Tabla 5: Estadísticos medios de los 5 mejores modelos, de elaboración propia

El problema de la complejidad continúa afectando al tiempo de procesamiento, por lo que el siguiente objetivo que surgió fue el de reducir la complejidad del proceso del entrenamiento sin que esto afectase a los resultados del modelo.

La primera de las formas para reducir los tiempos de ejecución está en reducir el número de combinaciones posibles para las configuraciones del modelo. De esta forma, se tendrán que ejecutar un número menor de veces el entrenamiento del modelo, reduciendo enormemente el tiempo de ejecución.

Los resultados de la primera arquitectura que se propuso, la que predecía solamente el día siguiente, mostraron unas conclusiones que podrían ayudar en este frente. Prácticamente todos los mejores modelos tomaban 5 como el número adecuado de días pasados a introducir en el modelo. Si a esto le sumamos el hecho de que varios estudios como [8] también se toma la información correspondiente a 5 días, se puede concluir que es probable que sea la información adecuada a introducir en las entradas del modelo.

La combinación de entradas que se consideró más adecuada para el modelo del artículo se muestra a continuación.

ANN of maximum prices	ANN of minimum prices
Opening price of day D	
Opening price of day $D-1$	Opening price of day D
Maximum price of day $D-1$	Opening price of day $D-1$
Closing price of day $D-1$	Minimum price of day $D-1$
Best sell bid of day $D-4$	Closing price of day $D-1$
Closing price of day $D-5$	Opening price of day $D-2$
WMA of American dollar	

Tabla 6: Entradas óptimas utilizadas para construir modelo, tomado de [8]

El sentido de esto puede venir de la importancia de los ciclos dentro de una semana, que en estos datos son 5 días debido a que los mercados están cerrados los fines de semana. Es probable que, en el mundo del trading a muy corto plazo (intradía), los traders sean más reticentes a comprar un viernes porque, en caso de comprar y mantener una posición abierta durante el día, entonces también estarían obligados a mantenerla abierta durante dos días más.

Esto implica que ese tipo de posiciones estarían más expuestas a la incertidumbre del fin de semana, lo que hace que sea ligeramente menos probable que se invierta un viernes, lo que disminuirá la demanda de acciones ese día y, consecuentemente, su precio de venta en el mercado.

Cabe destacar que el hecho de que el número adecuado de días de entrada sea solo de 5 y no mayor es contraintuitivo. Cuantos más datos útiles tenga un modelo, mejor debería ser

en principio, por lo que se induciría a pensar que hay algún tipo de error. Una forma de explicar este fenómeno sería que la calidad de los datos no fuese lo suficientemente buena, pero este no es el caso, la información de los días anteriores es a priori importante, y cuanto más se explore, mejores resultados debería obtener el modelo.

La auténtica explicación de por qué no es conveniente añadir más entradas al modelo es la estructura limitada de la arquitectura de la red del iMLP debido a que esta no permite añadir más de una capa. Esto reduce enormemente la complejidad y la capacidad del modelo para extraer reglas y deducciones de los datos.

Sin embargo, los buenos resultados mostrados en trabajos anteriores sobre el iMLP inducen a pensar que o bien el problema que se está estudiando en este conjunto de datos no requiere de Deep Learning, o, por el contrario, el aprendizaje profundo podría permitir aprovechar un elevado número de datos de entrada adicionales que enriquecerían el modelo pero, debido a las limitaciones del código en C con el que se está trabajando, este camino no se puede explorar, aunque sin duda valdría la pena estudiar la posibilidad de ampliar el código para que permita estas características.

De esta forma, se fijó el número de días anteriores a 5, por lo que se redujo significativamente el número de combinaciones de hiperparámetros y, en consecuencia, de iteraciones en los entrenamientos del modelo.

El único hiperparámetro que queda por definir es el del número de neuronas en la capa oculta. Aunque ya se posea un rango en torno al cual se puede saber en qué intervalo de valores se encontrará el óptimo, la información no es tan concluyente como la del caso del número de días, donde todos los mejores modelos mostraban un número igual a 5.

Para este parámetro, por tanto, el conjunto de valores utilizado para explorar posibles combinaciones fue muy similar al del caso anterior.

Con estos cambios introducidos se consiguió el objetivo de reducir el número de combinaciones de los entrenamientos mediante esas asunciones. El siguiente paso fue buscar formas de reducir la complejidad interna del modelo y, por tanto, su tiempo de entrenamiento en cada iteración.

Para ello es necesario hacer un análisis del problema, que proviene de la aplicación del horizonte de predicción. Esto es un incremento del número de variables de salida, por lo que se optó por reducir el horizonte de predicción.

La longitud del horizonte de predicción debería ser lo bastante alta como para aportar valor, pero lo bastante baja como para que no se produzcan problemas en los tiempos de ejecución.

Finalmente se optó por reducirlo a aproximadamente la mitad, de un mes (29 días) a dos semanas (14 días).

A raíz del desarrollo de esta versión se observó que había un problema en la retroalimentación del modelo. La métrica utilizada para valorar los modelos debe cambiar forzosamente. Cada intervalo predice un conjunto de intervalos y no uno solo, por lo que el cálculo de parámetros como el iARV no es posible si se quieren tener en cuenta los valores de todo el horizonte dentro de los datos de la predicción.

Todos los parámetros se podrían calcular para cada horizonte predicho a partir de la información de cada día, pero se pasaría de tener un único estadístico como en el caso inicial a un vector de estadísticos.

Tras un análisis de posibles estadísticos que utilizar, se optó por la solución más sencilla, tomar la media del vector de estadísticos calculado y usarlo como referencia para el modelo. Este proceso se realizó sobre el iARV y la u de Theill.

La estructura final de la arquitectura del modelo tiene la siguiente forma.

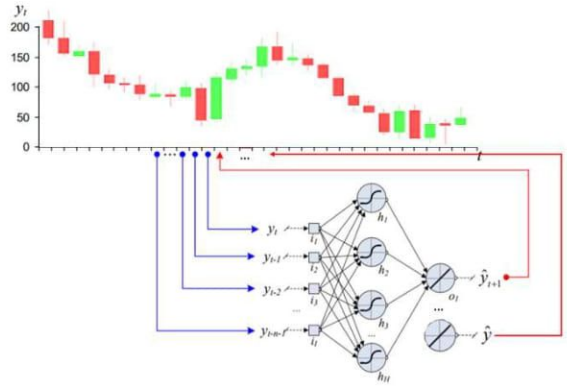

Ilustración 21: Arquitectura final del modelo. Con base en [12]

Con 5 entradas y 14 salidas.

Con todo esto aplicado, la diferencia en el procesamiento pudo ser compensada y se pudo entrenar de nuevo el modelo, dando lugar a unos resultados en las predicciones muy distintos.

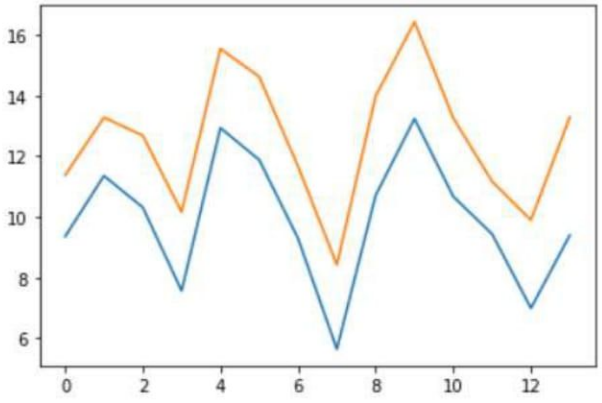

Ilustración 22: Predicciones máximo y mínimo con horizonte de 12 días, de elaboración propia

Ahora ya era posible ver qué forma realmente se había estimado con un tiempo de ejecución razonable y sin el problema de la degradación de la calidad del modelo debido a la realización de predicciones de predicciones.

Otro parámetro que se trató de ajustar fue el Nco, que corresponde al número de ciclos de optimización. Tiene una

función similar a las epochs de un modelo de Machine Learning convencional.

Para lograr establecer un criterio en torno al cual decidir cuál es el valor óptimo, se debe tener en cuenta la importancia de la velocidad de convergencia del modelo. Cuanto más lento sea, mayor será el número de ciclos necesario para que el modelo se aproxime con una cercanía razonablemente alta a la solución a la que se converja.

Si la diferencia es notable, entonces el modelo tiene amplias posibilidades de mejorar sus resultados, y será conveniente darle más margen y aumentar el número de ciclos, sacrificando tiempo de procesamiento a cambio de precisión.

El problema está en que no es posible ver hacia qué valor converge el modelo, por lo que resulta difícil ver la diferencia con respecto al valor óptimo. Lo que sí que se puede medir es la distancia entre las calidades de los modelos, razón por la cual se utilizó como método para comparar ambos modelos y poder así valorar la calidad y el valor aportado por los ciclos extra que se hayan añadido.

Las figuras con las predicciones para 500 y 200 se muestran respectivamente a continuación.

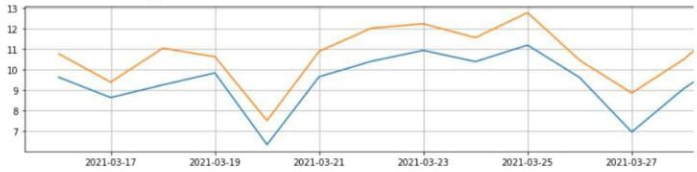

Ilustración 23: Predicciones máximo y mínimo con nco=500, de elaboración propia

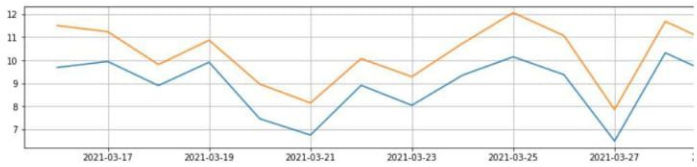

Ilustración 24: Predicciones máximo y mínimo con nco=200, de elaboración propia

La primera figura, de 500, será siempre mejor que la segunda. Lo que se mide no es la calidad de las predicciones, sino que las diferencias entre los resultados de las predicciones de ambos modelos.

Se pueden ver ciertas similitudes en varios tramos de ambas gráficas, lo que podría ser un buen indicador, pero también hay diferencias notables en las predicciones de la mayoría de los tramos. Esto significa que no hay una similitud suficientemente fuerte como para poder concluir que son iguales.

Por esta razón se optó por mantener una cantidad de 500 ciclos a pesar de que suponga un incremento significativo en los tiempos de ejecución.

VIII. OTROS MODELOS

El modelo que se ha estudiado ha dado lugar a un prototipo fruto de todo el desarrollo que ya puede hacer predicciones, pero se necesitará un conjunto de modelos más convencionales para poder comparar sus resultados.

El primero que se desarrolló fue el MLP. Es necesario remarcar que es no solo el modelo más sencillo, sino que también el más similar en arquitectura, metodología, etc, a la red iMLP que se acaba de desarrollar.

Para elaborar la arquitectura de entradas y salidas del MLP se utilizó una estructura igual a la del iMLP, con la diferencia de que los datos tratados estarán en formato crisp en vez de intervalo.

A continuación, se muestra un esquema de la arquitectura.

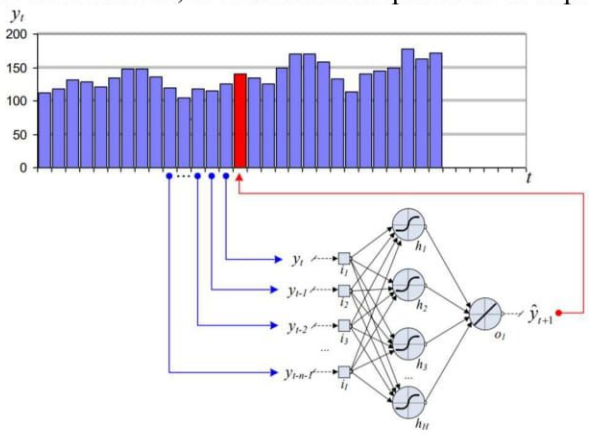

Ilustración 25: Esquema arquitectura MLP. Tomado de [12]

Estudio de independencia entre centros y radios

Una de las razones por las que se realizó este trabajo es para estudiar el iMLP y por qué tiene unos resultados tan buenos. Una de las posibilidades que podrían explicar este fenómeno son las relaciones presentes entre los centros y los radios de cada uno de los intervalos que se introducen en el modelo.

Por esta razón se decidió explorar esta vía de actuación a través de una representación de puntos de un conjunto que se había estudiado.

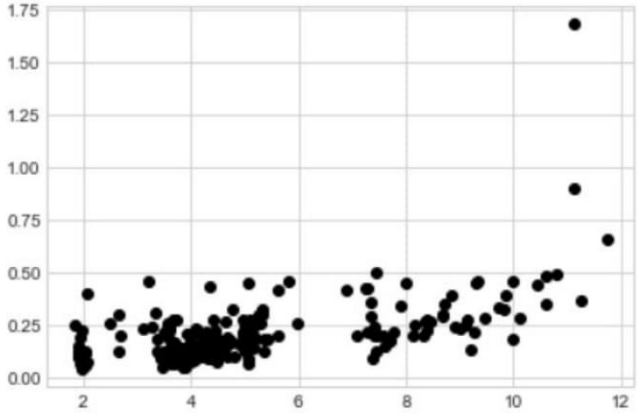

Ilustración 26: Centros y radios de los puntos del conjunto de datos, de elaboración propia

El objetivo era encontrar algún tipo de estructura en los datos. La independencia más pura implicaría que la representación tuviese forma de rectángulo, algo que, a priori, la representación no muestra en varias zonas.

Se van a analizar cada una de las diferencias con respecto al modelo teórico independiente rectangular para así poder llegar a concluir si son significativas.

Lo más llamativo, que arruina cualquier expectativa de conseguir la forma rectangular objetivo, son las observaciones anómalas con un radio y centro por encima de lo normal. Estos datos, aunque anómalos, no son representativos en los datos, por lo que se podrían despreciar y seguir adelante con la asunción de que los centros y radios son independientes sin ningún tipo de problema.

Otra característica que llama la atención son los cortes abruptos entorno a los puntos con centro igual a 3 y 6.5. Estos cortes tan notorios separan el conjunto de datos en tres clústeres aparentemente distintos.

Para poder analizar más en profundidad esta cuestión, es necesario visualizar el conjunto de datos que se presenta a continuación. Se trata de la acción de la empresa BCRX en el periodo de marzo de 2020 a marzo del 2021.

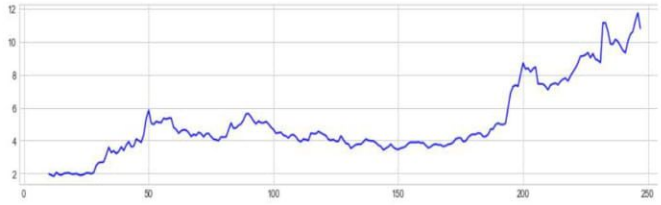

Ilustración 27: Histórico de la acción BCRX. Fuente: Bloomberg Finance L.P.

Si observamos la gráfica en torno a los valores que actúan como separadores, podemos entender la razón de los mismos. Al principio del histórico hubo una subida que supuso un cambio en la cotización de aproximadamente 2 a en torno a 3.5 y luego 3.75. La acción no volvió a bajar de ese valor, por lo que en un periodo corto hubo una transición a un nivel de precio superior con muy pocos puntos intermedios, lo que explica la ausencia de puntos en esos valores.

Otra subida incluso más abrupta cerca del final del periodo de análisis puede explicar de forma similar el segundo corte en torno a 6.5.

La razón de estos cortes es, por tanto, la ausencia de puntos y no la dependencia entre las variables.

Otra de las diferencias entre la estructura rectangular uniforma teórica y la gráfica es la diferencia entre las densidades de puntos en determinadas zonas. De nuevo esta diferencia se puede explicar visualizando el conjunto de datos, donde hay más datos en ciertas zonas porque son los valores en torno a los que ha oscilado la acción en ese periodo.

Tras analizar una a una cada característica diferencial del modelo independiente teórico con respecto a la gráfica resultado se ha concluido que las diferencias son explicables y que, por tanto, los datos no presentarían ninguna estructura, siendo así independientes entre sí a priori.

De todas formas, cabe destacar que se están comparando los centros y radios de un mismo intervalo, y no el centro de un día con el radio de otro. Por lo que se decidió ir un paso más y explorar las relaciones temporales entre centros y radios.

Para hacer esto se optó por construir una visualización que englobase tanto centros con radios pasados como radios con centros pasados, mostrando múltiples combinaciones.

El resultado fue una matriz de 5x5 con visualizaciones similares a la que se mostró anteriormente. A pesar de haber realizado este análisis exploratorio de la variable temporal, los resultados de todas las combinaciones mostraron unas características similares a la de la variable, con las observaciones anómalas a en puntos con tres y radios altos, la diferencia entre las densidades, así como los cortes o zonas con ausencia de puntos.

Los resultados se muestran a continuación.

Ilustración 28: Centros y radios para múltiples distancias temporales, de elaboración propia

A partir de todo esto se puede concluir que la independencia entre centros y radios también está presente, aunque se modifique la componente temporal.

Por esta razón se considera que la creación de dos redes MLP independientes, una para predecir los centros y otra para predecir los radios, podría estimar con cierta precisión los intervalos al juntar ambos resultados.

Surge así la cuestión de cómo generar el mejor modelo en lo que respecta a las entradas de la red. Demasiadas entradas que no aporten información podrían ser perjudiciales debido a que el aprendizaje no será profundo. Esto se debe a que debe haber una comparación consistente entre modelos, y las limitaciones del iMLP, que es principalmente el número reducido de capas ocultas, deben mantenerse en este modelo MLP base.

Asimismo, una cantidad muy baja de variables de entrada afectará a la precisión del modelo, por lo que debe producirse un equilibrio en el número de entradas.

Para poder responder a esta cuestión, se hizo un estudio preliminar, y se observó que en [8] se concluyó que una buena combinación de variables de entrada y de salida la que ya se mostró anteriormente en la tabla:

ANN of maximum prices	ANN of minimum prices
Opening price of day D	
Opening price of day $D-1$	Opening price of day D
Maximum price of day $D-1$	Opening price of day $D-1$
Closing price of day $D-1$	Minimum price of day $D-1$
Best sell bid of day $D-4$	Closing price of day $D-1$
Closing price of day $D-5$	Opening price of day $D-2$
WMA of American dollar	

Tabla 6: Entradas óptimas utilizadas para construir modelo, tomado de [8]

Cabe destacar que la información de la tabla se refiere a la predicción, de forma independiente, de los máximos y mínimos de un intervalo, y lo que se pretende es crear modelos que predigan los centros y los radios, lo que supone un reto dada la información de la tabla pero que puede servir como punto de partida.

Para afrontar este problema, se estudiaron varias alternativas:

1. Hacer predicciones sobre los máximos y mínimos en vez de sobre centros y radios para posteriormente traducir los resultados a un centro y un radio. Esto podría ser lo mejor para aumentar la precisión, pero el problema es que lo que se está estudiando es la independencia del centro y del radio, y ambos modelos predicen los extremos del intervalo, por lo que predicen una parte de información del radio y otra del centro dentro de un mismo modelo, sin separar las componentes.
2. Adaptar las entradas de máximo y mínimo a centro y radio. Para ello se podría considerar que, si se tienen modelos capaces de estimar los extremos de un intervalo, entonces la información necesaria para hallar la información del centro podría ser la intersección de las entradas de ambos modelos al tratarse de la parte común. Para predecir el radio se utilizaría el complementario de la intersección.
3. Hacer predicciones sobre centro y radio sin cambiar las entradas. Esta es la opción más sencilla, y no solucionaría ninguno de los problemas que vendrían, se reduciría la precisión y surgiría la duda de qué entradas convendría utilizar para el radio y cuáles para el centro.
4. Entrenar ambos modelos con todas las entradas. Esta es la solución menos eficiente en lo referente a precisión y complejidad, lo que incrementaría los tiempos de procesamiento.

La solución por la que finalmente se optó, debido a que lo que se estaba estudiando eran centros y radios, y no máximos y mínimos, es la opción 2, donde se utiliza la intersección de las entradas para los modelos de máximo y mínimo como entradas para la red que haga predicciones sobre los centros, y las restantes, correspondientes al complementario de la intersección, para el modelo predictor de radios.

Tras realizar todo el proceso de entrenamiento se llegaron a las siguientes predicciones, las cuales destacan por presentar un ancho de intervalo prácticamente constante.

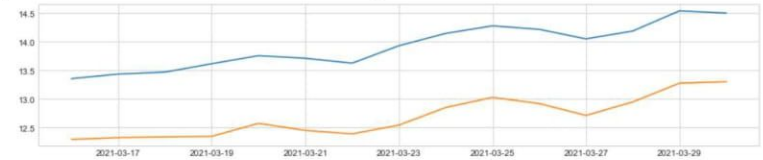

Ilustración 29: Predicciones MLP base, de elaboración propia

Con los centros y radios predichos, se pudo pasar de forma sencilla a máximo y mínimo utilizando las siguientes fórmulas:

$$Máximo = Centro + Radio$$
$$Mínimo = Centro - Radio$$

A primera vista, se podría decir que el modelo es un predictor centrado en las tendencias a largo plazo mucho más que en el corto, razón por la cual las variaciones son tan mínimas entre los días.

Si se analizan de forma más cuantitativa cada uno de los modelos, se puede ver que el error cuadrático medio es mucho menor en los radios que en los centros.

Centros:

```
model.evaluate(normed_test_data, test_labels)

2/2 [==============================] - 0s 5ms/step - loss: 0.83

[0.8359615802764893, 0.667266845703125, 0.8359615802764893]
```

Radios:

```
model.evaluate(normed_test_data, test_labels)

2/2 [==============================] - 0s 5ms/step - loss: 0.016

[0.01654953323304653, 0.0831475779414177, 0.01654953323304653]
```

Una vez se tuvieron los resultados del modelo MLP base, se decidió explorar otros modelos para comparar resultados. En primer lugar, se estudió la ampliación del número de capas ocultas de la red.

La única diferencia con respecto al modelo base fue el uso de dos capas en lugar de una, dando lugar a las siguientes predicciones.

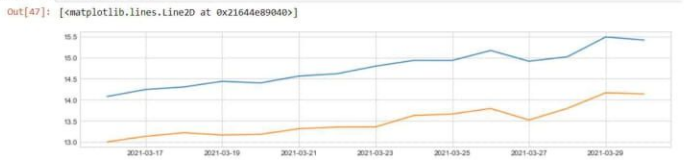

Ilustración 30: Predicciones MLP profundo, de elaboración propia

Los resultados son a priori muy similares a los que se obtuvieron con el modelo MLP básico. Si se analizan en detalle los errores cuadráticos medios, también se encuentran entorno a unos valores muy similares.

Centros:

```
model.evaluate(normed_test_data, test_labels)

2/2 [==============================] - 0s 2ms/step - loss: 0.840

[0.8403813242912292, 0.652463972568512, 0.8403813242912292]
```

Radios:

```
model.evaluate(normed_test_data, test_labels)
```

```
2/2 [==============================] - 0s 2ms/step - lc

[0.01630994863808155, 0.08114250749349594, 0.016309948€
```

Asimismo, también se llegó a la misma conclusión con respecto a la precisión en los centros y radios.

Una posible razón de que el error sea tan bajo en los radios que probablemente explique las diferencias es la diferencia entre los órdenes de magnitud de los radios con respecto a los centros. Los centros son mucho mayores que los radios, por lo que su error medio deberá ser significativamente mayor.

La razón más probable por la que los resultados en el modelo MLP simple sean similares a los del profundo es que se trata de un modelo sencillo. El número de entradas permaneció igual, por lo que el modelo no requería de complejidad adicional para ser entrenado, lo que le lleva al estancamiento de los resultados del modelo al principio y, finalmente, al sobreaprendizaje.

LSTM

Otro de los modelos que se quiso probar fueron las llamadas LSTM (Long Short Term Memory). Como ya se explicó anteriormente, estas redes tienen la cualidad de la memoria, son idóneas para datos secuenciales como texto o series temporales, por lo que es un modelo que podría aportar valor dado el tipo de conjunto de datos y datos con los que se están trabajando en el proyecto.

Esta memoria se muestra en la retroalimentación de las neuronas, que están conectadas con las entradas anteriores, de la siguiente forma.

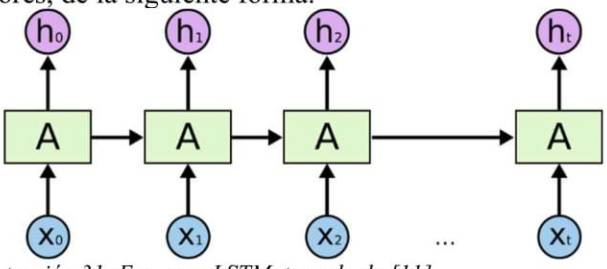

Ilustración 31: Esquema LSTM, tomado de [11]

A pesar de las expectativas, los resultados no fueron especialmente sobresalientes. Destaca el elevado error cuadrático medio del modelo predictor de centros.

Centros:

```
model.evaluate(normed_test_data, test_labels)
```

```
2/2 [==============================] - 0s 1ms/step - l

[6.279849052429199, 1.914023756980896, 6.2798490524291
```

Radios:

```
model.evaluate(normed_test_data, test_labels)
```

```
2/2 [==============================] - 0s 2ms/step - l

[0.022372864186763763, 0.09723648428916931, 0.02237286
```

La calidad a la hora de predecir radios es, por el contrario, similar a la mostrada en el resto de los modelos.

IX.MODELO DUDEK

En el artículo referenciado en [9] se sigue una metodología de descomposición del problema de modelaje en múltiples modelos para predecir series temporales con datos energéticos.

Un enfoque similar podría ser aplicado a la clase de datos que se está analizando, de múltiples mercados financieros. Para ello se tomaron datos del índice NASDAQ y se estudiaron las distintas transformaciones que podrían ser necesarias para realizar el caso análogo adaptado.

En primer lugar, se tomaron los valores de entrada de cada media hora en vez de cada día y se normalizaron mediante la siguiente fórmula:

$$x_{i,t} = \frac{L_{i,t} - \overline{L_i}}{D_i}$$

Donde:

$$D_i = \sqrt{\sum_{l=1}^{n} (L_{i,l} - \overline{L_i})^2}$$

Asimismo, también se normalizaron las salidas:

$$y_{i,t} = \frac{L_{i+\tau,t} - \overline{L_i}}{D_i}$$

Como se no se conoce $L_{i+\tau,t}$ se utiliza la aproximación a partir de la salida del modelo como estimador.

$$\hat{L}_{i+\tau,t} = \widehat{y_{i,t}} D_i + \overline{L_i}$$

Cabe destacar que las transformaciones aplicadas a los datos para conseguir las entradas del modelo son las usadas en los procesos de tipificación:

$$z = \frac{x - \hat{x}}{\sigma}$$

Estos procesos se utilizan para pasar a de una distribución normal cualquiera a otra con media nula y desviación típica igual a la unidad.

El siguiente paso es crear otro modelo que tome como entradas los días de la semana. Para ello se estudiaron las distintas codificaciones del día de la semana y finalmente se optó por el one hot encoding, que ya se usó anteriormente, de forma que se realizaría el siguiente mapeo:

- Lunes -> 10000
- Martes -> 01000
- Miércoles -> 00100
- Jueves -> 00010
- Viernes -> 00001
- Sábado y Domingo -> 00000

Los sábados y domingos se pondrá todo el vector a ceros por defecto, pero en principio no aplica porque el horario de apertura de la bolsa no incluye los fines de semana. Sin embargo, es conveniente definir ese default en caso de se produzca cualquier error en la lectura o extracción de los datos

Como se tienen datos de cada media hora del día, fue necesario el cálculo de los datos del día a través de un cálculo

agregado de todo el intervalo.

Dentro del artículo también se trató el tema de la representación del periodo del año como un vector unitario de componentes trigonométricas:

$$p = \left[\sin\left(2\pi\frac{\#(i+\tau)}{366}\right), \cos\left(2\pi\frac{\#(i+\tau)}{366}\right)\right]$$

La razón por la que esa inclusión de datos aportó valor al modelo está en la base de datos, que era de energía, la cual tiene un precio muy dependiente del periodo del año debido a que está íntimamente relacionado con la demanda de los consumidores.

Si se analizan los datos con los que se está tratando, se puede ver fácilmente cómo no existe a priori relación alguna entre el periodo del año y el valor de la acción. La razón es que el mercado se anticipa a todos estos cambios, incluso en negocios cuya facturación está fuertemente relacionada con las estaciones del año.

Un ejemplo podrían ser las cadenas hoteleras, las cuales facturan mucho más en los periodos de vacaciones como agosto, que en meses de temporada baja como noviembre. Pero la información del carácter cíclico de sus ingresos es conocida por todos sus inversores, por lo que el precio de la acción refleja el valor esperado, por parte del mercado, de los beneficios futuros que obtenga la empresa teniendo en cuenta esta correlación entre facturación y el periodo del año.

A pesar de esa correlación, el precio se muestra independiente, por lo que se cree que la codificación del año no aportaría valor, razón por la cual no se desarrolló ningún modelo en ese ámbito.

El modelo global que agrupe tanto al del día de la semana como al que particiona por hora tendrá como entradas las siguientes:

- Vector x con los datos normalizados.
- Codificación del día de la semana.
- Valor de la función de agregación de los puntos del día.

En primer lugar, se trabajó en el diseño y programación del modelo base centrado en los datos de cada media hora. Esto supuso un reto porque el índice temporal ahora sería variable, por lo que se debieron realizar una serie de ajustes.

El modelo se entrenó utilizando una red MLP que tuvo una estructura similar a la ya utilizada. Para realizar este entrenamiento se buscó optimizar el hiperparámetro del número de neuronas en la capa oculta para así encontrar la combinación adecuada que produjese los mejores resultados.

A continuación, se muestran las gráficas para múltiples entrenamientos.

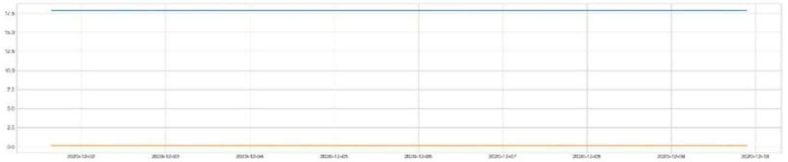

Ilustración 32: Predicciones para el primer entrenamiento, de elaboración propia

Ilustración 33: Predicciones para el segundo entrenamiento, de elaboración propia

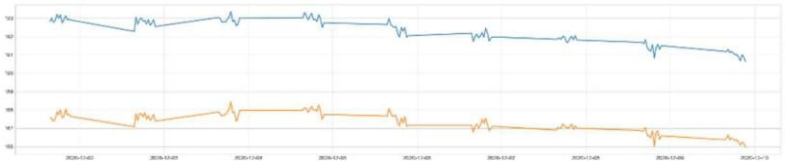

Ilustración 34: Predicciones para el último entrenamiento, de elaboración propia

Con la excepción del modelo más sencillo (0 neuronas en la capa oculta), todos los modelos mostraron unos resultados muy cercanos entre sí, con un MSE de 2.05 en Test y de 1.95 en entrenamiento.

Es importante remarcar que la forma de las gráficas tan anómala en comparación con la de modelos anteriores se debe a los ajustes requeridos para ajustarlo a la hora. Ahora hay zonas en las que no hay datos porque la bolsa no operaba en esas horas.

Tras este primer paso, se procedió a introducir en el modelo la información del día de la semana usando codificación one-hot.

El dataframe resultante del que se extrajeron las entradas y salidas del modelo es el que se muestra a continuación.

Date	Open	High	Low	Close	Volume	DayWeek	Mon	Tue	Wed	Thu	Fri
2021-02-01 15:30:00	242.52	268.8836	238.5000	260.0000	1.000000	0	1	0	0	0	0
2021-02-01 16:00:00	259.59	276.0000	258.3797	272.1999	0.505599	0	1	0	0	0	0
2021-02-01 16:30:00	271.20	275.0000	265.5000	267.6900	0.375499	0	1	0	0	0	0
2021-02-01 17:00:00	267.68	270.6000	264.4500	270.0850	0.172656	0	1	0	0	0	0
2021-02-01 17:30:00	270.03	270.0300	263.1800	266.4900	0.156574	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
2021-04-16 19:30:00	219.84	220.8500	219.0000	219.7800	0.037466	4	0	0	0	0	1
2021-04-16 20:00:00	219.59	223.5000	219.5670	223.1500	0.069367	4	0	0	0	0	1
2021-04-16 20:30:00	223.02	226.6700	221.5300	225.6800	0.101124	4	0	0	0	0	1
2021-04-16 21:00:00	225.70	225.9100	221.8200	223.8800	0.083174	4	0	0	0	0	1
2021-04-16 21:30:00	224.00	228.1000	223.6700	227.3500	0.202995	4	0	0	0	0	1

Tabla 7: Dataframe con día de la semana en formato onehot encoding, de elaboración propia

Sobre este modelo con las nuevas entradas se iteró de forma similar al caso anterior, buscando el óptimo número de neuronas en la capa oculta. Los resultados fueron curiosamente peores para todos los casos.

Las predicciones del mejor modelo son las siguientes, MSE = 2.795 en test y MSE = 2.672 en entrenamiento.

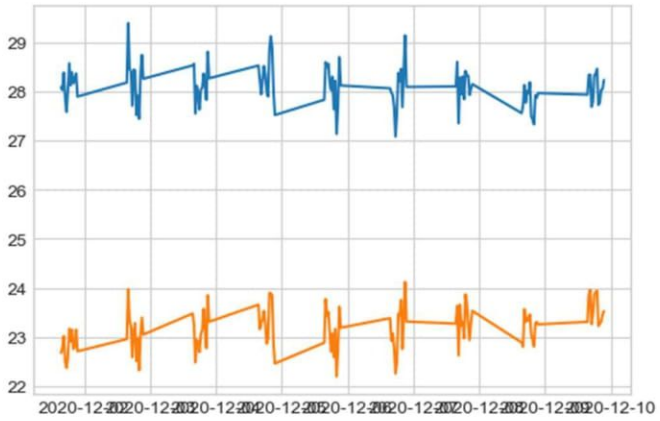

Ilustración 35: Predicciones del mejor modelo Dudek, de elaboración propia

Es necesario analizar qué razones puede haber para que se hayan producido peores resultados a pesar de ser en principio más rico en datos.

La única explicación por la que un modelo con más entradas no se haya vuelto más preciso es que la complejidad no es la adecuada o que la información nueva que se ha incluido no tenga de valor.

La idea de la complejidad es posible y vale la pena ser analizada, pero eso no explica por qué el máximo de precisión del modelo se ha producido en 9 neuronas de la capa oculta. Si la complejidad no fuese suficiente, entonces el óptimo encontrado debería estar en el máximo de neuronas o, al menos, en un punto cercano al máximo, pero este punto está por debajo de la mediana de valores posibles analizados, no tiene sentido.

Otra posibilidad de que el problema sea la complejidad es que no haya que crecer verticalmente (número de neuronas), sino que horizontalmente, a través de la inclusión de nuevas capas ocultas.

La aplicación de Deep Learning no tiene sentido porque se pretende que se comparen las predicciones del modelo con las del iMLP, y este no posee esa escalabilidad horizontal en el número de capas, como ya se ha explicado.

El siguiente paso fue proceder a entrenar el iMLP con el nuevo conjunto de datos y ver qué predicciones tendría.

La principal dificultad en este proceso está, de nuevo, en la complejidad y los tiempos de procesamiento. El nuevo horizonte de predicción que se escogió para este conjunto de datos fue de 9 días, pero, como los datos están dispuestos con una frecuencia de 1 registro cada media hora, esto son 13 registros por día, por lo que, en términos de datos, el horizonte se extendió a 117 intervalos.

Las consecuencias de ese cambio en la complejidad supusieron que el entrenamiento a través de la búsqueda de hiperparámetros ya no fuese posible. Cada una de las iteraciones del bucle de entrenamiento tardaba aproximadamente 2 horas y 30 minutos, por lo que probar todas las combinaciones, que eran iguales a un total de (5 valores de delt_t) x (30 valores del número de neuronas de la capa oculta) x (3 escenarios) = 450 ejecuciones.

Esto serían un total de 2.5 x 450 = 1125 horas o, lo que es lo mismo, 46.87 días.

Por esta razón, resultó imposible hacer el entrenamiento y, en su lugar, se tomó una combinación que había funcionado en modelos anteriores. Las predicciones de ese modelo fueron las siguientes.

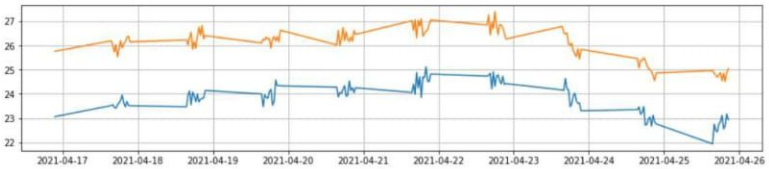

Ilustración 36: Predicciones iMLP sobre el conjunto de datos, de elaboración propia

El modelo sigue caracterizándose por sus variaciones en el corto plazo y la captación de ese tipo de tendencias como contraposición a los otros modelos, que presentaban una anchura de intervalo mucho más estable y estaban focalizados en el largo plazo.

X. ANÁLISIS DE RESULTADOS Y CONCLUSIONES

Los modelos desarrollados han mostrado una capacidad de predicción de los centros y radios que por lo general ha estado caracterizada por la carencia de variabilidad en los intervalos predichos.

En contraposición, el iMLP muestra unos resultados que parecen buscar predecir la componente especulativa de la acción, que al fin y al cabo es la que más interesa en el mundo del trading, con mucha más precisión, sin buscar tendencias alcistas o bajistas a largo plazo.

El modelo de Dudek desarrollado no logró los resultados deseados, probablemente debido a la falta de profundidad del modelo o a la calidad de la información introducida. La información del día de la semana aportaría a priori valor al modelo porque existe una leve tendencia de bajada de la demanda cerca de los fines de semana y una subida al principio de la semana.

Sin embargo, este valor que aporta la información no es significativo. La razón por la que aportó valor en [9] es probablemente el conjunto de datos.

Como ya se explicó, el conjunto de datos que se utilizó en ese trabajo buscaba predecir los valores de la energía, para lo que la información del día de la semana aportaba muchísimo más valor al existir picos significativos en la demanda durante los fines de semana debido a que los hogares están más ocupados.

El estudio de la independencia entre centros y radios ha concluido que las variables no presentan ninguna estructura ni patrón, por lo que se podrían considerar independientes si no se tuviesen en cuenta las observaciones anómalas o la ausencia de puntos en determinadas franjas de valores. Esto plantea más preguntas que respuestas. El objetivo era concluir que existía una relación entre centros y radios que pudiese explicar la idoneidad de la red iMLP.

Como no se ha logrado encontrar esa relación, se concluyó que la aplicación de modelos independientes debería ser

igualmente válida y de calidad similar, pero a pesar de ello, el modelo iMLP fue claramente mejor en predecir esa componente especulativa.

De todos los modelos desarrollados para competir con el iMLP, el que dio mejores resultados fue el más sencillo, la red MLP doble con una única capa oculta, y el número de neuronas no mostró cambios significativos en la precisión del modelo (aunque requeriría un análisis).

XI. TRABAJOS FUTUROS

En los modelos entrenados con el iMLP se calculan los estadísticos mencionados anteriormente: estos son el iARV, la u de Theill, el MSE, el ER y el CR.

Cada uno de estos estadísticos aporta una métrica que puede utilizarse para valorar la calidad del modelo, pero existe un problema una vez se tienen los resultados. El problema está en que no hay un criterio que defina objetivamente cuál es el mejor modelo. Se puede tomar el que tenga mejor iARV, CR, u de Theill, etc., pero no el mejor en todos los estadísticos.

Asimismo, hay otro inconveniente aun cuando se da el caso de que solo haya un estadístico para valorar el modelo. Para cada estadístico hay dos medidas y se valoran dos cosas: que el valor del estadístico en test sea lo más óptimo posible y que la diferencia entre el estadístico calculado en test y el calculado en entrenamiento sea tan baja como sea posible.

A raíz de este problema se propuso una fórmula para valorar cada estadístico.

$$f(S) = 0.5 * (S_{train} + S_{test}) - |S_{train} - S_{test}|^2$$

Donde S es el estadístico que se desea analizar para valorar sus resultados tanto en entrenamiento como en test. El primer componente de la función se refiere a la media entre los resultados en entrenamiento y test, que representa el rendimiento del modelo esperado tanto en entrenamiento como en test según el estadístico S.

La segunda parte de la fórmula contiene la diferencia de resultados en ambos conjuntos, que debe ser mínima porque, en caso contrario, podría implicar sobreentrenamiento en el modelo. Esta diferencia se ve amplificada por una función cuadrática porque es preferible que el modelo no esté sobreentrenado a que la medida del rendimiento medio del modelo sea mayor, caso en el que el modelo empieza a perder valor.

Cabe destacar que los valores S_{train} y S_{test} representan el rendimiento del modelo según el estadístico. Esto es el valor del estadístico S para el conjunto en concreto sobre el que se esté evaluando el modelo, pero adaptando su signo de forma que si un estadístico bajo implica mejores resultados se multiplicará el estadístico por (-1) y se dejará sin alterar en el caso contrario.

Tras probar esta fórmula se comprobó que hubo una suposición errónea a la hora de calcularla, la idea del término elevado al cuadrado es la de aumentar la importancia de ese término, pero esto no tiene sentido para valores menores a 1 porque los reduce, y se da la circunstancia de que todas las diferencias de todos los estadísticos son menores que la unidad.

Por esta razón se modificó la fórmula a otra donde el término, en lugar de estar elevado al cuadrado, fuese la raíz cuadrada. La razón es que es la operación inversa, reduce la entrada cuando es mayor a la unidad y la incrementa en el caso contrario, que es el que siempre se da para este problema.

$$f(S) = 0.5 * (S_{train} + S_{test}) - \sqrt{|S_{train} - S_{test}|}$$

Otra alternativa en la que se pensó fue la de hacer el primer término cuadrático y el segundo lineal, pero esto no funcionaba porque el primer término sí que podía tomar valores mayores que 1, por lo que optó por la nueva versión.

Una vez tenemos una forma de medir la calidad del modelo según un estadístico, queda por resolver el problema de agrupar a todos los estadísticos, por lo que se podría utilizar la siguiente fórmula.

$$F(\Omega) = \sum_{S \in \Omega} f(S) = \sum_{S \in \Omega} \left(0.5 * (S_{train} + S_{test}) - \sqrt{|S_{train} - S_{test}|}\right)$$

Donde Ω es el conjunto de todos los estadísticos utilizados para valorar el modelo. Debido a que en principio ningún estadístico toma mayor importancia que otro, no tendría sentido aplicar pesos en el cálculo del sumatorio.

De esta forma se obtiene una manera de agrupar todos los términos de los distintos estadísticos en la función F. Para tomar un ranking con los mejores modelos basta con calcular la función y buscar maximizarla.

	Unnamed: 0	delt_t	Nhidden	iARV_Train	iARV_Test	iuTheill_Train	iuTheill_Test	Score_iARV	Score_iuTheill
0	5	5	6	1.397897	1.090440	3.088576	5.049870	-1.798657	-5.469685
1	6	5	7	1.126499	1.113992	5.563670	6.145020	-1.232083	-6.616808
2	7	5	8	1.248549	1.142087	3.795826	5.584439	-1.521604	-6.027522
3	9	5	10	1.320953	1.165296	3.412219	5.635781	-1.637658	-6.015161
4	12	5	13	1.290203	1.195224	3.579906	4.692311	-1.550900	-5.190815
5	13	5	14	1.339436	1.219520	3.409205	4.460292	-1.625768	-4.959973
6	8	5	9	1.373418	1.236385	3.228595	3.974273	-1.675081	-4.464961
7	10	5	11	1.349289	1.241158	3.244438	3.945976	-1.624056	-4.432785
8	3	5	4	1.381195	1.272978	3.161209	3.793291	-1.656050	-4.272285
9	4	5	5	1.373398	1.293128	3.228182	3.443021	-1.616583	-3.799109
10	11	5	12	1.397875	1.363140	3.138237	3.276572	-1.566881	-3.579339
11	1	5	2	1.408573	1.364502	3.196013	3.293607	-1.596468	-3.557210
12	0	5	1	1.398314	1.392252	3.200310	3.140253	-1.473139	-3.415347
13	2	5	3	1.407026	1.398250	3.197879	3.124167	-1.496316	-3.432523

Tabla 8: Dataframe con resultados de múltiples modelos, de elaboración propia

Cabe destacar que la baja precisión reflejada en los valores de los estadísticos calculados se debe a que se están estimando con un horizonte de predicción alto, de unos 15 días, por lo que, sobre todo en un conjunto de datos tan cambiante e imprevisible como son los mercados financieros, la calidad de las predicciones se ve seriamente perjudicada.

Debido a las altas diferencias en las puntuaciones de los estadísticos, es recomendable realizar una normalización de los parámetros una vez hayan pasado por las funciones f de puntuación de estadísticos.

Dicha normalización podría ser como la propuesta en [10], que utiliza la siguiente fórmula.

$$S' = \frac{S - S_{min}}{S_{max} - S_{min}}$$

Esta es una línea de trabajo que posiblemente valdría la pena explorar para trabajos posteriores.

REFERENCIAS

[1] Teja Reddy, B., & J.C., U. (2021). *Prediction of Stock Market using Stochastic Neural Networks*.

[2] Volodymyr Turchenko1 , Patrizia Beraldi2 , Francesco De Simone2 and Lucio Grandinetti2 1 Research Institute of Intelligent Computer Systems ,"Short-term Stock Price Using MLP in Moving Simulation Mode", Ternopil National Economic University.

[3] San Roque, A. M., Maté, C., Arroyo, J., & Sarabia, Á. (2007). iMLP: Applying multi-layer perceptrons to interval-valued data. *Neural Processing Letters*, *25*(2), 157-169.

[4] Majid Vafaeipour, Omid Rahbari, Marc A. Rosen, Farivar Fazelpour, Pooyandeh Ansarirad, Application of sliding window technique for prediction of wind velocity time series., Springerlink.com

[5] All About the Regression Learner App. (2021). Retrieved 8 June 2021, from https://explore.mathworks.com/all-about-regression-learner-app

[6] All About Model Validation. (2021). Retrieved 8 June 2021, from https://explore.mathworks.com/all-about-model-validation

[7] Jiménez del Campo, L. (2020). Neural networks for crisp and interval-valued data. Application to forecasting financial markets. TFG. Universidad Pontificia Comillas

[8] Laboissiere, L. A., Fernandes, R. A. S., & Lage, G. G. (2015). Maximum and minimum stock price forecasting of Brazilian power distribution companies based on artificial neural networks. *Applied Soft Computing*, *35*, 66–74. https://doi.org/10.1016/j.asoc.2015.06.005

[9] Dudek, G. (2019). Multilayer perceptron for short-term load forecasting: from global to local approach. *Neural Computing and Applications*, *32*(8), 3695–3707. https://doi.org/10.1007/s00521-019-04130-y

[10] Wang, Y., Wang, L., Yang, F., Di, W., & Chang, Q. (2021). Advantages of direct input-to-output connections in neural networks: The Elman network for stock index forecasting. Information Sciences, 547, 1066–1079. https://doi.org/10.1016/j.ins.2020.09.031

[11] Mañas, A. M. (n.d.). Notas sobre pronóstico del flujo de tráfico en la ciudad de Madrid. Capítulo 8 Métodos basados en Deep Learning. Arquitectura RNN expandida [Figura] https://bookdown.org/amanas/traficomadrid/m%C3%A9todos-basados-en-deep-learning.html.

[12] Crone, S. F., & Kourentzes, N. (2010). Feature selection for time series prediction–A combined filter and wrapper approach for neural networks. Neurocomputing, 73(10-12), 1923-1936.

[13] Puyol Lombos, L. (2021). Neural Networks to forecast interval time series. Applications to stock markets. TFM. Universidad Pontificia Comillas