MASTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO FIN DE MASTER

Neural Networks to forecast interval time series. Applications to stock markets

Autor: Luis Puyol Lombos

Director: Carlos Maté Jiménez

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Neural Networks system to forecast interval time series. Applications to stock markets

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2020/21 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Luis Puyol Lombos Fecha: 19/07/21

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Carlos Maté Jiménez Fecha: 19/07/2021

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Luis Puyol Lamban

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Neural Networks to forecast interval time series applications to stock markets que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducirla y almacenarla en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna [illegible] para que los usuarios hagan un uso posterior de las obras no conforme con la [illegible] posterior, más allá de la copia privada, requerirá que se cite la fuente [illegible] que no se obtenga beneficio comercial, y que no se realicen obras [illegible]
- La Universidad no revisará [illegible] y en todo caso permanecerá bajo la responsabilidad exclusiva [illegible] ejercitar acciones legales en nombre del autor en el supuesto de [illegible] propiedad intelectual derivados del depósito y archivo de las obras. El [illegible] reclamación frente a la Universidad por las formas no ajustadas a [illegible] los usuarios hagan uso de las obras.
- La Universidad adoptará [illegible] para la preservación de la obra en un futuro.
- La Universidad se reserva [illegible] la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 19 de julio de 2021

ACEPTA

Fdo. Lucía Puyal Lombor

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

MASTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO FIN DE MASTER

Neural Networks to forecast interval time series. Applications to stock markets

Autor: Luis Puyol Lombos

Director: Carlos Maté Jiménez

Madrid

Agradecimientos

A Jim Simons, por ser un modelo inspirador de toda una generación y un ejemplo de que nada es imposible si se le dedica tiempo y esfuerzo.

NEURAL NETWORKS SYSTEM TO FORECAST INTERVAL TIME SERIES. APPLICATIONS TO STOCK MARKETS

Autor: Puyol Lombos, Luis**.**
Director: Maté Jiménez, Carlos.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

A lo largo de este proyecto se han desarrollado múltiples modelos con el objetivo de predecir valores de varias acciones. El principal modelo en el que se ha profundizado fue el iMLP, del cual se han estudiado diversas arquitecturas. Los resultados han mostrado una mayor precisión en la predicción del corto plazo en contraste a seguir la tendencia a largo plazo.

Palabras clave: iMLP, redes neuronales, predicción de acciones, LSTM, MLP

1. Introducción

Las nuevas tecnologías de redes neuronales que ha aportado el Machine Learning suponen un importante avance, un cambio de paradigma que lleva trayendo innovaciones a lo largo de los últimos años. Uno de los campos en los cuales la aplicación fue más directa fue el de los mercados financieros, particularmente en la predicción de valores futuros de acciones y otros productos financieros.

La aplicación de modelos para la predicción del mercado de valores ha sido uno de los principales objetivos que se han estudiado desde prácticamente su creación. Para poder realizarla correctamente se han de seguir una serie de pasos.

En primer lugar, se ha de tener lugar una recolección de datos, independientemente de lo buena que sea una lógica de un modelo, esta pierde utilidad si no se aplican los datos adecuados. Algunas formas de conseguirlos serían mediante APIs, comprándolos, midiéndolos por medio de un sistema de medida especializado al problema como un sensor, a través del diseño de una aplicación de web scraping, etc.

El siguiente paso consiste en limpiar los datos, en eliminar los valores nulos que hayan podido aparecer por cualquier error o circunstancia. La razón es que, si se introdujesen en un modelo durante el entrenamiento, no sabemos cómo podría afectar, es posible que reproduzca una observación anómala que sesgue el modelo o cualquier otro tipo de fallo no deseable.

En algunos casos también es necesario preparar los datos antes de su introducción en el modelo para construir estadísticos que se sospeche que puedan ser de especial interés para el modelo en cuestión. Un ejemplo podría ser introducir la media de un vector de muestras para poder hacer un modelo que calcule la desviación típica de los datos, cálculo para el cual es indispensable la media.

El último paso antes del entrenamiento es el escalado de los datos. La razón es que los modelos están preparados para trabajar con datos normalizados y son más eficientes.

Este proceso no supone la pérdida de nada de información, por lo que no tiene ninguna desventaja, es un procedimiento habitual.

Una vez se tienen los datos con el formato adecuado y sin NAs se procede a entrenar el modelo. Para ello se seleccionan 3 grupos en los que se subdivide el conjunto de datos. El conjunto de entrenamiento se utiliza para ir probando variantes del modelo de forma que para una entrada se obtiene su salida y se compara con la salida objetivo. A partir de ahí se elabora una función de error que se intentará minimizar, si el error es alto se ajustarán los pesos del modelo para que, en las próximas salidas, el error sea menor.

El principal modelo sobre el que se ha trabajado es el iMLP, el cual es una variante del perceptrón multicapa, pero con la cualidad especial de que tiene como entradas y salidas datos de intervalo en lugar de datos crisp. Se trata de un tipo de modelo relativamente novedoso sobre el que aún no se han realizado tantos estudios ni se ha implantado al nivel de otros como el MLP o el LSTM. Este modelo se ha utilizado en trabajos como el desarrollado en [4], que corresponde a una revista que es Q1.

2. Definición del proyecto

El primer objetivo fue 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 buscó 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.

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

Con respecto a los modelos adicionales a crear, se elaboró una lista y se fue uno por uno buscando cuáles son los parámetros idóneos para obtener la mayor precisión posible. Cabe destacar que se utilizaron ventanas de tiempo sobre los modelos. La aplicación de estas ventanas temporales al trabajo con redes neuronales ha mostrado resultados prometedores, como se muestra en [1].

También se elaboró un modelo que agrupaba las salidas de varios modelos entrenados con acciones distintas de forma que se introdujeron en una red iMLP adicional para predecir las acciones con información de la propia acción, así como la de las otras, que pertenecen a un mismo mercado. Esto se basó en el trabajo referenciado en [3]

Asimismo, se realizaron estudios de independencia para los centros y radios de los puntos con el objetivo de descubrir si se podrían hacer las predicciones de forma separada a través de 2 modelos independientes.

3. Descripción del modelo/sistema/herramienta

Dada la naturaleza de las salidas en forma de intervalo y no de un único valor del iMLP, se tomaron unos estadísticos alternativos a los más comunes como el MSE para así poder valorar mejor los resultados.

$$iARV = \frac{\sum(\widehat{F_L}-F_L)^2+\sum(\widehat{F_U}-F_U)^2}{\sum(\widehat{F_{L,t}}-F_{L,t})^2+\sum(\widehat{F_{U,t}}-F_{U,t})^2} \quad (1)$$

$$U\ de\ Theill = \frac{\sum(\widehat{F_L}-F_L)^2+\sum(\widehat{F_U}-F_U)^2}{\sum(F_{L,t}-F_{L,t-1})^2+\sum(F_{U,t}-F_{U,t-1})^2} \quad (2)$$

$$CR = \frac{\mathrm{Min}(\widehat{F_U},F_U)-\mathrm{Max}(\widehat{F_L},F_L)}{F_U-F_L} \quad (3)$$

$$ER = \frac{\mathrm{Min}(\widehat{F_U},F_U)-\mathrm{Max}(\widehat{F_L},F_L)}{\widehat{F_U}-\widehat{F_L}} \quad (4)$$

El modelo principal mostraba la siguiente arquitectura.

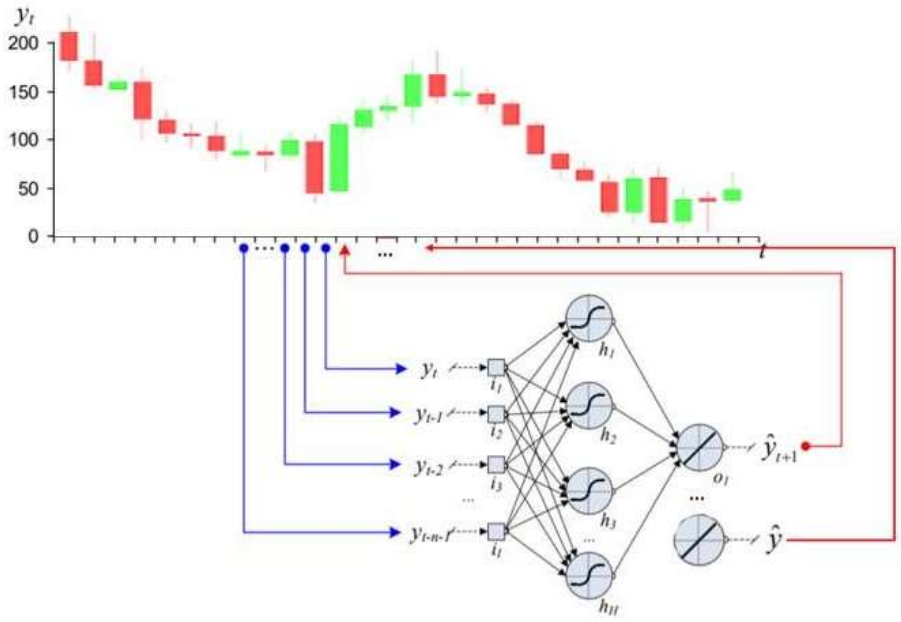


Ilustración 1: Arquitectura final del modelo. Con base en [2]

Sobre el modelo base se hicieron múltiples modificaciones en las entradas para desarrollar nuevas variantes con el iMLP:

- Introducción de una variable de tendencia esperada (alzista, bajista o lateral).
- Modelo con información del día de la semana.

- Modelo con varias acciones de un mismo índice.

4. 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.
- El modelo iMLP mostró una precisión mayor en el corto plazo en contraposición a otros como el MLP.

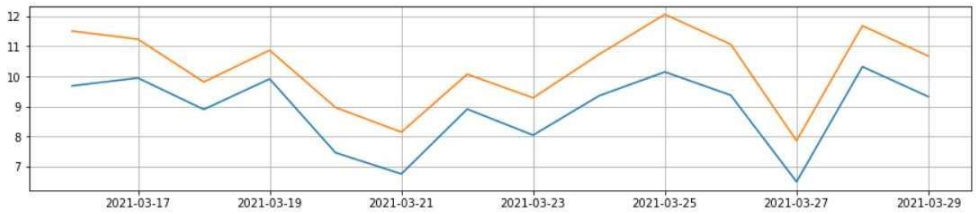


Ilustración 2: Predicciones máximo y mínimo del modelo iMLP, de elaboración propia

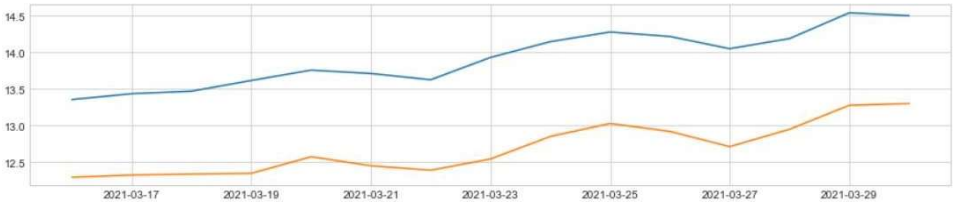


Ilustración 3: Predicciones MLP base, de elaboración propia

- Los centros y radios no mostraron ningún tipo de estructura, por lo que serían independientes. También se estudiaron dependencias y correlaciones entre centros y radios con varias separaciones temporales, llegando a la misma conclusión.

Para futuros trabajos sería posible estudiar la posibilidad de desarrollar un código iMLP que permitiese modelos con mayor número de capas ocultas. También se podría desarrollar un estadístico único que agregue otros estadísticos para que así sea posible valorar resultados con una única métrica resumida. Por último, cabe destacar la importancia de la información de los mercados de renta fija, que se complementan con los de renta variable y que podrían aportar valor si se incluyera en un modelo.

5. Referencias

[1] 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

[2] 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.

[3] FISTER, Dušan; PERC, Matjaž; JAGRIČ, Timotej. Two robust long short-term memory frameworks for trading stocks. Applied Intelligence, 2021, p. 1-19.

[4] Maté, C., & Jiménez, L. (2021). Forecasting exchange rates with the iMLP: New empirical insight on one multi-layer perceptron for interval time series (ITS). Engineering Applications of Artificial Intelligence, 104, 104358.

NEURAL NETWORKS SYSTEM TO FORECAST INTERVAL TIME SERIES. APPLICATIONS TO STOCK MARKETS

Author: Puyol Lombos, Luis**.**
Supervisor: Maté Jiménez, Carlos.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

Throughout this project, multiple models have been developed in order to predict values of various stocks. The main model that has been studied in depth was the iMLP, of which various architectures have been studied. The results have shown greater precision in predicting the short term in contrast to following the long term trend.

Keywords: iMLP, neural network, stock prediction, LSTM, MLP

1. Introduction

The new neural network technologies that Machine Learning has provided represent an important advance, a paradigm shift that has brought innovations over the last few years. One of the fields in which the application was most direct was the financial markets, particularly in the prediction of future values of stocks and other financial products.

The application of models for the prediction of the stock market has been one of the main objectives that have been studied since practically its creation. To be able to do it correctly, a series of steps must be followed.

In the first place, a data collection has to take place, regardless of how good a model logic is, it loses usefulness if the appropriate data is not applied. Some ways to achieve them would be through APIs, buying them, measuring them through a specialized measurement system for the problem such as a sensor, through the design of a web scraping application, etc.

The next step is to clean the data, to eliminate the null values that may have appeared due to any error or circumstance. The reason is that, if they were introduced into a model during training, we do not know how it could affect, it is possible that it reproduces an anomalous observation that skews the model or some other type of undesirable failure.

In some cases it is also necessary to prepare the data before its introduction into the model to construct statistics that are suspected of being of special interest to the model in question. An example could be to introduce the mean of a vector of samples to be able to make a model that calculates the standard deviation of the data, a calculation for which the mean is essential.

The last step before training is scaling the data. The reason is that the models are ready to work with normalized data and are more efficient. This process does not mean the loss of any information, so it does not have any disadvantage, it is a common procedure.

Once the data is in the appropriate format and without NAs, the model is trained. For this, 3 groups are selected into which the data set is subdivided. The training set is used to test variants of the model in such a way that its output is obtained for an input and it is compared with the target output. From there, an error function is elaborated that we will try to minimize. If the error is high, the model weights will be adjusted so that, in the next outputs, the error is lower.

The main model that has been worked on is the iMLP, which is a variant of the multilayer perceptron, but with the special quality that it has interval data as inputs and outputs instead of crisp data. It is a relatively new type of model on which so many studies have not yet been carried out nor has it been implemented at the level of others such as the MLP or the LSTM.This model has been used in several projects such as the article [4], which is from a Q1 magazine.

2. Project definition

The first objective was to create an iMLP-based model adapted to Python to facilitate future work and to make it easier to apply and modify.

It was also sought to establish a quality metric of a model in order to be able to vary the basic parameters of the network and find the best configuration to make the model more accurate.

Create other multiple models from different methodologies when building a neural network. It is expected that the models ranged from the most basic to more complex ones like LSTM.

Finally, we will try to compare the results of all the models and draw conclusions about which ones have the most desirable qualities in relation to the mean square error and other precision parameters.

To meet the first objective, previous works of adapting the iMLP code to languages such as Matlab were studied and they were used as a basis to see what errors to avoid and what steps to follow.

With regard to the additional models to create, a list was drawn up and one by one went looking for the ideal parameters to obtain the highest possible precision. It should be noted that the time window methodology was used on the models. The application of time windows to work with neural networks has shown promising results, as shown in [1].

A model was also developed that grouped the outputs of several models trained with different actions in such a way that they were entered into an additional iMLP network to predict actions with information from the action itself as well as from the others, which belong to the same market. This was based on work referenced in [3]

Likewise, independence studies were carried out for the centers and radii of the points in order to discover if the predictions could be made separately through 2 independent models.

3. Model description

Given the nature of the outputs in the form of an interval and not a single iMLP value, alternative statistics to the most common ones, such as the MSE, were taken in order to better assess the results.

$$iARV = \frac{\sum(\widehat{F_L}-F_L)^2+\sum(\widehat{F_U}-F_U)^2}{\sum(F_{L,t}-F_{L,t-1})^2+\sum(F_{U,t}-F_{U,t-1})^2} \quad (1)$$

$$U\ de\ Theill = \frac{\sum(\widehat{F_L}-F_L)^2+\sum(\widehat{F_U}-F_U)^2}{\sum(F_{L,t}-F_{L,t-1})^2+\sum(F_{U,t}-F_{U,t-1})^2} \quad (2)$$

$$CR = \frac{Min(\widehat{F_U},F_U)-Max(\widehat{F_L},F_L)}{F_U-F_L} \quad (3)$$

$$ER = \frac{Min(\widehat{F_U},F_U)-Max(\widehat{F_L},F_L)}{\widehat{F_U}-\widehat{F_L}} \quad (4)$$

The main model showed the following architecture.

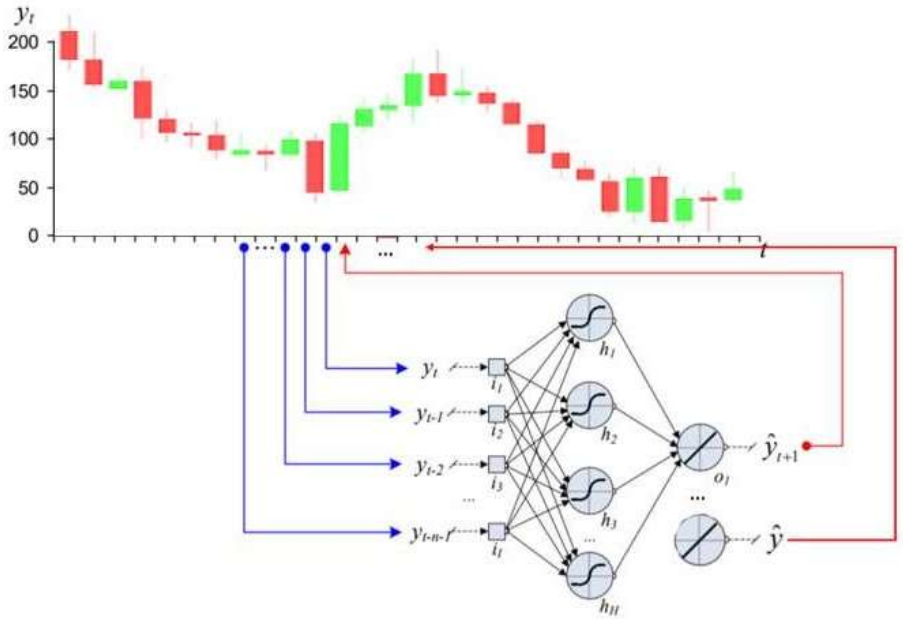


Figure 1: Final architecture. Based on [2]

On the base model, multiple modifications were made to the inputs to develop new variants with the iMLP:

- Introduction of an expected trend variable (bullish, bearish or lateral).
- A Model with information on the day of the week.
- A Model with several shares of the same index.

4. Results and conclusions

• The models developed have shown a prediction capacity of the centers and radii that has generally been characterized by the lack of variability in the predicted intervals.

• The iMLP model showed greater precision in the short term compared to others such as the MLP.

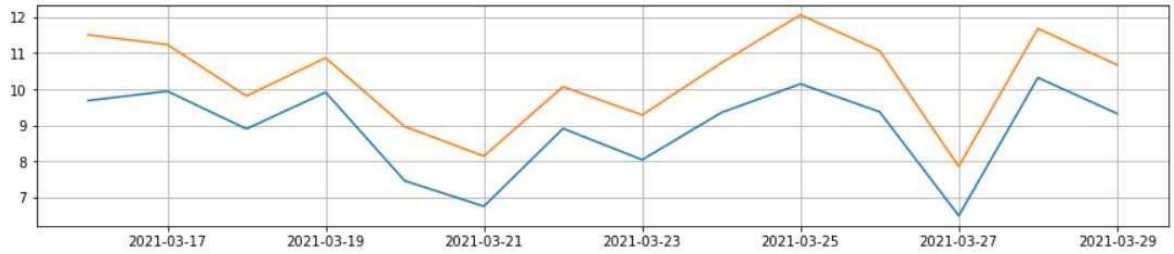


Figure 2: Predictions from iMLP model, made during project

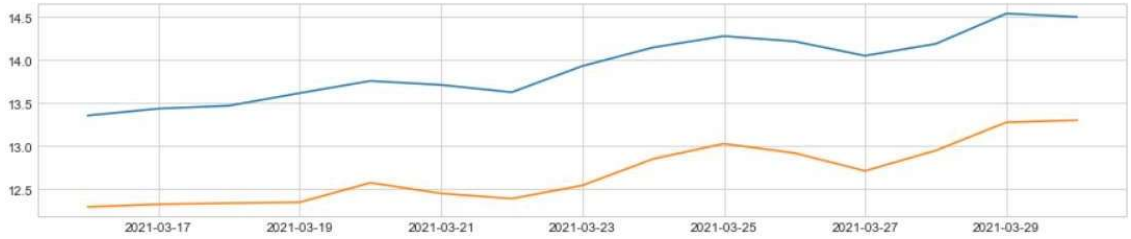


Figure 3: Predictions from basic MLP model, made during project

- The centres and radios did not show any type of structure, so they would be independent. Dependencies and correlations between centres and radii with various temporal separations were also studied, reaching the same conclusion.

For future work it would be possible to study the possibility of developing an iMLP code that would allow models with a greater number of hidden layers. A single statistic could also be developed that aggregates other statistics so that results can be assessed with a single summary metric. Lastly, it is worth highlighting the importance of information from fixed income markets, which are complemented by equity markets and which could add value if included in a model.

5. References

[1] 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

[2] 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.

[3] FISTER, Dušan; PERC, Matjaž; JAGRIČ, Timotej. Two robust long short-term memory frameworks for trading stocks. Applied Intelligence, 2021, p. 1-19.

[4] Maté, C., & Jiménez, L. (2021). Forecasting exchange rates with the iMLP: New empirical insight on one multi-layer perceptron for interval time series (ITS). Engineering Applications of Artificial Intelligence, 104, 104358.

Capítulo 1. INTRODUCCIÓN

Las nuevas tecnologías de redes neuronales que ha aportado el Machine Learning suponen un importante avance, un cambio de paradigma que lleva trayendo innovaciones a lo largo de los últimos años. Uno de los campos en los cuales la aplicación fue más directa fue el de los mercados financieros, particularmente en la predicción de valores futuros de acciones y otros productos financieros.

La razón por la que se desarrolló tan rápidamente en este campo es el alto potencial lucrativo que podría tener un modelo de predicción, con mejor calidad y resultados, al aplicarse sobre el mercado. A esto también le podemos sumar la alta competitividad y la lucha constante por innovar características del sector como factores propicios para que sea un caldo de cultivo idóneo para la rápida adaptación de todas las tecnologías que puedan aportar algún tipo de valor.

La ventaja que aporta la aplicación de las tecnologías de redes neuronales es que hace que los criterios con los que se toma una decisión puedan ser objetivos. Antes de que se pudieran automatizar, por medio de estas metodologías, las tareas, todo se basaba en confiar en una persona o equipo de personas para que tomasen las decisiones según sus criterios personales, adquiridos mediante la experiencia o educación.

Este sistema no es óptimo dada su alta sensibilidad al error humano, que puede generar altas pérdidas para sus clientes o incluso contribuir a crear corrientes especulativas al hacer estimaciones erróneas o sesgadas en el valor de una acción, lo que hace que el mercado valore erróneamente una empresa, lo que, a su vez, hace que las compañías comparables bajen su valor al bajar su media de mercado sobre múltiplos usados para la valoración de empresas como puede ser el EV/EBITDA, lo que puede causar un efecto acumulativo que acabe desembocando en consecuencias desastrosas para los inversores.

Otra razón por la que estas tecnologías tienen un desarrollo cada vez más fácil es que no dependen del talento a la hora de idear un algoritmo lógico para resolver un problema, sino que están fuertemente ligadas a los datos, y con la digitalización de muchos ámbitos, así como el auge de empresas que se dedican a generarlos, se está produciendo un alto crecimiento del volumen de información que se genera y registra. Esto puede dar lugar a muchas y nuevas oportunidades para usar datos adecuadamente y generar valor. El sector financiero no se queda atrás, y ya se han visto compañías como Bloomberg que recopilan los datos más actualizados para dar servicio a posibles aplicaciones de estos.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Describir las tecnologías, protocolos, herramientas específicas, etc. que se vayan a tratar durante el proyecto para facilitar su lectura y comprensión.

Hablar de Java no procede aquí porque todo el mundo sabe lo que es, pero si en el proyecto hablo continuamente del protocolo Baseband, debo especificar en este capítulo qué es y para qué sirve.

Para desarrollar el proyecto será necesario aplicar tecnologías de Machine Learning. Por esta razón uno de los primeros pasos fue realizar un análisis exploratorio de los distintos paquetes y funcionalidades dentro de lenguajes de programación como Python y Matlab que fuesen de utilidad para el propósito del proyecto.

Lo que se buscaba eran modelos de redes neuronales como el perceptrón multicapa, que fuesen fáciles de implementar y computacionalmente no muy costosos.

Python

Dentro de Python existen un conjunto de funciones ampliamente extendidas para el uso de redes neuronales, se trata de Tensorflow, de Google. Dentro de este paquete existe la función Dense, que se puede utilizar para generar redes MLP. Para esto será necesario adaptar los datos y realizar algunos ajustes a su configuración.

Arguments	
`units`	Positive integer, dimensionality of the output space.
`activation`	Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: `a(x) = x`).
`use_bias`	Boolean, whether the layer uses a bias vector.
`kernel_initializer`	Initializer for the `kernel` weights matrix.
`bias_initializer`	Initializer for the bias vector.
`kernel_regularizer`	Regularizer function applied to the `kernel` weights matrix.
`bias_regularizer`	Regularizer function applied to the bias vector.
`activity_regularizer`	Regularizer function applied to the output of the layer (its "activation").
`kernel_constraint`	Constraint function applied to the `kernel` weights matrix.
`bias_constraint`	Constraint function applied to the bias vector.

Ilustración 1: Documentación función Dense. Extraida de la documentación de Tensorflow

Un ejemplo de código en el que se crea una función que utiliza la función Dense anterior para construir un modelo con varias capas de neuronas es el siguiente:

```
def build model():
  model = keras.Sequential([
    layers.Dense(64, activation='relu',
input shape=[len(train dataset.keys())]),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
  ])

optimizer = tf.keras.optimizers.RMSprop(0.001)

  model.compile(loss='mse',
                optimizer=optimizer,
                metrics=['mae', 'mse'])
  return model
```

Uno de los modelos que parecen más necesarios para el desarrollo del trabajo está también presente en la API de Tensorflow. Las redes LSTM se deberían aplicar a través de funciones

ya programadas debido a que la complejidad del modelo dificultaría enormemente la programación en caso de que se decidiese diseñarla desde cero.

Arguments	
`units`	Positive integer, dimensionality of the output space.
`activation`	Activation function to use. Default: hyperbolic tangent (`tanh`). If you pass `None`, no activation is applied (ie. "linear" activation: `a(x) = x`).
`recurrent_activation`	Activation function to use for the recurrent step. Default: sigmoid (`sigmoid`). If you pass `None`, no activation is applied (ie. "linear" activation: `a(x) = x`).
`use_bias`	Boolean (default `True`), whether the layer uses a bias vector.
`kernel_initializer`	Initializer for the `kernel` weights matrix, used for the linear transformation of the inputs. Default: `glorot_uniform`.
`recurrent_initializer`	Initializer for the `recurrent_kernel` weights matrix, used for the linear transformation of the recurrent state. Default: `orthogonal`.
`bias_initializer`	Initializer for the bias vector. Default: `zeros`.
`unit_forget_bias`	Boolean (default `True`). If True, add 1 to the bias of the forget gate at initialization. Setting it to true will also force `bias_initializer="zeros"`. This is recommended in Jozefowicz et al..
`kernel_regularizer`	Regularizer function applied to the `kernel` weights matrix. Default: `None`.
`recurrent_regularizer`	Regularizer function applied to the `recurrent_kernel` weights matrix. Default: `None`.
`bias_regularizer`	Regularizer function applied to the bias vector. Default: `None`.
`activity_regularizer`	Regularizer function applied to the output of the layer (its "activation"). Default: `None`.
`kernel_constraint`	Constraint function applied to the `kernel` weights matrix. Default: `None`.
`recurrent_constraint`	Constraint function applied to the `recurrent_kernel` weights matrix. Default: `None`.
`bias_constraint`	Constraint function applied to the bias vector. Default: `None`.
`dropout`	Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs. Default: 0.
`recurrent_dropout`	Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state. Default: 0.
`implementation`	Implementation mode, either 1 or 2. Mode 1 will structure its operations as a larger number of smaller dot products and additions, whereas mode 2 will batch them into fewer, larger operations. These modes will have different performance profiles on different hardware and for different applications. Default: 2.
`return_sequences`	Boolean. Whether to return the last output. in the output sequence, or the full sequence. Default: `False`.
`return_state`	Boolean. Whether to return the last state in addition to the output. Default: `False`.
`go_backwards`	Boolean (default `False`). If True, process the input sequence backwards and return the reversed sequence.
`stateful`	Boolean (default `False`). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.

Ilustración 2: Documentación función LSTM. Extraida de la documentación de Tensorflow

Otro paquete que también se encontró fue la librería Pytorch, dentro de la cual existe la función torch.nn.RNN(). Esta función genera una red neuronal recurrente de varias capas que puede ser utilizada para construir un modelo.

Arguments	
input_size	The number of expected features in the input *x*
hidden_size	The number of features in the hidden state *h*
num_layers	Number of recurrent layers. E.g., setting `num_layers=2` would mean stacking two RNNs together to form a *stacked RNN*, with the second RNN taking in outputs of the first RNN and computing the final results. Default: 1
nonlinearity	The non-linearity to use. Can be either `'tanh'` or `'relu'`. Default: `'tanh'`
bias	If `False` , then the layer does not use bias weights *b_ih* and *b_hh*. Default: `True`
batch_first	If `True`, then the input and output tensors are provided as *(batch, seq, feature)*. Default: `False`

dropout	If non-zero, introduces a *Dropout* layer on the outputs of each RNN layer except the last layer, with dropout probability equal to `dropout`. Default: 0
bidirectional	If `True`, becomes a bidirectional RNN. Default: `False`

Tabla 1: Documentacióin función RNN. Tomada de la documentación de Pytorch

Matlab

Dentro de Matlab se han encontrado múltiples tipos de capas para aplicárselos al modelo que se desee crear. A continuación, se muestran las capas convolucionales y totalmente conectadas. También se presentan las capas en secuencia.

Layer	Description
convolution2dLayer	A 2-D convolutional layer applies sliding convolutional filters to the input.
convolution3dLayer	A 3-D convolutional layer applies sliding cuboidal convolution filters to three-dimensional input.
groupedConvolution2dLayer	A 2-D grouped convolutional layer separates the input channels into groups and applies sliding convolutional filters. Use grouped convolutional layers for channel-wise separable (also known as depth-wise separable) convolution.
transposedConv2dLayer	A transposed 2-D convolution layer upsamples feature maps.
transposedConv3dLayer	A transposed 3-D convolution layer upsamples three-dimensional feature maps.
fullyConnectedLayer	A fully connected layer multiplies the input by a weight matrix and then adds a bias vector.

Ilustración 3: Capas convolucionales Matlab. Extraída de la documentación de Matlab

Layer	Description
sequenceInputLayer	A sequence input layer inputs sequence data to a network.
lstmLayer	An LSTM layer learns long-term dependencies between time steps in time series and sequence data.
bilstmLayer	A bidirectional LSTM (BiLSTM) layer learns bidirectional long-term dependencies between time steps of time series or sequence data. These dependencies can be useful when you want the network to learn from the complete time series at each time step.
gruLayer	A GRU layer learns dependencies between time steps in time series and sequence data.
sequenceFoldingLayer	A sequence folding layer converts a batch of image sequences to a batch of images. Use a sequence folding layer to perform convolution operations on time steps of image sequences independently.
sequenceUnfoldingLayer	A sequence unfolding layer restores the sequence structure of the input data after sequence folding.
flattenLayer	A flatten layer collapses the spatial dimensions of the input into the channel dimension.
wordEmbeddingLayer (Text Analytics Toolbox)	A word embedding layer maps word indices to vectors.

Ilustración 4: Capas secuenciales Matlab. Tomada de la documentación de Matlab

Otra de las funcionalidades que permite, al igual que otras herramientas, es definir la función de activación. Cabe destacar que está presente la función de tangente hiperbólica, que se utilizó en el modelo del artículo sobre MLP aplicado a datos en intervalo, por lo que presumiblemente será adecuada.

La base utilizada para programar el modelo iMLP será el código utilizado para el artículo [3], sobre el que se construirá el modelo.

Dentro de Matlab existe la herramienta RegressionLearner, que permite entrenar, sin necesidad de programación, múltiples modelos sobre los que hacer predicciones y medir su precisión. En [5] se explica detalladamente su uso, y en [6] se discute cómo adaptar un conjunto de datos en forma de series temporales además de cómo realizar la optimización de los hiperparámetros de los modelos para lograr así la máxima precisión.

Capítulo 3. ESTADO DE LA CUESTIÓN

La aplicación de modelos para la predicción del mercado de valores ha sido uno de los principales objetivos que se han estudiado desde prácticamente su creación. Para poder realizarla correctamente se han de seguir una serie de pasos.

En primer lugar, se ha de tener lugar una recolección de datos, independientemente de lo buena que sea una lógica de un modelo, esta pierde utilidad si no se aplican los datos adecuados. Algunas formas de conseguirlos serían mediante APIs, comprándolos, midiéndolos por medio de un sistema de medida especializado al problema como un sensor, a través del diseño de una aplicación de web scraping, etc. Existen opiniones de que se está produciendo un cambio de paradigma en el que este paso es el más importante. Esto es debido a la facilidad y sencillez a la hora de utilizar las tecnologías de modelado, antes el talento para diseñar un algoritmo específico del problema era lo realmente complejo. Actualmente, con las técnicas de Machine Learning, ya no supone un problema.

El siguiente paso consiste en limpiar los datos, en eliminar los valores nulos que hayan podido aparecer por cualquier error o circunstancia. La razón es que, si se introdujesen en un modelo durante el entrenamiento, no sabemos cómo podría afectar, es posible que reproduzca un outlayer que sesgue el modelo o cualquier otro tipo de fallo no deseable.

En algunos casos también es necesario preparar los datos antes de su introducción en el modelo para construir estadísticos que se sospeche que puedan ser de especial interés para el modelo en cuestión. Un ejemplo podría ser introducir la media de un vector de muestras para poder hacer un modelo que calcule la desviación típica de los datos, cálculo para el cual es indispensable la media.

El último paso antes del entrenamiento es el escalado de los datos. La razón es que los modelos están preparados para trabajar con datos normalizados y son más eficientes. Este

proceso no supone la pérdida de nada de información, por lo que no tiene ninguna desventaja, es un procedimiento habitual.

Una vez se tienen los datos con el formato adecuado y sin NAs se procede a entrenar el modelo. Para ello se seleccionan 3 grupos en los que se subdivide el conjunto de datos. El conjunto de entrenamiento se utiliza para ir probando variantes del modelo de forma que para una entrada se obtiene su salida y se compara con la salida objetivo. A partir de ahí se elabora una función de error que se intentará minimizar, si el error es alto se ajustarán los pesos del modelo para que, en las próximas salidas, el error sea menor.

Cuando la etapa de entrenamiento ha concluido, entran en juego los conjuntos de validación y test. Se comprueba el error del conjunto de test y se valida con el de validación. El objetivo de esta fase es asegurarse de que no se ha producido sobreentrenamiento, si el error en el conjunto de entrenamiento es mucho menor que el de test, entonces el modelo se ha aprendido los datos de entrenamiento, la muestra, y no la estructura de la función aleatoria, que era el auténtico objetivo, si modelamos la función aleatoria como la aplicación que va desde el conjunto de vectores que componen las entradas al conjunto de elementos que está formado por las salidas. Es de esperar que el error en entrenamiento sea menor, eso es prácticamente inevitable, pero no mucho menor. En caso de que la fase de validación y test muestre indicios de sobreentrenamiento o los resultados generales muestren un error muy elevado indicando que el modelo es de mala calidad, se recomendará que se replantee el paso de entrenamiento, aumentando 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 uno de los estudios referenciados [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). Estas RNN son de utilidad RNN son de utilidad y han demostrado una buena precisión, sin embargo, surgió la idea de las LSTM para información que siga una estructura de serie temporal. La idea está en hacer que la información del pasado también se tenga en consideración, para ello la idea inicial sería simplemente añadir las entradas de días anteriores, sin embargo, esto produce limitaciones en cuántos días sería necesario añadir. También requiere que los datos sigan una secuencia, por lo que el orden se debe conservar y no se puede aleatorizar. Esto produce un problema porque existe riesgo de que, si los datos han sido tomados en periodos alcistas se recomiende siempre comprar. Para solucionar esto se subdividen los datos en intervalos que sí mantendrán el orden. Esta técnica hace que se pueda alcanzar un punto de aleatorización de los datos respetando las necesidades de orden del modelo, aunque introduce una nueva limitación, el número de días que se van a tener en cuenta dentro del modelo. Esto implica que se tiene que alcanzar un equilibrio entre la cantidad de puntos pasados que tendrá el modelo como memoria y la cantidad de intervalos desordenados que se podrán introducir en el modelo para entrenarlo. Ahora bien, un modelo LSTM utiliza una pequeña variación sobre el modelo descrito, en lugar de introducirse las entradas de los puntos pasados directamente, se pasa cada punto pasado por la red y se obtiene un resultado (cabe destacar que alcanzar este resultado no es computacionalmente costoso dado que ya se calcularon los valores para predicciones anteriores). Con la salida de la red guardada se introduce como entrada en el punto presente que se estaba analizando. En la figura siguiente se ilustra la estructura de la red a alto nivel.

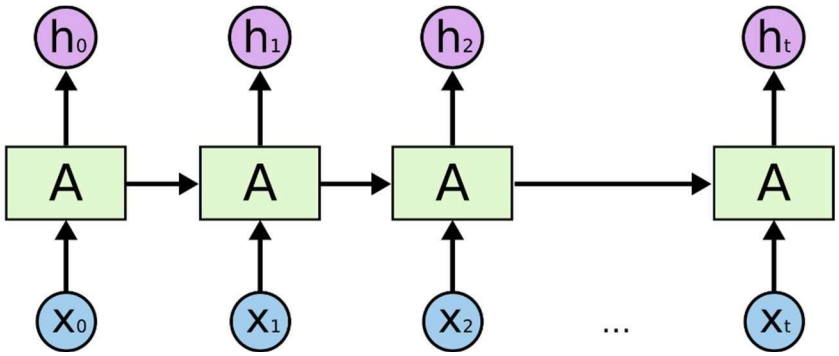


Ilustración 5: Estructura LSTM, tomado de [11]

Este tipo de modelos han mostrado unos muy buenos resultados para la información que esté compuesta por series temporales, la razón es que está diseñada con esa finalidad. 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. La razón por la que esto supone tanta diferencia es que, suponiendo que se desee minimizar una función de error E con la que se mide la calidad del modelo al final de cada iteración del entrenamiento, muchos modelos habituales encontrarían mínimos locales. Sin embargo, el componente aleatorio añadido de una red estocástica permite que, si se encuentra estabilizado en un mínimo local, salte a otros puntos aleatorios que podrían dar un resultado menor y a partir de los cuales el modelo convergería al mínimo de su entorno local, el cual, en caso de no ser el mínimo absoluto, volvería a probar otros puntos aleatorios que darían lugar a otros mínimos locales hasta que se encontrase la combinación adecuada que proporcionase el mínimo global de la función de error.

Estos modelos se han aplicado a la predicción del mercado de valores con unos resultados relativamente altos 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.

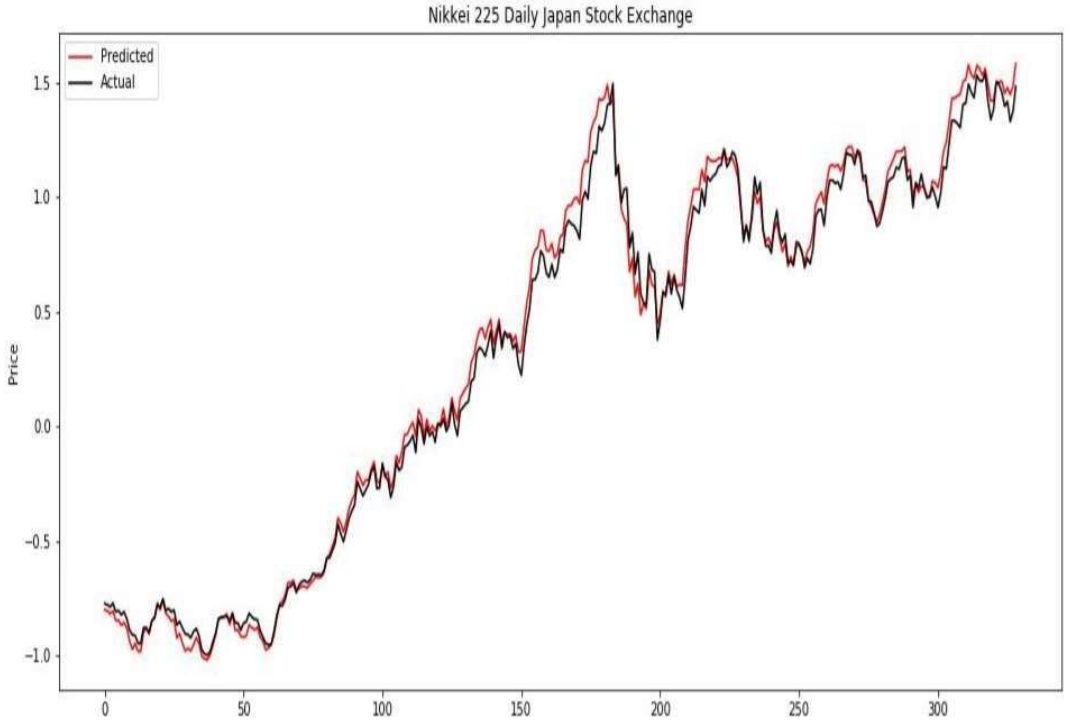


Ilustración 6: Predicción de SNN en comparación con el valor real del Nikkei. Tomado de [1]

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. Adicionalmente se calcularon múltiples indicadores mediante operaciones aritméticas sencillas a partir de las entradas iniciales de la red. Cabe destacar que estos indicadores, al poder deducirse a partir de la información de las entradas básicas, deberían ser omisibles, sin embargo, es probable que se

requiera de complejidad adicional para deducir esa información aportada, lo que tal vez podría provocar sobre entrenamiento.

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:

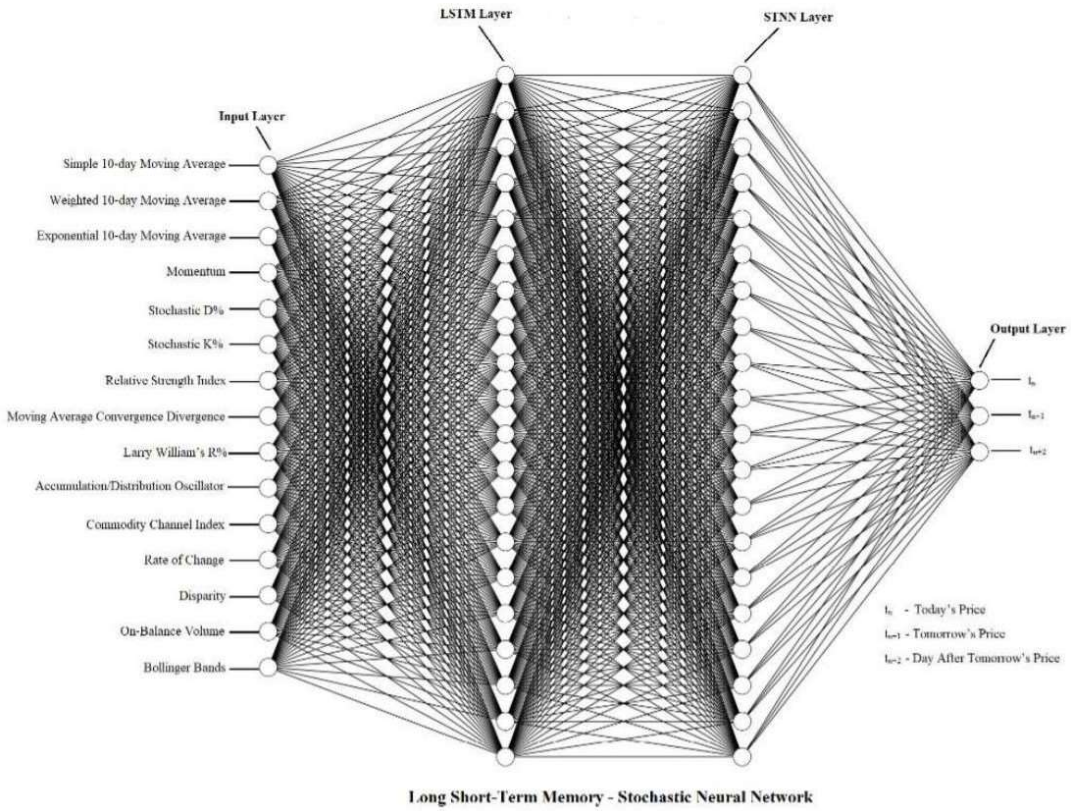


Ilustración 7: Diagrama red LSTM+SNN. Tomado de [1]

Cuando se construyó el modelo y se entrenó con datos del Nikkei se llegó a una predicción ligeramente superior a la del modelo puramente estocástico, de un 86.28% de precisión en

la predicción de la tendencia. Aunque la diferencia de precisión sea pequeña, es importante tener en cuenta que, cuanta más precisión se tenga, más difícil es incrementarla, en otras palabras, el coste marginal de incrementar la precisión cuando es ya muy alta es elevado. Por esta razón, esa diferencia que aparentemente era insignificante al tratarse de un incremento pequeño es, en realidad, significativo y un logro y un muy buen resultado. Una representación más visual se puede ver en la figura siguiente, donde se muestran las predicciones y los valores a predecir del modelo desarrollado.

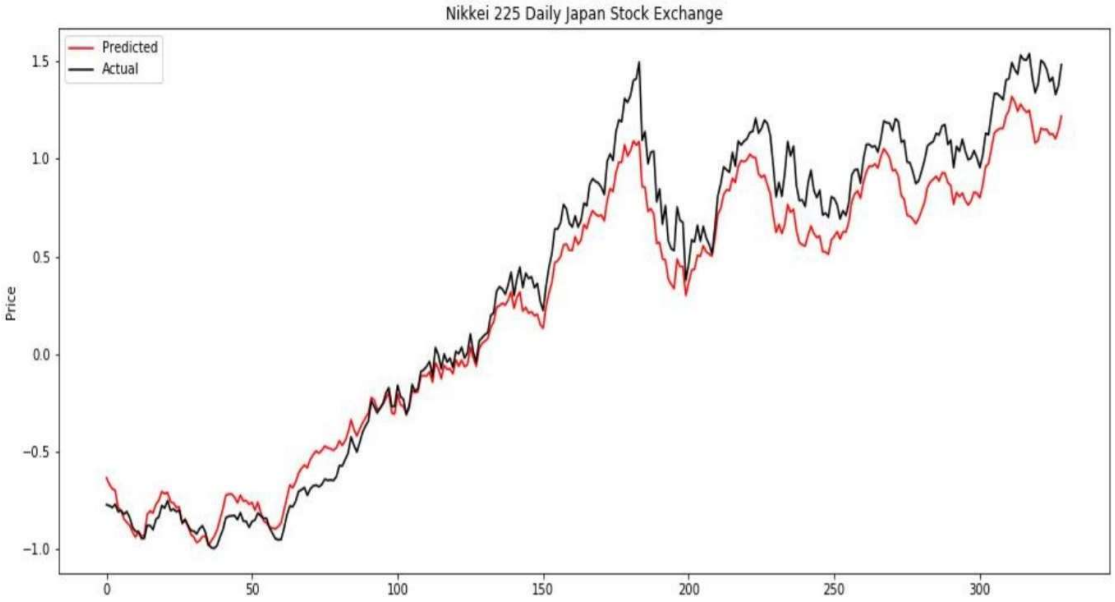


Ilustración 81: Predicción modelo LSTM+SNN vs datos objetivo. Tomado de [1]

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. Las salidas de cada neurona van a las de la capa siguiente hasta alcanzar la salida. Para el aprendizaje utiliza funciones con parámetros a elección del responsable, como el ratio de aprendizaje. 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 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.

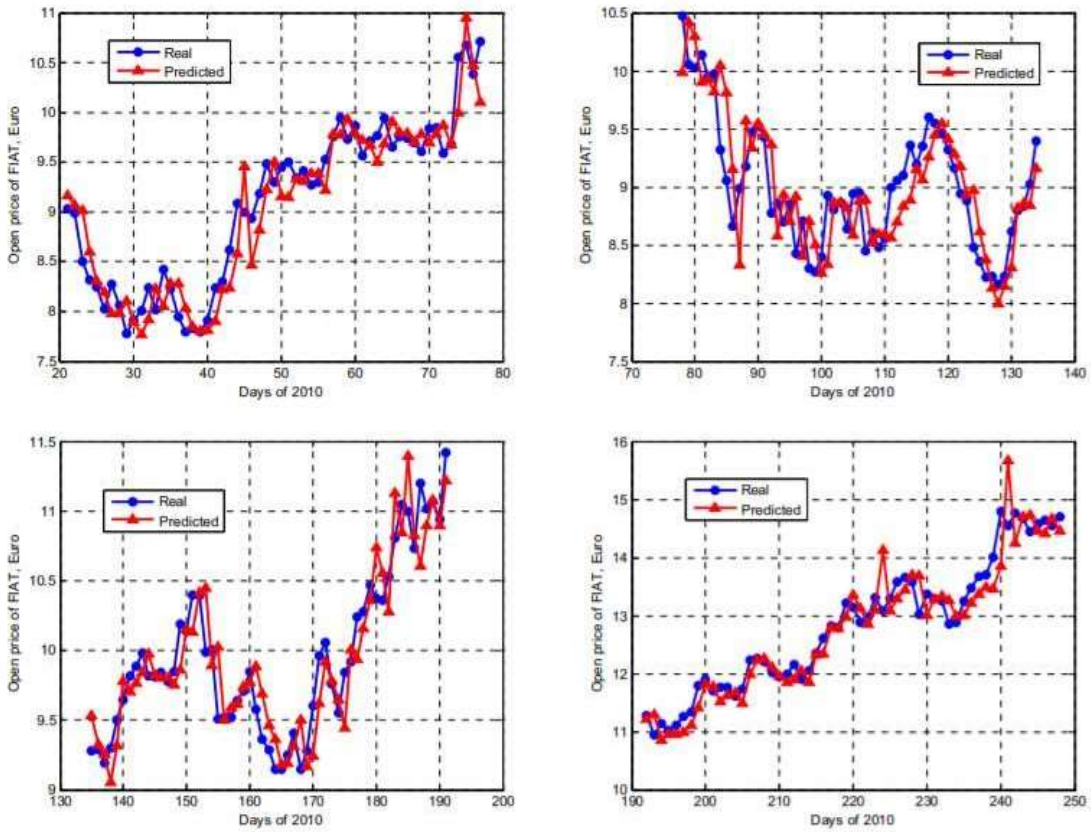


Ilustración 9: Valor real vs Predicción usando MLP. Tomado de [2]

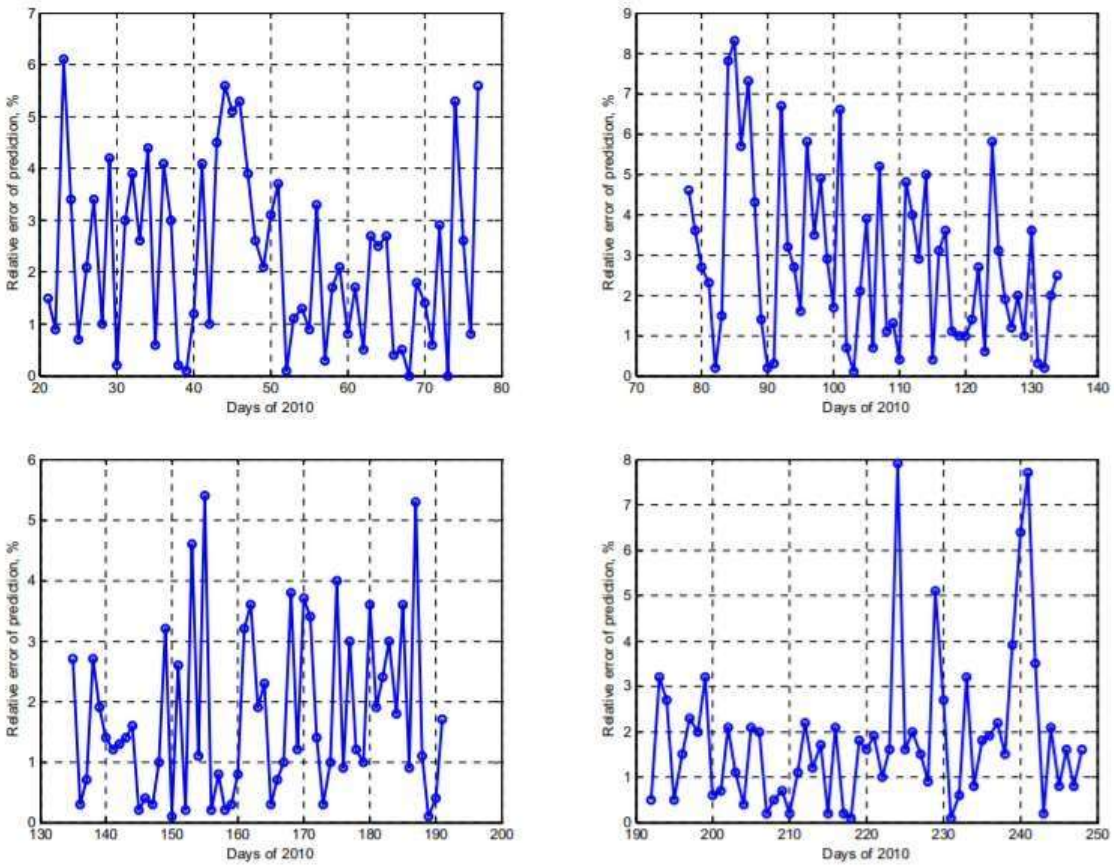


Ilustración 2: Error relativo a las predicciones en MLP. Tomado de [2]

Si utilizamos 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. Esta novedosa arquitectura del modelo ha sido utilizada en varios trabajos como el referente a [14], que se trata de una revista que es Q1.

El uso de los datos de este tipo en vez de los singulares ha producido múltiples variaciones de modelos que usan redes neuronales. 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. La más sencilla consiste en dar los puntos correspondientes a los extremos del intervalo, de forma que el elemento dado sería un vector de dimensión 2, perteneciendo cada componente a los números reales. La segunda forma que se utiliza es dar la información del centro del intervalo y su radio. Este último método es el que más se usa porque permite ver los errores del modelo en los ámbitos de forma separada. 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 semisumas y semidiferencias. 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 2: Pesos y sesgos con valores singulares. Con base [3]

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

Tabla 3: Pesos y sesgos con valores de intervalo. Con base [3]

En cada celda se muestran los autores que han desarrollado un modelo de esas características y que se han recopilado. Cabe destacar la ausencia de los autores Conan-Guez, quienes hicieron múltiples propuestas para aplicarlas a los modelos que utilizan datos en intervalo y que no entrarían en ninguna de las categorías. La metodología que se propone 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.

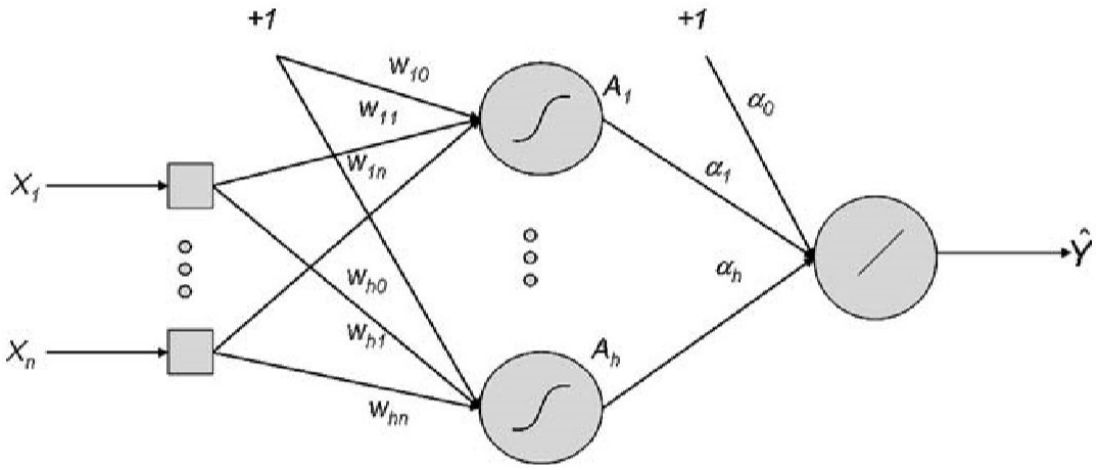


Ilustración 31: Estructura del modelo iMLP. Tomado de [3]

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:

$$S_j = (w_{j0} + \sum_{1=1}^{n} w_{ji} x_i^C \mid \sum_{i=1}^{n} |w_{ji}| \, x_i^R) \qquad (1)$$

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:

$$A_j = \left(\frac{\tanh(S_j^C - S_j^R) + \tanh(S_j^C + S_j^R)}{2} \mid \frac{\tanh(S_j^C + S_j^R) - \tanh(S_j^C - S_j^R)}{2}\right) \qquad (2)$$

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:

$$S_j = (\alpha_0 + \sum_{i=1}^{n} \alpha_i a_i^C \mid \sum_{i=1}^{n} |\alpha_i| \, a_i^R) \qquad (3)$$

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.

$$E = \frac{1}{2} \sum_{t=1}^{p} d(F(t), \hat{F}(t)) + \lambda \phi(\hat{F}) \qquad (4)$$

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.

$$d(A,B) = \beta(a^{C} - b^{C})^2 + (1-\beta)(a^{R} - b^{R})^2 \qquad (5)$$

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

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

A la vista del capítulo anterior debemos realizar un análisis crítico sobre los trabajos previos realizados y saber ya claramente por qué se va a desarrollar el proyecto. La motivación de la introducción puede ser una cuestión más filosófica ("las soluciones *e-health* tienen una gran impacto en la sociedad actualmente…"), mientras que la justificación es una cuestión técnica y de mercado ("ya que no hay ningún *wearable* basado en BLE en el mercado que mida las constantes y las procese en la nube mediante técnicas de *Machine Learning* para predecir enfermedades, yo voy a desarrollar…").

Aquí hay que vender el proyecto y responder a la pregunta ¿Por qué alguien puede querer comprar este proyecto? Si fueseis comerciales, aquí es donde hay que convencer al cliente o emprendedores, a vuestro inversor.

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.

4.2 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.

4.3 METODOLOGÍA

¿Cuál es el camino que se usa para conseguir los objetivos?

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].

El plan de trabajo tendrá la siguiente estructura:

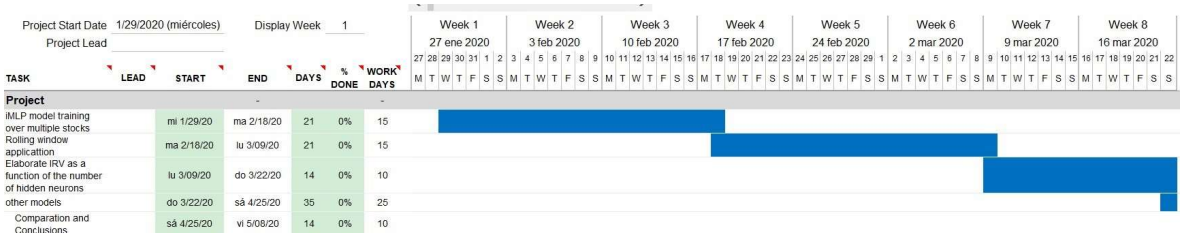


Ilustración 42: Plan de trabajo (1). De elaboración propia

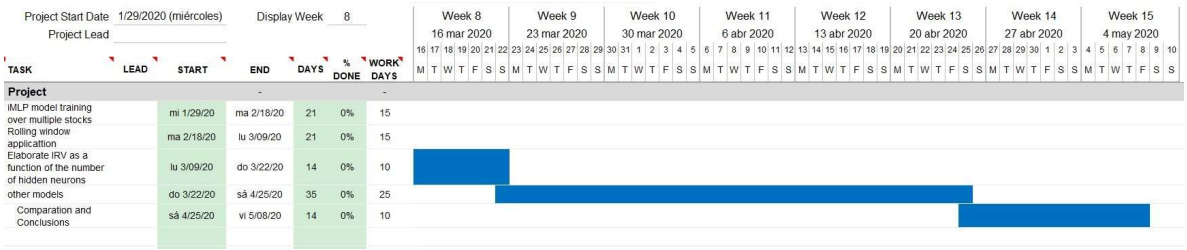


Ilustración 53: Plan de trabajo (2). De elaboración propia

Capítulo 5. 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 están programadas 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'].shift(-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.

5.1 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.

Estaba el intervalo de apertura a cierre, que tomaba gran importancia al definir valores concretos al final del día. Asimismo, también aporta información del crecimiento y la tendencia. Sin embargo, toma un valor aleatorio en el intervalo que puede ser muy poco representativo en el mismo.

El intervalo de mínimo a máximo, por el contrario, toma valores muy representativos que están influenciados por la volatilidad y otras componentes que pueden aportar información importante a la hora de tomar una decisión de inversión. Sin embargo, no informa del crecimiento, de si ha sido un día positivo o negativo.

Finalmente se optó por el intervalo de mínimo a máximo. Las razones positivas son las anteriormente explicadas, pero 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. Si se estima que el intervalo mínimo-máximo va a ser siempre mayor o igual al valor de cierre del día anterior, entonces el inversor estará mucho más confiado si el modelo es de calidad. Para el caso análogo, en el que se predice que el precio de cierre del día siguiente será mayor al del actual, el inversor invertirá igualmente, pero será consciente de las probabilidades de que se den altas fluctuaciones a lo largo del periodo, por lo que su confianza y seguridad serán menores.

Con este dilema resuelto y tomada la decisión, se pudieron adaptar los datos al formato de centro y radio requerido por el iMLP a través de una sencilla transformación en la que los centros se calculaban como la media de los extremos del intervalo y los radios como la diferencia entre cualquiera de los extremos del intervalo y los centros que se acaban de calcular.

Una de las posibles aplicaciones potenciales de este trabajo es la de ser una herramienta para el trading o la inversión. Esta herramienta se debería poder usar de varias formas, es posible que se utilizase exclusivamente para la predicción por sí misma, pero también se podría ver como un complemento para el trader.

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

La primera solución que se planteó para el formato de la variable de entrada fue el más sencillo: una variable entera que cambie su valor en función del tipo de comportamiento mostrado. El uso de este enfoque traería problemas porque para el modelo sería más complicado asociar un único valor a 5 distintos posibles comportamientos, si la variable aumenta en una unidad, el modelo probablemente supondría un incremento o decremento en la intensidad de ese comportamiento cuando, en realidad, significa un cambio completamente distinto.

Por esta razón 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. Este formato soluciona el problema al ser entendido por el modelo como variables independientes dicotómicas del tipo “es o no es decreciente suave”.

Aunque la complejidad aumente al tratarse de cinco nuevas entradas en lugar de una, esto es necesario como se acaba de explicar. Al tratarse de entradas de tipo intervalo se deberá asignar un radio ínfimo, el que se escogió fue de 10^(-10).

Cabe destacar que, si el vector introducido no tiene ninguna componente no nula, entonces se supondrá que no presenta ninguno de los escenarios descritos.

Junto con este nuevo parámetro de entrada viene la necesidad de identificar estos periodos a lo largo del histórico de la acción estudiada para así poder entrenar correctamente el modelo. Esto supondrá lógica de preprocesado añadida.

En el caso de que no se desee usar esta aplicación bastaría con no aplicar esta lógica y poner todas las componentes vectoriales a cero. Otra alternativa más sencilla sería no introducir el vector de entrada, aunque se hubiesen procesado los datos.

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 conjuntos de entrenamiento, validación y test 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.

De manera análoga se haría lo mismo con las particiones basadas en los días de la semana y los momentos del año.

El siguiente paso fue hacer combinaciones con las posibles descomposiciones: se hizo una con 24 modelos para cada hora del día, otro con 7 para cada día de la semana, otro con 168 modelos para cada día y hora de la semana, y otro que utilizaba un único modelo para cada tarea.

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.

$$\mathbf{p} = \left[\sin\left(2\pi\frac{\#(i+\tau)}{366}\right) \quad \cos\left(2\pi\frac{\#(i+\tau)}{366}\right)\right] \qquad (1)$$

Esta transformación ha sido utilizada en múltiples estudios para caracterizar esa variable de entrada, es un formato válido y probado.

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, por lo que el tiempo requerido para el proceso de aprendizaje y optimización es menor. Asimismo, la baja complejidad del modelo lo hace más resistente al problema del overfitting. En cuanto al rendimiento, se han mostrado unos resultados competentes si se utilizan los modelos adecuados para las particiones.

Una de las ideas que surge de la segmentación es la de la paralelización. Dado que hay que entrenar múltiples modelos sencillos, es probable que facilite la división del trabajo en múltiples hilos que trabajen en paralelo. Para poder conseguir esto sería necesario coordinar cada hilo y poder entrenar y utilizar el modelo para hacer predicciones puntuales desde un programa principal.

Si se valora aplicar esta metodología al proyecto, se puede ver que uno de los posibles criterios de particionado podría ser el del comportamiento anteriormente descrito. Se desconoce si se producirían buenos resultados como los mostrados en el artículo, por ello se deberían probar las dos alternativas.

El principal hará las predicciones sobre el conjunto de datos completo mientras que, en el alternativo, se desarrollarán Tantos modelos como categorías y otros tantos subconjuntos de los datos.

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.

La metodología utilizada para hallar el valor óptimo de ambos hiperparámetros fue la de 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.

Cabe destacar que se debía repetir este proceso de prueba de todas las combinaciones cada vez que se estudiase un conjunto de datos, constituyendo una parte adicional del entrenamiento 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 gde representar el error.

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

$$iARV = \frac{\sum(\widehat{F_L}-F_L)_2^2+\sum(\widehat{F_U}-F_U)^2_2}{\sum(\widehat{F_L}-\widehat{F_{L,t}})^2+\sum(\widehat{F_U}-\widehat{F_{U,t}})^2} \quad (2)$$

$$U\ de\ Theill = \frac{\sum(\widehat{F_L}-F_L)^2+\sum(\widehat{F_U}-F_U)^2}{\sum(F_{L,t}-F_{L,t-1})^2+\sum(F_{U,t}-F_{U,t-1})^2} \quad (3)$$

$$CR = \frac{Min(\widehat{F_U},F_U)-Max(\widehat{F_L},F_L)}{F_U-F_L} \quad (4)$$

$$ER = \frac{Min(\widehat{F_U},F_U)-Max(\widehat{F_L},F_L)}{\widehat{F_U}-\widehat{F_L}} \quad (5)$$

Siguiendo esa metodología para el entrenamiento y con las métricas de calidad del modelo bien definidas, se definieron una amplia variedad de escenarios concretos sobre un conjunto de datos en el que se probó el modelo.

1. COCP
    a. Suponiendo comportamiento similar a junio de 2020
        i. IARV:
            1. Según Nhidden

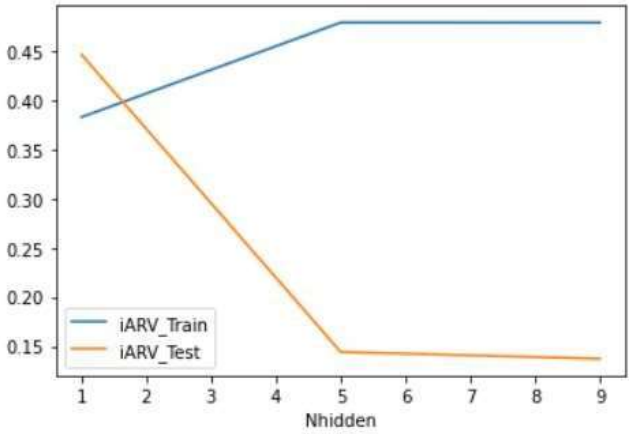


Ilustración 64: De elaboración propia

2. Según delt_t

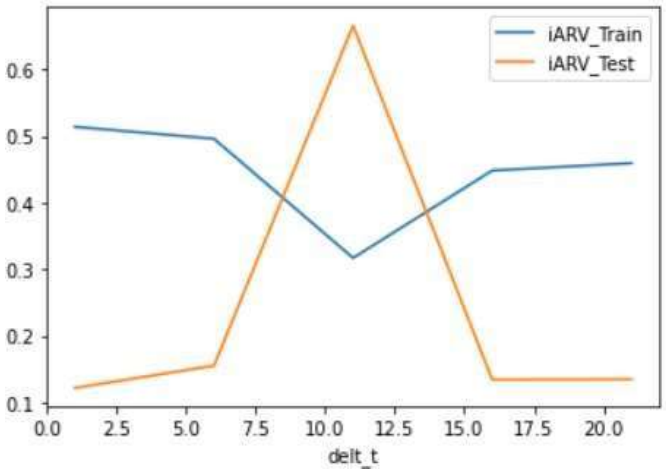


Ilustración 75: De elaboración propia

ii. iuTheill

1. Según Nhidden

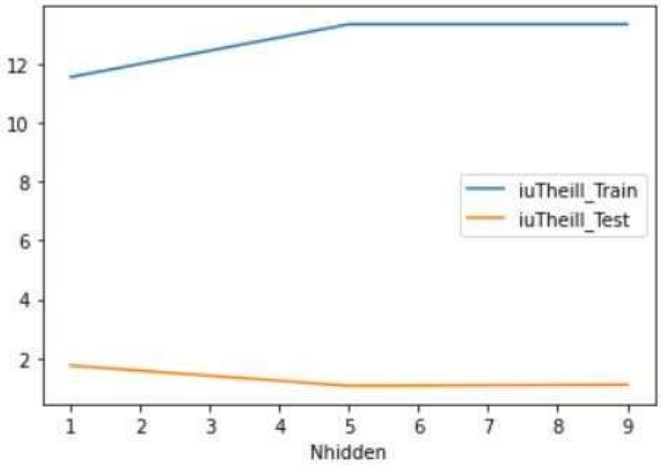


Ilustración 86: De elaboración propia

2. Según delt_t

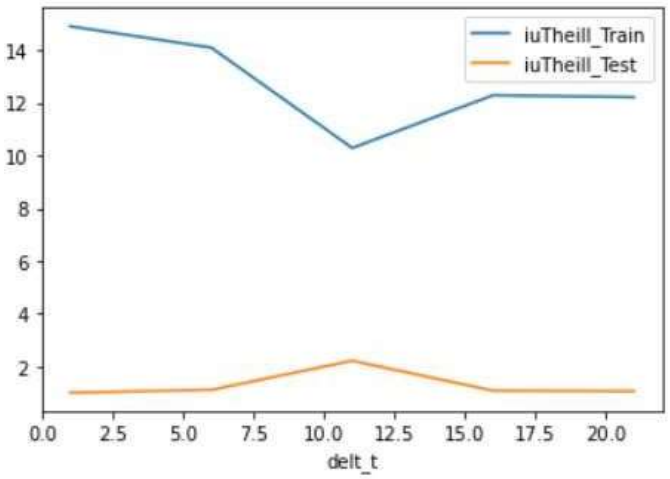


Ilustración 97: De elaboración propia

iii. CR, ER

1. Según Nhidden

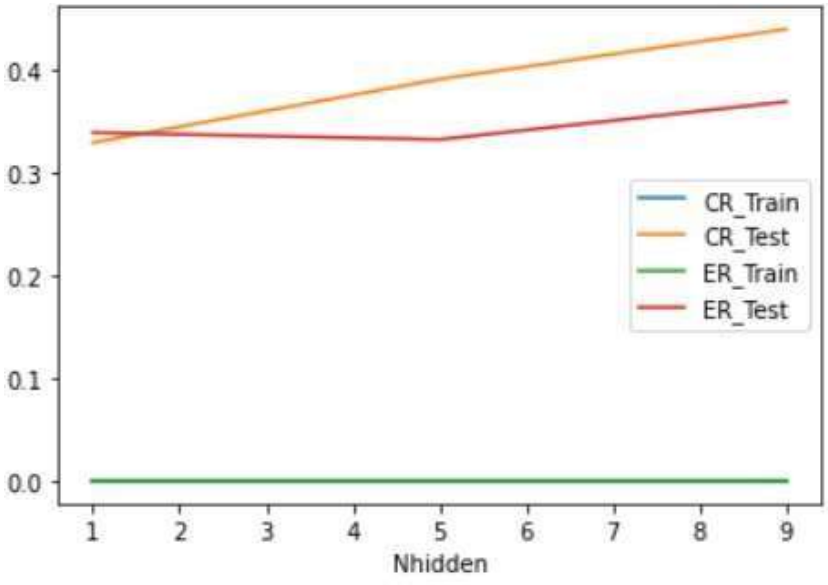


Ilustración 108: De elaboración propia

2. Según delt_t

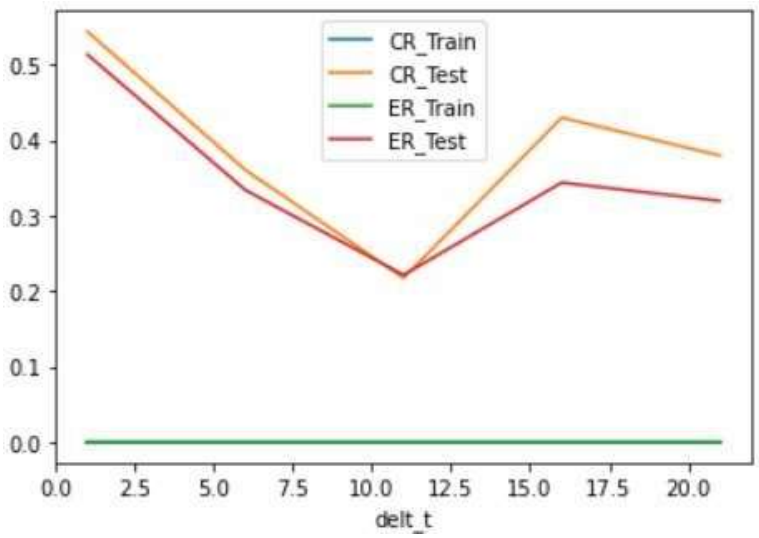


Ilustración 119: De elaboración propia

iv. Modelos seleccionados

	delt_t	Nhidden	iARV_Train	iARV_Test	iuTheill_Train	iuTheill_Test	CR_Train	CR_Test	ER_Train	ER_Test
0	1	9	0.514354	0.116374	14.916935	1.035944	0	0.649214	0	0.554754
1	1	5	0.515079	0.124597	14.943298	0.995401	0	0.437282	0	0.419871
2	1	1	0.512323	0.125152	14.841503	1.007734	0	0.545014	0	0.565709
3	21	1	0.458065	0.129760	12.201957	1.062410	0	0.385741	0	0.341409
4	6	1	0.492873	0.129954	14.047638	1.025648	0	0.361442	0	0.363548

Tabla 5: De elaboración propia

v. Predicciones

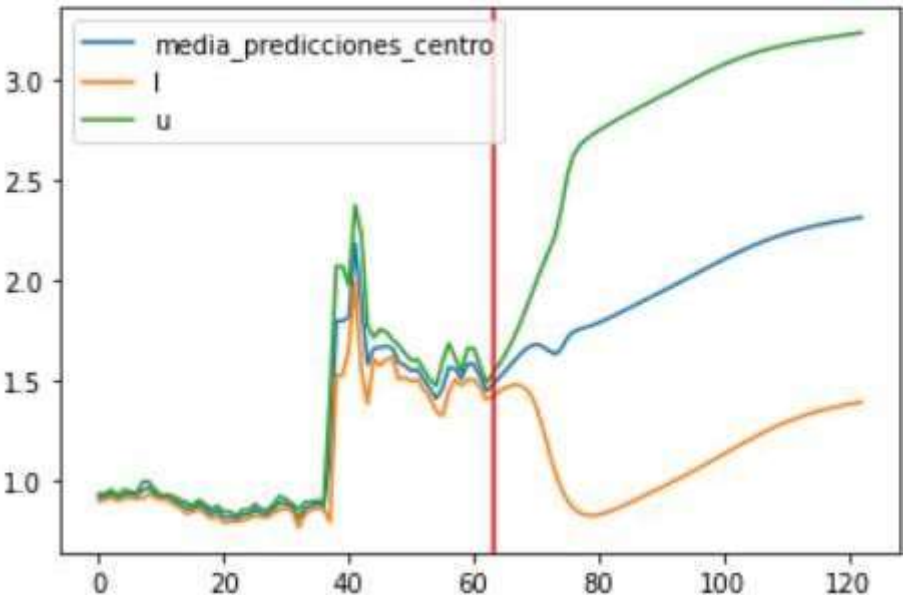


Ilustración 20: De elaboración propia

Como se puede apreciar, las predicciones muestran una tendencia claramente alcista. Esto es debido a que la tendencia esperada introducida corresponde a un periodo alcista, lo que ha hecho que el modelo tenga ese sesgo introducido artificialmente a hacer predicciones de subida.

Cabe destacar la alta precisión del modelo en lo referente a las métricas de entrenamiento en contraposición a las de test, lo que podría inducir a pensar que existe algún tipo de sobreentrenamiento.

b. Suponiendo comportamiento similar a abril-mayo-junio de 2020
    i. IARV:
        1. Según Nhidden

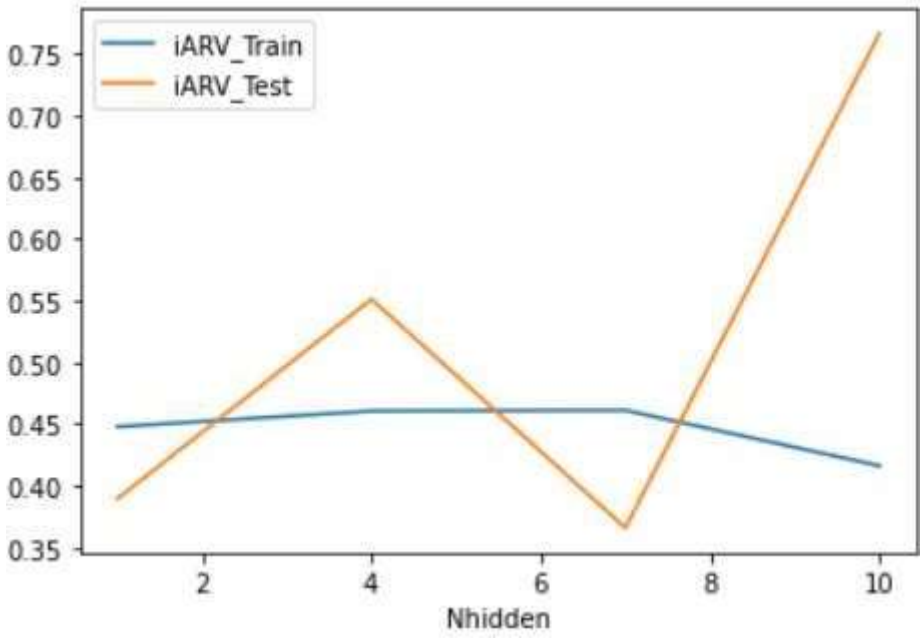


Ilustración 21: De elaboración propia

2. Según delt_t

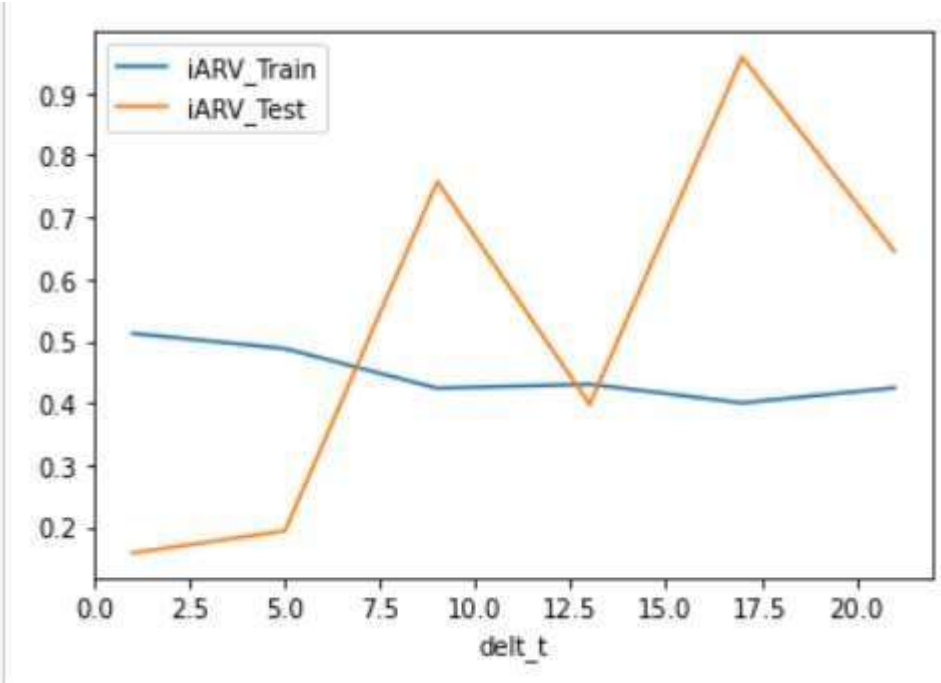


Ilustración 22: De elaboración propia

ii. iuTheill
1. Según Nhidden

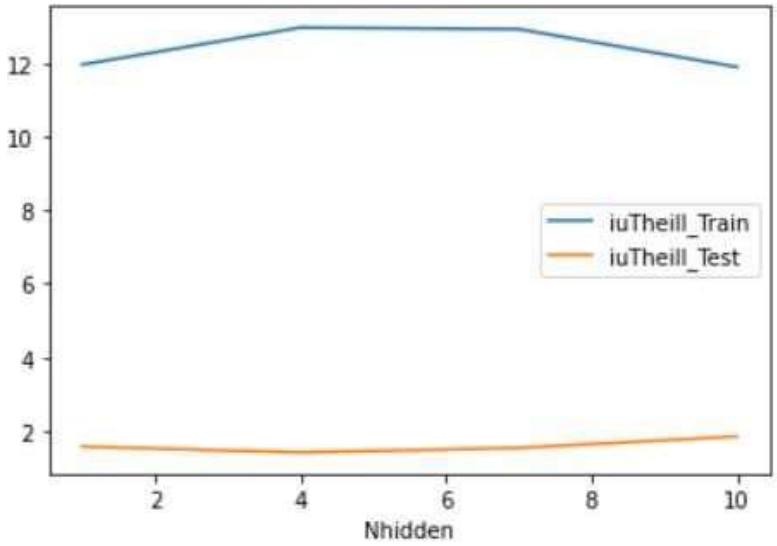


Ilustración 23: De elaboración propia

2. Según delt_t

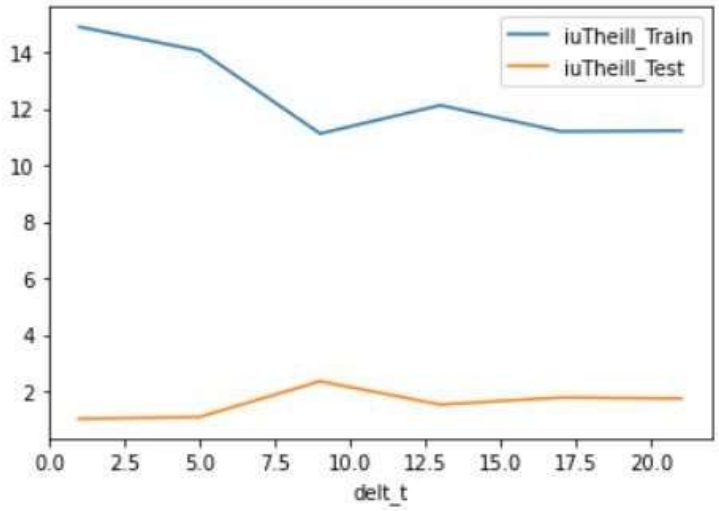


Ilustración 24: De elaboración propia

iii. CR, ER

1. Según Nhidden

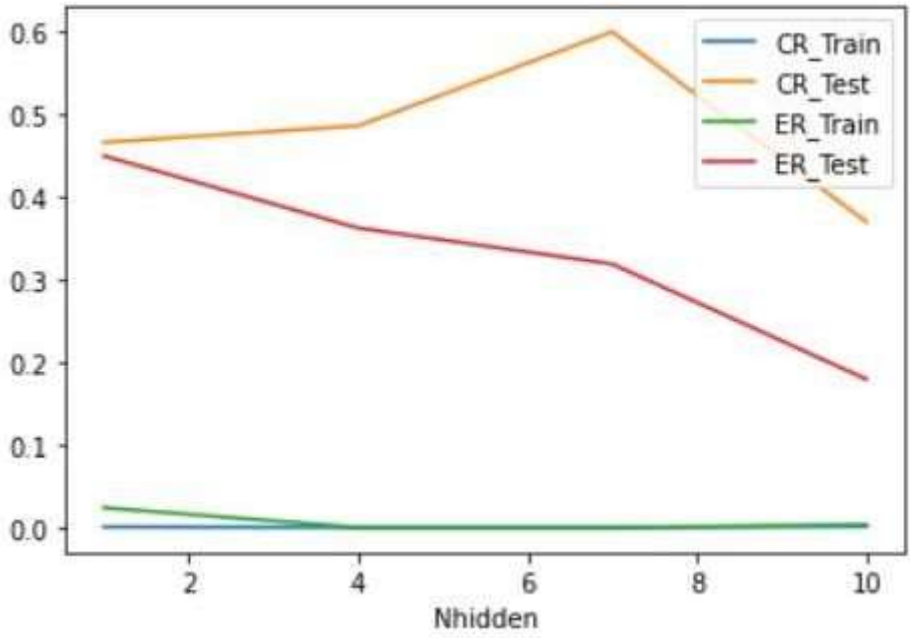


Ilustración 25: De elaboración propia

2. Según delt_t

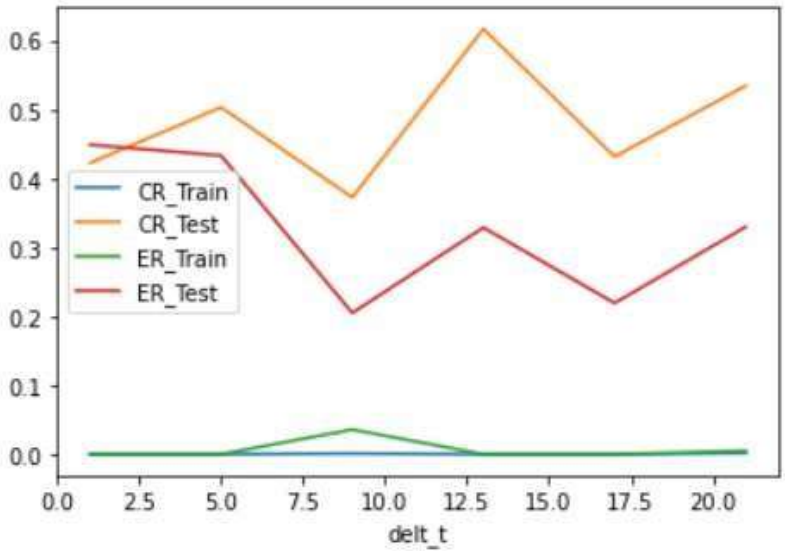


Ilustración 26: De elaboración propia

iv. Modelos seleccionados

	delt_t	Nhidden	iARV_Train	iARV_Test	iuTheill_Train	iuTheill_Test	CR_Train	CR_Test	ER_Train	ER_Test
0	1	1	0.514829	0.120428	14.929437	1.001029	0.0	0.524097	0.0	0.534695
1	13	1	0.464597	0.124304	12.838623	1.018808	0.0	0.580644	0.0	0.511512
2	5	1	0.499636	0.124701	14.245329	1.002926	0.0	0.496868	0.0	0.497570
3	17	1	0.463362	0.126253	12.525730	1.016884	0.0	0.572548	0.0	0.495681
4	21	1	0.467658	0.128229	12.351793	1.021036	0.0	0.579107	0.0	0.488548

Tabla 6: De elaboración propia

v. Predicciones

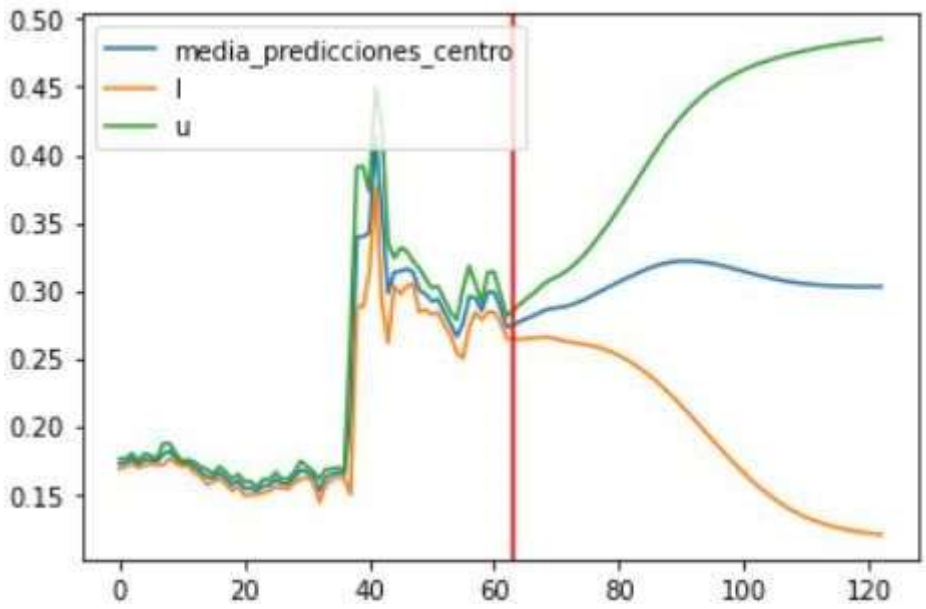


Ilustración 27: De elaboración propia

Para este caso, la tendencia introducida tenía unas características más bajistas, aunque no tan pronunciadas como las del periodo introducido en el caso anterior. Esto ha producido unas predicciones que de media son más bajistas tanto en la media de los centros como en el mínimo. Las diferencias entre los estadísticos tomados del entrenamiento y del test vuelven a ser altas, especialmente en el caso del CR y la ER.

c. Suponiendo comportamiento similar a marzo de 2020
    i. IARV:
        1. Según Nhidden

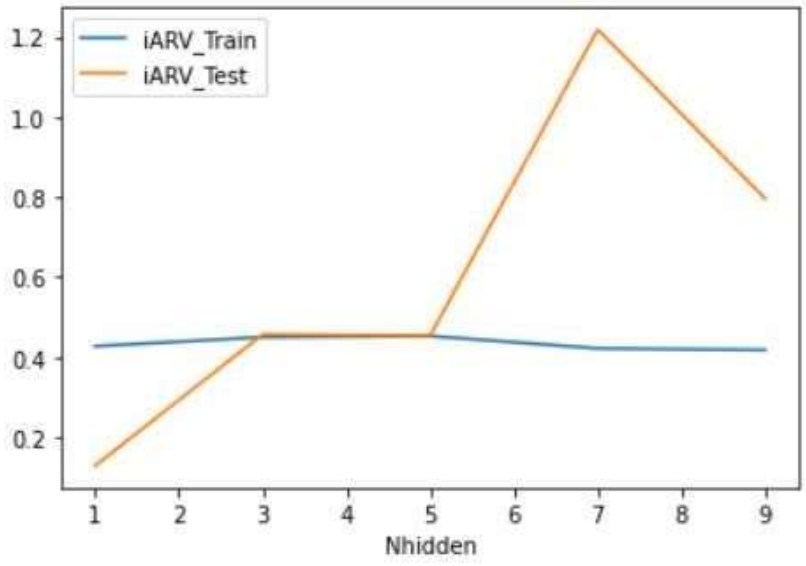


Ilustración 28: De elaboración propia

2. Según delt_t

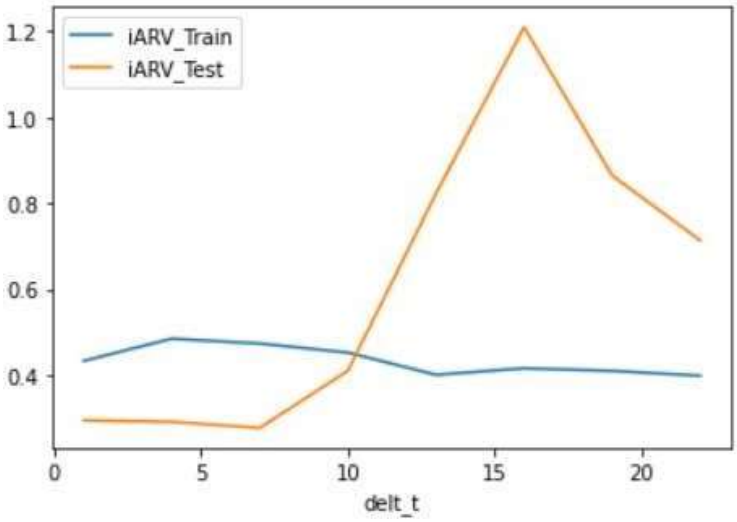


Ilustración 29: De elaboración propia

ii. iuTheill

1. Según Nhidden

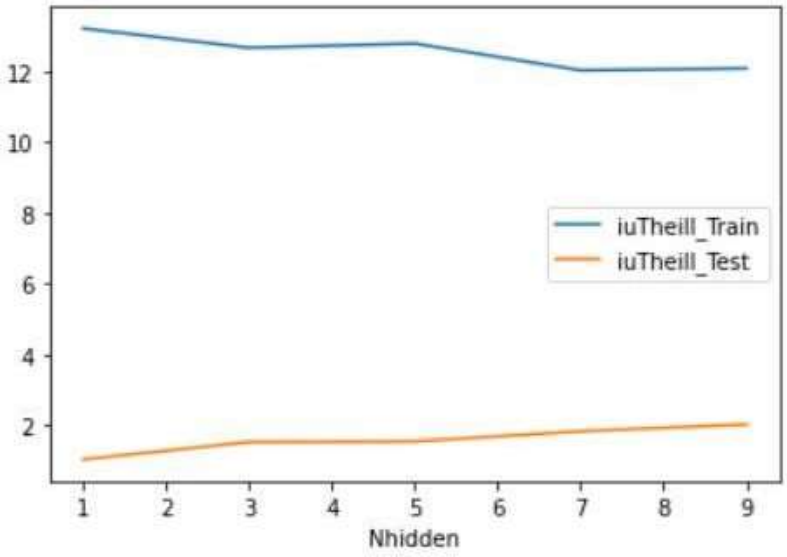


Ilustración 30: De elaboración propia

2. Según delt_t

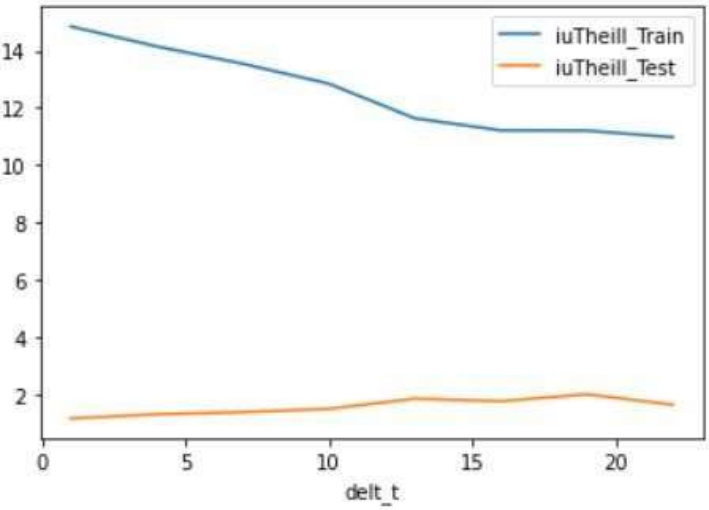


Ilustración 31: De elaboración propia

iii. CR, ER

1. Según Nhidden

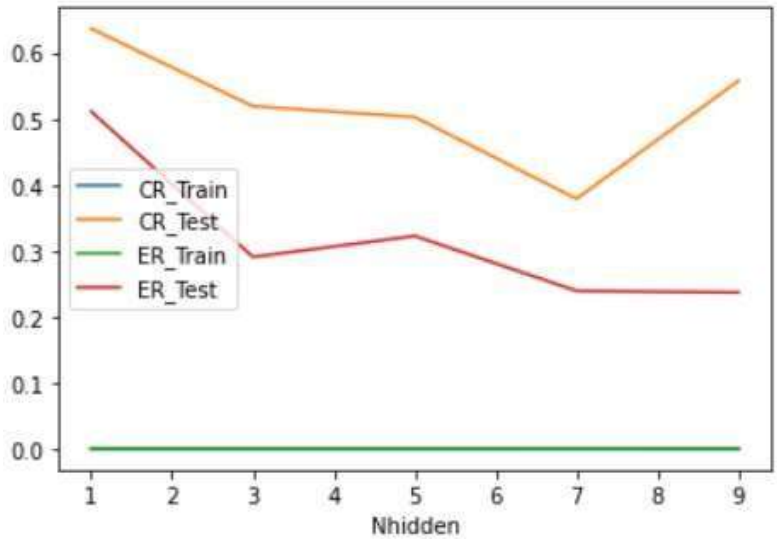


Ilustración 32: De elaboración propia

2. Según delt_t

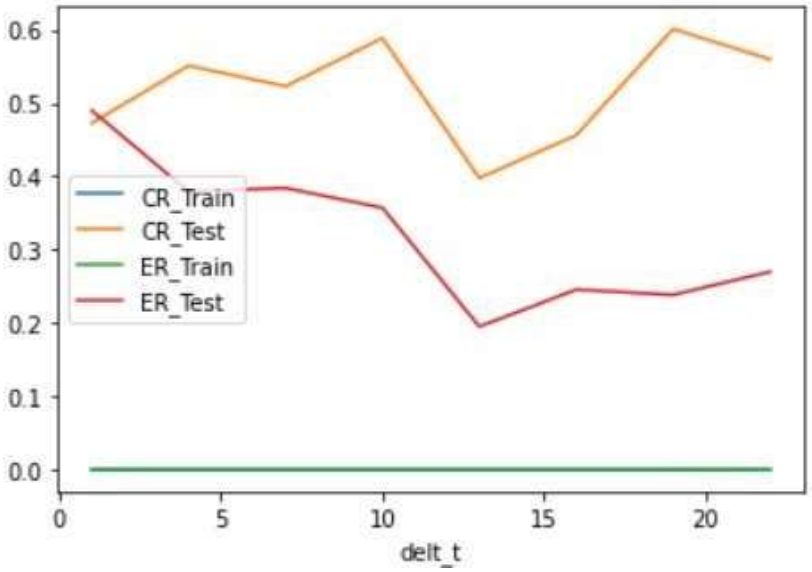


Ilustración 33: De elaboración propia

iv. Modelos seleccionados

	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 7: De elaboración propia

v. Predicciones

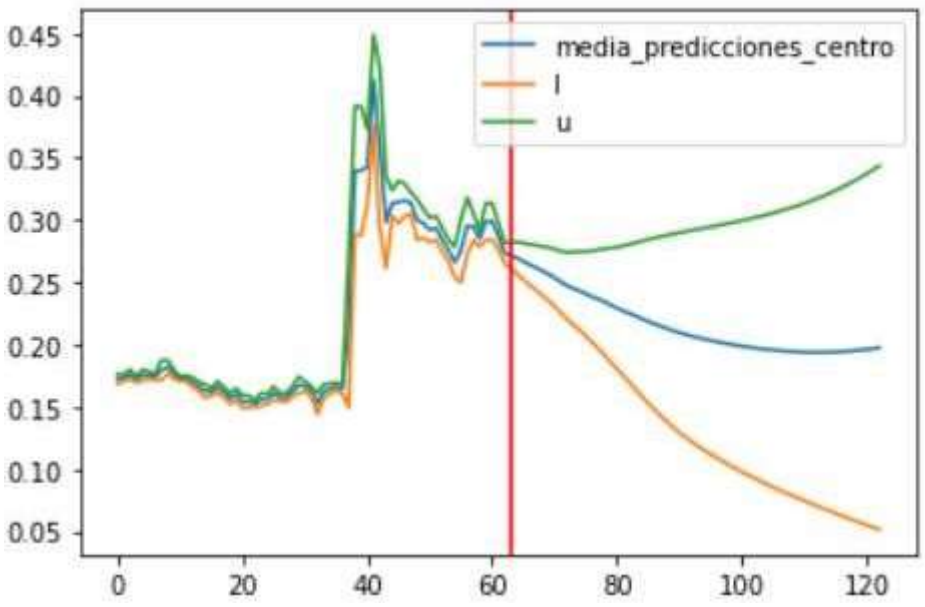


Ilustración 34: De elaboración propia

En este último escenario las predicciones muestran una tendencia por lo general más estable, lo que tendría sentido para un comportamiento esperado del tipo lateral, caracterizado por subidas y bajadas sin una tendencia clara.

2. SAB
    a. Suponiendo comportamiento similar a marzo de 2020
        i. IARV:
            1. Según Nhidden

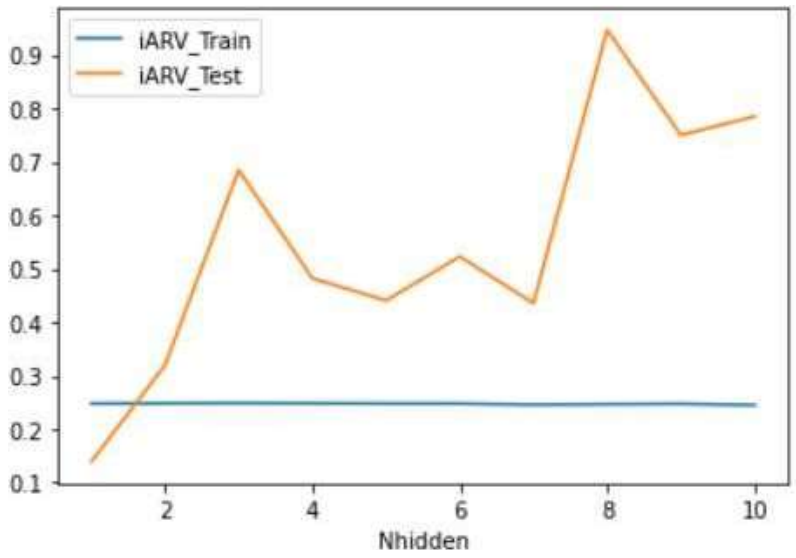


Ilustración 35: De elaboración propia

2. Según delt_t

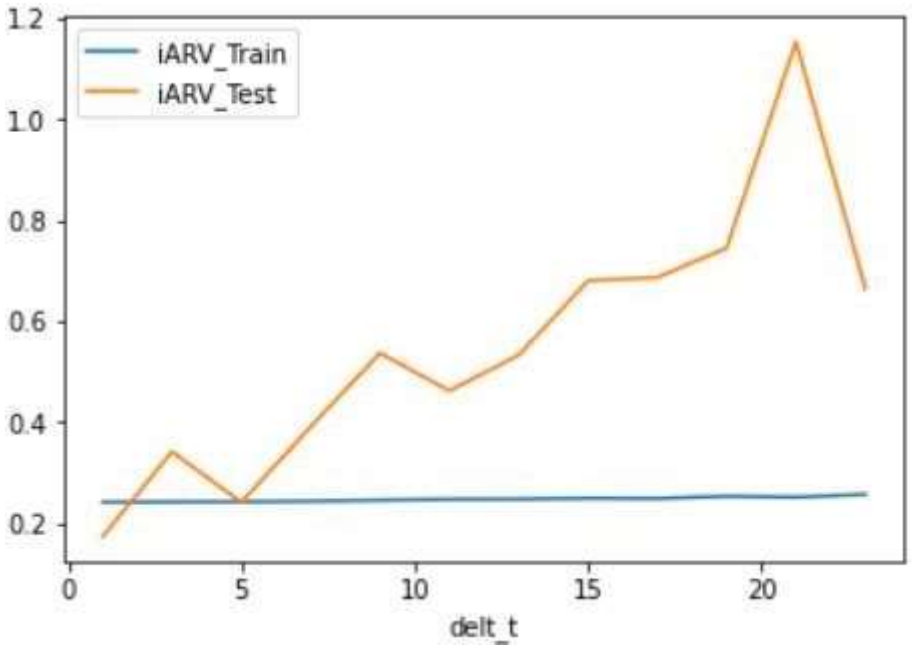


Ilustración 36: De elaboración propia

ii. iuTheill

1. Según Nhidden

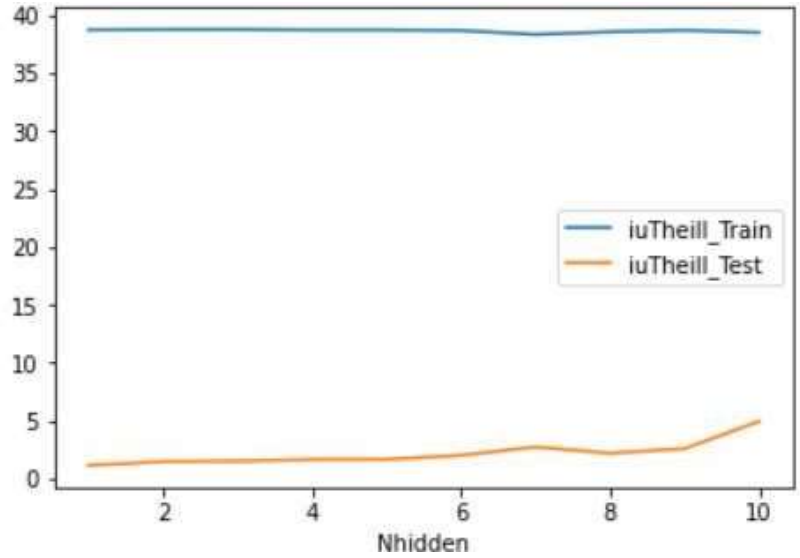


Ilustración 37: De elaboración propia

2. Según delt_t

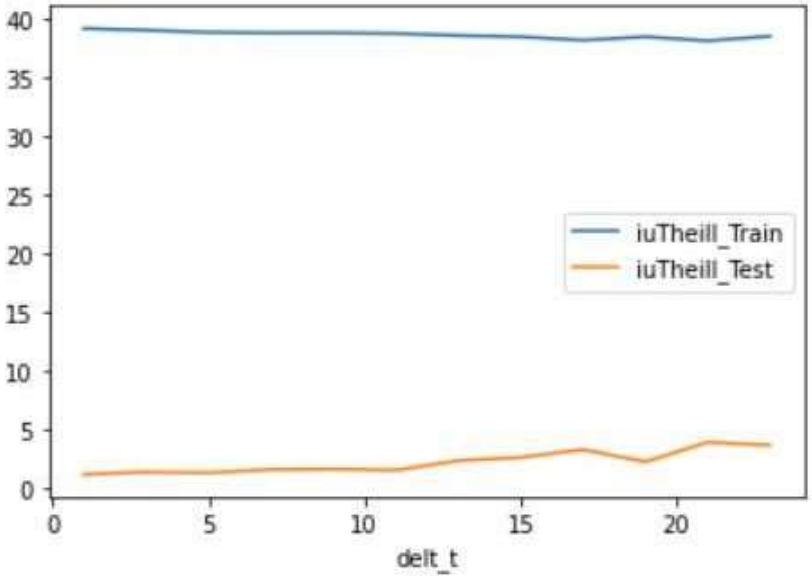


Ilustración 38: De elaboración propia

iii. CR, ER

1. Según Nhidden

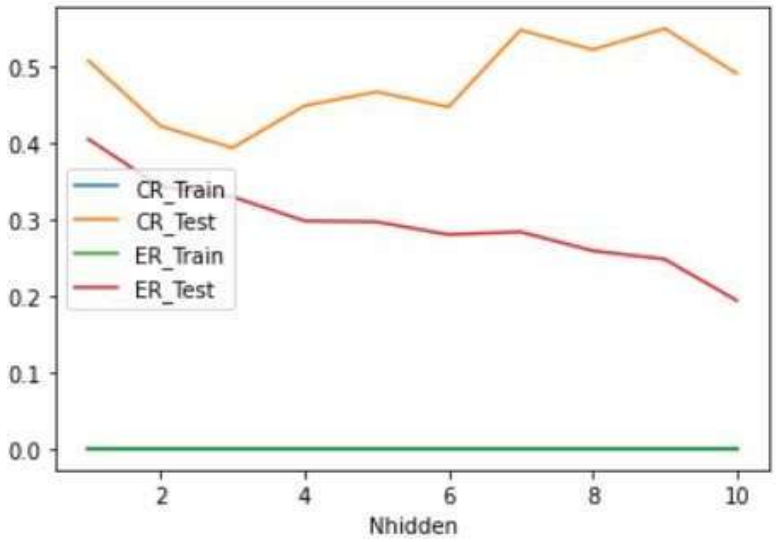


Ilustración 39: De elaboración propia

2. Según delt_t

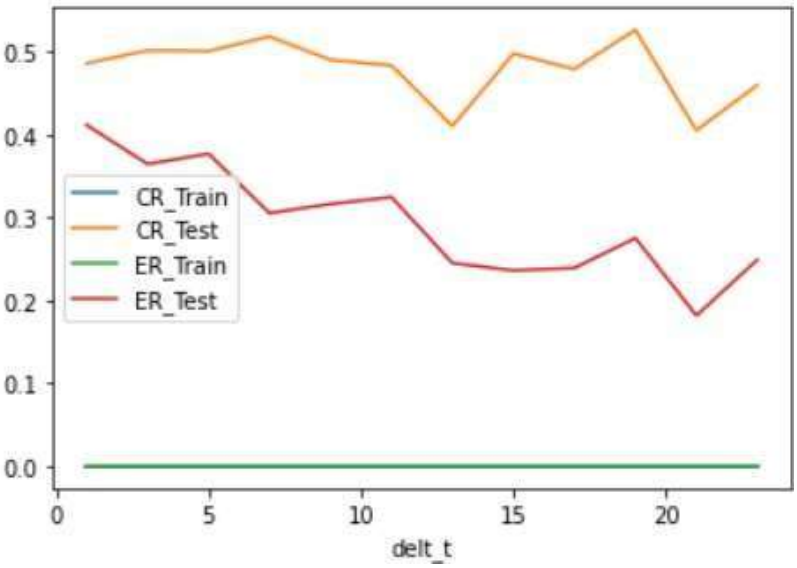


Ilustración 40: De elaboración propia

iv. Modelos seleccionados

	delt_t	Nhidden	iARV_Train	iARV_Test	iuTheill_Train	iuTheill_Test	CR_Train	CR_Test	ER_Train	ER_Test
0	1	1	0.240126	0.095767	38.900550	1.040570	0	0.700663	0	0.446364
1	1	2	0.241115	0.097048	39.333858	0.966348	0	0.647221	0	0.490425
2	1	7	0.241299	0.108240	39.169604	0.979610	0	0.599921	0	0.483637
3	11	3	0.246558	0.112797	38.748661	1.064797	0	0.632772	0	0.464197
4	3	1	0.241696	0.115864	38.978884	0.982861	0	0.534737	0	0.477533

Tabla 8: De elaboración propia

v. Predicciones

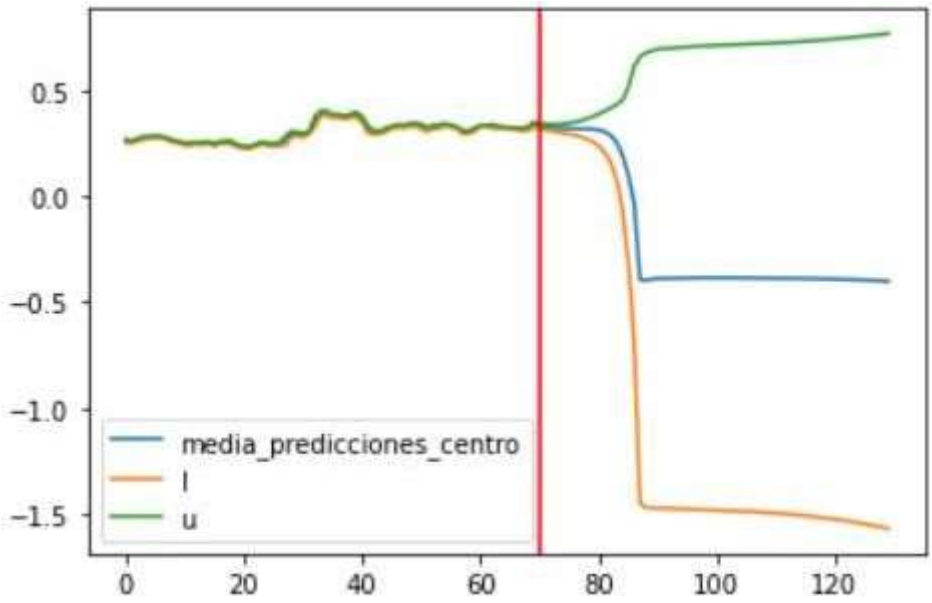


Ilustración 41: De elaboración propia

El periodo bajista seleccionado como referencia a lo que el modelo esperaría se corresponde a uno de los periodos de bajada más abruptos de la historia de la acción. Esto se refleja en una caída inicial prácticamente instantánea que se estanca en torno a un valor absurdo, negativo.

b. Suponiendo comportamiento similar a octubre de 2019
   i. IARV:
      1. Según Nhidden

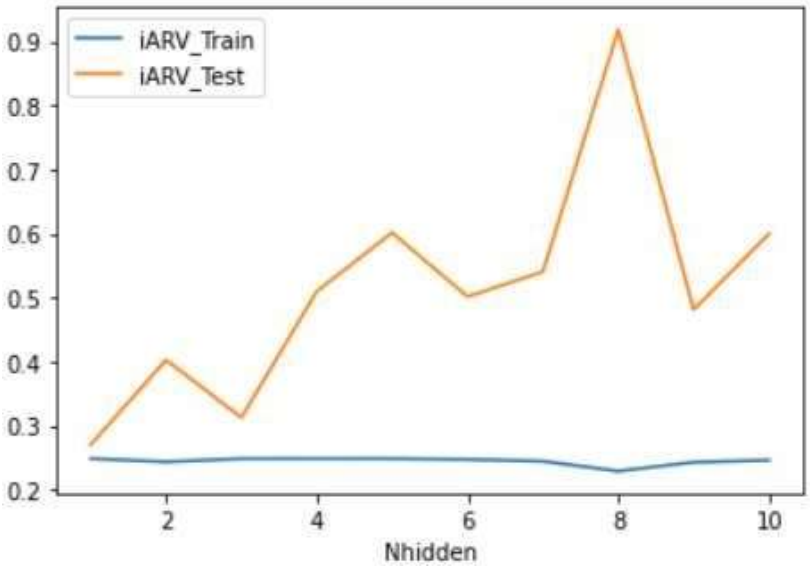


Ilustración 42: De elaboración propia

2. Según delt_t

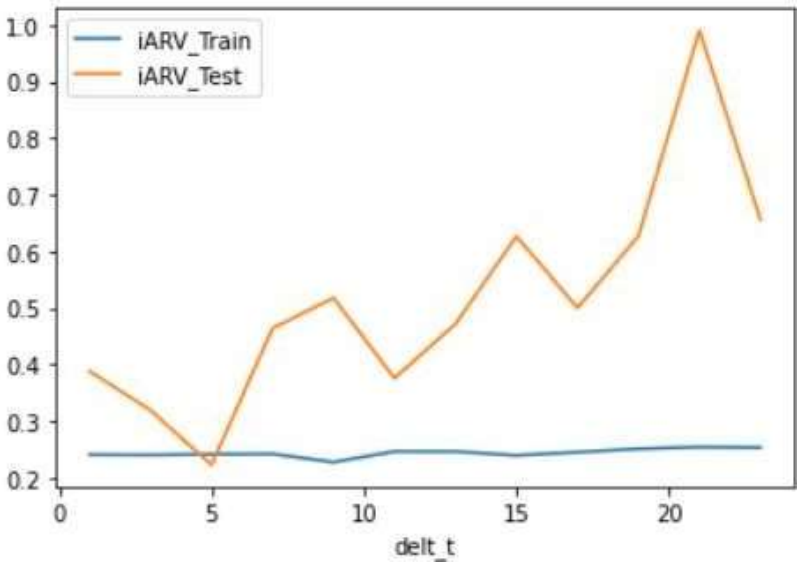


Ilustración 43: De elaboración propia

ii. iuTheill

1. Según Nhidden

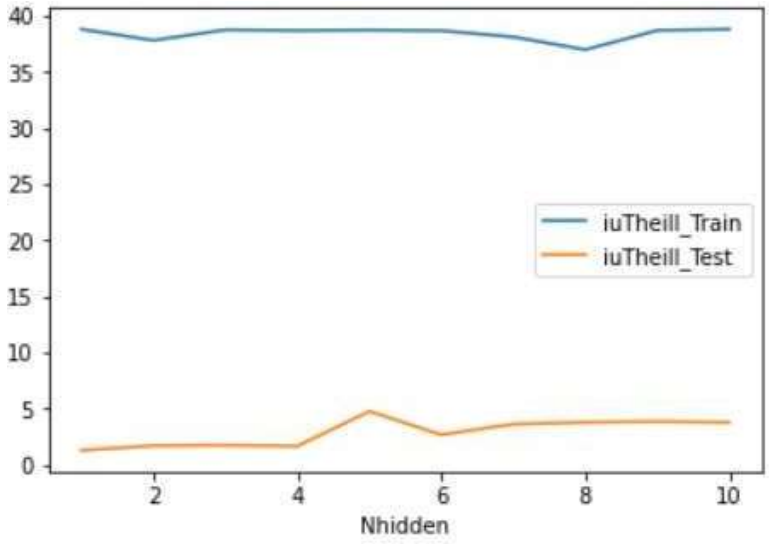


Ilustración 44: De elaboración propia

2. Según delt_t

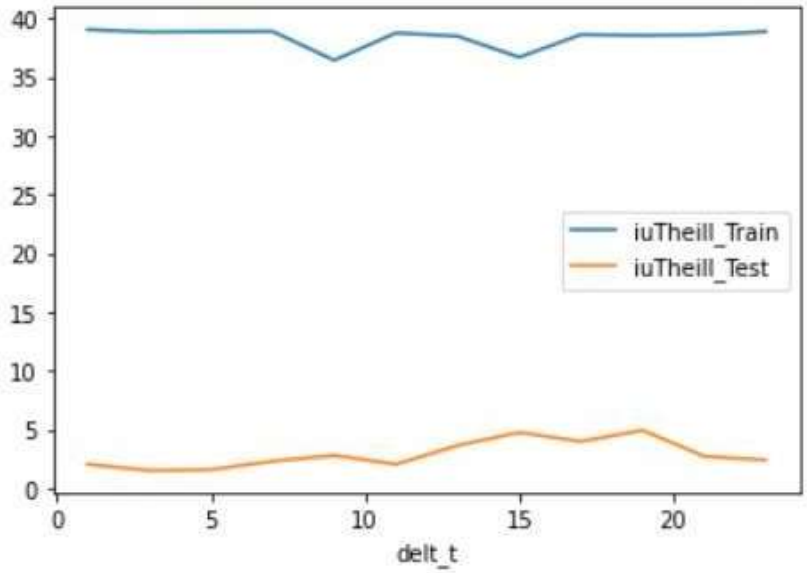


Ilustración 45: De elaboración propia

iii. CR, ER

1. Según Nhidden

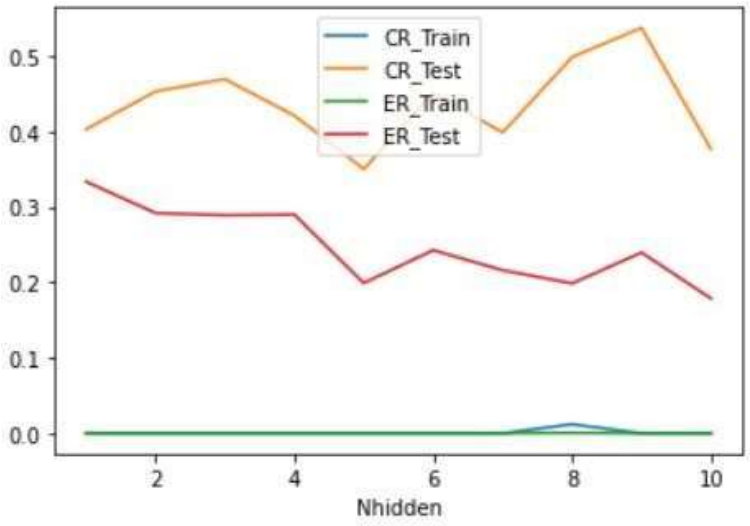


Ilustración 46: De elaboración propia

2. Según delt_t

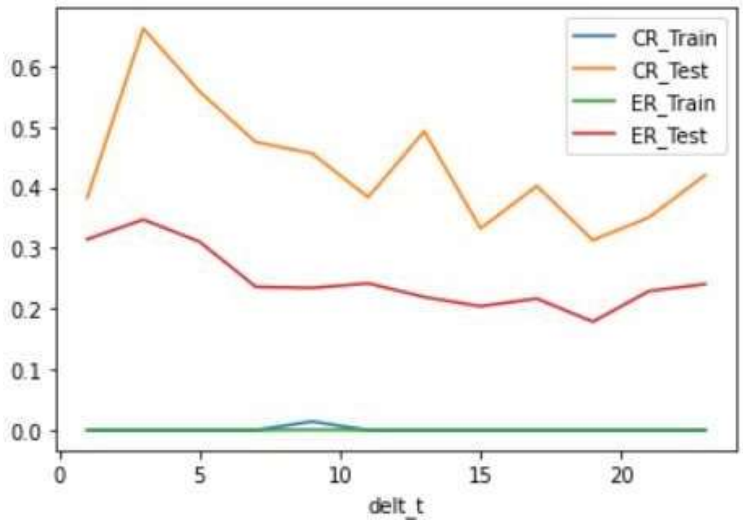


Ilustración 45: De elaboración propia

iv. Modelos seleccionados

	delt_t	Nhidden	iARV_Train	iARV_Test	iuTheil_Train	iuTheil_Test	CR_Train	CR_Test	ER_Train	ER_Test
0	1	1	0.241213	0.105087	39.168409	0.975835	0.0	0.539596	0.0	0.509393
1	1	2	0.241360	0.105593	39.149024	0.973499	0.0	0.548100	0.0	0.505381
2	3	1	0.241581	0.109495	39.033830	0.966391	0.0	0.645374	0.0	0.467858
3	5	6	0.242476	0.130085	38.924918	1.211540	0.0	0.662674	0.0	0.389702
4	3	10	0.241400	0.133430	38.972307	1.090050	0.0	0.777771	0.0	0.429600

Tabla 9: De elaboración propia

v. Predicciones

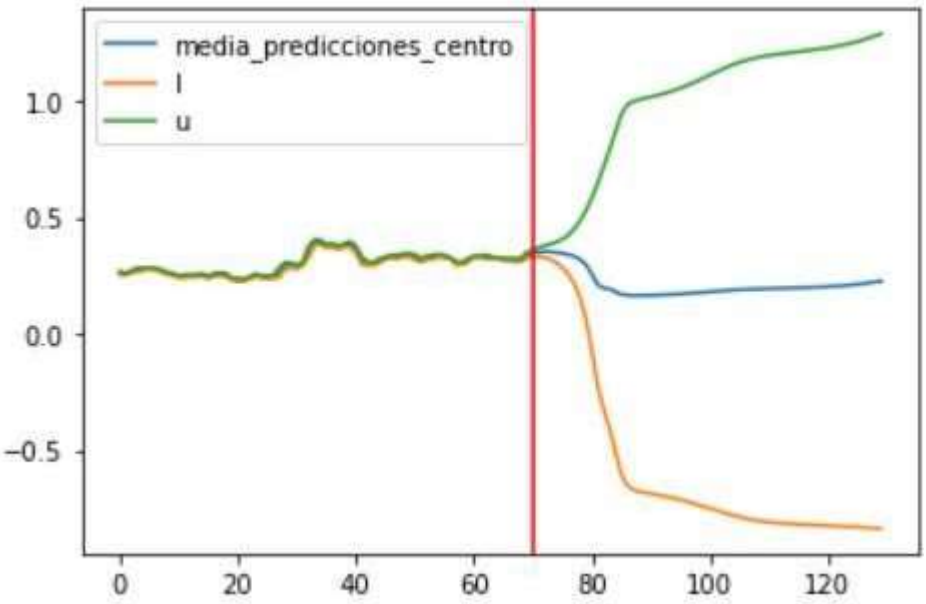


Ilustración 48: De elaboración propia

En este caso, el comportamiento esperado sería alcista, esto se refleja en la tendencia levemente alcista que presentan las predicciones al final del periodo. El bajo valor de los estadísticos CR y ER se confirma al igual que en todos los ejemplos anteriores.

c. Suponiendo comportamiento similar mayo de 2020

i. IARV:
   1. Según Nhidden

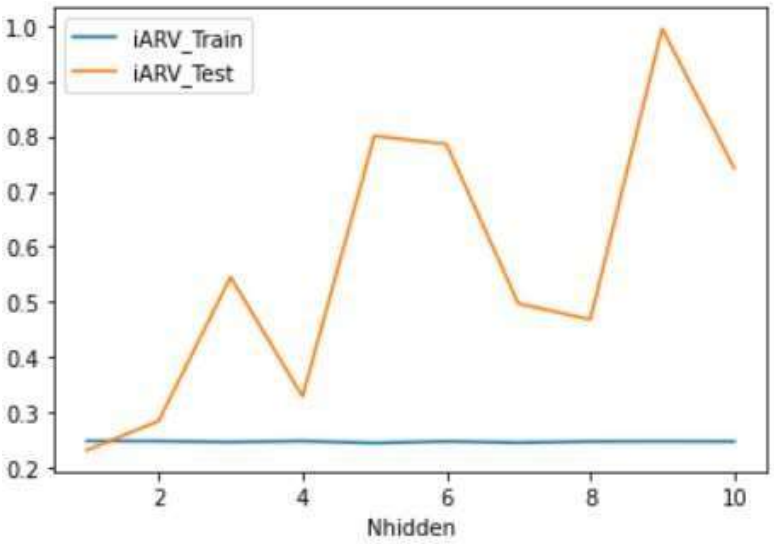


Ilustración 49: De elaboración propia

   2. Según delt_t

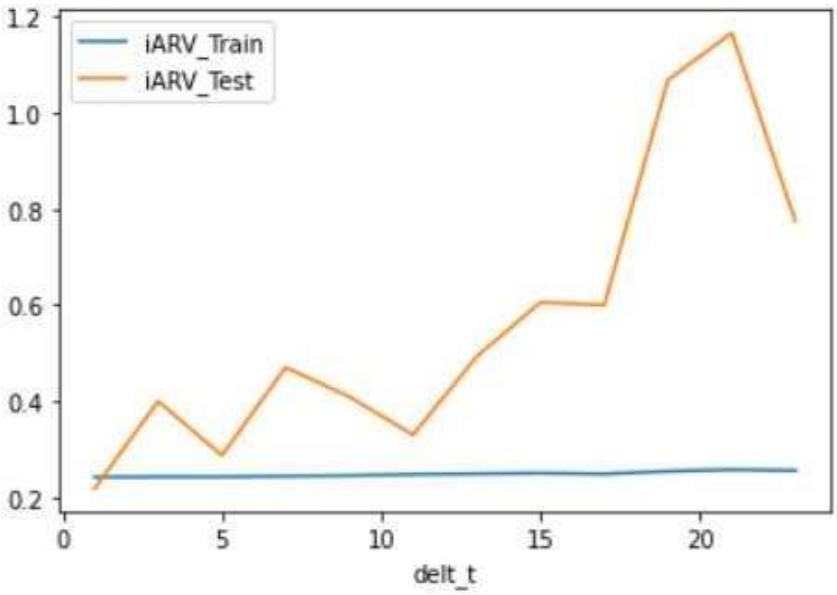


Ilustración 49: De elaboración propia

ii. iuTheill

1. Según Nhidden

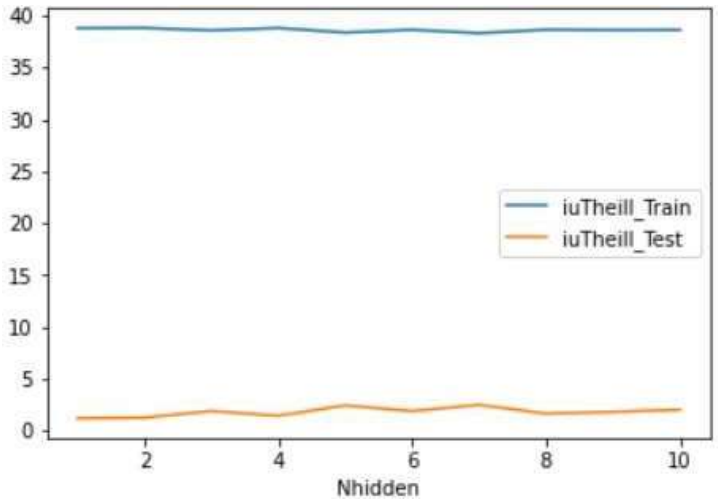


Ilustración 50: De elaboración propia

2. Según delt_t

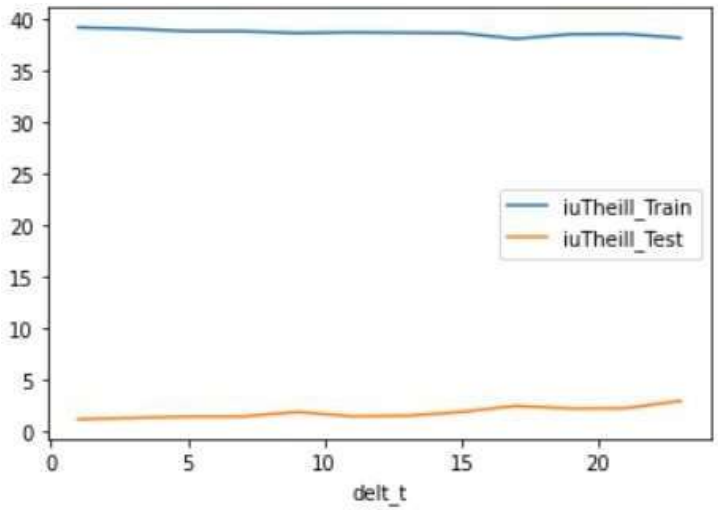


Ilustración 51: De elaboración propia

iii. CR, ER

1. Según Nhidden

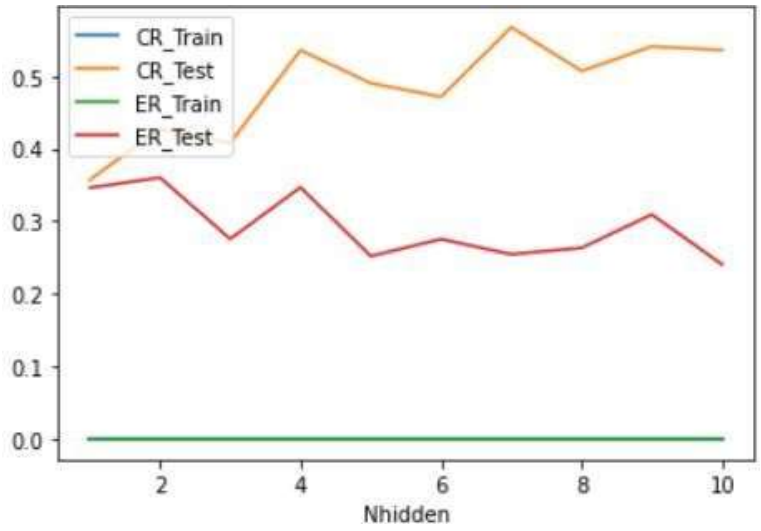


Ilustración 52: De elaboración propia

2. Según delt_t

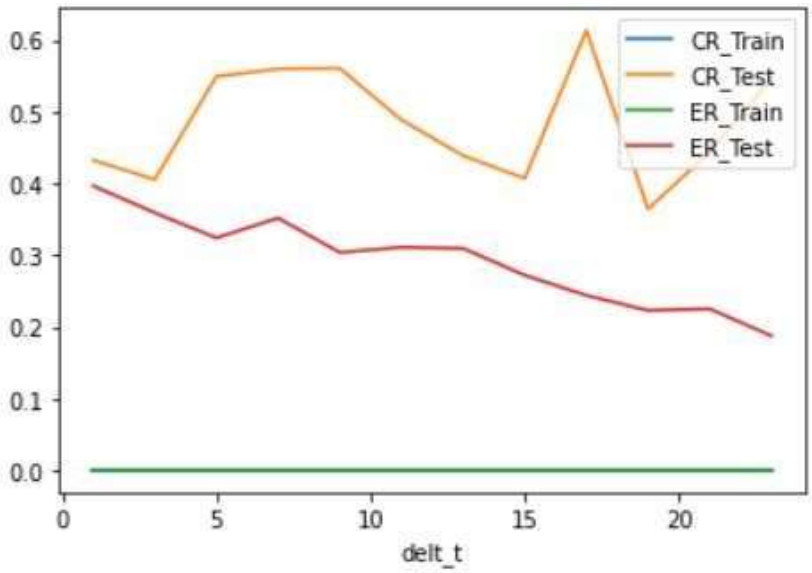


Ilustración 53: De elaboración propia

iv. Modelos seleccionados

	delt_t	Nhidden	iARV_Train	iARV_Test	iuTheil_Train	iuTheil_Test	CR_Train	CR_Test	ER_Train	ER_Test
0	1	7	0.241129	0.101933	39.166350	0.990018	0	0.610310	0	0.496968
1	1	9	0.241289	0.104807	39.144022	1.014467	0	0.585999	0	0.492567
2	1	4	0.241432	0.118754	39.223570	1.000703	0	0.534029	0	0.489977
3	7	3	0.243215	0.120688	38.834452	1.076721	0	0.564569	0	0.461742
4	7	2	0.243209	0.123016	38.824355	1.024127	0	0.499836	0	0.474264

Tabla 10: De elaboración propia

v. Predicciones

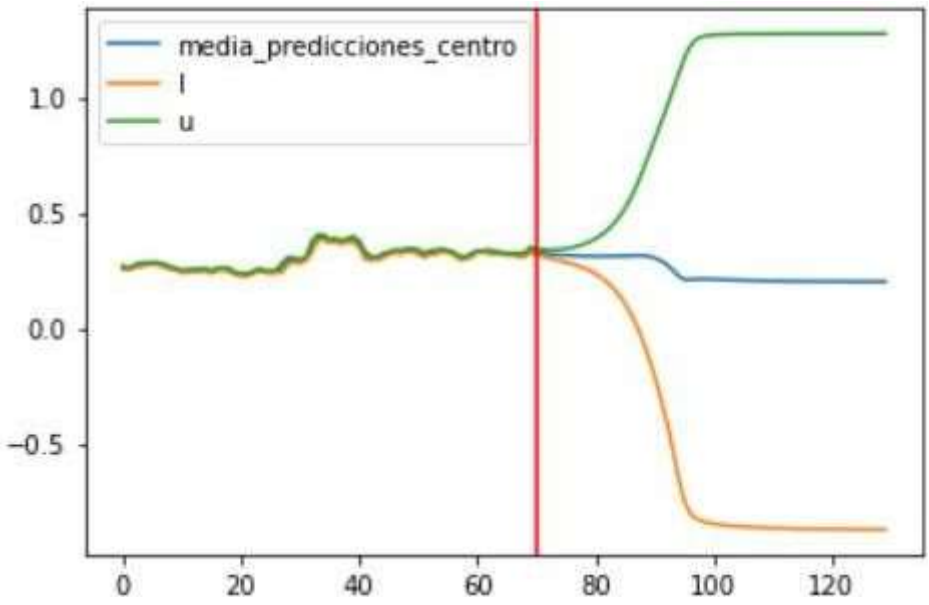


Ilustración 54: De elaboración propia

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.

Como se puede observar, en todos los casos la degradación del modelo comienza en ese momento.

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 lo más amplios posible. De aquí se concluye que el modelo desarrollado no es el más adecuado si se pretende predecir un número elevado de días. No se sabe hasta qué punto sería considerado el número de días aceptable, pero se puede intuir por la degradación presente en todas las gráficas de resultados que es un punto considerablemente bajo, menor a 10.

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 los outliers que afecte a la medida.

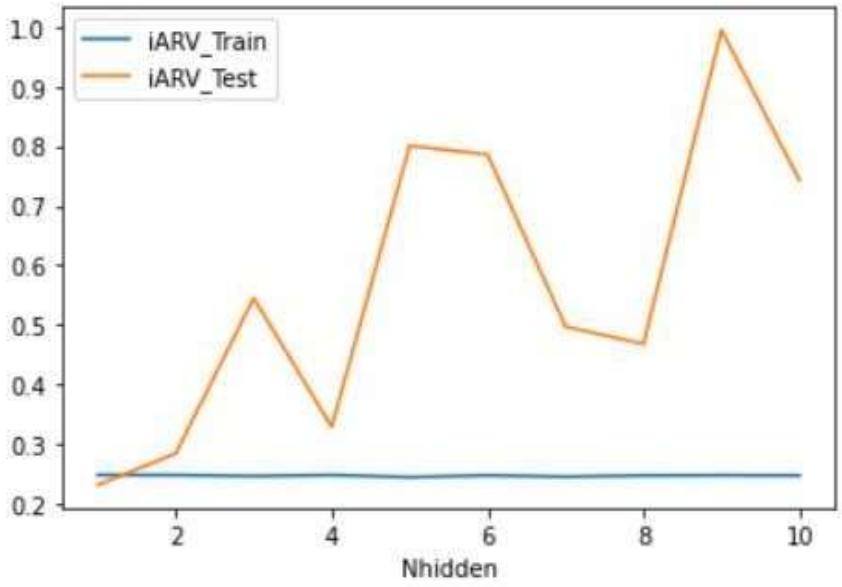


Ilustración 55: iARV vs neuronas ocultas De elaboración propia

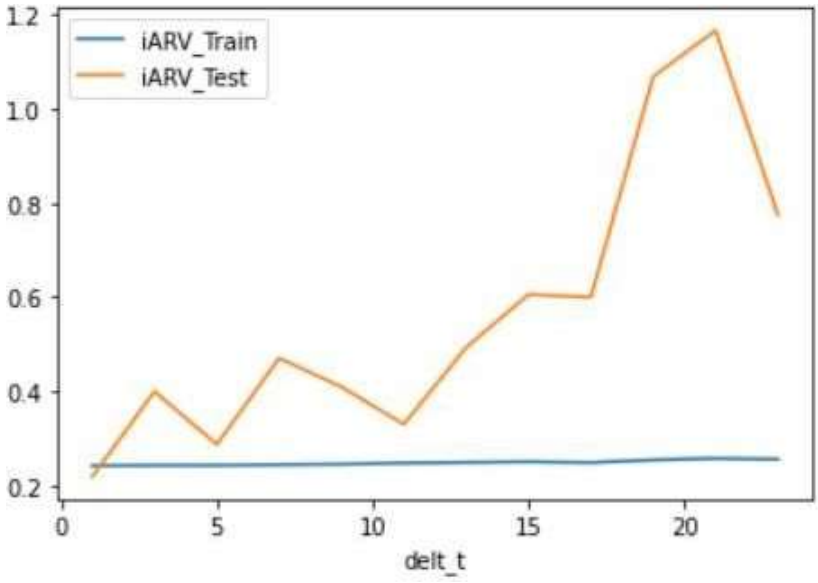


Ilustración 56: iARV vs horizonte De elaboración propia

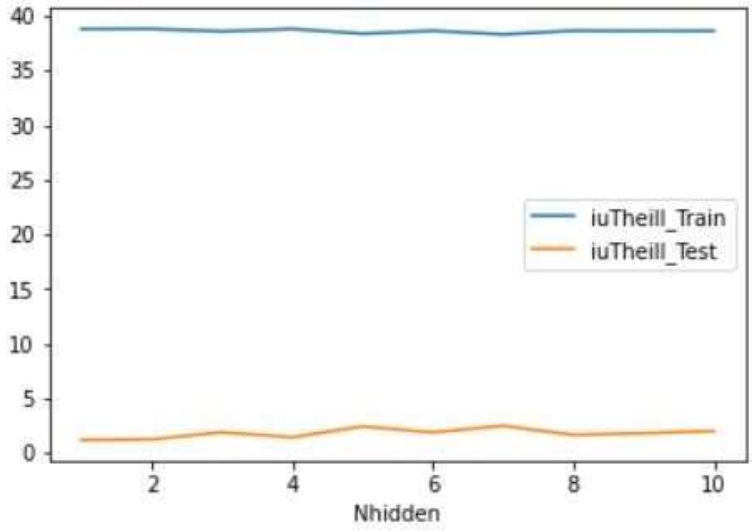


Ilustración 57: U de Theill vs neuronas ocultas De elaboración propia

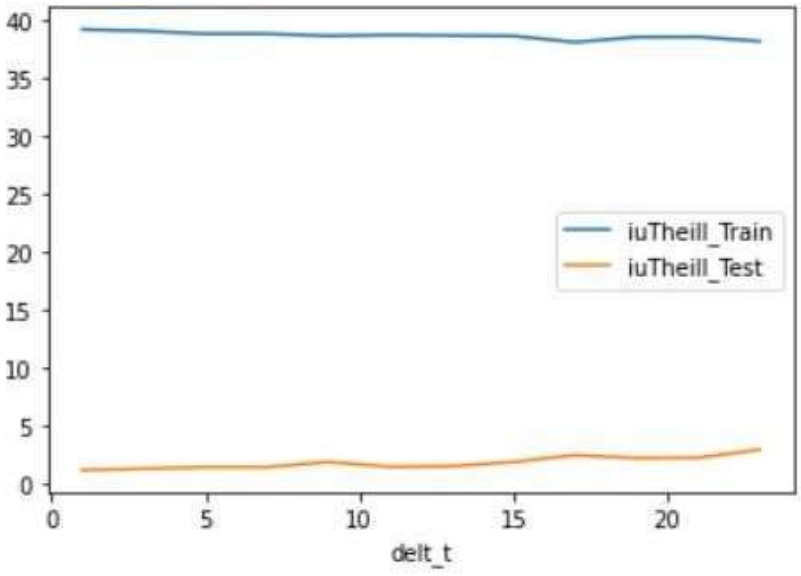


Ilustración 58: U de Theill vs horizonte. De elaboración propia

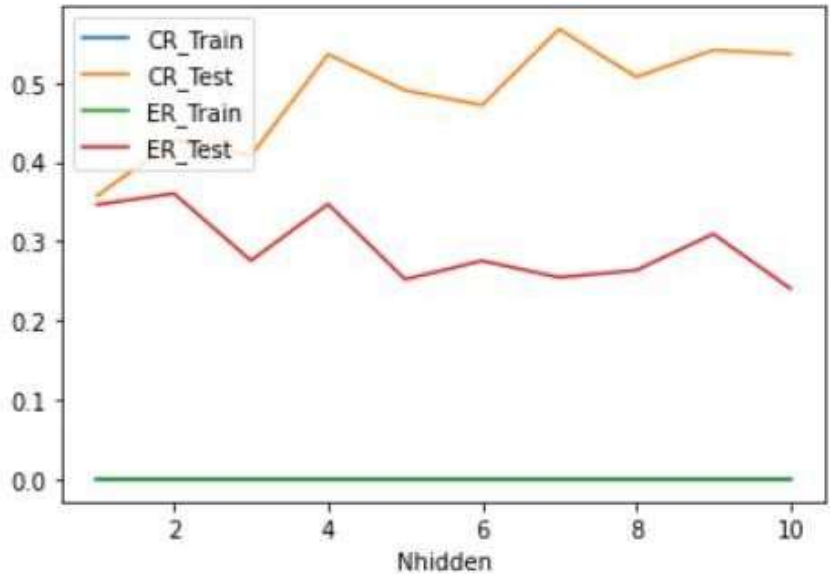


Ilustración 59: CR y ER vs neuronas ocultas De elaboración propia

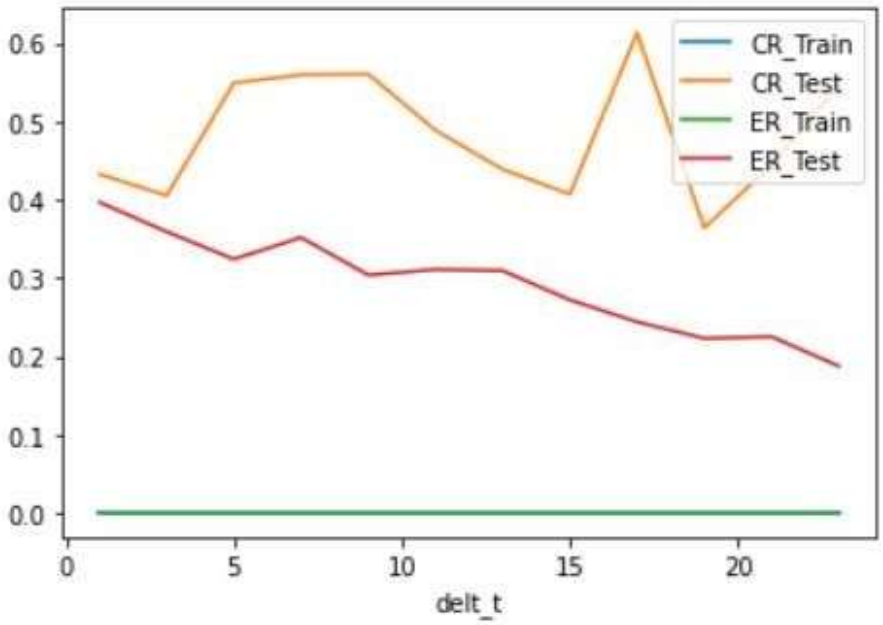


Ilustración 60: CR y ER vs horizonte. De elaboración propia

	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 11: Modelos seleccionados. 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	

Ilustración 61: Inputs utilizados en [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, mejor 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.

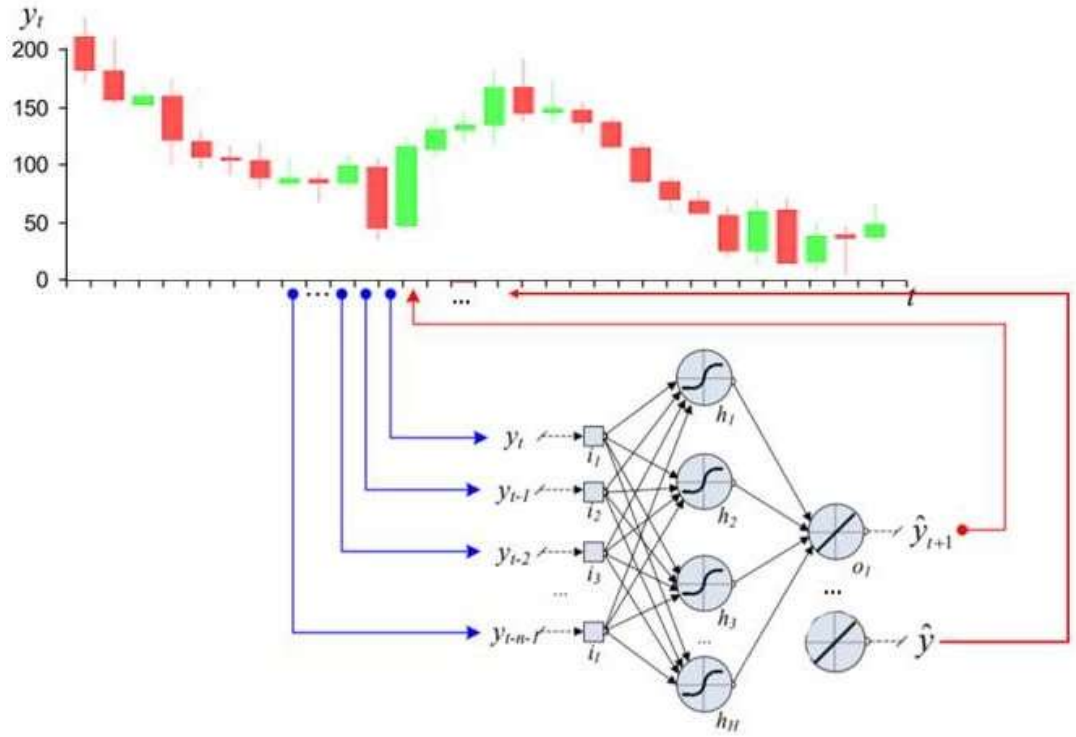


Ilustración 62: Arquitectura final del modelo, con base en [12]

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.

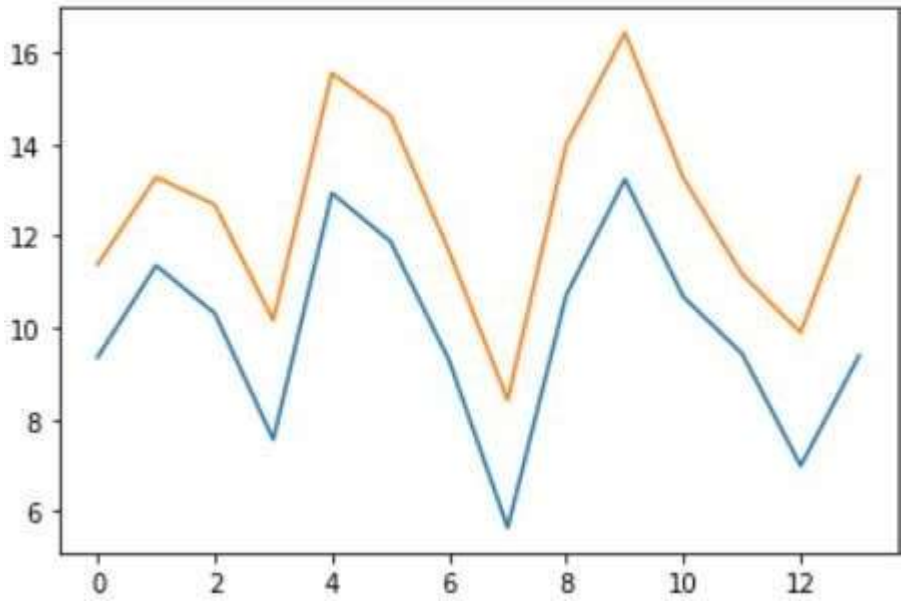


Ilustración 63: Predicciones modelo iMLP final. De elaboración propia.

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.

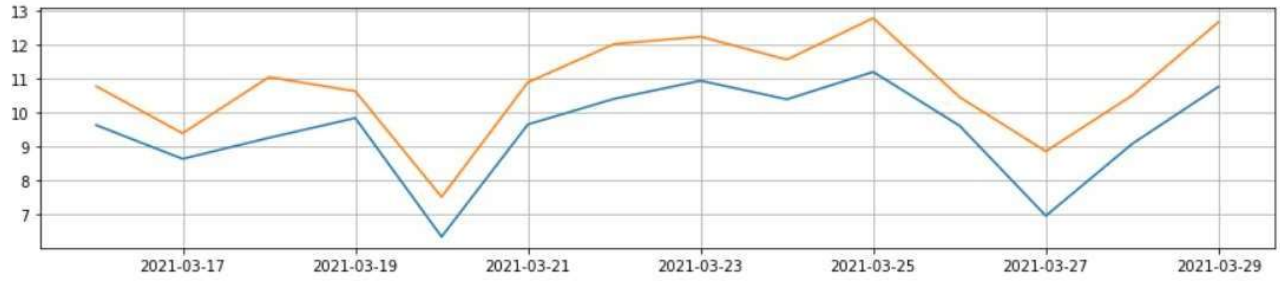


Ilustración 64: Predicciones máximo y mínimo con nco=500, de elaboración propia

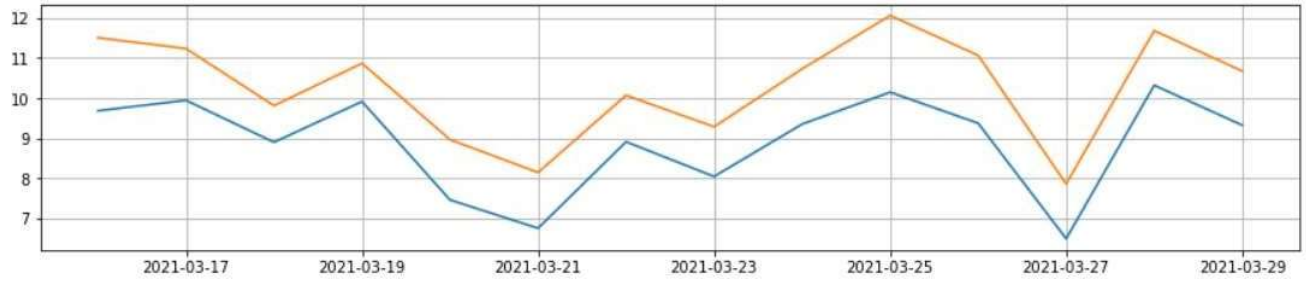


Ilustración 65: Predicciones máximo y mínimo con nco=200, de elaboración propia

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.

Capítulo 6. 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.

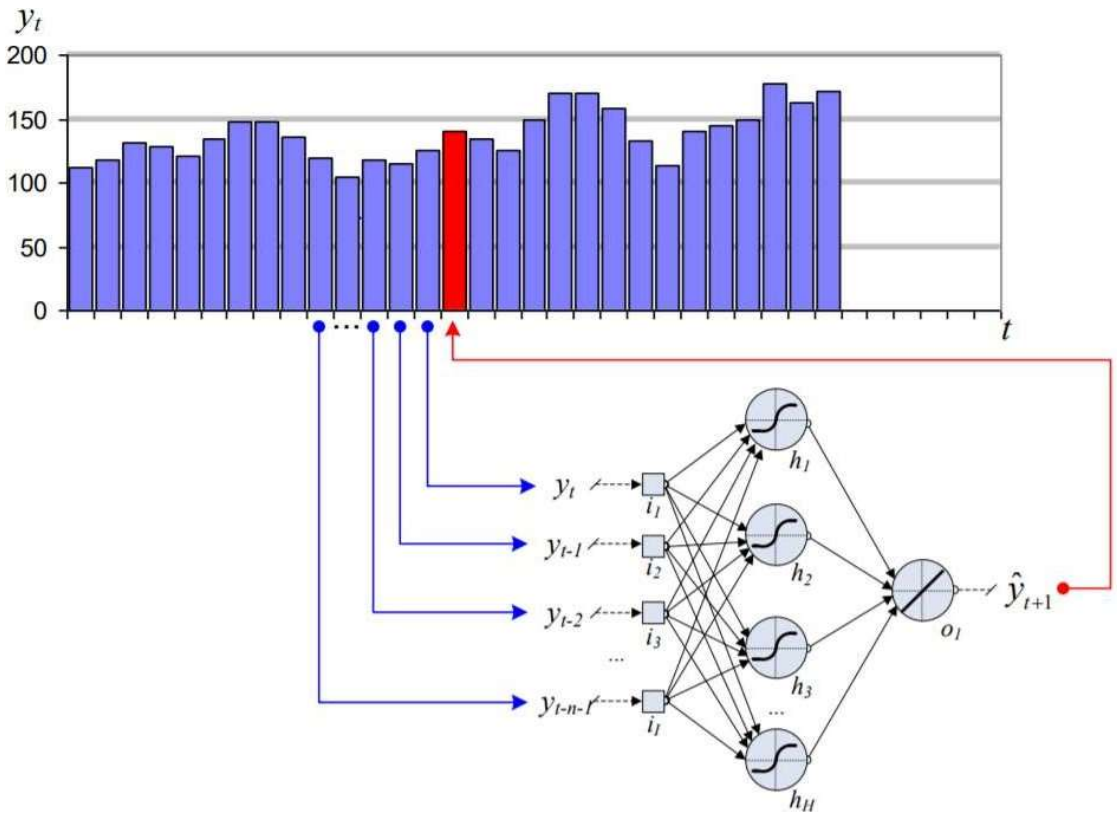


Ilustración 66: Esquema de la arquitectura MLP, tomado de [12]

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.

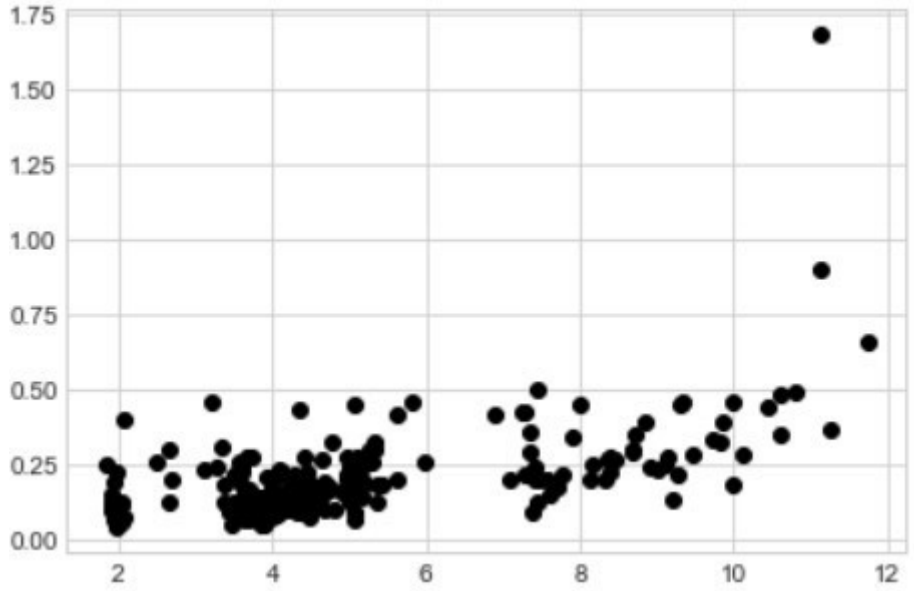


Ilustración 67: : Centros y radios de los puntos del conjunto de datos, de elaboración propia

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 los outliers 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 dataset en tres clústeres aparentemente distintos.

Para poder analizar más en profundidad esta cuestión, es necesario visualizar el dataset 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 68: 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 dataset, 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 los outliers 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.

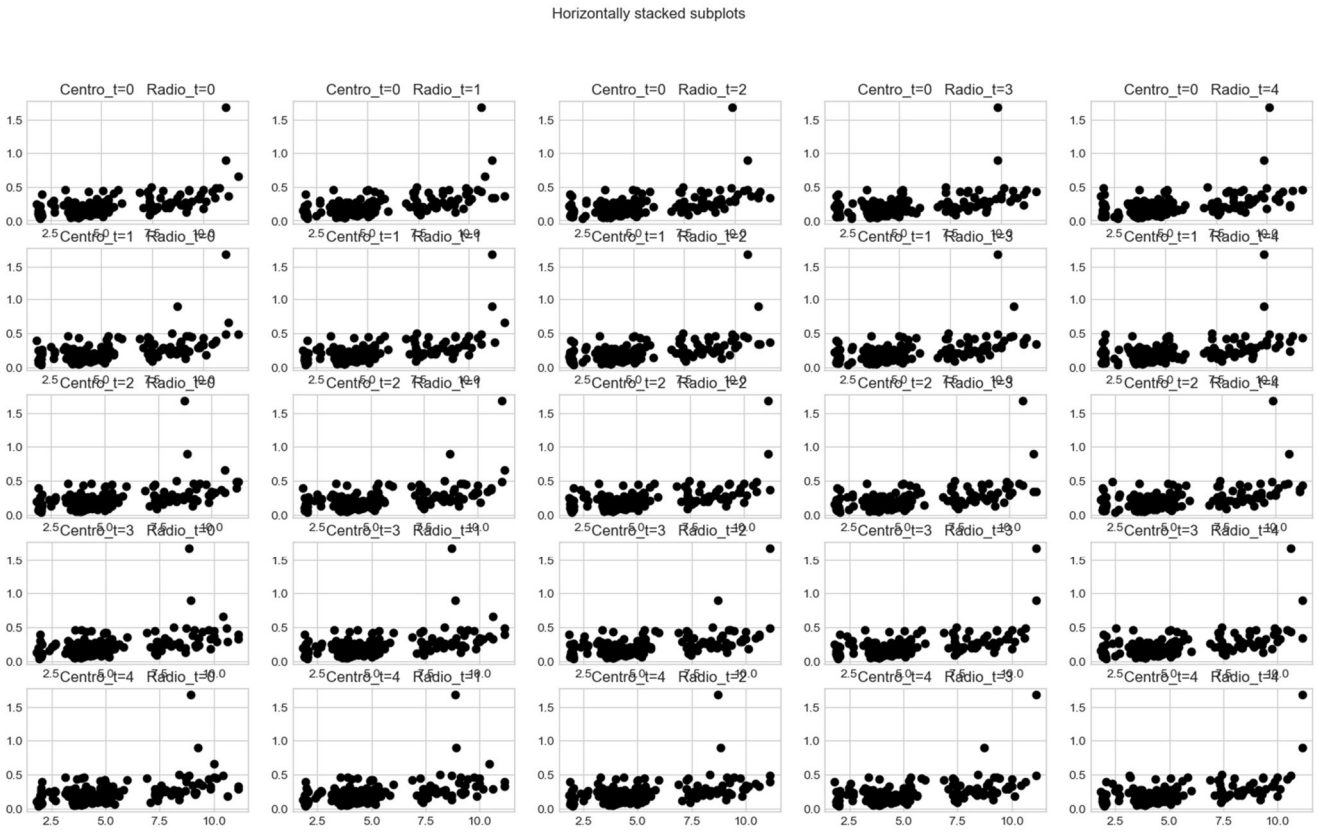


Ilustración 69: Centros y radios para múltiples distancias temporales, de elaboración propia.

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 12: Entradas utilizadas en [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 lidiar con 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.

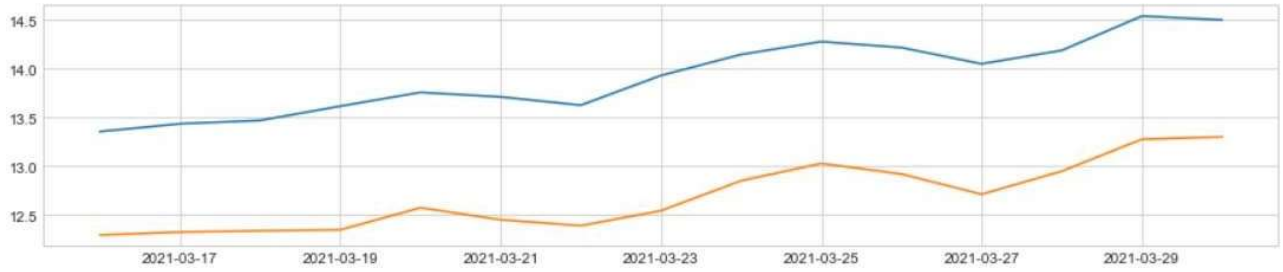


Ilustración 70: Predicciones MLP base, de elaboración propia.

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

$$Máximo = Centro + Radio \qquad (6)$$

$$Mínimo = Centro - Radio \qquad (7)$$

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: 0.8360 - mae: 0.6673 - mse: 0.8360
[0.8359615802764893, 0.667266845703125, 0.8359615802764893]
```

Radios:

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

```
2/2 [==============================] - 0s 5ms/step - loss: 0.0165 - mae: 0.0831 - mse: 0.0165
[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.

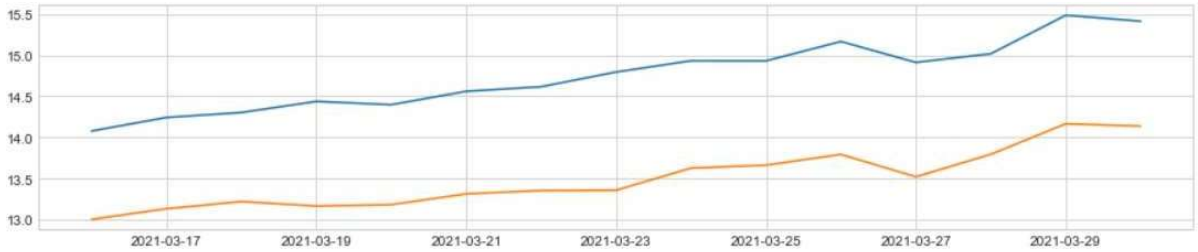


Ilustración 71: Predicciones MLP profundo, de elaboración propia.

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: 0.8404 - mae: 0.6525 - mse: 0.8404
[0.8403813242912292, 0.652463972568512, 0.8403813242912292]
```

Radios:

```
model.evaluate(normed_test_data, test_labels)
2/2 [==============================] - 0s 2ms/step - loss: 0.0163 - mae: 0.0811 - mse: 0.0163
[0.01630994863808155, 0.08114250749349594, 0.01630994863808155]
```

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 sobre-aprendizaje.

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 dataset 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.

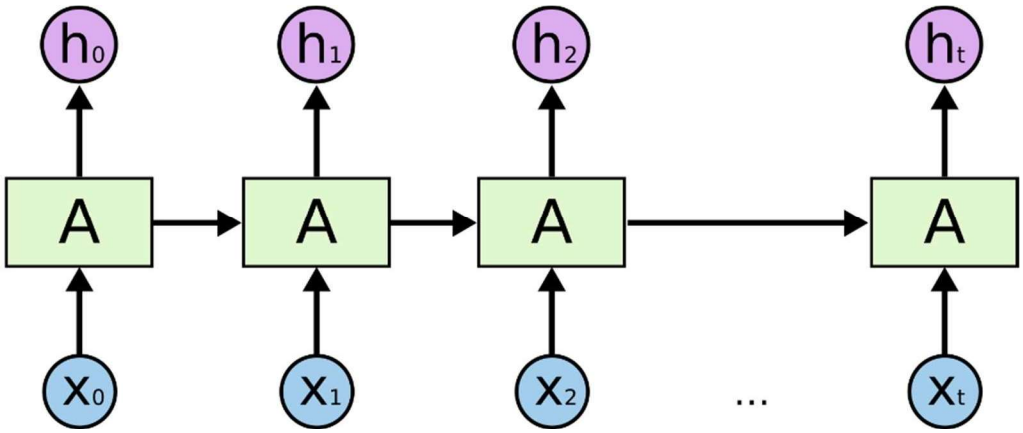


Ilustración 72: Esquema LSTM, tomado de [11].

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 - loss: 6.2798 - mae: 1.9140 - mse: 6.2798
[6.279849052429199, 1.914023756980896, 6.279849052429199]
```

Radios:

```
model.evaluate(normed_test_data, test_labels)
2/2 [==============================] - 0s 2ms/step - loss: 0.0224 - mae: 0.0972 - mse: 0.0224
[0.022372864186763763, 0.09723648428916931, 0.022372864186763763]
```

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

Capítulo 7. 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:

$$x_{i,t} = \frac{L_{i,t} - \bar{L_7}}{D_i} \qquad (8)$$

Donde:

$$D_i = \sqrt{\sum_{l=1}^{n} (L_{i,l} - \bar{L_7})^2} \qquad (9)$$

Asimismo, también se normalizaron las salidas:

$$y_{i,t} = \frac{L_{i+r,t} - \bar{L_7}}{D_i} \qquad (10)$$

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

$$\hat{L}_{i+c,t} = \hat{y_{7,t}} D_i + \bar{L_7} \qquad (11)$$

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

$$z = \frac{x - \bar{x}}{\sigma} \qquad (12)$$

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:

$$p = [\sin(2\pi \frac{\#(i+c)}{366}) , \cos(2\pi \frac{\#(i+c)}{366})] \qquad (13)$$

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.

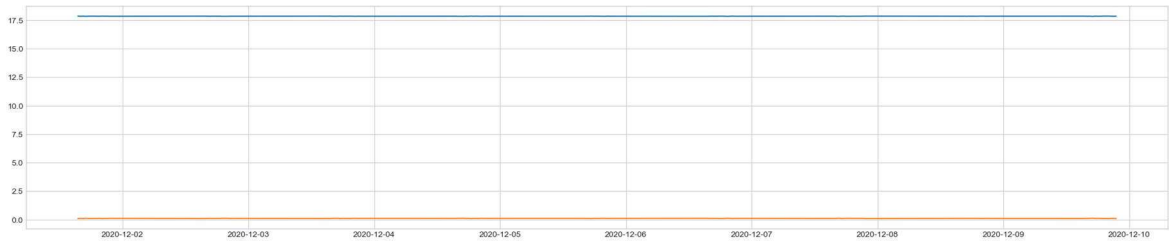
El modelo global que agrupe tanto al del día de la semana como al que particiona por hora tendrá como entradas las siguientes:

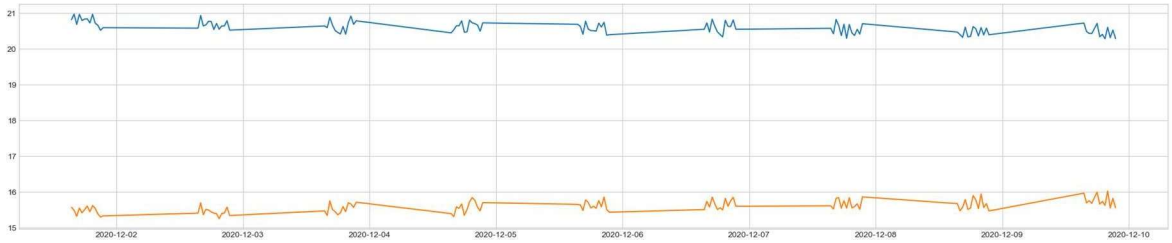
- Vector x con los datos normalizados.
- Codificación del día de la semana.
- Valor de la función de agregación de los puntos del día.

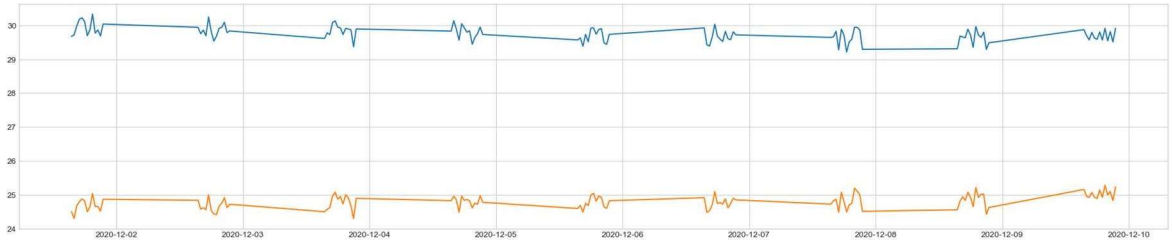
En primer lugar, se trabajó en el diseño y programación del modelo base centrado en los datos de cada media hora. Esto supuso un reto porque el índice temporal ahora sería variable, por lo que se debieron realizar una serie de ajustes.

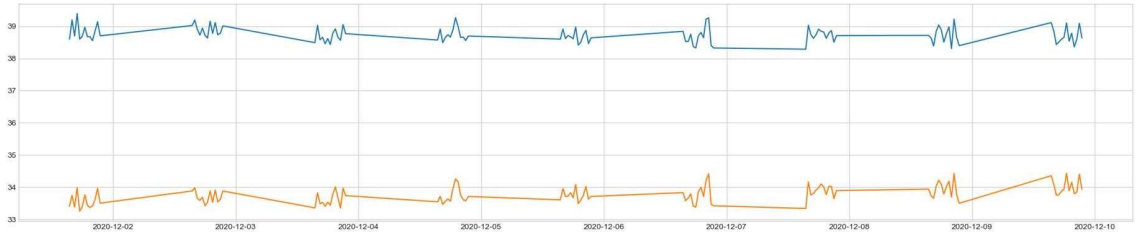
El modelo se entrenó utilizando una red MLP que tuvo una estructura similar a la ya utilizada. Para realizar este entrenamiento se buscó optimizar el hiperparámetro del número de neuronas en la capa oculta para así encontrar la combinación adecuada que produjese los mejores resultados.

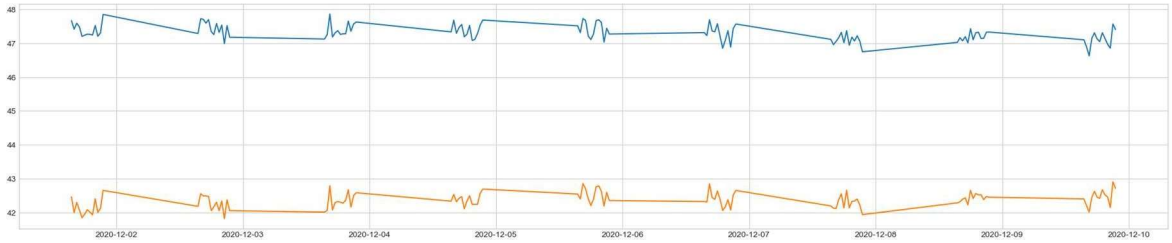
A continuación se muestran las gráficas para los distintos entrenamientos.



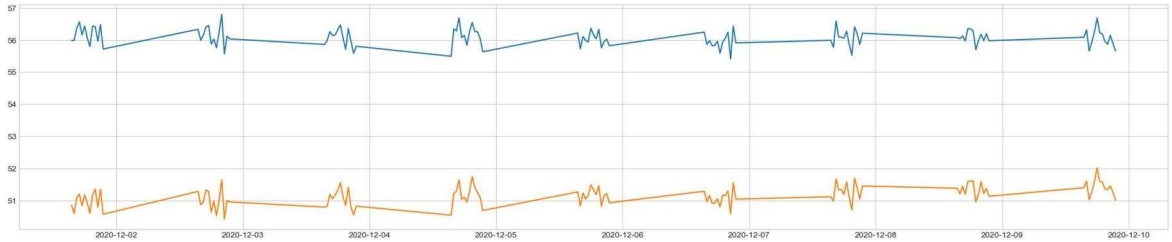


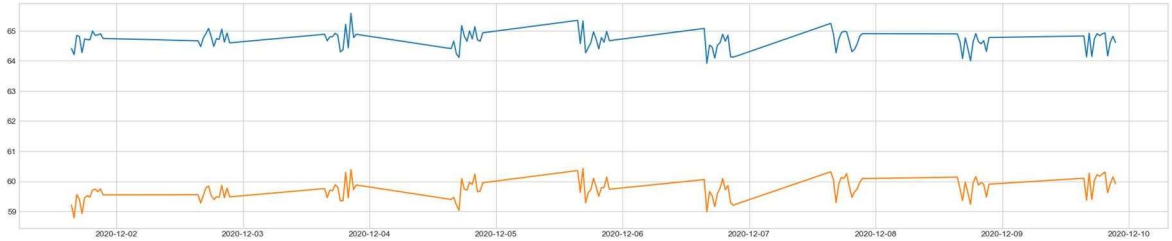


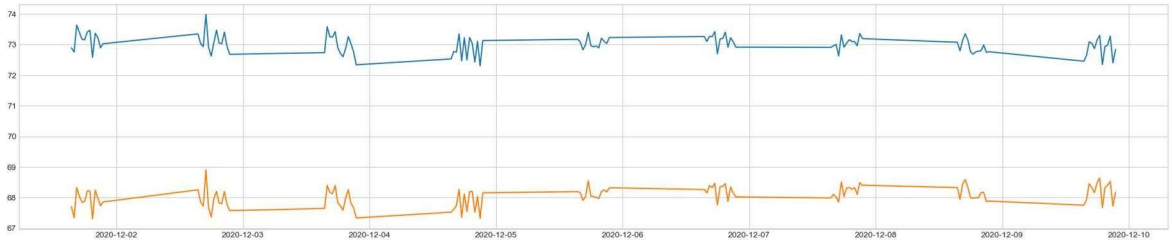


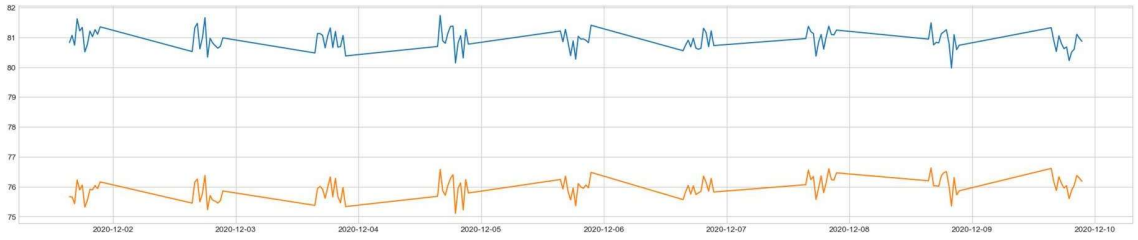


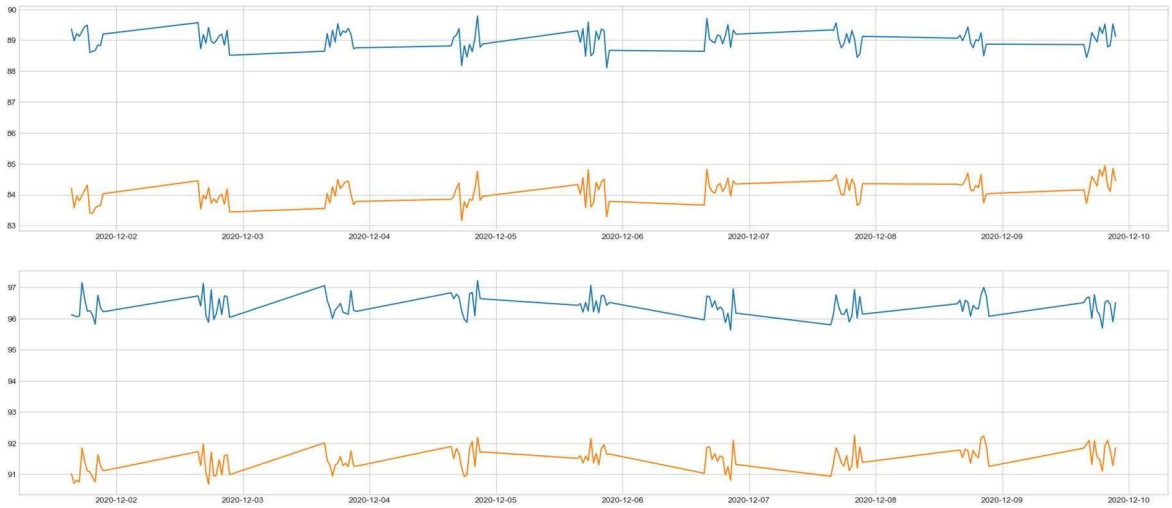
Ilustraciones 73-77: Predicciones modelo Dudek de los entrenamientos 1-5. De elaboración propia.



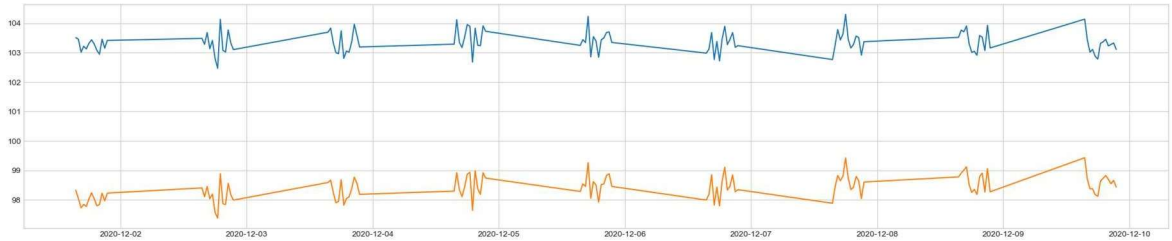


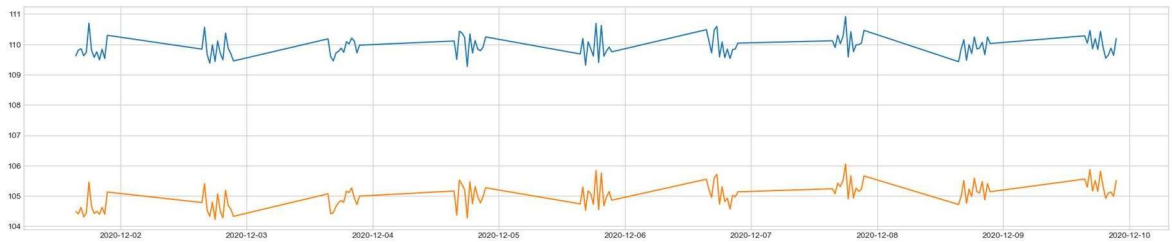


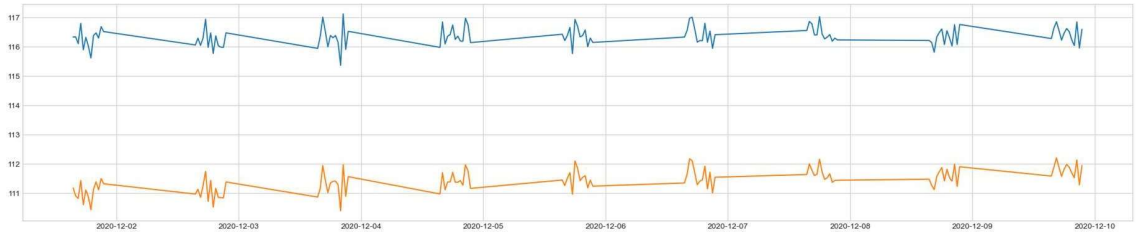


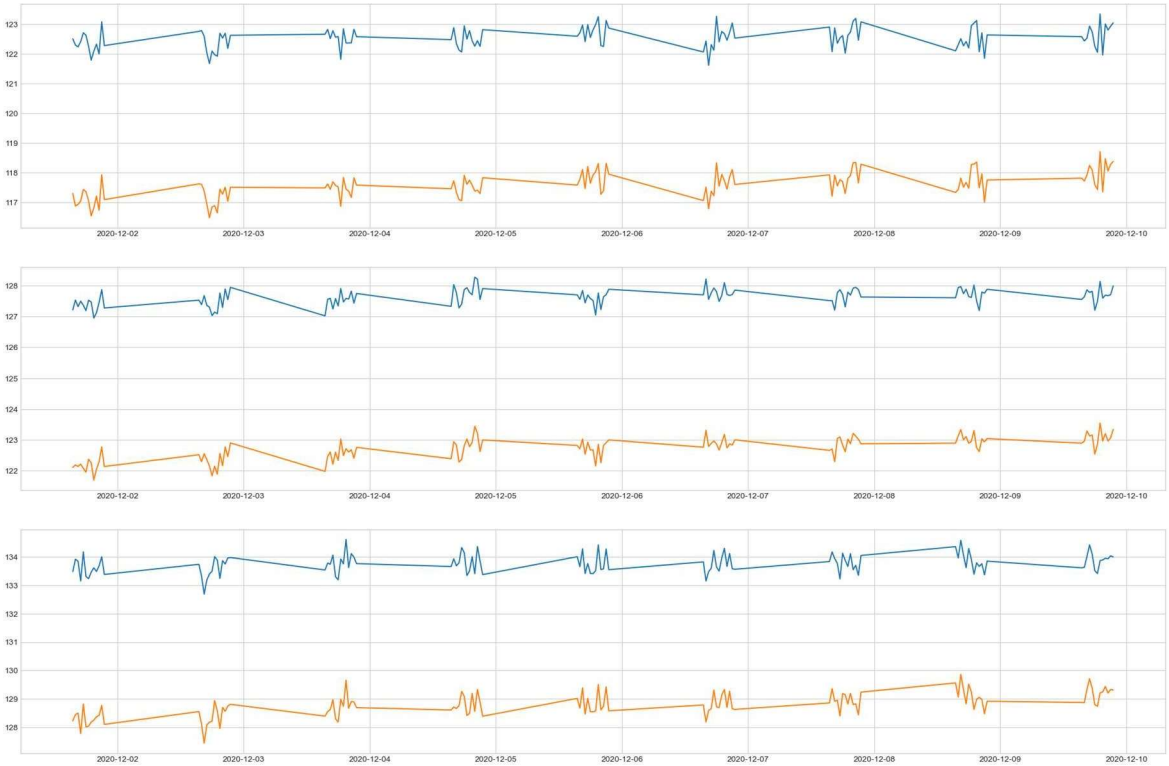


Ilustraciones 78-83: Predicciones modelo Dudek de los entrenamientos 6-11. De elaboración propia.

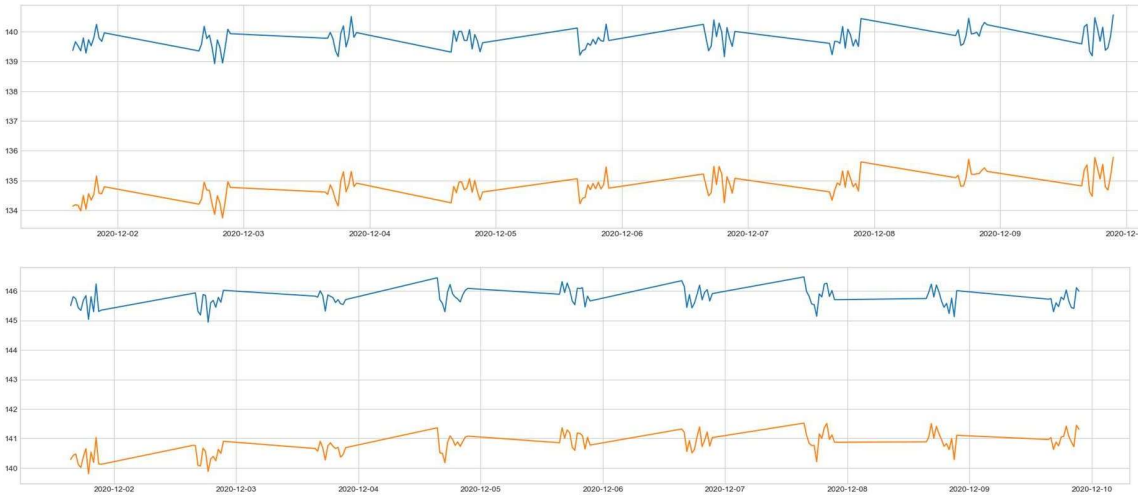


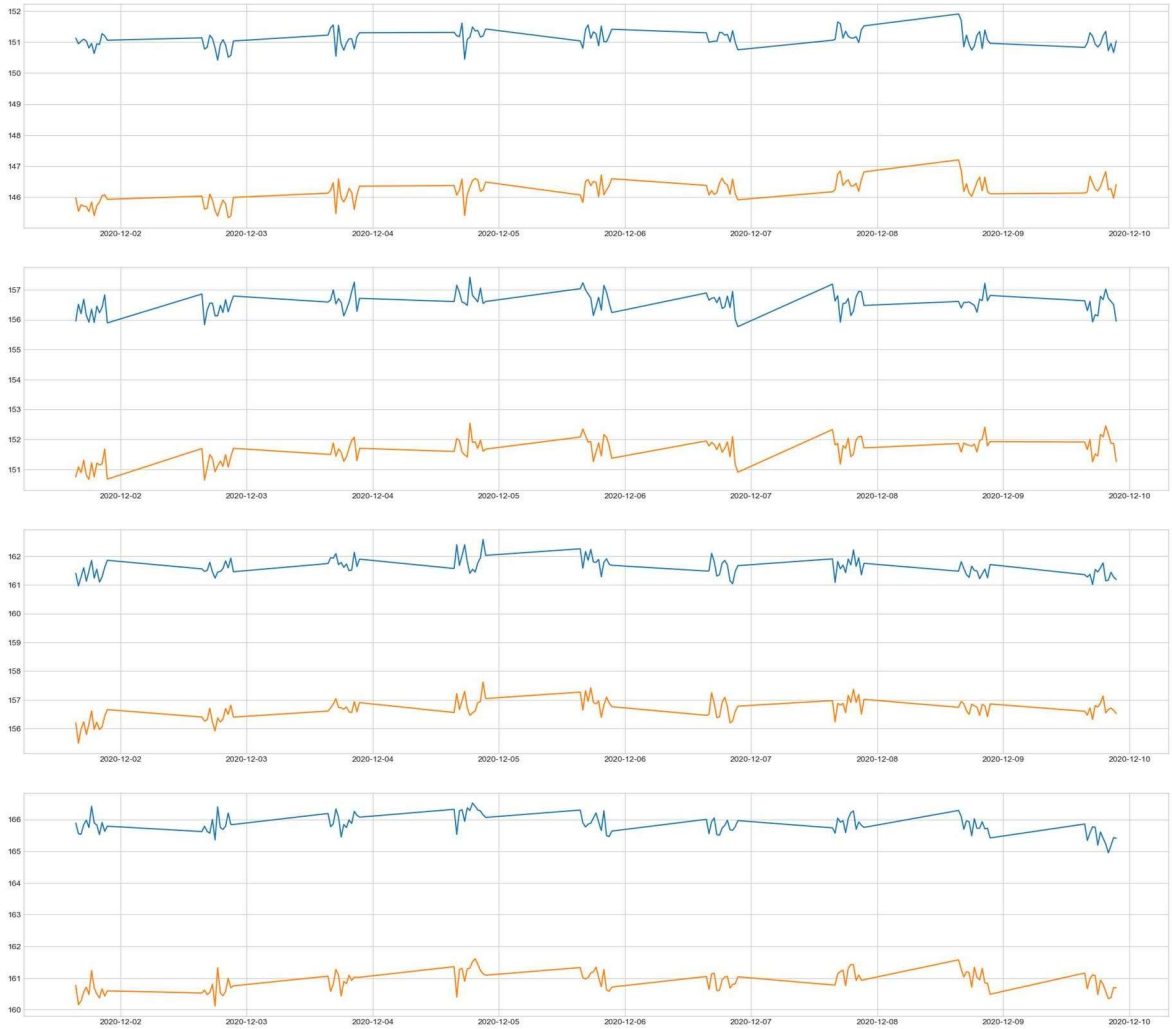




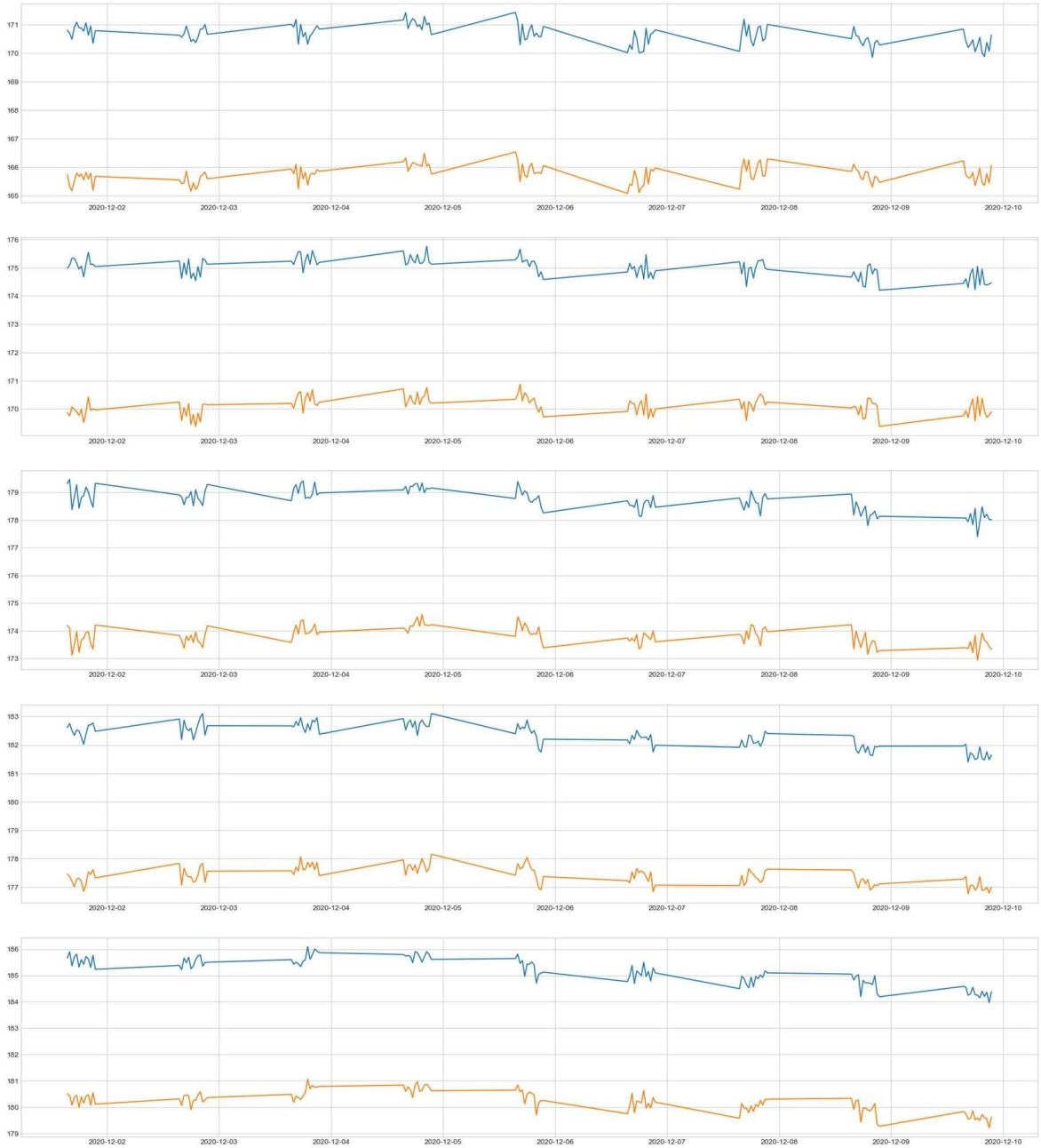


Ilustraciones 84-89: Predicciones modelo Dudek de los entrenamientos 12-17. De elaboración propia.





Ilustraciones 90-95: Predicciones modelo Dudek de los entrenamientos 18-23. De elaboración propia.



Ilustraciones 96-100: Predicciones modelo Dudek de los entrenamientos 24-28. De elaboración propia.

Ilustraciones 101-102: Predicciones modelo Dudek de los entrenamientos 29 y 30. 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 Train.

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.

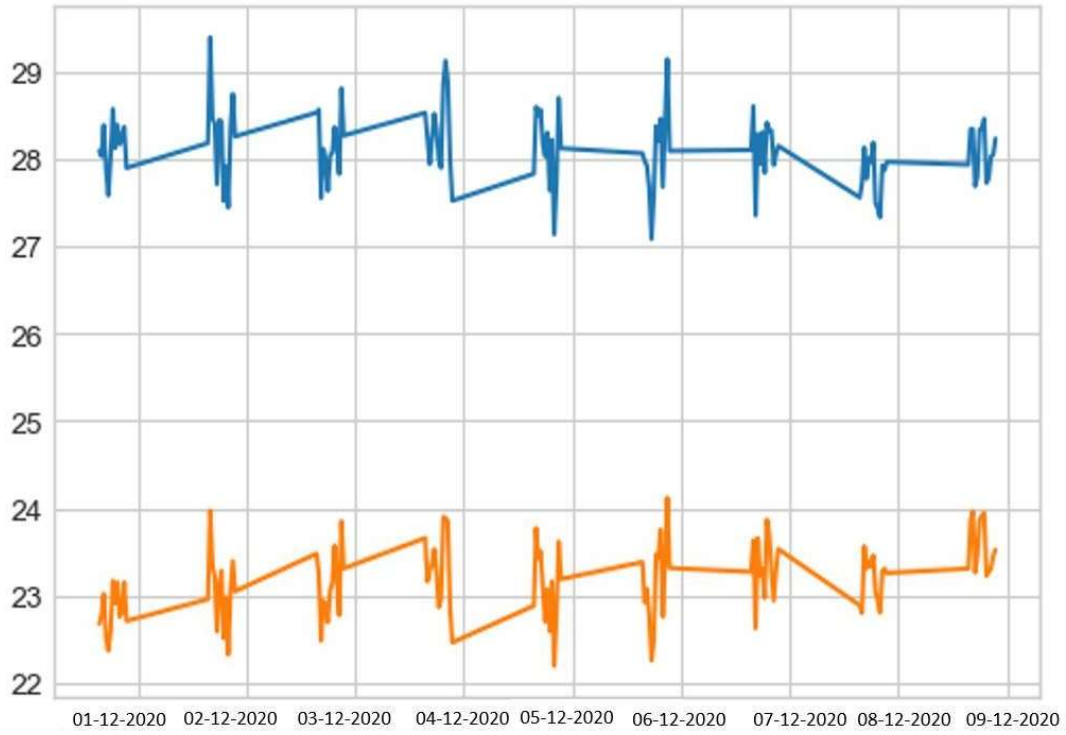
El dataframe resultante del que se extrajeron las entradas y salidas del modelo es el que se muestra a continuación.

	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

Ilustraciones 103: Dataframe resultante de la codificación por día de la semana. 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 train.



Ilustraciones 104: 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.

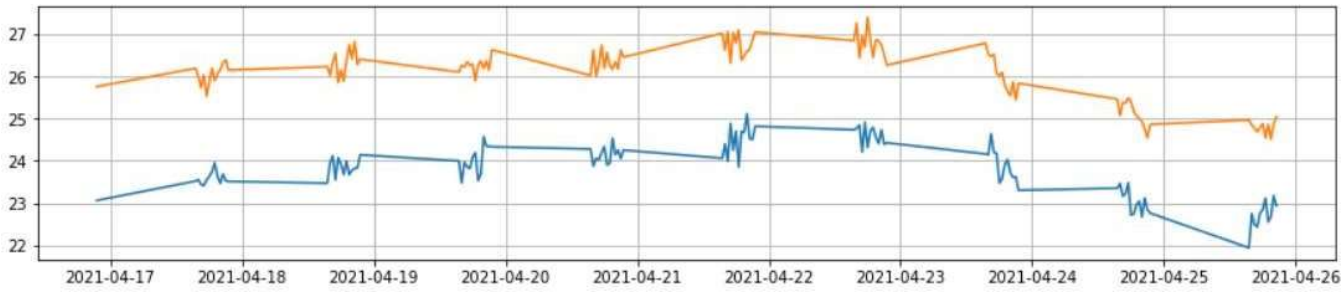
El siguiente paso fue proceder a entrenar el iMLP con el nuevo dataset y ver qué predicciones tendría.

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 dataset 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.

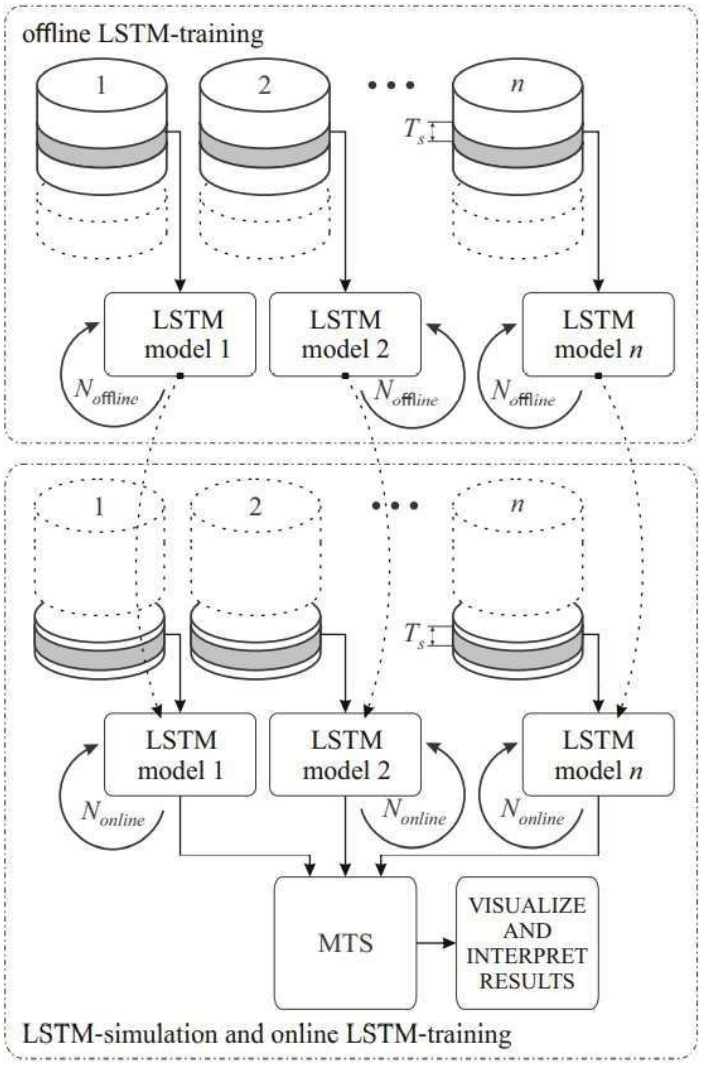
Por esta razón, resultó imposible hacer el entrenamiento y, en su lugar, se tomó una combinación que había funcionado en modelos anteriores. Las predicciones de ese modelo fueron las siguientes.



Ilustraciones 105: 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.

Capítulo 8. MODELO CON MÚLTIPLES ACCIONES



Ilustraciones 106: Esquema arquitectura del modelo multi acción. Tomado de [13]

Una de las formas en las que se podría esperar ver el potencial de las redes iMLP sería elaborando un sistema basado en el que se realizó en [13], donde se crearon múltiples modelos que predecían acciones del índice DAX.

Cada uno de estos modelos se encargaba de predecir una acción del índice, para lo que se entrenaba utilizando una red LSTM por modelo. El resultado fueron una serie de predicciones dispuestas en forma de vector.

Finalmente, utilizando ese último resultado se modeló otra red LSTM adicional para predecir las acciones del índice. Los resultados fueron notablemente mejores que los de los modelos individuales.

Lo que se propone es reproducir ese mismo modelo aplicado sobre un índice, aunque no sería necesariamente el mismo. De hecho, se podrían tomar empresas estadounidenses ya que se ha trabajado con ellas anteriormente en el proyecto.

La principal diferencia estaría en la utilización de redes iMLP en lugar de LSTM para entrenar los distintos modelos creados tanto para predecir cada una de las acciones como para agruparlos y hacer predicciones sobre el índice.

Cabe destacar que la limitación de la profundidad de aprendizaje debido al número de capas reducido podría llevar a problemas y a que el modelo no sea capaz de extraer información al completo de tanta información sin más capas ocultas.

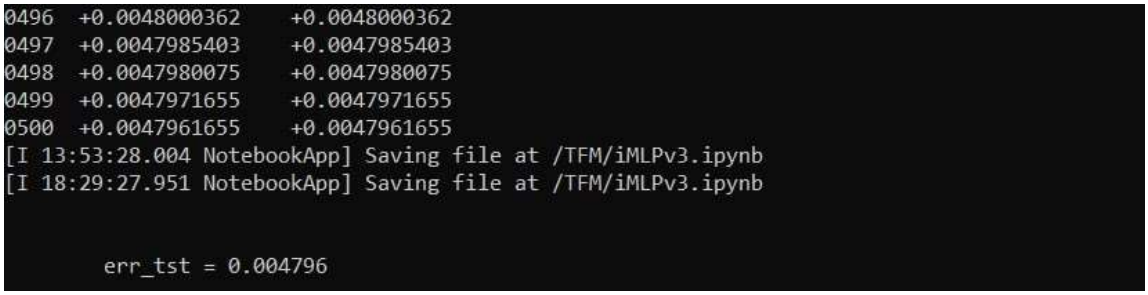
Para poder replicar un modelo análogo al realizado en el trabajo anterior, fue necesario realizar un entrenamiento para cada una de las acciones de un conjunto de acciones americanas. Estas acciones fueron las siguientes:

- ABT US Equity
- ACX SM Equity
- AGRX US Equity
- BCRX US Equity

- BNGO US Equity
- CABK SM Equity

Una vez se realizó el entrenamiento, se hicieron una serie de predicciones de días siguientes que se almacenarían en un archivo independiente. Este proceso se realizó para todas las acciones escogidas.

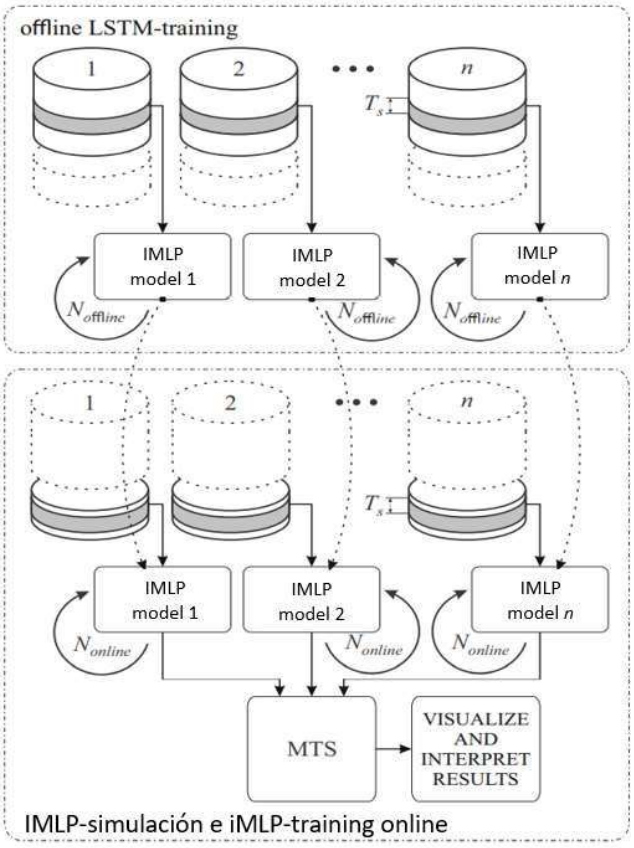
Con todos los datos resultantes, se pudieron juntar en un modelo unificador de cada una de las predicciones surgidas de los modelos para así poder utilizar la información de otras acciones adicionales para predecir la de otra.



Ilustraciones 107: Resultado de la ejecución del modelo

El error que se mostró en el conjunto de validación fue de 0.004796.

El modelo final tendría una arquitectura como la que se muestra a continuación.



Ilustraciones 108: Esquema arquitectura del modelo multi acción iMLP. Basado en [13]

Se trata de un esquema similar al estudiado con la diferencia de que se utilizaron redes iMLP en lugar de LSTM, lo que permitió testear este tipo de agregación de modelos utilizando una variación distinta del modelo.

La principal limitación fue la inflexibilidad del código del iMLP, que no permitía conseguir la arquitectura deseada para el modelo.

Capítulo 9. CONCLUSIONES Y TRABAJOS FUTUROS

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 tiende a ser la que más interesa en el mundo del trading intradía, 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 dataset.

Como ya se explicó, el dataset 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 los outliers 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.

A pesar de que la red LSTM parecía que iba a ser a priori el modelo con mejores resultados debido a su idoneidad para la predicción de series temporales, los resultados no fueron especialmente buenos dado que acabaron siendo similares a los mostrados en el MLP.

El desarrollo del código que sirvió de puente entre C y Python aporta mayor flexibilidad y la oportunidad de aprovechar la facilidad para la carga y manipulación de datos con códigos sencillos de Python con la rapidez que proporciona C a bajo nivel. Una forma de ampliar el trabajo sería hacer desarrollos sobre el código para incrementar el número de capas ocultas del modelo iMLP.

Los modelos entrenados sobre los que se introdujo la información codificada de la tendencia esperada fueron capaces de captar esa expectativa de tendencia en el largo plazo. Esto se podía visualizar con claridad en las predicciones más degradadas en calidad, las de horizonte más lejano.

El reto de adaptar el modelo que predice un solo punto futuro supuso un aumento de la complejidad de la red. Esto puede suponer un dilema a la hora de decidir si hacer un modelo que haga predicciones solamente para un instante inmediatamente posterior o si es mejor que prediga varios. Es cierto que predecir más puntos aumenta significativamente los tiempos de procesamiento y que, dependiendo de las posibles aplicaciones, lo que interese sea hacer predicciones rápidas del instante siguiente para poder actuar mejor más rápido, por lo que tendría sentido hacer predicciones con un horizonte mínimo.

Sin embargo, también cabe destacar que si se obtienen más predicciones del futuro entonces se tendrá una medida relativa de la calidad de las predicciones en base a su dispersión.

El modelo desarrollado que combinaba las salidas de todas las acciones agregándolas en un único modelo resultó ser muy costoso de probar debido, de nuevo, a las limitaciones del código utilizado. Si se utilizara un horizonte amplio y suficientes acciones como, por ejemplo, las 35 empresas del Ibex 35, entonces el tiempo de entrenamiento podría ser muy elevado, llegando, muy probablemente, a varios días.

Es importante remarcar que la incompatibilidad de las métricas de valoración de modelos entre los que utilizan datos crisp y los que están adaptados al dato en forma de intervalo ha supuesto un problema constante a lo largo de todo el proyecto.

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.

$$f(S) = 0.5 * (S_{\text{train}} + S_{\text{test}}) - |S_{\text{train}} - S_{\text{test}}|^2 \qquad (14)$$

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.

$$f(S) = 0.5 * (S_{\text{train}} + S_{\text{test}}) - \sqrt{|S_{\text{train}} - S_{\text{test}}|} \qquad (15)$$

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.

$$F(\Omega) = \sum_{S\in\Omega} f(S) = \sum_{S\in\Omega}(0.5 * (S_{\text{train}} + S_{\text{test}}) - \sqrt{|S_{\text{train}} - S_{\text{test}}|}) \qquad (16)$$

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 13: 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.

$$S' = \frac{S - S_{min}}{S_{max} - S_{min}} \tag{17}$$

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

Capítulo 10. BIBLIOGRAFÍA

[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.

[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] FISTER, Dušan; PERC, Matjaž; JAGRIČ, Timotej. Two robust long short-term memory frameworks for trading stocks. Applied Intelligence, 2021, p. 1-19.

[14] Maté, C., & Jiménez, L. (2021). Forecasting exchange rates with the iMLP: New empirical insight on one multi-layer perceptron for interval time series (ITS). Engineering Applications of Artificial Intelligence, 104, 104358.

ANEXO: ALINEAMIENTO CON ODS

El trabajo desarrollado tiene aplicaciones en el sector financiero, está orientado al uso de los datos públicos de la forma más óptima posible para que el inversor pueda beneficiarse de ello.

Sin embargo, existe otro grupo de stakeholders sobre el que generalmente no se consideran los efectos de esta clase de proyectos. Se trata de las empresas analizadas por el inversor. El objetivo de este proyecto consiste en proporcionar información al inversor sobre qué empresa está infravalorada y cuál sobrevalorada. Si una empresa se encuentra infravalorada, entonces el mercado le pondrá unas condiciones de financiación más difíciles y exigentes que entorpecerán y ralentizarán su desarrollo y crecimiento, aunque esta empresa tenga un mayor potencial.

De manera análoga, si una empresa está sobrevalorada, las ventajosas condiciones de financiación de las que dispondrá debido a esta sobrevaloración serán negativas para los inversores en un primer lugar, pero también para el resto de las empresas del mercado, que podrían crear más valor y de forma más eficiente con el capital que se ha destinado a la financiación de la empresa sobrevalorada.

En definitiva, el proyecto busca ayudar a que se mejore la información de la que dispone el inversor, contribuyendo a que el mercado se aproxime a uno más perfecto. Si se dispone de mejor información del mercado, los inversores sabrán destinar más financiación a las empresas con mayor potencial de crecimiento y menos a las que tengan un potencial menor. Esto aceleraría el desarrollo económico porque se haría un uso más eficiente del dinero de los inversores para los proyectos que más valor generen.

Entre estos proyectos habría algunos innovadores que contribuirían mediante técnicas disruptivas al algunos de los ODS de forma directa, como una empresa dedicada a la potabilización del agua (ODS 6: Agua Limpia y Saneamiento)

El incremento del crecimiento de estas empresas que generan valor de forma más eficiente aumentaría el ratio de beneficio/empleado, esto es la productividad de la mano de obra. El aumento de la productividad por hora de trabajo traería consecuentemente un aumento del gasto que la empresa estaría dispuesta a asumir por hora de trabajo de cada empleado. Esto se podría manifestar mediante una reducción de la jornada laboral, una mejora de los seguros y dietas que se les proporcionen a los empleados, o simplemente a través de una subida de los salarios.

En definitiva, se mejorarían las condiciones laborales así como el crecimiento económico, lo que afecta directamente al ODS 8 (Trabajo Decente y Crecimiento Económico) y al 1 (Fin de la Pobreza)