Long Short Term Memory (LSTM) para predicción de la Criptomoneda Bitcoin.

**Introducción**

**Proyecto Final**

**Ramírez Andrade Darío, 436963,** [**d.ramirezandrade@ugto.mx**](mailto:d.ramirezandrade@ugto.mx)

¿La historia de los datos cuenta? Absolutamente sí, podemos extraer bastante información y relevancia del orden secuencial de los datos.

Un ejemplo del que podemos partir es:

- "El caballo blanco corre por la pradera libre y rápido".

"Libre" nos da mucho contexto y significados que podemos interpretar.

- "Las nuevas elecciones fueron necesarias, aunque pensé que iban a ser un desastre"

¿Cómo hacemos para analizar el sentimiento del enunciado?

Podemos tener una red neuronal para cada palabra, este conjunto de redes se le denomina time-step.

# Objetivo

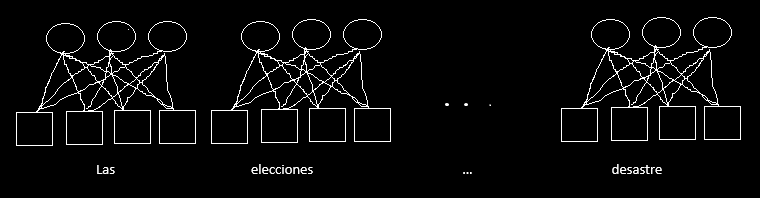
El objetivo será poder hacer una predicción del valor Close (USD) de la criptomoneda Bitcoin usando su serie temporal con los atributos Date, High, Low, Open, Close, Volume, Marketcap usando una LSTM, modelo del Deep Learning.

# Justificación

Este proyecto es importante ya que ayuda a introducirnos a un campo complejo donde desde hace tiempo, en la creación del mercado bursátil se ha querido predecir valores futuros tanto como de acciones y monedas. Este proyecto ayudará a tener una mejor visión sobre este campo, de como las herramientas del Deep Learning nos pueden ayudar a predecir valores; el comprender y analizar este ámbito obtendremos un beneficio personal, científico, empresarial y profesional.

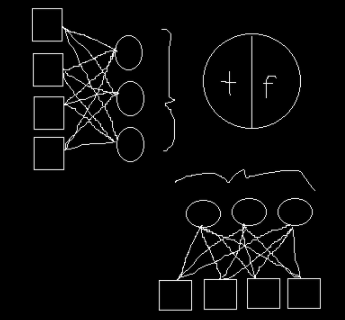
# Marco Teórico

Retomando el problema de la introducción, si estas redes de alguna manera ya funcionasen y tienen toda su implementación, la palabra desastre sin todo el contexto la clasificaría como "mala", sentimiento negativo; es ahí donde entra las Redes Neuronales Recurrentes.



¿Qué podemos hacer para no tener esta pérdida de contexto? Hacemos "un vector con contexto" que incluye toda la información previa, es por eso que recuerda información del pasado.

Agregamos una capa densa el cual tendrá toda la información histórica. La suma de esta capa histórica y de la capa secuencial es una RNN cell.



En cada time-step vamos a tener una estructura similar a esta.

De manera descriptiva este sería el funcionamiento de una RNN donde en su estructura tiene varias celdas RNN y al final, dependiendo del problema puede hacer una regresión o una clasificación.

Puntos clave:

* Los pesos iniciales de la capa densa oculta es ceros.
* Normalmente se usa tanh como función de activación en las RNN cell

Se actualizan los pesos como normalmente se suele hacer para capas densas y/o CNN, usando un tipo de retro propagación ( Backpropagation throug time).

El problema y solución:

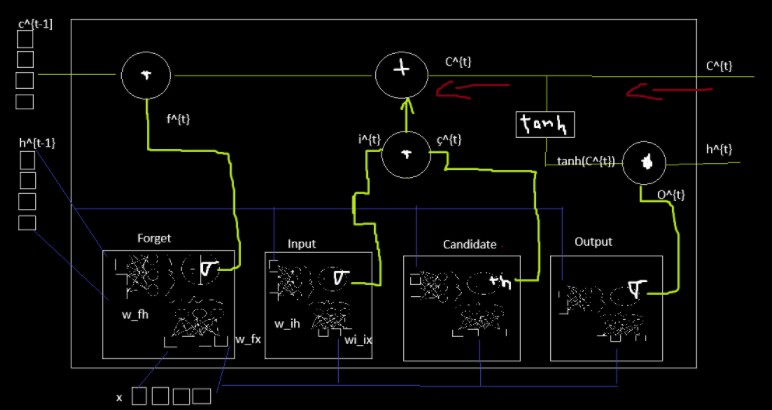
* Vanishing Gradient: si las derivadas en cada paso son pequeñas (por ejemplo menor que 1), al multiplicarlas muchas veces (una por cada paso en el tiempo), el gradiente se hace cada vez más pequeño, hasta que es casi cero. Eso impide que la red aprenda dependencias lejanas.
* Entra LSTM (variante de RNN), usa una conexión directa para el estado de memoria (el cell-state), que permite que el gradiente fluya sin apagarse.

Long Short Term Memory (LSTM):

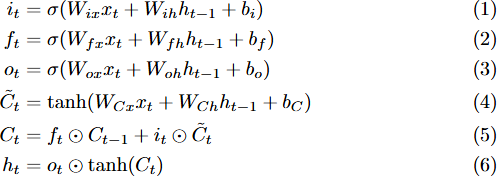
Agrego un "camino", un camino fácil para el backpropagation donde además tenemos el control de Gates (que tanto conservamos o dejamos pasar del estado pasado de la celda).

El paper original de LSTM se presentó en 1997, pasaron 20 años para ser relevante, definitivamente adelantado a su época.

Para entender las LSTM, podemos decir que es una estructura con varias sub-celdas RNN dentro de una sola celda LSTM.

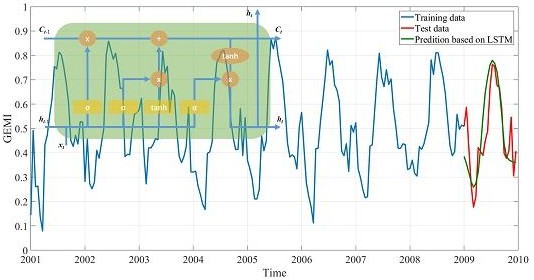


Puntos clave y ecuaciones:



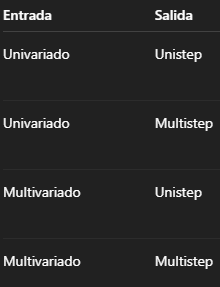
1. Input Gate: Un valor cercano a 1 significa "dejar pasar" o "agregar mucha" de la nueva información potencial. Un valor cercano a 0 significa "casi no agregar" nada de la nueva información. Este valor i\_t es una señal de cuánta atención se le debe prestar a la nueva información entrante.
2. Forget Gate: Un valor cercano a 1 para un elemento en particular significa "recordar" o "mantener" esa parte del estado de memoria anterior. Un valor cercano a 0 significa "olvidar" o "descartar" esa parte. Es la señal que controla el flujo de información hacia adelante desde el estado de celda anterior.
3. Output Gate: Un valor cercano a 1 para un elemento significa que esa parte del estado de celda actual es muy importante para la salida. Un valor cercano a 0 significa que no es tan relevante. Este valor O\_t es una señal que controla cuánta información del estado de la celda se expone para convertirse en el nuevo estado oculto.
4. Los valores de Ç representan la información nueva y fresca que la red ha decidido que es potencialmente importante para el estado de memoria, ya sea para agregarla (valores positivos) o para restarla (valores negativos).
5. Cell State: La LSTM decide explícitamente que olvidar de la memoria pasada f\_t *\* C\_t y que nueva información añadir i\_t \** Ç\_t para formar la memoria actual. El producto Hadamard (punto) es clave aquí, ya que actúa como un "filtro" elemento a elemento.
6. La compuerta de salida O\_t decide qué partes del estado de memoria filtrado (tanh(C\_t)) son relevantes para ser expuestas como la salida de esta celda en este paso de tiempo. Esto permite que la LSTM mantenga información compleja en su estado de celda C\_t sin tener que exponerla toda en su salida inmediata h\_t, lo que la hace muy potente para modelar secuencias.

El siguiente artículo profundiza un poco más sobre las ecuaciones y nos quedamos con la siguiente imagen proporcionada. [https://www.mdpi.com/2072-4292/10/3/452](https://www.google.com/url?q=https%3A%2F%2Fwww.mdpi.com%2F2072-4292%2F10%2F3%2F452)



Ventanas y tipos de series temporales:

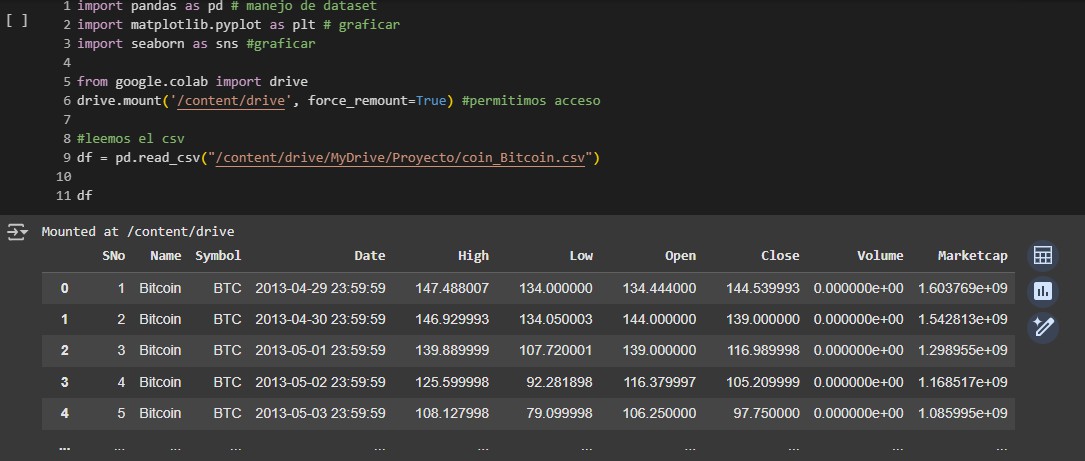
Dependiendo de nuestro problema y serie temporal, es que podemos hacer diferentes tipos de "ventanas", como ya visto, son los time-step.



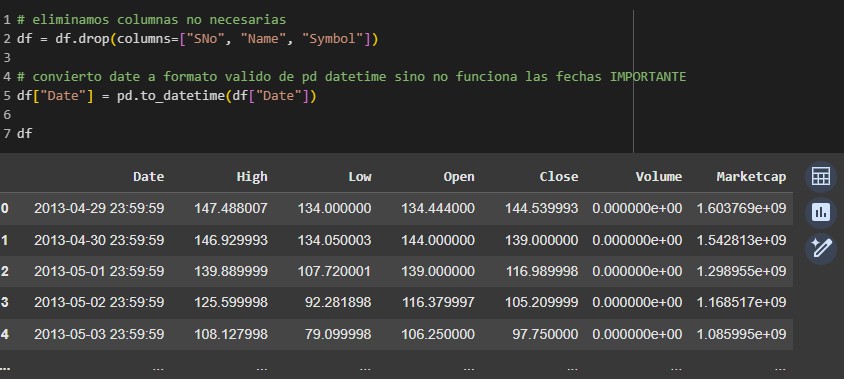
# Diagrama de bloques

Como primer componente se hizo la Implementación del dataset. El objetivo será predecir los valores Close de la Criptomoneda Bitcoin. Donde usando diferentes atributos (Multivariado) haremos una predicción de una sola clase y un instante de tiempo, en este caso "Close" en dólares .

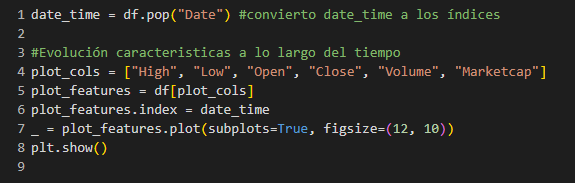
1. Cargar el dataset obtenido en Keggle [https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehis](https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory?select=coin_Bitcoin.csv) [tory?select=coin\_Bitcoin.csv](https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory?select=coin_Bitcoin.csv)



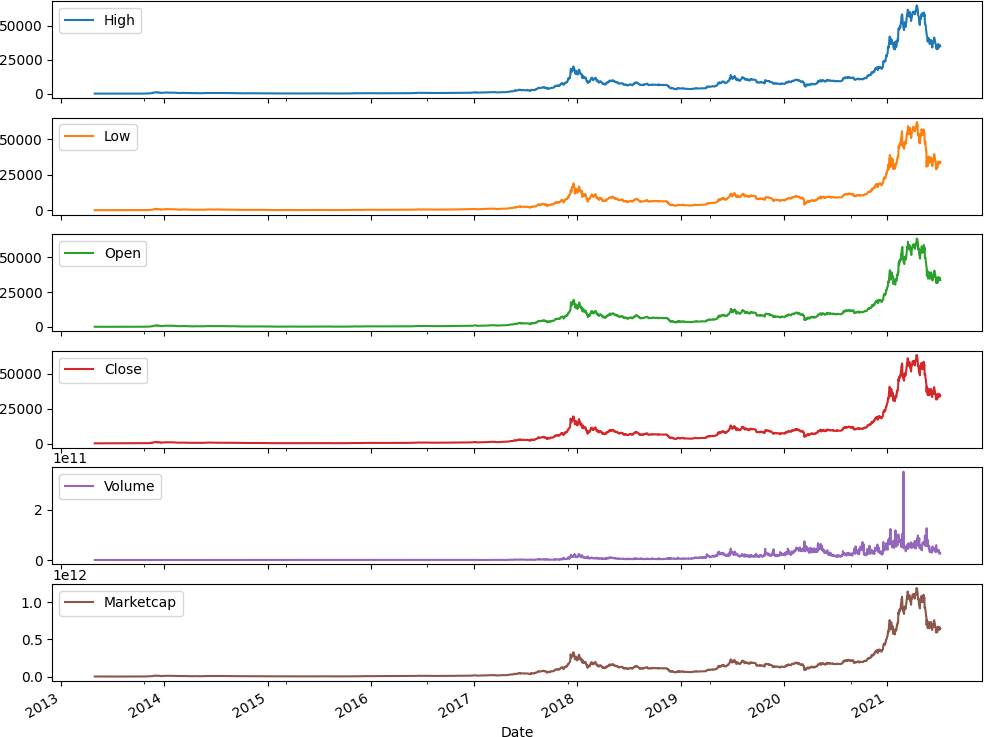
Después se pasó a un preprocesamiento en el que se eliminó columnas innecesarias y se modificó el índice.



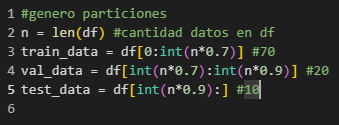
Se obtuvo las gráficas con:



Podemos observar de mejor manera las series de tiempo que contamos.

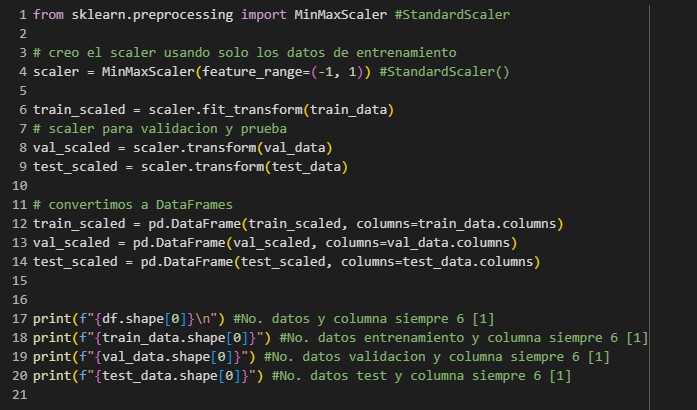


Hacemos una partición 70/20/10 para entrenamiento, validación y prueba. IMPORTANTE: Respetar las particiones cronológicamente.

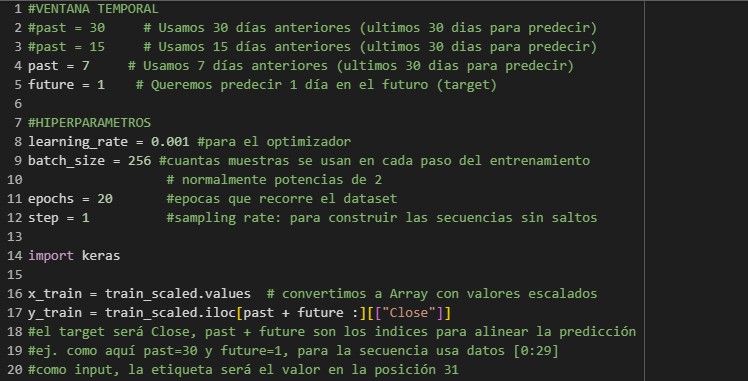


Implemento sklearn para usar un normalizador, ya que como los modelos ML se requiere normalizar los datos y evitar sesgos y sensibilidades.

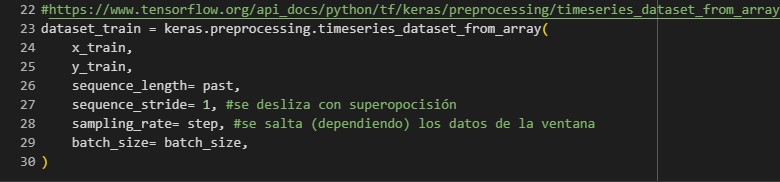
En este punto, varios programadores usan StandardScaler o MinMaxScaler; pro prueba y error MinMax obtenía un val\_loss menor. (si tuve en consideración la función de activación tanh).



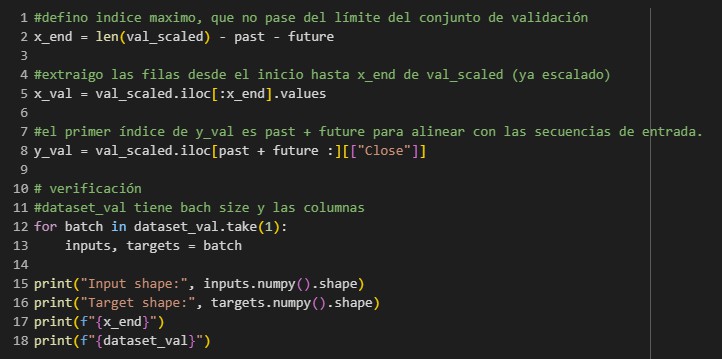
Procedemos a hacer la ventana temporal antes vista en la metodología:



Usamos el método ya implementado por Keras.



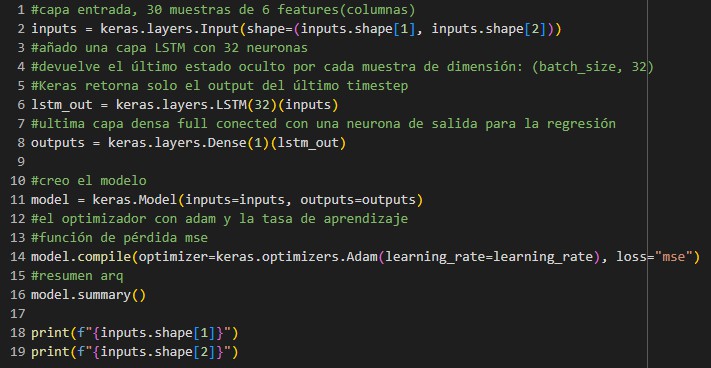
Defino índices para que no haya desfaces y los arrays tengan correctos índices y steps.



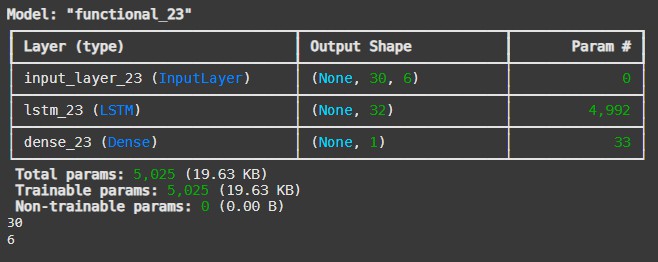
Constructor:

¿Dónde quedan todas las ecuaciones antes vistas? Todas las ecuaciones de la LSTM, como la input gate, forget gate, output gate etc., están embebidas dentro de la capa keras.layers.LSTM; Keras las implementa internamente cuando definimos esa capa. Nosotros solo preparamos la entrada como una secuencia temporal (en mi caso la variable past=7), y luego esa capa se encarga de aplicar todos los pasos recurrentes. Finalmente, usamos una capa densa para proyectar el estado oculto final a una predicción del dia futuro.

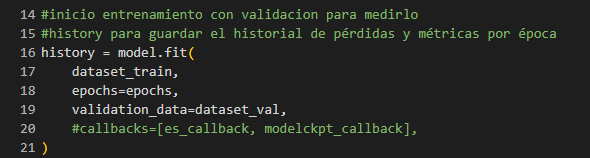
A la salida final, me da el estado oculto final h\_t, con dimensión (batch\_size, 32). Ese 32 es cada célula LSTM con 32 unidades. Y cada unidad LSTM tiene sus propias puertas.



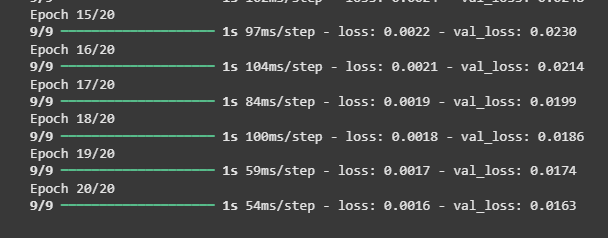
Resumen del modelo:



Finalmente entrenamos con la ayuda de las ventanas.

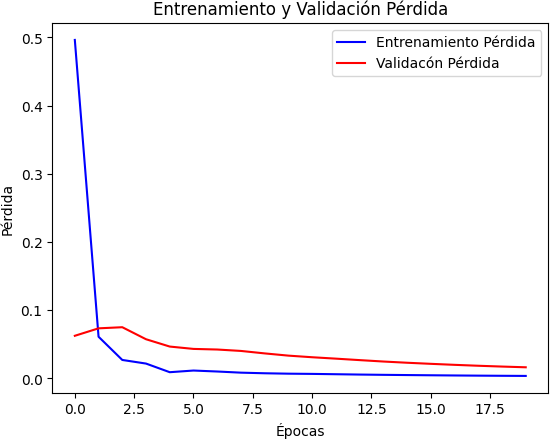


Observamos las últimas épocas (relevantes):



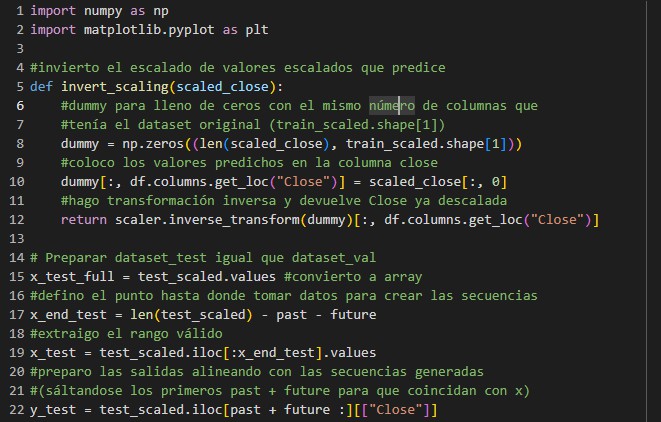
Finalmente hacemos evaluaciones.

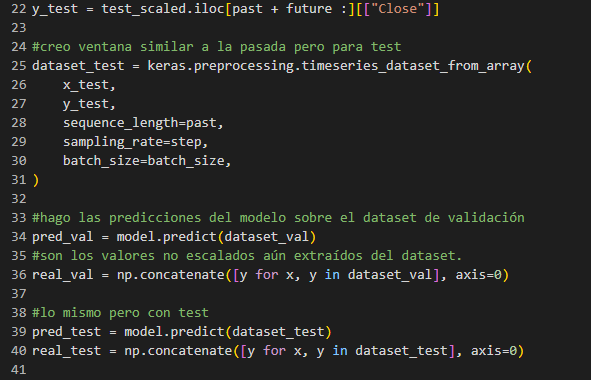


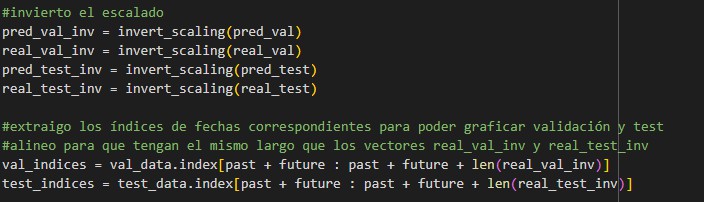


El objetivo es que el modelo converja al final del entrenamiento con un nivel de pérdida bajo.

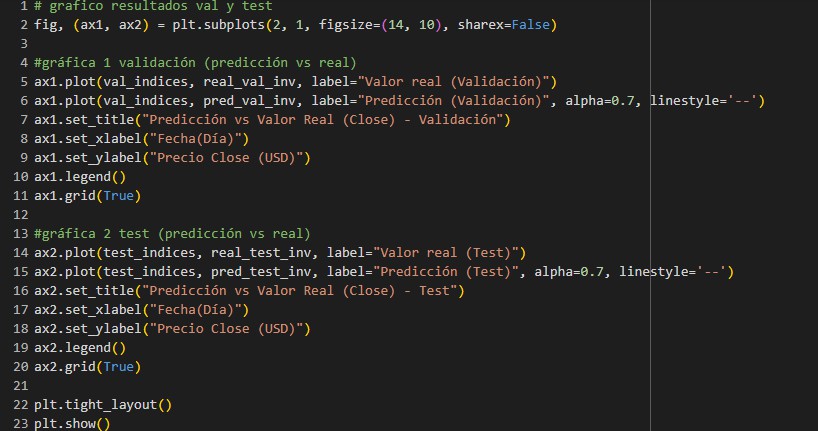
Para hacer las gráficas debemos des escalar para que tengan los mismos tamaños.

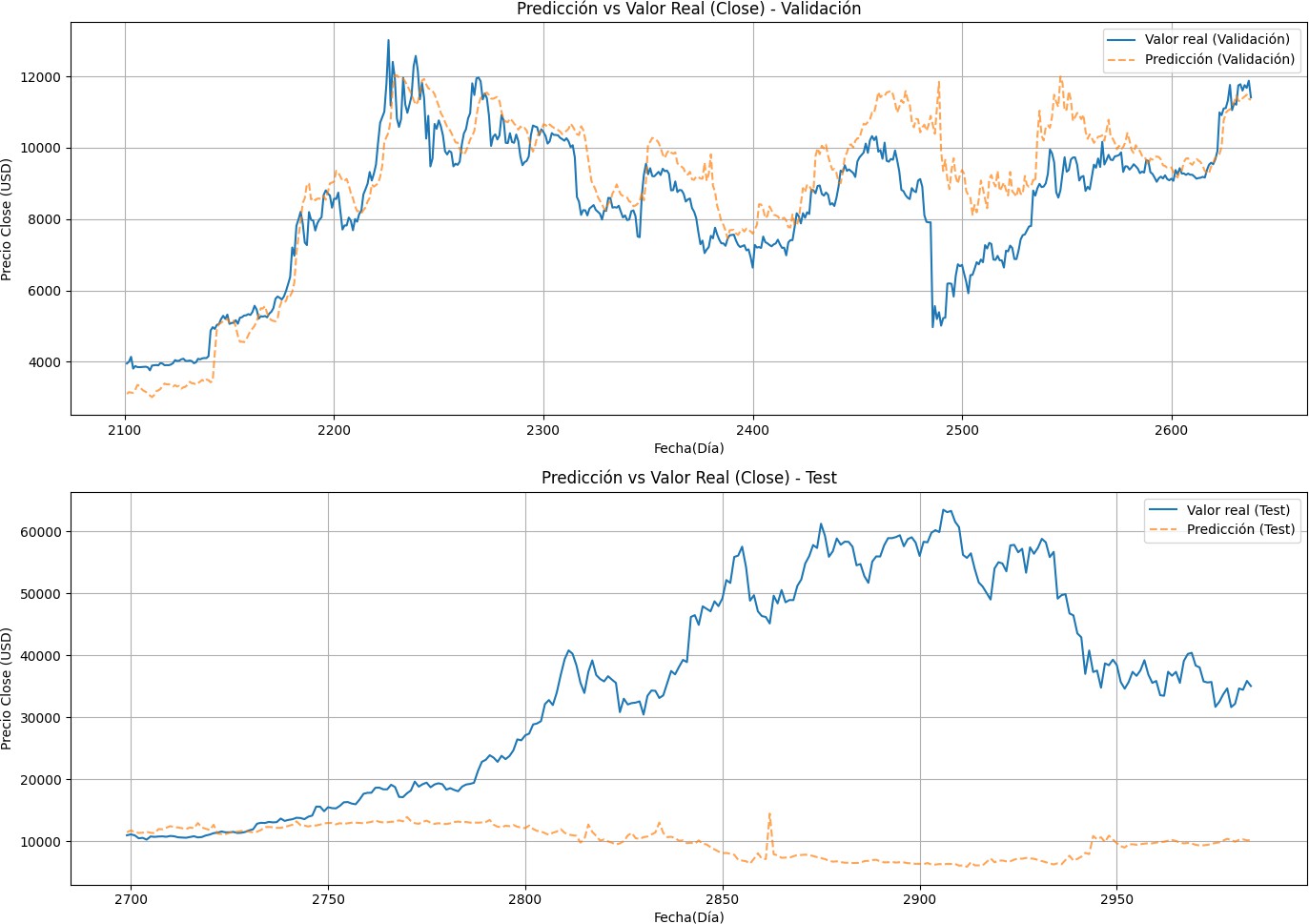






Una vez hecho esto podemos hacer las gráficas y obtener métricas.





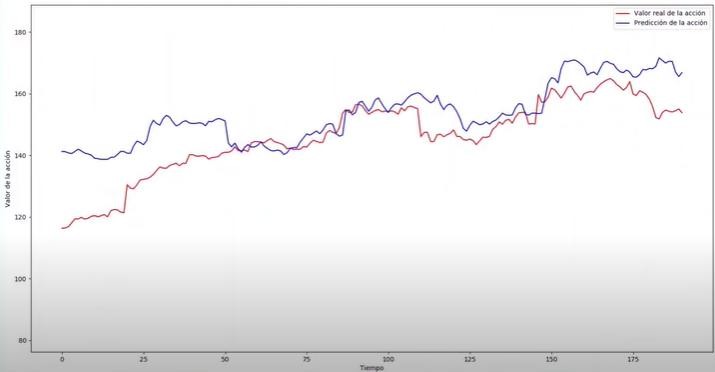
La validación fue buena, pero el test falla. Pueden ocurrir varias cosas:

* El modelo aprendió muy bien el set de entrenamiento/validación, pero no generaliza bien al test. Esto lo podemos mantener ya que ajustando hiperparametros no cambia demasiado, de hecho, todas las actualizaciones dan bien la validación, pero no el testeo.

Igual podemos dar alguna explicación, pero será más a detalle adelante.

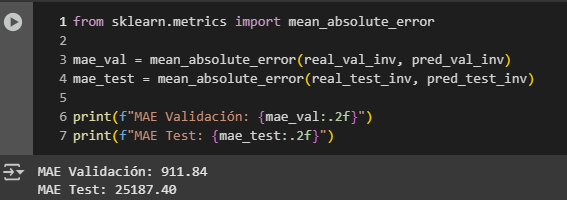
* Error en el escalado o des escalado. Esto también lo podemos descartar ya que viendo otras fuentes parecidas, donde analizan igual "acciones" mediante LSTM vemos gráficas de validación y valor real con una naturaleza similar.
* Predicciones lejanas (future). En este caso usamos el mínimo (1), justo para evitar complicaciones en cuestión de predecir más instantes de tiempo.

Predicción de acciones en la bolsa con PYTHON (tutorial redes LSTM) por Codificando Bits: <https://www.youtube.com/watch?v=3kXj6VgxbP8> Min. 4:00:



Mean Absolute Error (MAE)

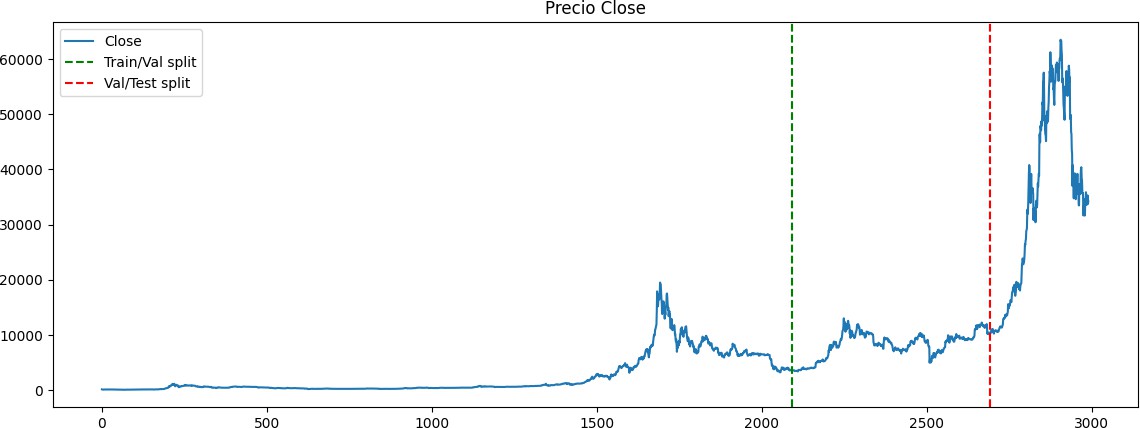
Usamos métricas cuantitativas, una común para regresión es MAE que mide el error promedio en valor absoluto.



En mejor de los casos tuvimos un MAE de Validación de 694.72 y MAE en Test de 22188.24

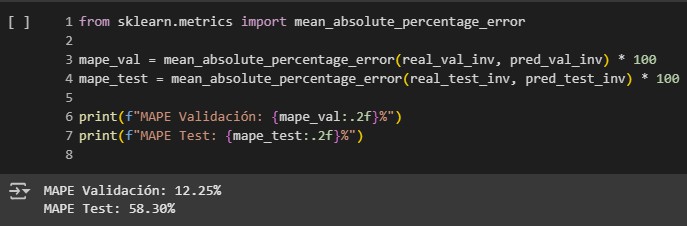
Esto indica que entre el error de validación y de test dice que el modelo aprendió bien en entrenamiento/validación, pero cuando ve datos nuevos (test) falla mucho.

Esto puede deberse a que por ejemplo y de hecho los precios en testeo son mucho más altos que en validación (por eso el MAE es tan alto). Por lo que asumimos que tal vez hubo una subida fuerte en el mercado y el modelo nunca vio o "vivió" algo parecido. Que de hecho fue cuando fue más el "BOOM" de la cripto.



Para ver un poco mejor: El Error Porcentual Absoluto Medio (MAPE)

Es una métrica para evaluar la precisión de los modelos de regresión. Se calcula como el promedio de los porcentajes absolutos de diferencia entre los valores previstos y los valores reales (que ya lo hicimos arriba), expresados como un porcentaje. El MAPE es útil para comprender la desviación promedio de las predicciones en relación con los valores reales.



Como el anterior, el mejor MAPE fueron de 7.81% para validación y para MAPE Test de 51.80%

Conclusiones:

La LSTM aprendió bien las secuencias de entrenamiento/validación.

Los errores en test son casi del 50%, similar a lo que se nos comentó en su día en la clase, ya que los modelos donde se predicen cripto, acciones, el mercado bursátil en general es algo muy completo; ya que tiene tantos factores y métricas que se pueden agregar, los cuales no tenemos o se encuentra alguna relevancia o significado, como por ejemplo lo que diga algún presidente, guerras, etc. cosas que claramente no están presentes en el modelo y su implementación seria por no decir imposible.

Sin embargo, el propósito en general del modelo fue implementado correctamente, dando a explicar las métricas, sus conceptos clave como las ventanas, etc.

En consideración, mi principal idea del proyecto era incluir un API el cual día con día se actualicen, carguen y entrene el modelo ya que así tendrá siempre en constante actualización la capa histórica, claro, seguirá siendo un modelo falible ante las cuestiones ya mencionadas(política), pero así evitaríamos el error de que los datos de entrada son cosas que no ha "vivido" ya que al final esa es la ventaja de las LSTM.

# Referencias

* + Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2018). Deep learning and its applications to machine health monitoring. *Remote Sensing*, *10*(3), 452. <https://doi.org/10.3390/rs10030452>
  + Pepe Cantoral. (2021, junio 3). *Qué es una LSTM | Redes neuronales recurrentes* [Video]. YouTube. <https://www.youtube.com/watch?v=f6PaCo-NfJA>
  + Pepe Cantoral. (2021, junio 3). *Cómo funcionan las LSTM paso a paso | Parte práctica* [Video]. YouTube. <https://www.youtube.com/watch?v=x6E44DDWg5Q>
  + Codificando Bits. (2017, diciembre 1). *Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM)* [Video]. YouTube. <https://www.youtube.com/watch?v=3kXj6VgxbP8>
  + Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735– 1780.
  + TensorFlow. (n.d.). *tf.keras.preprocessing.timeseries\_dataset\_from\_array*. TensorFlow. <https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/timeseries_dataset_from_array>
  + Keras. (n.d.). *Time series forecasting using weather data*. Keras. <https://keras.io/examples/timeseries/timeseries_weather_forecasting/>
  + IBM. (n.d.). *Graph training models*. IBM Documentation. [https://www.ibm.com/docs/es/masv-and-](https://www.ibm.com/docs/es/masv-and-l/maximo-vi/cd?topic=models-graph-training) [l/maximo-vi/cd?topic=models-graph-training](https://www.ibm.com/docs/es/masv-and-l/maximo-vi/cd?topic=models-graph-training)
  + OpenAI. (2025). *ChatGPT (June 2025 version) [Large language model]*. <https://chat.openai.com/> Utilizado para generar imágenes, ecuaciones y consultas.
  + Kumar, S. (2018). *Cryptocurrency Historical Market Price Data* [Dataset]. Kaggle. <https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory>