UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e ingenierías

Evaluación de arquitecturas de aprendizaje automático en predicción de precio de acciones integrado con análisis de sentimiento y factores macroeconómicos.

Juan Martín Sánchez Bardellini
ingeniería en Ciencias de la computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 25 de noviembre de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Evaluación de arquitecturas de aprendizaje automático en predicción de precio de acciones integrado con análisis de sentimiento y factores macroeconómicos.

Juan Martin Sánchez Bardellini

Nombre del profesor, Título académico **Felipe Leonel Grijalva Arévalo, PhD**

Quito, 25 de noviembre de 2023

© DERECHOS DE AUTOR

Nombres y apellidos: Juan Martin Sanchez Bardellini

Código: 00213868

Cédula de identidad: 1719506139

Lugar y fecha: Quito, 25 de noviembre de 2023

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

La implementación de inteligencia artificial ha tenido mucha atención estos últimos años, pero no es algo reciente. Ha existido desde hace muchos años y se ha encontrado en constante cambio, lo cual ha dado espacio a muchas herramientas evolucionen. Por esta constate evolución es importante analizar y tener claro los mejores usos de recursos sin sacrificar precisión. Este proyecto busca en analizar la eficiencia y eficacia de tres arquitecturas de aprendizaje automático: LSTMs, GRUs, TCNs, adicionalmente, también se evaluaron sus contrapartes con componentes bidireccionales. Esto se hizo con el fin de encontrar la arquitectura más eficiente para realizar tareas de predicción de series de tiempo multivariable, con un énfasis en análisis de sentimientos en redes sociales y factores macroeconómicos para predecir el precio de acciones en la bolsa de valores. Mediante evaluaciones de error y pruebas de hipótesis se logra concluir que la arquitectura más eficiente en este contexto en específico seria las LSTMs, que presenta un desempeño más contante con una menor cantidad de parámetros a comparación de las otras arquitecturas que se evaluaron.

Palabras clave: LSTM, GRU, TCN, multivariable, acciones, predicción

ABSTRACT

Even though Artificial intelligence has been very prominent in recent years, it wasn't introduced recently. It has been regularly evolving, which has resulted in new tools being constantly developed. Due to this constant change, it is important to conduct analysis and have a better understanding on the best ways to utilize resources without sacrificing accuracy. This project focuses on analyzing the efficiency and efficacy of three different machine learning architectures: LSTMs, GRUs, TCN, additionally their counterparts with Bidirectional inputs were also evaluated. This was done to find the better performing architecture for multivariate time series forecasting, with an emphasis on sentiment analysis and macroeconomic factors for stock price forecasting. Utilizing error metrics and hypothesis tests it is concluded that the better performing architecture in this context is the LSTM architecture, this is due to its more constant performance and the lower number of parameters it uses compared to the other architectures that where tested.

Key words: LSTM, GRU, TCN, multivariate, stock, forecasting

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Dentro del campo financiero, existen numerosas herramientas utilizadas para proyecciones, entre ellas se encuentran las herramientas tradicionales como los modelos estadísticos. Estos modelos han sido la norma para la evaluación de series de tiempo durante muchos años y continúan siendo empleados en la actualidad para diversas tareas de proyecciones financieras. Son fáciles de entender e implementar y presentan una notable adaptabilidad para distintos tipos de tareas. Sin embargo, enfrentan un desafío: cuanto más compleja es la correlación entre los datos a evaluar, más complicado se vuelve el desarrollo del modelo estadístico y aun así es difícil que refleje le verdadera correlación que existe entre los datos. Esto se debe, en gran medida, a que muchos modelos estadísticos se basan en correlaciones lineales entre los datos. Aunque si existen modelos estadísticos que abordan correlaciones no lineales, presentan limitaciones en su aplicación.

En la actualidad, se han introducido nuevas herramientas en el ámbito financiero para lograr un mejor desempeño en las proyecciones con datos más complejos. Estas herramientas son modelos de aprendizaje automático, específicamente arquitecturas basadas en redes neuronales que pertenecen a la rama de aprendizaje automático denominado: aprendizaje profundo. Las redes neuronales muestran una superioridad al manejar correlaciones complejas, no lineales, lo cual se asemeja más a la realidad de factores financieros, como el precio de acciones.

Durante las últimas décadas, se ha generado una notable explosión en la implementación de modelos de redes neuronales en el ámbito financiero. Entre estas hay arquitecturas que sobresalen, y una de las principales son las Red Neuronal Recurrentes (RNNs, por sus siglas en inglés). Estas redes son particularmente destacadas porque tienen la

capacidad de "recordar" entradas de momentos anteriores, lo que las hace excelentes para interpretar dependencias temporales.

Sin embargo, las RNNs presentan limitaciones al tratar dependencias a largo plazo. Es eso por lo que, en la implementación de proyecciones de series de tiempo, se suelen utilizar las RNN de Memoria de Corto y Largo Plazo (LSTM, por sus siglas en inglés). Las LSTMs están especialmente diseñadas para manejar estas dependencias a largo plazo, convirtiéndolas en herramientas ideales para modelar series de tiempo complejas, como los precios de acciones, predicciones climáticas o para otras secuencias como en el procesamiento del lenguaje natural (NLP, por sus siglas en inglés).

Hoy en día, las LSTM son consideradas una de la herramienta más comunes en la modelación de series de tiempo y han estado en el campo por más de una década, manteniendo su relevancia en modelación de series temporales. Sin embargo, el ámbito del aprendizaje profundo está en constante evolución, dando paso a nuevas herramientas, que podrías ser alternativas a arquitecturas previamente establecidas en el campo.

Surge así una interrogante: si hay herramientas novedosas, ¿podrían estas nuevas herramientas superar a las anteriores en las predicciones de series de tiempo? Varios estudios [1,2] han buscado responder esta pregunta, y los resultados sugieren que, en ciertos contextos, nuevas arquitecturas como los Transformers puede superar en desempeño a los modelos tradicionales de proyecciones de series de tiempo, incluso a aquellos basados en LSTMs. Ante el cambio constante que se genera en el campo de aprendizaje automático, es esencial explorar las herramientas disponibles y determinar la arquitectura óptima para la tarea deseada. Para esto, es importante identificar problemas específicos donde podamos medir y comparar el rendimiento de estas distintas arquitecturas.

Este proyecto plantea el reto de evaluar diferentes arquitecturas de aprendizaje automático para proyectar series de tiempo de precios de acciones, tomando en cuenta tanto sentimientos en redes sociales como factores macroeconómicos. A pesar de que existen fundamentos lógicos para creer que existe una relación entre estos factores, determinar su relación es un tema complejo, e incluso podría darse el caso que en verdad no exista una relación entre ellos. Dada esta complejidad, se busca evaluar el desempeño de distintas arquitecturas de aprendizaje profundo en la predicción de precios de cierre de acciones de diversas empresas.

El objetivo principal es identificar la arquitectura más adecuada para esta tarea específica en la predicción de acciones. Este análisis es crucial, ya que permite limitar el uso ineficiente de recursos. Se evaluarán tres arquitecturas principales:

1. LSTM, reconocida como uno de los métodos comunes en el ámbito del aprendizaje profundo para predicciones de series de tiempo.
2. Unidades Recurrentes con Compuertas (GRUs, por sus siglas en inglés) Una evolución en el campo de RNN que son estructuralmente más simples y nuevas que las LSTMs.
3. Redes Convolucionales Temporales (TCN, por sus siglas en inglés), derivadas de las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés). Estas son más recientes que las GRUs y están especializadas en predicciones de series de tiempo.

Adicionalmente, se evaluarán las mismas arquitecturas con componentes bidireccionales en sus entradas.

Atreves de diversas métricas, se busca llevar a cabo una evaluación rigurosa del desempeño de cada arquitectura con el fin de determinar cuál es la más eficaz para anticipar

los cambios en precios de acciones. Para esto la construcción del conjunto de datos, consolidará diversas fuentes para capturar dimensiones relacionadas con el sentimiento, el historial de precios de acciones y el registro de factores macroeconómicos.

ESTADO DEL ARTE

En la década de los 80, Jeff Elman y Michael I. Jordan teorizaron el concepto de redes que conservan una "memoria" de entradas previas. Esto marcó el nacimiento de lo que a principios de los 90s se denominarían RNNs (Redes Neuronales Recursivas). Estas redes ayudan a modelar dependencias temporales, pero enfrentaron un problema conocido como el desvanecimiento y explosión de gradiente. Este desafío surgió porque las RNNs encontraban difícil mantener de manera estable las dependencias a largo plazo. Para solucionar esto, en 1997 se propusieron las LSTMs (Long Short-Term Memory), una variante de las RNNs que solucionó sus problemas asociados con dependencias a largo plazo. En 2010, comenzó un auge en la implementación de modelos basados en RNNs, siendo utilizados en tareas como el NLP y la proyección de series temporales (TSF, por sus siglas en inglés). Durante muchos años, las LSTMs se emplearon en tareas de TSF debido a su versatilidad para manejar dependencias a largo plazo.. Años después el panorama con relación a las series temporales comenzó a evolucionar, transaccionando hacia diferentes arquitecturas que lograban resolver mejor estos problemas. En 2014 surgió otra arquitectura basada en las RNNs las GRUs, son más computacionalmente eficientes que las LSTMs porque tienen una estructura más simple, debido a esto no son tan eficientes como las LSTMs para capturar dependencias a largo plazo [3,4]

En 2018, otra rama del aprendizaje profundo exploró TSF. A partir de una investigación [5], surgió la idea de las TCNs, que, aunque están basadas en la arquitectura de las CNNs, tienen un enfoque exclusivo en series de tiempo. Estas herramientas, más recientes que las LSTMs y

GRUs, han mostrado un buen rendimiento en áreas de TSF. Gracias a su énfasis en la paralelización debido a sus raíces en CNNs, las TCNs son computacionalmente eficientes. Además, esta arquitectura ha demostrado tener un mejor desempeño que las LSTMs en determinadas tareas [6,7], lo que las posiciona como un potencial candidato destacado para el futuro del modelado de series temporales.

Dada la información presentada sobre tecnologías emergentes y significativas en el ámbito de TSF, es crucial realizar más estudios para determinar en qué tareas una herramienta destaca sobre la otra y en qué condiciones ocurre esto. Tal investigación podría ofrecernos detalles para comprender mejor las características específicas de cada una de estas herramientas de aprendizaje profundo. El proyecto mencionado anteriormente ofrece una oportunidad perfecta para evaluar la eficacia de estas herramientas en un área específicamente vinculada al sector financiero. Además, brinda la posibilidad de combinar datos de series numéricas, como el precio de las acciones, con otros factores influyentes, tales como el sentimiento hacia la empresa y factores macroeconómicos.

Adicionalmente, el proyecto se basa en el estándar ISO/IEC 22989:2022 “Information technology — Artificial intelligence — Artificial intelligence concepts and terminology.” Para seguir terminología actual y estandarizada acerca del campo de inteligencia artificial/aprendizaje de automatico.

DESARROLLO DEL TEMA

Fundamentos Teóricos:

LSTMs[9]:

Las Celdas de LSTMs están conformadas por dos secciones principales que interactúan entre ellas: las memorias a largo (C) y a corto (H) plazo. Estas 2 memorias

están en constante cambio debido a 3 compuertas principales: compuerta de olvido, compuerta de recuerdo y la compuerta de salida.

Primer paso: C y H se inicial con algún valor bajo o en cero y se ingresa una variable de entrada x. (En los siguientes pasos estarán marcados con subíndices temporales como t y t-1, esto para denotar el flujo y cambio de los datos)

Segundo paso: C_{t-1}, H_{t-1} y x_t ingresan a la celda LSTM y se utilizan para calcular cuanta memoria a largo plazo se va a olvidar mediante el factor de olvido (f_t) y mediante la función (U y W son representaciones de pesos):

$$f_t = \sigma(x_t * U_f + H_{t-1} * W_f)$$

Tercer paso: Seguido por esto se determina cuanta memoria a largo plazo es retenida mediante la siguiente función:

$$C_t^* = C_{t-1} * f_t$$

Cuarto paso: Se determina el factor de recuerdo de las nuevas entradas:

$$i_t = \sigma(x_t * U_i + H_{t-1} * W_i)$$

Quinto paso: Se determina la información que potencialmente se agregaría a la memoria a largo plazo:

$$N_t = tanh(x_t * U_c + H_{t-1} * W_c)$$

Sexto paso: se actualiza la memoria a largo plazo con la siguiente función:

$$C_t = C_t^* + i_t * N_t$$

Séptimo paso: se determina el valor de salida mediante la siguiente función:

$$o_t = \sigma(x_t * U_o + H_{t-1} * W_o)$$

Octavo paso: se actualiza la memoria a corto plazo:

$$H_t = o_t * tanh(C_t)$$

Este proceso se repite por cada segmento de tiempo de la secuencia hasta llegar a la última unidad LSTM en la cual se espera la predicción, en este se retorna el ultimo valor de H.

GRUs[10]:

Las celdas de una GRU son más simples que las de una LSTM. Estas están Conformadas por solo un componente de memoria (H) y dos compuertas importantes: la compuerta de reset y la compuerta de actualización, que cumplen propósitos similares a las compuertas de olvido y compuertas de recuerdo de las LSTMs.

Primer paso: Se inicializa H en 0 y se ingresa la primera entrada x. (En los siguientes pasos estarán marcados con subíndices temporales como t y t-1, esto para denotar el flujo y cambio de los datos)

Segundo paso: Utilizando las entradas H_{t-1} y x_t, se utiliza la siguiente función para determinar el factor de reset :

$$r_t = \sigma(x_t * U_r + H_{t-1} * W_r)$$

Tercer paso: Utilizando las entradas H_{t-1} y x_t, se utiliza la siguiente función para determinar el factor de actualización

$$z_t = \sigma(x_t * U_z + H_{t-1} * W_z)$$

Cuarto paso: Se genera un nuevo estado de memoria (H_t^*) mediante la siguiente función:

$$H_t^* = tanh(x_t * U_h + r_t * (W_h * H_{t-1}))$$

Quinto paso: Se genera el estado de memoria final en t utilizando la siguiente función:

$$H_{t-1} = \big(z_t * H_{t-1} + (1 - z_t)\big) * H_t^*$$

Este proceso se repite por cada segmento de tiempo de la secuencia temporal hasta llegar a la unidad de tiempo en la cual se espera la predicción, en este se retorna el ultimo valor de *H*

TCNs[11,12]:

Las Redes Convolucionales Temporales (TCNs) presentan una estructura significativamente diferente en comparación con las previamente discutidas. Esto se debe a que están basadas en otra familia de redes neuronales conocidas como redes neuronales convolucionales, que utilizan filtros convolucionales para identificar características cruciales en los datos. A diferencia de las estructuras previas basadas en RNNs, estas no capturan dependencias de manera secuencial, como lo hacen las celdas de LSTMs y GRUs. En cambio, las TCNs se centran en ventanas completas de la serie temporal para determinar las dependencias relevantes.

Dado que las TCNs evalúan subseries de tiempo conjuntamente en lugar de secuencialmente, es necesario modificar un tensor de entrada para introducir el concepto de causalidad en el sistema de las TCNs. Esto se logra mediante un relleno de ceros al inicio del tensor de entrada, y el tamaño de dicho relleno depende de otros factores que se abordarán más adelante.

Primer paso: Proporcionar un tensor de entrada que puede estar compuesto por la serie de tiempo completa o solo una porción de esta, dependiendo de la tarea deseada.

Segundo paso: dentro del tensor de entrada, se utiliza una subsecuencia de tamaño k, y se aplica la función de producto punto para multiplicar este vector con otro denominado kernel, compuesto por pesos. Esta multiplicación tiene como objetivo determinar la predicción para el tiempo t+1. El vector de tamaño k representa la ventana de tiempo utilizada para predecir el

siguiente valor de la serie temporal. El hiperparámetro k se conoce como el tamaño del kernel (en este ejemplo representado por un vector compuesto de pesos w). El siguiente ejemplo describe el proceso con k=3:

$$< \boldsymbol{x_{t1}}, \boldsymbol{x_{t2}}, \boldsymbol{x_{t3}} > \bullet < \boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3} >= \boldsymbol{o_{t4}}$$

Tercer paso: la ventana de tiempo se desplaza un paso a la derecha para continuar evaluando la serie temporal y generando las siguientes predicciones de manera consecutiva, como se muestra en el siguiente ejemplo para k=3:

$$< \boldsymbol{x_{t1}}, \boldsymbol{x_{t2}}, \boldsymbol{x_{t3}} > \bullet < \boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3} >= \boldsymbol{o_{t4}}$$

$$< \boldsymbol{x_{t2}}, \boldsymbol{x_{t3}}, \boldsymbol{x_{t4}} > \bullet < \boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3} >= \boldsymbol{o_{t5}}$$

El proceso anterior explica cómo se determinan las salidas dentro de una capa convolucional de la TCN. Sin embargo, es importante mencionar que existen dos conceptos adicionales esenciales para comprender la estructura de una TCN.

El primero de estos conceptos es el campo de recepción. Al igual que en las CNNs, las TCNs buscan determinar dependencias evaluando un gran número de entradas en conjunto y filtrando las características más importantes. Esto se logra mediante la utilización de más de una capa convolucional y tomando el tensor de salida de una capa como tensor de entrada para la siguiente capa, como se muestra a continuación:

Primera capa:

$$< \mathrm{x_{t1}}, \mathrm{x_{t2}}, \mathrm{x_{t3}} > \bullet < \mathrm{w_1}, \mathrm{w_2}, \mathrm{w_3} >= \mathrm{x'_1}$$

$$< \mathrm{x_{t2}}, \mathrm{x_{t3}}, \mathrm{x_{t4}} > \bullet < \mathrm{w_1}, \mathrm{w_2}, \mathrm{w_3} >= \mathrm{x'_2}$$

$$< x_{t3}, x_{t4}, x_{t5} > \bullet < w_1, w_2, w_3 >= x'_3$$

Segunda capa:

$$< x'_1, x'_2, x'_3 > \bullet < w_4, w_5, w_6 >= o_{t6}$$

De esta forma, podemos visualizar cómo, al tener más capas convolucionales, podemos determinar dependencias a largo plazo y proyectar la salida con más contexto de toda la serie de tiempo. Se puede observar que para k=3 y dos capas (n=2), utilizamos un total de 5 pasos de la serie de tiempo para llegar a una predicción, y esto se denomina campo de recepción.

Para determinar cuántas capas son necesarias para lograr una convergencia total, considerando un tamaño de tensor de entrada l, se utiliza la siguiente función:

$$n = [(l-1)/(k-1)]$$

Sin embargo, esta convergencia total puede ser costosa, ya que podría requerir un gran número de capas, lo que a su vez podría llevar a problemas de gradiente. Por lo tanto, se implementa un sistema de dilatación, que introduce saltos en las entradas teniendo en cuenta la capa en la que se encuentran. Una capa con una dilatación de dos (d=2) tendría el vector de subsecuencia de la siguiente forma:

$$< x_{t1}, x_{t3}, x_{t5} > \bullet < w_1, w_2, w_3 >= x'_1$$

Para $d = 3$:

$$< x_{t1}, x_{t4}, x_{t6} > \bullet < w_1, w_2, w_3 >= x'_1$$

Para que la dilatación disminuya el número de capas requeridas para la convergencia total, se suele utilizar un crecimiento exponencial para d, dependiendo del nivel en el que se encuentra:

$$d_i = d^{2*i}$$

Según la función anterior, el factor de dilatación aumenta cuadráticamente según la capa en la que se encuentra. Para la primera capa (i=0), con d=2 (valor base de la dilatación), la dilatación sería 1, y en la tercera capa, la dilatación sería de 4. Esto

permite lograr una convergencia total con un menor número de capas, y ahora se determina el número de capas requeridas mediante la siguiente función:

$$n = [\log_d(\frac{(l-1)*(d-1)}{(k-1)} + 1)]$$

Debido a la variabilidad en el concepto de dilatación y cómo se determina la subserie de tiempo en cada capa convolucional, es necesario calcular el relleno de ceros teniendo en cuenta estos factores, como se determina mediante la siguiente función:

$$p = d^i * (k-1)$$

Componentes Bidireccionales [16]:

De forma adicional a las 3 arquitecturas anteriores, se implementarán las tres arquitecturas mencionadas anteriormente con componentes bidireccionales. Esto significa que la secuencia temporal observada se evaluará de forma normal e inversa. Este concepto funciona de igual manera para las arquitecturas basadas en RNNs y la arquitectura TCN. Pero, en los 3 casos, resulta en una arquitectura diferente a la original (unidireccional), esto se puede dividir en 3 diferencias claves.

La primera diferencia es en la estructura de capas del modelo. A diferencia de una capa unidireccional, el modelo bidireccional utiliza dos capas idénticas.

La segunda diferencia es que cada capa recibe la misma información, pero en otro orden secuencial, una recibe la secuencia en orden temporal normal y la otra en orden temporal inverso.

Finalmente, la última diferencia es que cada capa dentro del conjunto bidireccional va a generar sus propios valores de salida. Por lo cual debe existir una combinación de los valores generados a partir de la secuencia normal e inversa. Hay diferentes métodos para

generar una combinación, entre ellos están la concatenación de las salidas para generar un vector, promediar las salidas e incluso aplicar funciones de activación.

La implementación de componentes bidireccionales en los modelos facilita que el modelo identifique dependencias temporales complejas que se pueden pasar por alto en modelos unidireccionales. Los componentes bidireccionales en un modelo también generan que este sea más complejo que un modelo unidireccional y tiene más riesgo de generar sobreajuste del modelo debido a su complejidad adicional.

Materiales y métodos

El objetivo principal es determinar la arquitectura óptima, entre las evaluadas, para realizar predicciones en series de tiempo con un enfoque en factores macroeconómicos y sentimiento en redes sociales. Este proceso de selección de la arquitectura ideal no solo se basa en la evaluación del modelo a través de métricas de desempeño, sino también en su eficiencia en términos de tamaño (número de parámetros).

Este proceso se llevará a cabo utilizando las arquitecturas previamente mencionadas: LSTMs, GRUs y TCNs y sus contrapartes bidireccionales. El entrenamiento y la evaluación de estas arquitecturas se realizarán utilizando un conjunto de datos diseñado específicamente para este propósito. Más adelante, se detallará el proceso de creación y estructura de este conjunto de datos y de las arquitecturas utilizadas en este proyecto.

Conjunto de Datos

El primer enfoque en la construcción del conjunto de datos se centró en el análisis de sentimientos. Para esto, fue necesario definir la fuente de esta información y se determinó que Twitter (X) sería la fuente adecuada. La elección se basó en el hecho de que X proporciona una excelente fuente de textos cortos de opinión, lo que resulta una herramienta valiosa para capturar el sentimiento social hacia ciertos temas o, en

este caso, a empresas. Lastimosamente, debido a un exceso de minería de datos, X cerró su API detrás de un plan mensual con un costo elevado. Como consecuencia, se decidió no utilizar el API para construir el conjunto de datos, sino buscar un conjunto de datos que ya contuviera tweets relacionados con empresas. Esto resultó ser un desafío, ya que fue difícil encontrar un conjunto de datos lo suficientemente completo que abarcara una gran cantidad de tiempo.

Finalmente, se encontró un conjunto de datos en un artículo de investigación presentado en la Conferencia Internacional de Big Data de la IEEE [13]. Este conjunto de datos contenía más de 3 millones de tweets relacionados con 5 empresas: Apple (AAPL), Amazon (AMZN), Microsoft (MSFT), Tesla (TSLA) y Google (GOOG y GOOGL). Los tweets abarcaban un período de 5 años, desde el 1 de enero de 2015 hasta el 1 de enero de 2020. Debido a todos los factores mencionados anteriormente, se decidió utilizar este conjunto de datos para el proyecto.

Una vez obtenidos los datos textuales, se procedió a realizar el análisis de sentimiento para cada uno de los tweets disponibles. Para llevar a cabo esta tarea, se utilizó un modelo pre-entrenado de análisis de sentimiento obtenido de la plataforma Hugging Face [14], el cual había demostrado un buen rendimiento en trabajos anteriores. Se procesaron más de 3 millones de tweets mediante este modelo, lo que resultó en una distribución de probabilidad de positivo/neutral/negativo para cada tweet.

Dado que el único interés radicaba en determinar si el texto era positivo o negativo, se decidió considerar solo las salidas positivas y negativas, generando una nueva distribución de probabilidad. El conjunto de datos que encapsula la información sobre el sentimiento social con respecto a las empresas se denominará *Ds* para futuras referencias. *Ds* se compone de la siguiente manera:

Tabla 1: Dimensiones contenidas en conjunto de datos de sentimiento (Ds)

Date	Ticker Symbol	Sentiment	Likes	Retweets

El siguiente conjunto de datos se relaciona con las acciones de las empresas incluidas en *Ds*, y se buscó obtener datos en el mismo período de tiempo que está disponible en *Ds*. Para lograr esto, se utilizó la base de datos históricos de acciones de Yahoo Finance, que proporciona datos históricos diarios acerca de acciones de las empresas. Se descargaron los datos históricos de cada empresa, que estaban compuestos de la siguiente manera:

Tabla 2: Dimensiones contenidas en históricos de Yahoo Finance (por empresa)

Date	Open	High	Low	Close	Adj Close	Volume

Se creó un solo conjunto de datos utilizando los datos históricos de cada empresa y se seleccionaron únicamente los factores cruciales para la serie de tiempo. Como resultado de esta decisión, se eliminó la columna "Adj Close" que representa el valor de cierre de la acción en el contexto de otros factores que no son relevantes para esta propuesta. Por lo tanto, se excluyó esa dimensión de los datos. Para futuras referencias, este conjunto de datos se denominará *Dst* y se compone de la siguiente manera:

Tabla 3: Dimensiones contenidas en conjunto de dantos de acciones (Dst)

Date	Ticker Symbol	Open	High	Low	Close	Volume

Dst este compuesto de 7544 filas, lo que representa aproximadamente 1250 saltos de tiempo por empresa. El tamaño de la serie de tiempo varía ligeramente de una empresa a otra debido a factores particulares para cada una. Este número de filas será

el mismo que el número de filas del conjunto de datos final, ya que el resto de los datos están diseñados para complementar este subconjunto.

El siguiente conjunto de datos se enfocó en los factores macroeconómicos que se utilizarían en el análisis. Dado que los datos de las acciones están relacionados con el mercado de valores de los Estados Unidos, los factores macroeconómicos también pertenecen al mismo país. Afortunadamente, el Bureau of Economic Analysis (BEA) de los Estados Unidos proporciona todos los datos históricos de múltiples factores económicos de forma pública. Se encontró en la plataforma Kaggle un conjunto de datos que reúne diferentes factores económicos clave que se derivan de los informes del BEA [15]. Debido a la naturaleza de los datos, los factores macroeconómicos se evalúan en intervalos de tiempo más amplios en comparación con las unidades de tiempo de los conjuntos de datos anteriores. Por lo tanto, los intervalos de tiempo de estos factores son mensuales en lugar de diarios. Para futuras referencias, este conjunto de datos se denominará *Dm* y se compone de la siguiente manera:

Tabla 4: Dimensiones contenidas en conjunto de datos macroeconómicos (Dm)

Date	unrate	psr	m2	dspic	pce	reer	ir	ffer	tcs	indropro	ccpi

Tabla 5: Factores Macroeconomicos

unrate	Tasa de desempleo
psr	Tasa de ahorro personal
m2	Oferta Monetaria
dspic	Ingreso real disponible
Pce	Gastos de consumo personal
reer	Tipo de Cambio Efectivo Real

Ir	Tasa de interés para bonos federales a 10 años
Ffer	Tasa Efectiva de los Bonos Federales
Tcs	Inversión total en construcción
Indropro	Índice total de producción industrial
ccpi	Índice de Precios al Consumidor Subyacente

Es importante destacar que existen una gran cantidad de otros factores macroeconómicos que podrían ser utilizados. La selección de factores que se realizó proviene de un conjunto de datos utilizado originalmente para otro propósito, pero estos factores cumplen con la función de proporcionar un contexto económico al conjunto de datos final.

Finalmente, a partir de *Ds, Dst* y *Dm* podemos construir el conjunto de datos final (*Df*). El paso más complejo para lograr este objetivo es la ponderación del sentimiento diario por empresa. Esto se debe a que existe una gran cantidad de tweets por día para cada empresa y cada tweet tiene un valor de sentimiento asociado. Por lo tanto, es necesario ponderar los tweets para obtener un valor diario de sentimiento por empresa.

Para llevar a cabo la ponderación, primero se agrupan los datos de cada empresa en conjuntos separados para facilitar el proceso. Luego, se determina el "Factor de Consenso de Tweet" o TAS (por sus siglas en inglés). El TAS es un factor diseñado específicamente para este proyecto, que se utiliza para ponderar los "likes" (l) y los "retweets" (r), con el propósito de tomar en cuenta el alcance y las interacciones que

cada tweet tiene al momento de calcular el puntaje de sentimiento diario. El TAS se determina mediante la siguiente función:

$$tas = \sigma(0.4 * l + 0.6 * r)$$

El valor de los pesos son una distribución para representar que un retweet tiene más peso que un like, porque este genera mas alcance al tweet original. A partir del TAS de un tweet podemos obtener el factor de multiplicación del ese tweet en particular (M_t), para eso se utiliza la siguiente ecuación:

$$M_t = \lfloor tas * d\_max \rfloor$$

Donde d_max representa el valor máximo de duplicaciones para los tweets. Por ejemplo, para TAS=0.5 y d_max=6, esa entrada en *Ds* se repetiría 3 veces. De esta manera, se busca cuantificar el alcance y el apoyo adicional que tiene ese tweet en particular. Finalmente, se calcula un promedio diario de sentimiento y se vuelven a unir los datos de las diferentes empresas.

En su estado actual, las entradas de *Ds* y *Dst* están emparejadas temporalmente, lo que significa que para un momento t dado, se dispone de información sobre las acciones y el sentimiento en el mismo punto en el tiempo. Sin embargo, este emparejamiento temporal es un problema, ya que se busca establecer una relación de causalidad entre el sentimiento en Twitter de la empresa en t-1 y el comportamiento de la acción en t. Para generar esta relación, se desplazaron todas las fechas de sentimiento un día hacia adelante. De esta manera, se introduce el concepto de causalidad al unir el conjunto de datos por fecha y empresa. El conjunto de datos final preliminar estaría compuesto por las siguientes dimensiones:

Tabla 6: Dimensiones preliminares para conjunto de datos final(Df)

Date	Sentimiento Ponderado	Ticker Symbol	Open	High	Low	Close	Volume

Finalmente, se busca agregar a *Dm* al conjunto de datos preliminar. Esto se logra duplicando las entradas de *Dm* (intervalos de tiempo mensuales) para que coincidan con las unidades de tiempo del conjunto preliminar (intervalos de tiempo diarios). Esto implica que los mismos valores de los factores macroeconómicos se repiten dentro de un mes, para todos los días, en el conjunto de datos completo. Las dimensiones del conjunto de datos completo están compuestas de la siguiente manera:

Tabla 7: Dimensiones de conjunto de datos final (Df)

Date	Sentimiento Ponderado	Ticker Symbol	Open	High	Low	Close	Volume	unrate	psr

m2	dspic	pce	reer	ir	ffer	tcs	indropro	ccpi

Esto quiere decir que el conjunto de datos final este compuesto por una dimensión total de 19 columnas por 7544 filas.

Este conjunto de datos final será transformado para generar el conjunto de entrada y de objetivo del modelo, lo que resultará en la eliminación de 2 dimensiones del conjunto de datos final: ‘Date’ y ‘Ticker’ symbol. Date será eliminada, ya que solo se utilizaría para ordenar los datos en orden cronológico y una vez que se cumpla ese objetivo, ya no es requerida. Ticker symbol será eliminado porque cada arquitectura se entrenará con las series de tiempo respectivas para cada empresa, por lo tanto, el modelo no necesita saber a qué empresa pertenece. Por último, se formará un agrupamiento en

3 diferentes dimensiones (Identificador temporal, Ventana de retroceso, características).

Debido a que se está evaluando información solo de una empresa, el tamaño total de saltos de tiempo por empresa será de alrededor de 1257 por empresa. Este valor se reduce por la cantidad de saltos de tiempo que se consideren dentro de la ventana de retroceso, en este caso, será de siete días. Por cada salto de tiempo, se evaluarán siete días consecutivos para la predicción del octavo día, y por cada día que se evalúe para la predicción del octavo, existen 17 características (las características del conjunto de datos final menos Date y Ticker symbol). Esto resultará en que el conjunto de variables de entrada por empresa tenga la dimensionalidad (1250 aprox., 7, 17).

El conjunto de variable objetivo estará compuesto del octavo valor en secuencia por cada salto de tiempo del conjunto de entrada, pero solo se enfocará en la dimensión 'Close'. Por lo tanto, el conjunto de datos de la variable objetivo tendrá la dimensionalidad de (1250 aprox., 1).

Los conjuntos de variables de entrada y variables objetivo se dividirán en entrenamiento, validación y prueba, respectivamente, para el entrenamiento y evaluación de cada modelo.

Configuración de arquitecturas:

Debido a factores de tiempo el número de hiper parámetros que se optimizo fue reducido y en un rango relativamente bajo por arquitectura. Adicionalmente se optimizaron los hiper parámetros de todas las arquitecturas por empresa, de esa forma cada arquitectura tiene una configuración que genere mejor rendimiento para cada empresa.

LSTM:

Tabla 8: Hiper parámetros optimizados en LSTM

Hiper parámetro	**Valores**
Unidades	En un rango de 2-240 en saltos de 2

Tabla 9: Hiper parámetros constantes en LSTM

Hiper parámetro	**Valores**
Capas	(2) [LSTM, Dense]
Activación LSTM	ReLu
Optimizador	Adam (tasa de aprendizaje=1e-4)

BiLSTM:

Tabla 10:Hiper parámetros optimizados en BiLSTM

Hiper parámetro	**Valores**
Unidades	En un rango de 2-240 en saltos de 2

Tabla 11:Hiper parámetros Constantes en BiLSTM

Hiper parámetro	**Valores**
Capas	(2) [biLSTM, Dense]
Activación LSTM	ReLu
Optimizador	Adam (tasa de aprendizaje=1e-4)

GRU:

Tabla 12: Hiper parámetros optimizados en GRU

Hiper parámetro	**Valores**
Unidades	En un rango de 2-240 en saldos de 2

Tabla 13:Hiper parámetros constantes en GRU

Hiper parámetro	**Valores**
Capas	(2) [GRU, Dense]
Activación GRU	ReLu
Optimizador	Adam (tasa de aprendizaje=1e-4)

BiGRU:

Tabla 14:Hiper parámetros optimizados en BiGRU

Hiper parámetro	**Valores**
Unidades	En un rango de 2-240 en saltos de 2

Tabla 15:Hiper parámetros constantes en BiGRU

Hiper parámetro	**Valores**
Capas	(2) [biGRU, Dense]
Activación GRU	ReLu
Optimizador	Adam (tasa de aprendizaje=1e-4)

Los hiper parámetros no mencionados en ninguna de las tablas anteriores corresponden a hiper parámetros constantes que tienen los valores predeterminados en la implementación de “Keras” para cada capa.

TCN:

Tabla 16:Hiper parámetros optimizados en TCN

Hiper parámetro	**Valores**

Filtros	En un rango de 16-256 en saltos de 16
Tamaño de Kernel	En un rango de 3-7 en saltos de 1
Dropout	0.0, 0.2 y 0.3
Dilataciones	Se calcula mediante la ventana de retroceso el tamaño del kernel y el factor de dilatación para conseguir la secuencia de para que se logre capturar toda la secuencia de tiempo ingresada.

Tabla 17:Hiper parámetros constantes en TCN

Hiper parámetro	**Valores**
Capas	(2) [TCN*, Dense]
Activación GRU	ReLu
Optimizador	Adam (tasa de aprendizaje=1e-4)
Padding	Causal

biTCN:

Tabla 18: Hiper parámetros optimizados en BiTCN

Hiper parámetro	**Valores**
Filtros	En un rango de 16-256 en saltos de 16
Tamaño de Kernel	En un rango de 3-7 en saltos de 1
Dropout	0.0, 0.2 y 0.3
Dilataciones	Se calcula mediante la ventana de retroceso el tamaño del kernel y el factor

	de dilatación para conseguir la secuencia de para que se logre capturar toda la secuencia de tiempo ingresada.

Tabla 19: Hiper parámetros constantes en BiTCN

Hiper parámetro	**Valores**
Capas	(2) [biTCN*, Dense]
Activación GRU	ReLu
Optimizador	Adam (tasa de aprendizaje=1e-4)
Padding	Causal

Los hiper parámetros no mencionados en ninguna de las tablas anteriores corresponden a hiper parámetros constantes que tienen los valores predeterminados en la implementación de Keras-TCN [18] para la capa TCN. Es importante aclarar que, a pesar de que la implementación de Keras-TCN permite agregar la arquitectura TCN como una sola capa, la arquitectura TCN está compuesta por bloques residuales que internamente contienen capas convolucionales de 1 dimensión.

Utilizando las configuraciones obtenidas dentro de la optimización de hiper parámetros, en total se obtuvo un conjunto de 36 configuraciones diferentes que se detallan en la tabla a continuación.

Tabla 20: Numero de parámetros por modelo

	AAPL	**AMZN**	**GOOG**	**GOOGL**	**MSFT**	**TSLA**	**Promedio**
LSTM	54,541	145,783	52,683	47,301	114,081	60,307	79,116
BiLSTM	369,565	472,213	68,713	163,213	57,305	297,713	238,120.33
GRU	7,729	66,921	84,057	20,721	76,201	38,481	49,018.33

BiGRU	54,353	367,473	156,257	263,201	39,457	141,121	170,310.33
TCN	459,841	104,641	68,353	579,569	148,609	291,809	275,470.33
BiTCN	691,969	775,457	2,133,601	1,779,841	961,601	400,513	1,123,830.33

Métricas y métodos de evaluación:

Se utilizará "Time Series Split" (TSS), que es un método de validación cruzada específico para modelos de series de tiempo. Este método dividirá el conjunto de datos en 10 partes y mediante el método de ventana creciente, se evaluará el desempeño del modelo un total de 10 veces. El TSS maneja la división entre el conjunto de entrenamiento y el conjunto de prueba. Adicionalmente, se implementó una división adicional para que el 20% del conjunto de entrenamiento se utilice para validación.

Para los 36 modelos discutidos anteriormente, se evaluará su desempeño mediante la medición de cinco métricas: error cuadrado medio (MSE), error absoluto medio (MAE), error porcentual absoluto medio (MAPE), Error cuadrático medio raíz (RMSE) y error porcentual medio (MPE). Además, se determinará estadísticamente si un modelo se desempeña mejor que el resto de forma significativa. Esto se hará mediante una prueba de hipótesis utilizando la prueba de rango con signos de Wilcoxon. Esta evaluación se llevará a cabo con un nivel de significancia (α) de 0.05. Para la evaluación se tomara en cuenta la significancia estadística de todas las métricas de manera individual.

Resultados y Discusión:

A continuación, se presentan los resultados de la evaluación de las arquitecturas mediante las métricas de error preestablecidas en un gráfico de barras. Mas adelante,

también se detallarán los valores exactos con desviación estándar en las pruebas de hipótesis. Estas evaluaciones se llevaron a cabo sobre el conjunto de prueba. Para tener una mejor perspectiva del entrenamiento y desempeño por empresa se recomienda ver el Anexo B y C.

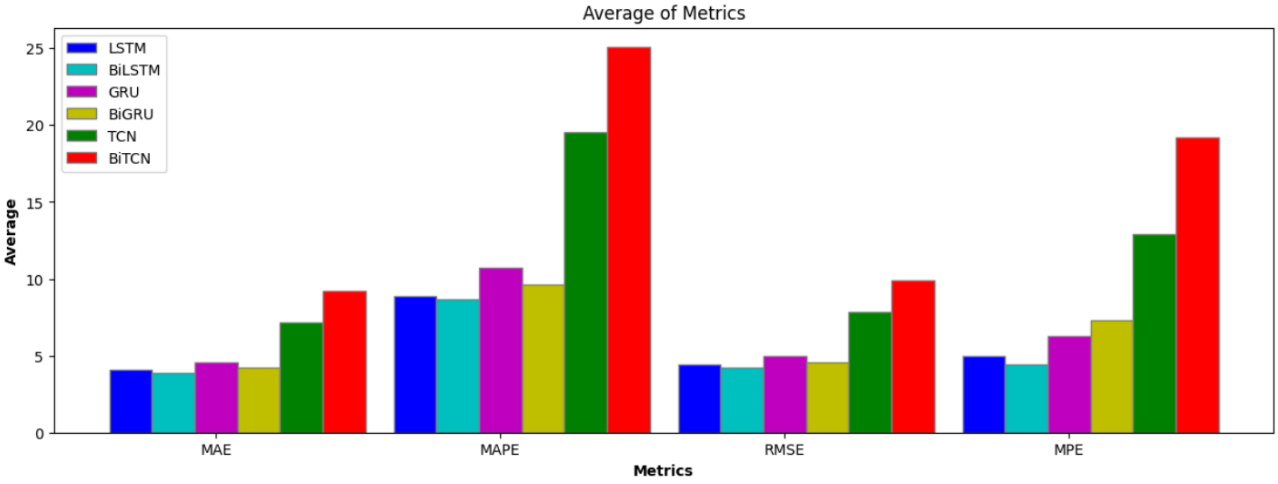


Ilustración 1: visualización de diversas métricas

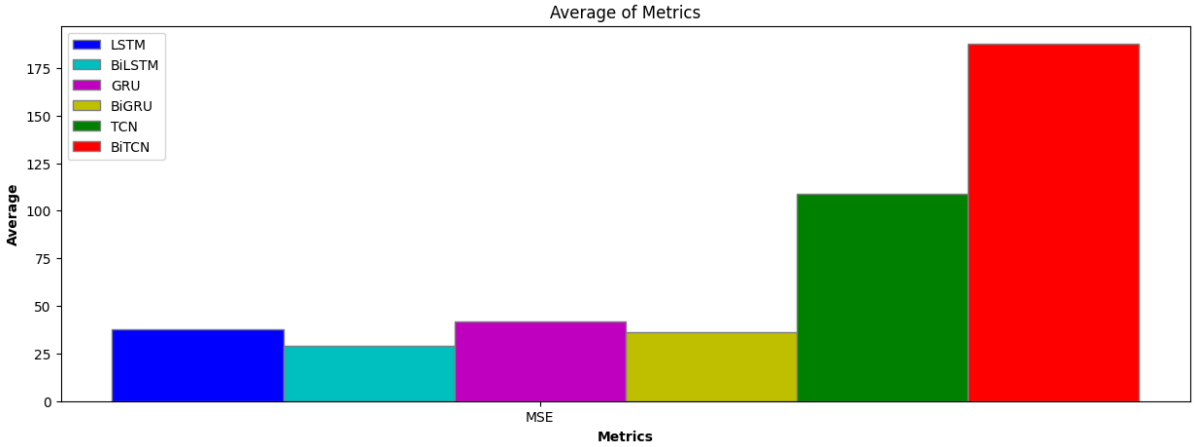


Ilustración 2: visualización de MSE

A través de las ilustraciones, podemos apreciar el rendimiento de cada arquitectura. Esto evidencia dos aspectos importantes. Primero, se observa que el rendimiento de las arquitecturas TCN es notablemente inferior al resto de las arquitecturas basadas en RNNs. En segundo lugar, es crucial señalar que, entre las

RNNs evaluadas, su rendimiento es relativamente uniforme. Así mismo, es importante notar que las RNNs muestran una mayor divergencia entre ellas al evaluar el MPE.

Mediante los datos presentados en los puntos anteriores se puede empezar a formar una idea de desempeño de las arquitecturas, pero es importante determinar qué tan significativa es la diferencia entre las arquitecturas evaluadas. Para esto se presentan los resultados de la prueba de rango con signos de Wilcoxon, con un nivel de significancia de 0.05. La prueba se realizó con la arquitectura con BiLSTM como base, ya que presentan el mejor rendimiento en las métricas evaluadas anteriormente.

Tabla 21: Resultados de prueba de hipótesis con métricas de error promedio entre empresas

- *"*" representa las métricas que no tienen diferencias estadísticas significativas*
- *Alado del promedio de cada métrica se representa la desviación estándar encerradas en paréntesis.*

Arquitecturas	**MSE**	**MAE**	**MAPE**	**RMSE**	**MPE**	**Params**
LSTM	**37.64* (47.06)**	**4.055* (2.575)**	**8.851* (2.498)**	**4.422* (2.710)**	**4.978* (4.713)**	79,116
BiLSTM	28.96 (27.49)	3.848 (1.900)	8.666 (1.885)	4.248 (2.059)	4.416 (3.521)	238,120
GRU	41.93 (42.53)	4.560 (2.400)	10.70 (3.384)	4.999 (2.539)	**6.305* (4.832)**	49,018
BiGRU	**36.38* (36.96)**	**4.190* (2.329)**	**9.654* (2.560)**	**4.577* (2.489)**	7.283 (3.736)	170,310
TCN	108.9 (109.7)	7.153 (3.860)	19.56 (9.685)	7.869 (4.188)	12.92 (11.24)	275,460
BiTCN	187.7 (171.1)	9.203 (4.916)	25.08 (7.678)	9.923 (5.203)	19.19 (7.055)	1,123,830

A través de las pruebas de hipótesis realizadas, logramos obtener una comprensión más precisa del rendimiento de las arquitecturas evaluadas. Se evidencia que, en orden de mejor a peor rendimiento, las arquitecturas destacadas son BiLSTM, LSTM y BiGRU. Esta conclusión se basa en los resultados de las métricas de error consideradas. Sin embargo, al someter el análisis a una prueba de hipótesis, el panorama se aclara aún más.

Para las BiGRUs, no se observa una diferencia estadísticamente significativa en las primeras cuatro métricas en comparación con las BiLSTMs. No obstante, en el caso del MPE, no solo se identifica una diferencia significativa, de forma similar a TCNs y BiTCNs. Estos resultados indican que las BiGRUs entrenadas en este proyecto tienden a sobreestimar de manera significativa, descartándolas como la arquitectura más eficiente.

En cuanto a LSTMs y BiLSTMs, la situación se torna más compleja. Si bien se evidencia un mejor rendimiento de las BiLSTMs en comparación con las LSTMs, la prueba de hipótesis revela que, en ninguna métrica evaluada, las BiLSTMs superan de manera significativa a las LSTMs. Este análisis lleva a considerar la cantidad de parámetros en juego. Dado que las LSTMs tienen un rendimiento estadísticamente similar a las BiLSTMs con aproximadamente un tercio de los parámetros, se concluye que la arquitectura más eficiente entre las evaluadas seria la LSTM.

Adicionalmente, se logró determinar una aparente relación entre el rango de la serie de tiempo y el error en su predicción. Para ilustrar esto se utilizará la métrica de MSE por empresa para los modelos de LSTM y BiLSTM.

Tabla 22: Rangos de precios y MSE por empresa

	AAPL	**AMZN**	**GOOG**	**GOOGL**	**MSFT**	**TSLA**
LSTM	9.03	48.79	14.07	13.12	137.66	3.21
BiLSTM	12.72	42.25	23.07	7.67	83.75	4.36
Rango de precios	20.83-70.58	14.31-101.63	24.80-68.75	24.85-68.03	35.06-153.15	9.86-28.72

Magnitud rango de precios	49.75	87.32	43.95	43.18	118.09	18.86

Al analizar la tabla anterior se puede identificar una posible relación entre la magnitud del rango de precios y su magnitud de error. Cabe recalcar que a pesar de solo estar visualizando una métrica para dos arquitecturas este fenómeno se pude identificar en la gran mayoría de métricas por empresa que se obtuvieron. Esto sugiere una relación entre error y el rango de los precios del conjunto de datos, que necesitaría una investigación dedicada para probarse.

CONCLUSIONES

El propósito fundamental de este proyecto fue determinar la arquitectura más eficiente, considerando tanto su eficacia para predicción de precio de cierre de acciones como su utilización de recursos. A través de la utilización de datos provenientes de diferentes contextos, se llevó a cabo un análisis extenso utilizando diversas métricas de evaluación. El enfoque consistió en evaluar 3 arquitecturas distintas con componentes unidireccionales y bidireccionales, para las cuales se construyeron un total de 36 modelos de aprendizaje profundo. Este proceso condujo a la evaluación de 5,040 configuraciones diferentes, con el objetivo de determinar los modelos óptimos para cada arquitectura.

Los resultados obtenidos en este proyecto alcanzan el objetivo principal planteado, ofreciendo una visión clara sobre qué arquitectura destaca como la más eficiente en el conjunto evaluado. Además, se presentan oportunidades adicionales de seguir explorando campo de TSF

expandiendo el análisis realizado, con el propósito de entender mejor las herramientas presentes.

En la culminación de este proyecto se logró determinar que estructura más eficiente evaluada en este contexto. Las BiLSTMs lograron el mejor desempeño evaluando simplemente las métricas de error, sin embargo, mediante pruebas de hipótesis se logró determinar que la diferencia de desempeño entre LSTMs y BiLSTMs no es estadísticamente significativa. Debido a que en promedio las LSTMs utilizan alrededor de un tercio de los parámetros utilizados en las BiLSTMs, debido a esto, se determinó que las LSTMs fueron la arquitectura más eficiente en el contexto evaluado.

A pesar de la conclusión de esta investigación aún existe un campo muy diverso por explorar ya sea investigaciones como esta centradas en otro tipo de tareas como NLP, visión de computador o dentro del campo de TSF. Hay un gran número de herramientas que se pueden seguir analizando. Dentro de mismo proyecto surgieron aspectos, que por cuestiones de tiempo quedaron para posibles investigaciones futuras, la primera seria explorar la arquitectura de Transformers en el mismo contexto. Los Transformers han logrado ser más eficientes que las LSTMs en tareas de NLP y seria interesante evaluar su desempeño en TSF en este contexto también. Por igual, también existe la posibilidad de realizar un análisis más a profundidad para determinar la verdadera relación entre la magnitud del rango de precios y la magnitud el error de predicción sugerido en este proyecto.

Mas allá de cumplir con el objetivo del proyecto, en lo personal, se logró obtener un conocimiento mas amplio del campo del aprendizaje profundo y de TSF, areas que no son profundizadas en la malla curricular. Esto conllevo a una ardua cantidad de investigación de los temas tratados en este proyecto, que resulto en tener mejores capacidades para entender y desarrollar más proyectos como este.

En definitiva, se logró cumplir con el objetivo principal del proyecto y se determinó que las LSTMs fueron la arquitectura ms eficiente en el contexto evaluado. A pesar de concluir con este proyecto, se abrieron nuevas oportunidades de investigaciones futuras debido a más herramientas por evaluar, los resultados del proyecto y las habilidades obtenidas en el mismo.

REFERENCIAS BIBLIOGRÁFICAS

[1] J. Shi, “Time Series Forecasting (TSF) using various deep learning models,” *arXiv.org*, Apr. 23, 2022. https://arxiv.org/abs/2204.11115

[2] R. Cholakov, “Transformers predicting the future. Applying attention in next-frame and time series forecasting,” *arXiv.org*, Aug. 18, 2021. https://arxiv.org/abs/2108.08224

[3] J. Chung, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv.org*, Dec. 11, 2014. https://arxiv.org/abs/1412.3555

[4] Jiang, T. (2023, June 3). Financial sentiment analysis using FinBERT with application in predicting stock movement. arXiv.org. https://arxiv.org/abs/2306.02136

[5] D. Li, T. Ya, Y. Zhang, S. Miao, and S. He, “Probabilistic forecasting method for mid-term hourly load time series based on an improved temporal fusion transformer model,” *International Journal of Electrical Power & Energy Systems*, vol. 146, p. 108743, Mar. 2023, doi: 10.1016/j.ijepes.2022.108743.

[6] Bai, S. (2018, March 4). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv.org. https://arxiv.org/abs/1803.01271

[7] Fan, J., Zhang, K., Huang, Y., Zhu, Y., & Chen, B. (2021). Parallel spatio-temporal attention-based TCN for multivariate time series prediction. *Neural Computing and Applications*, *35*(18), 13109–13118. https://doi.org/10.1007/s00521-021-05958-z

[8] P. Hewage *et al.*, “Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station,” *Soft Computing*, vol. 24, no. 21, pp. 16453–16482, Apr. 2020, doi: 10.1007/s00500-020-04954-0.

[9] Saxena, S. (2023, October 25). *What is LSTM? Introduction to Long Short-Term Memory*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

[10] Kostadinov, S. (2019, November 10). Understanding GRU Networks - towards Data science. *Medium*. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[11] “Temporal Convolutional Networks and Forecasting - Unit8,” *Unit8*, Dec. 12, 2022. https://unit8.com/resources/temporal-convolutional-networks-and-forecasting/

[12] S. Bai, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” *arXiv.org*, Mar. 04, 2018. https://arxiv.org/abs/1803.01271

[13] “Speculator and influencer evaluation in stock market by using social media,” *IEEE Conference Publication | IEEE Xplore*. https://ieeexplore.ieee.org/document/9378170

[14] Barbieri, Francesco, et al. "Tweeteval: Unified benchmark and comparative evaluation for tweet classification." arXiv preprint arXiv:2010.12421 (2020).

[15] *USA key economic indicators*. (2022, January 8). https://www.kaggle.com/datasets/calven22/usa-key-macroeconomic-indicators

[16] R. Aggarwal, “Bi-LSTM - Raghav Aggarwal - Medium,” *Medium*, Dec. 10, 2021. [Online]. Available: https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0

[17] F. Kayim and A. Yilmaz, "Time Series Forecasting With Volatility Activation Function," in IEEE Access, vol. 10, pp. 104000-104010, 2022, doi: 10.1109/ACCESS.2022.3211312.

[18] Philipperemy. (n.d.). *GitHub - philipperemy/keras-tcn: Keras Temporal Convolutional Network.* GitHub. https://github.com/philipperemy/keras-tcn/tree/master

ANEXO A: REPOSITORIO DE GITHUB:

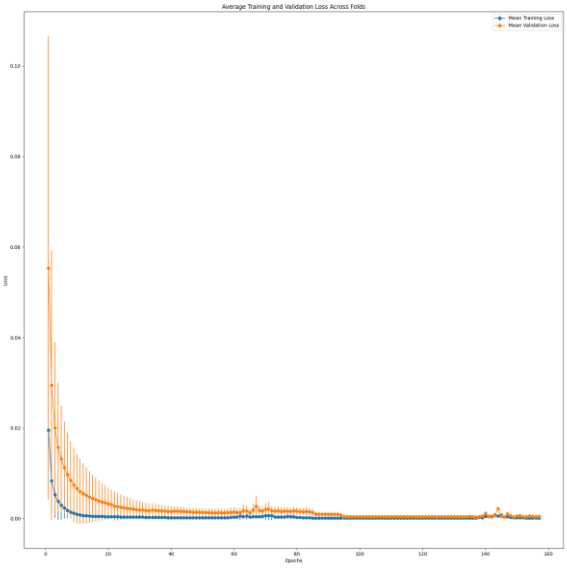
En este repositorio se encontrará todo el código utilizado para realizar este proyecto.

ANEXO B: PROMEDIOS DE CURVAS DE APRENDIZAJE POR FOLD:

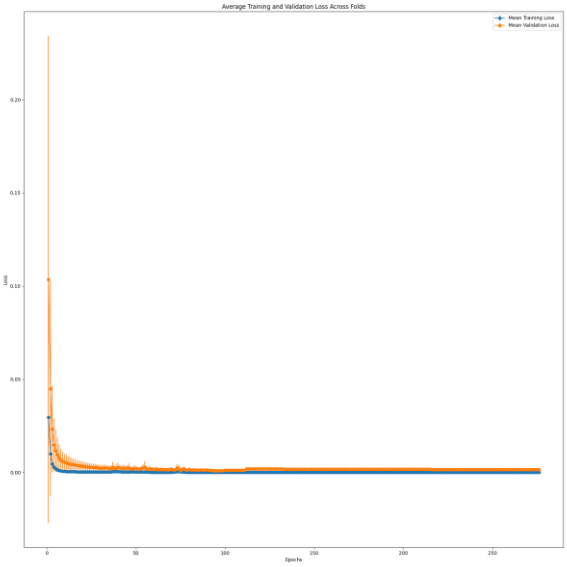
A continuación, se presentan las curvas de aprendizaje promedio de los 10 Time Series Split de entrenamiento de cada modelo con sus desviaciones estándares correspondientes.

LSTM:

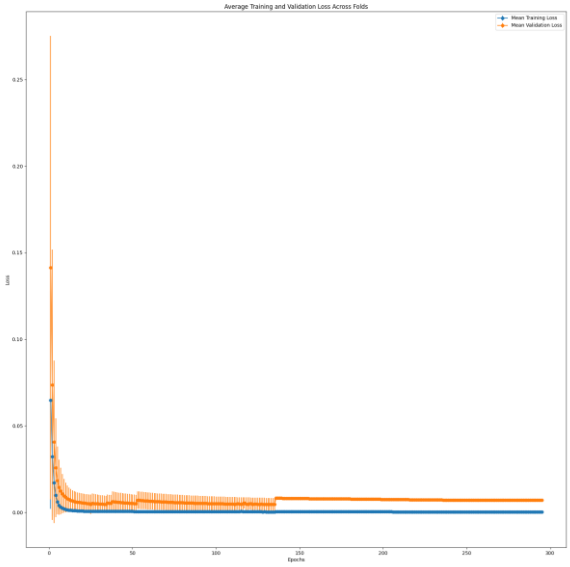
AAPL:



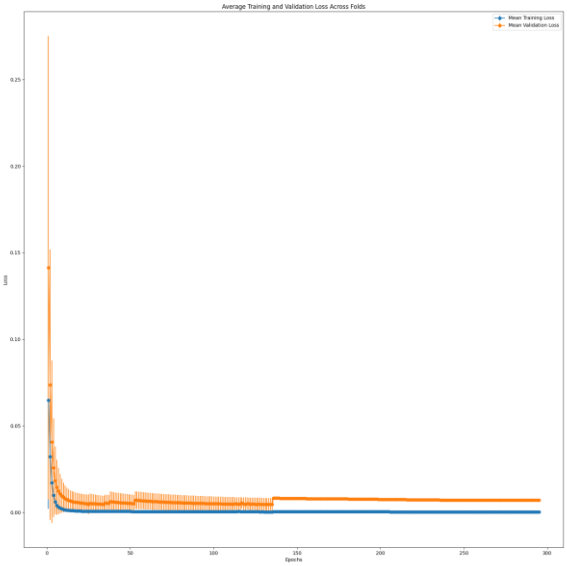
AMZN:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epochs

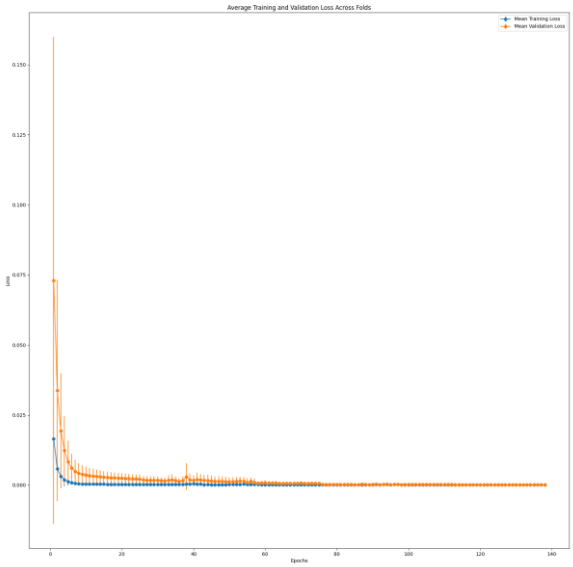
GOOG:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epochs

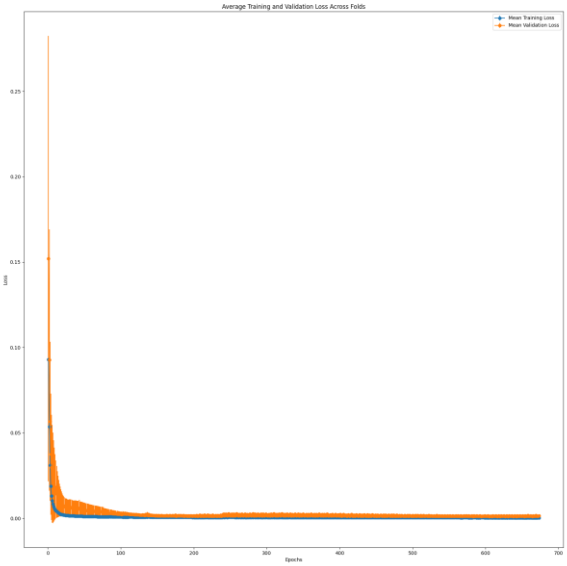
GOOGL:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epochs

MSFT:

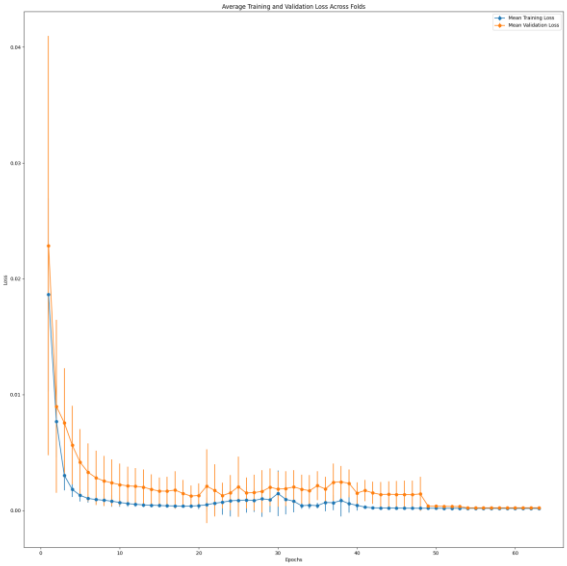

Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epochs

TSLA:

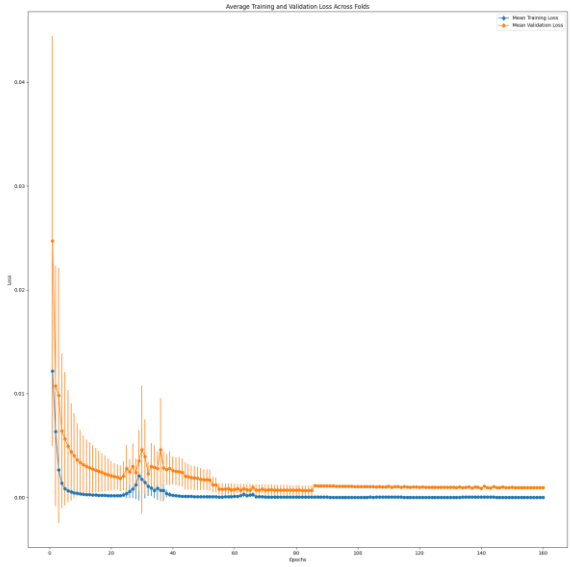


BiLSTM:

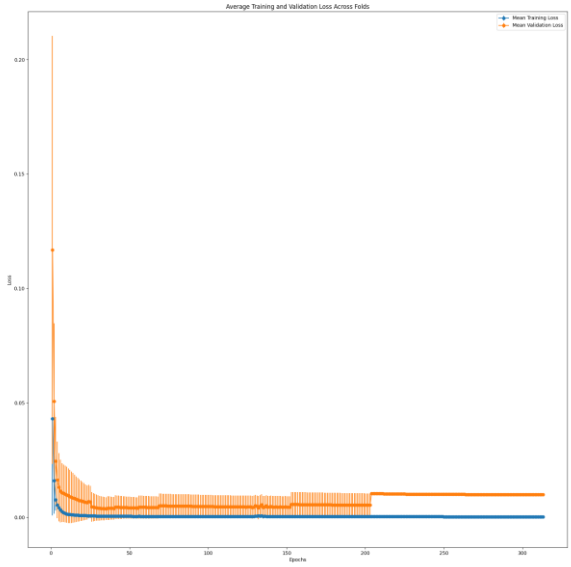
AAPL:



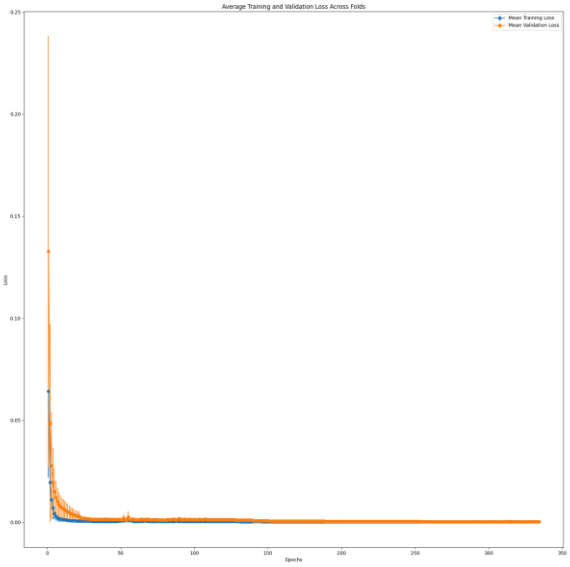
AMZN:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

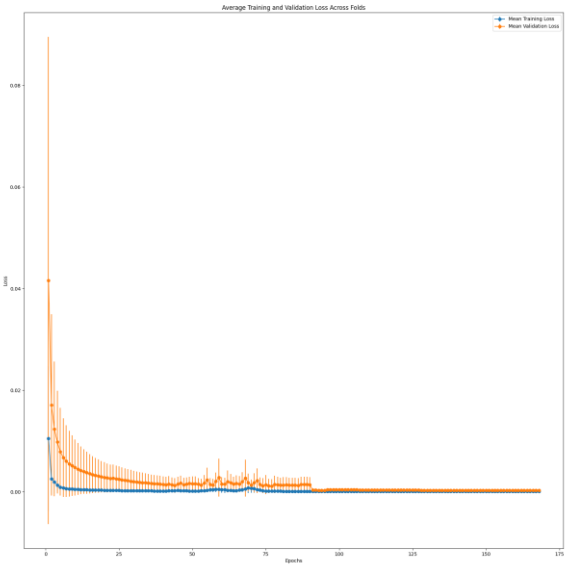
GOOG:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

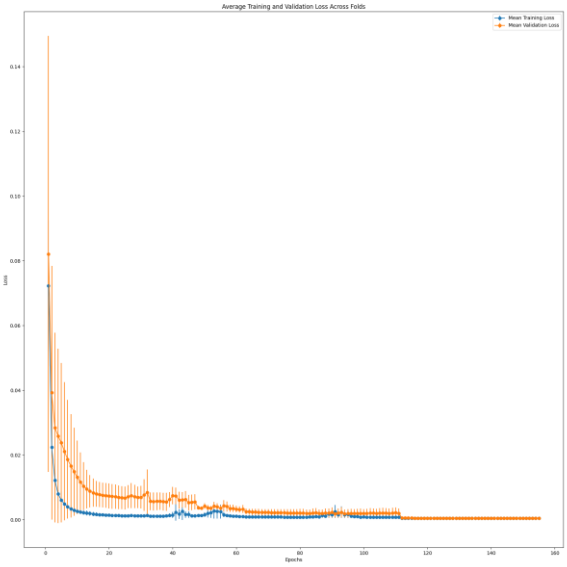
GOOGL:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

MSFT:

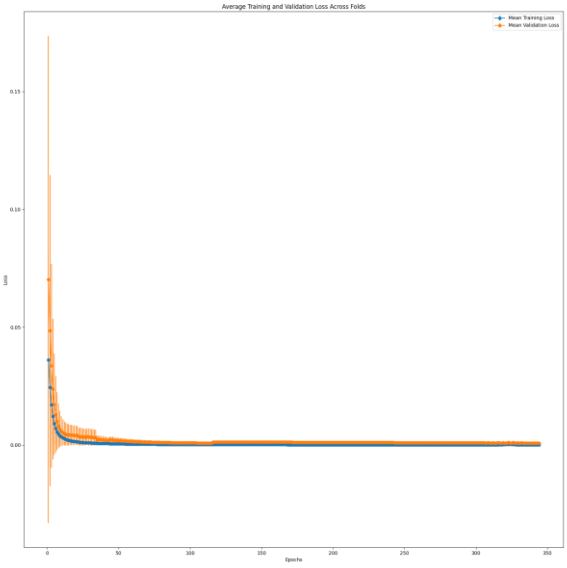

Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

TSLA:

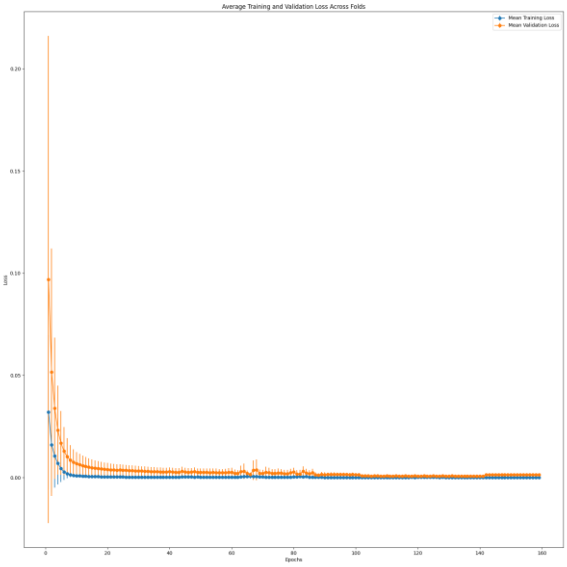


GRU:

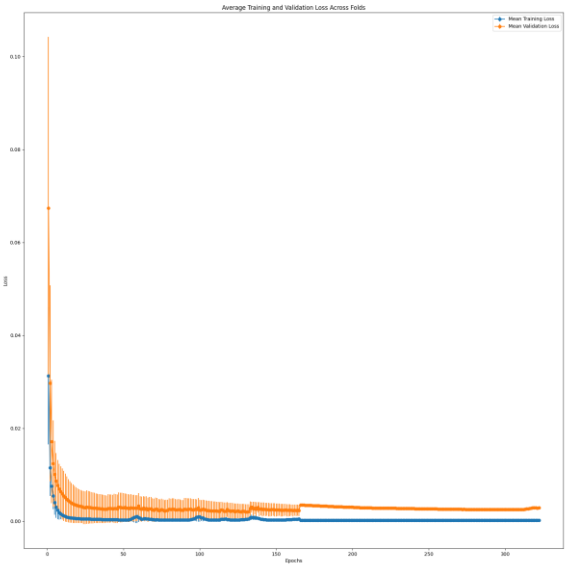
AAPL:



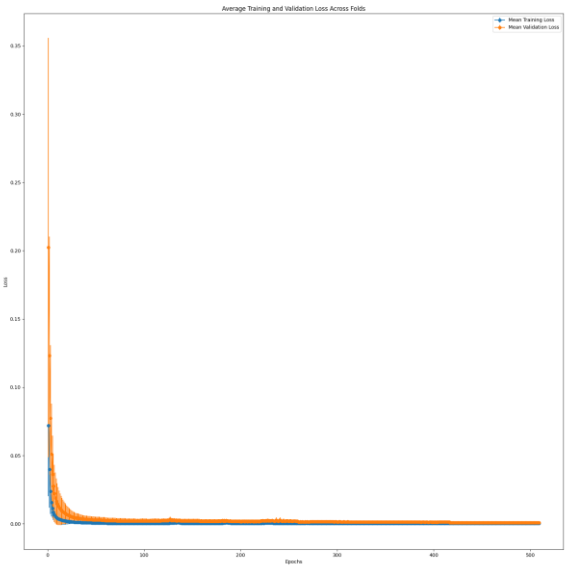
AMZN:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

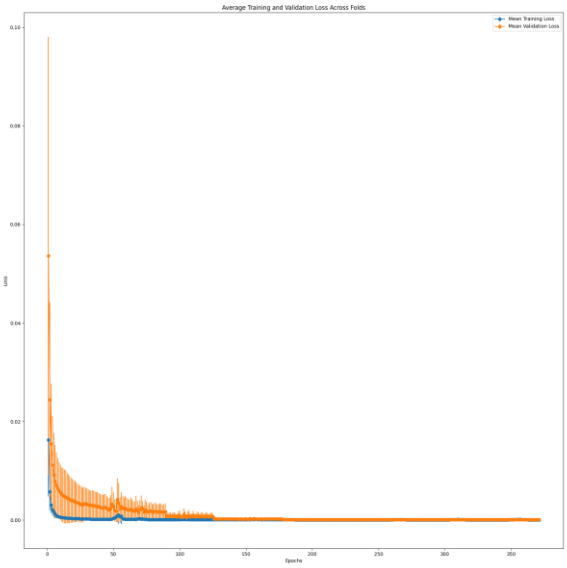
GOOG:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

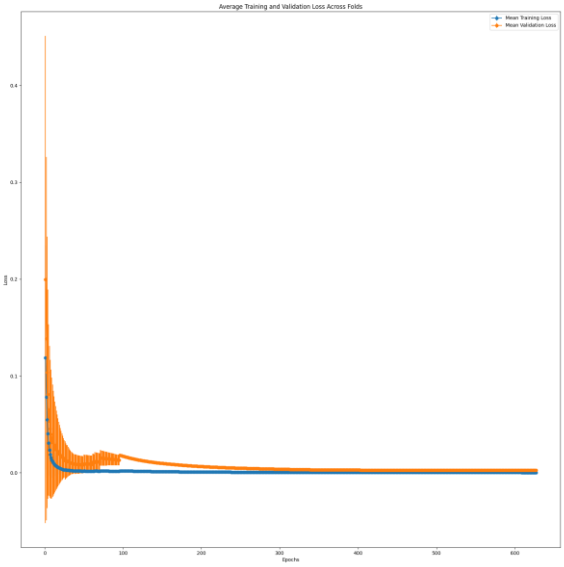
GOOGL:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoch

MSFT:

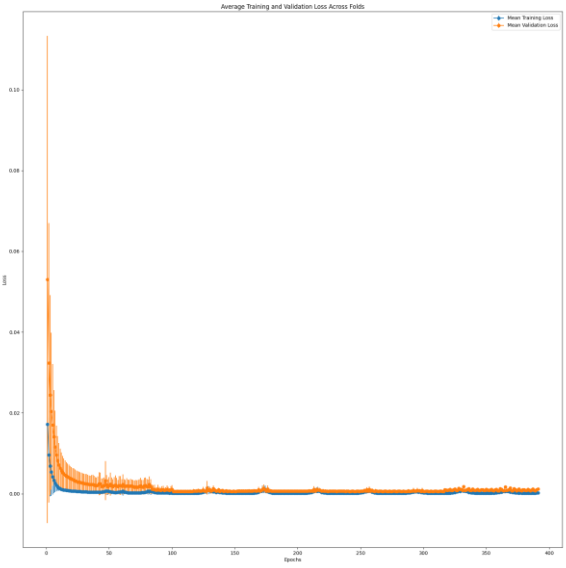

Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoch

TSLA:

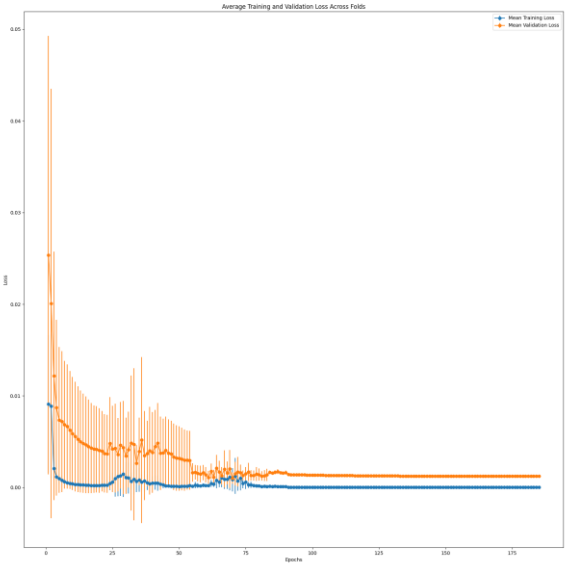


BiGRU:

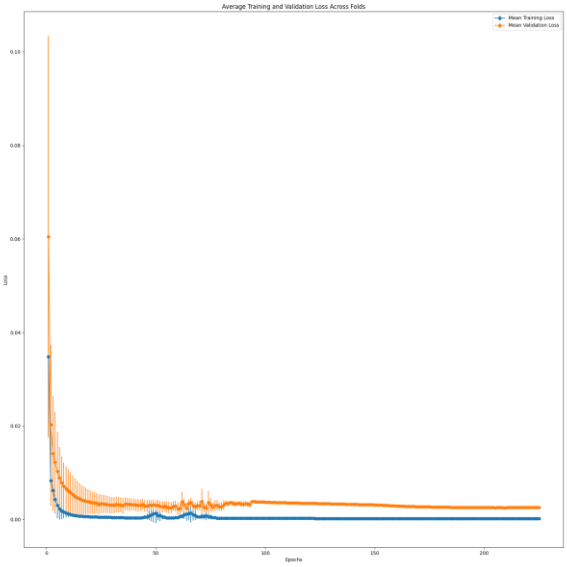
AAPL:



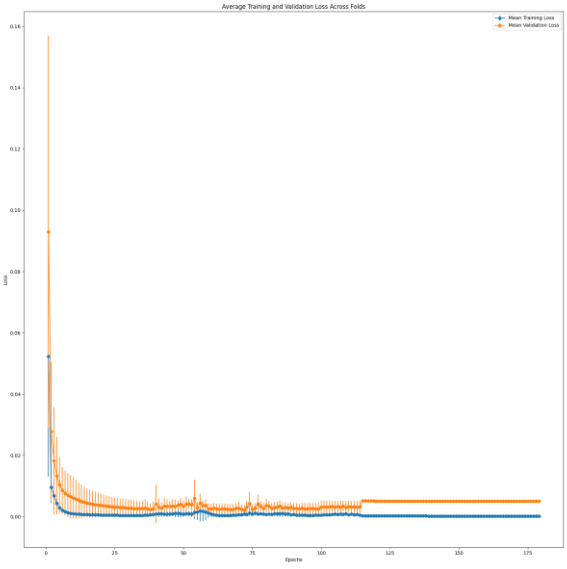
AMZN:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

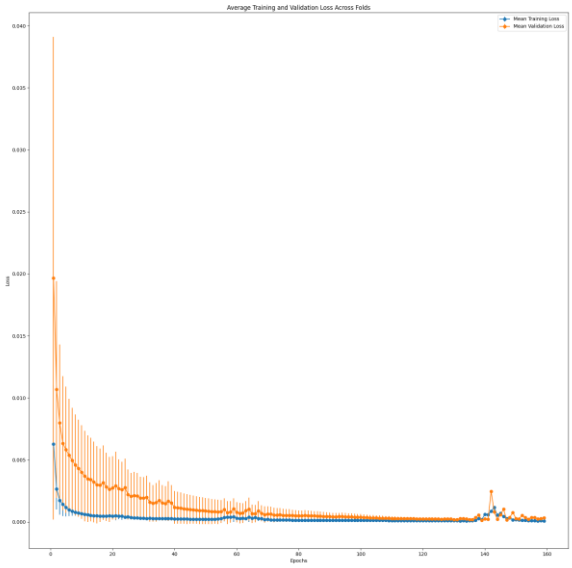
GOOG:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

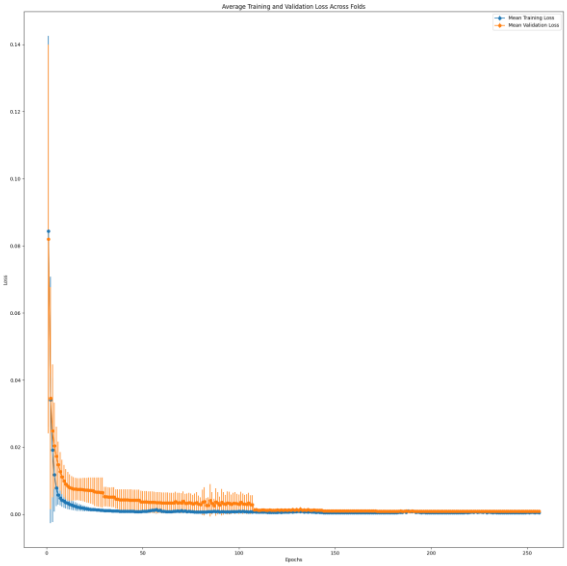
GOOGL:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

MSFT:

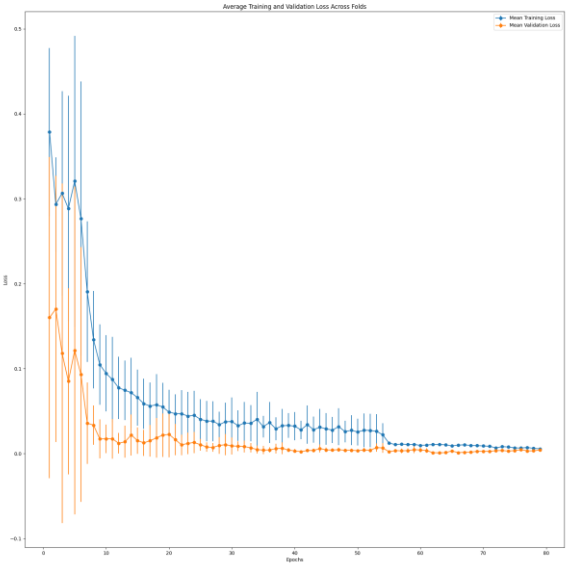

Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

TSLA:

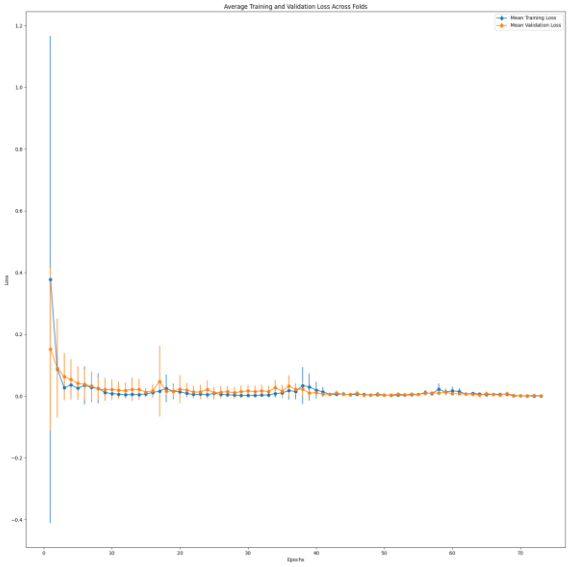


TCN:

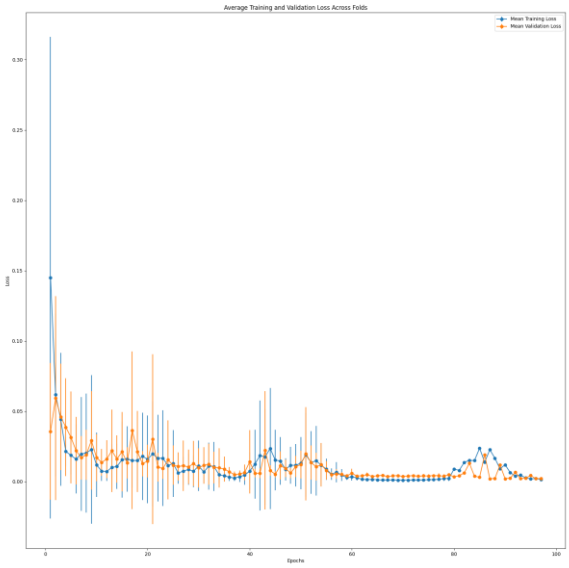
AAPL:



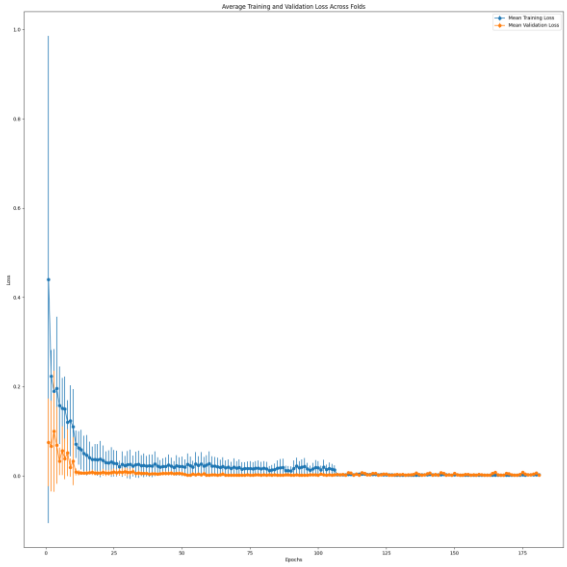
AMZN:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epochs

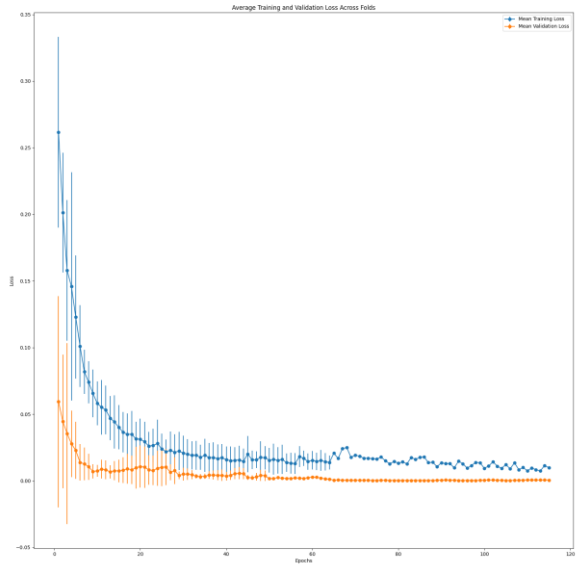
GOOG:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epochs

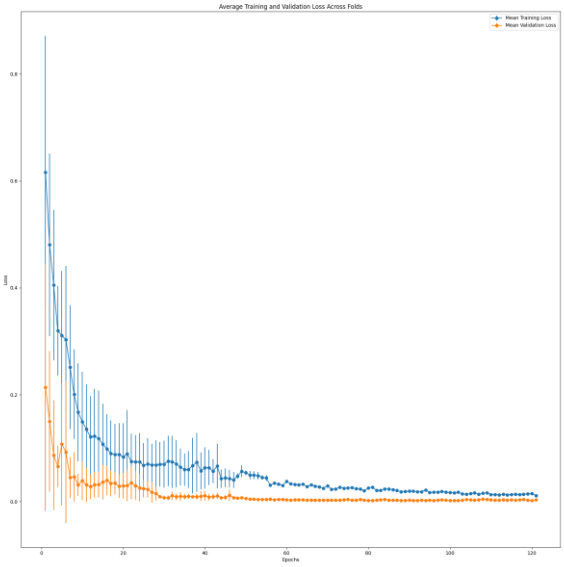
GOOGL:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoch

MSFT:

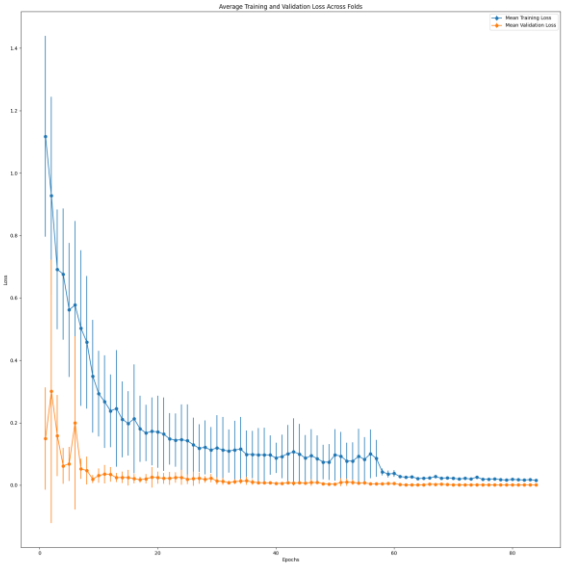

Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoch

TSLA:

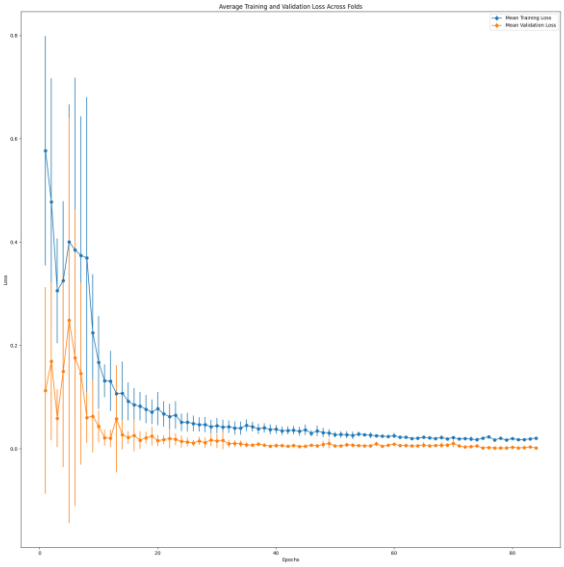


BiTCN:

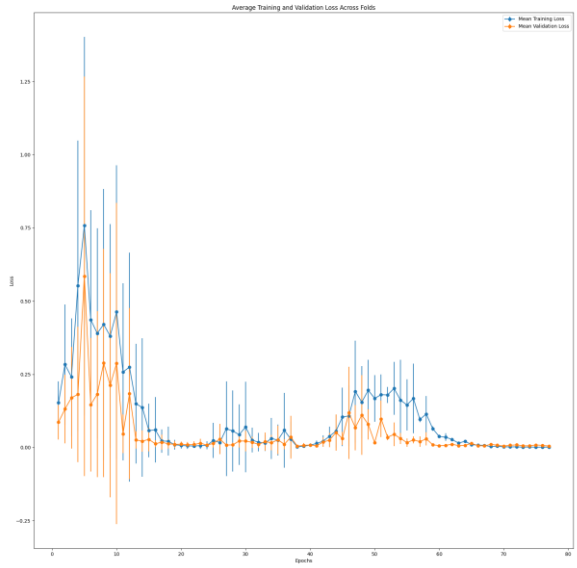
AAPL:



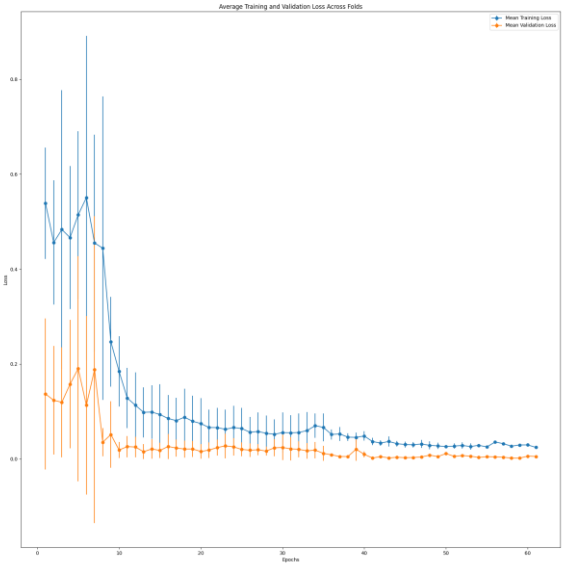
AMZN:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

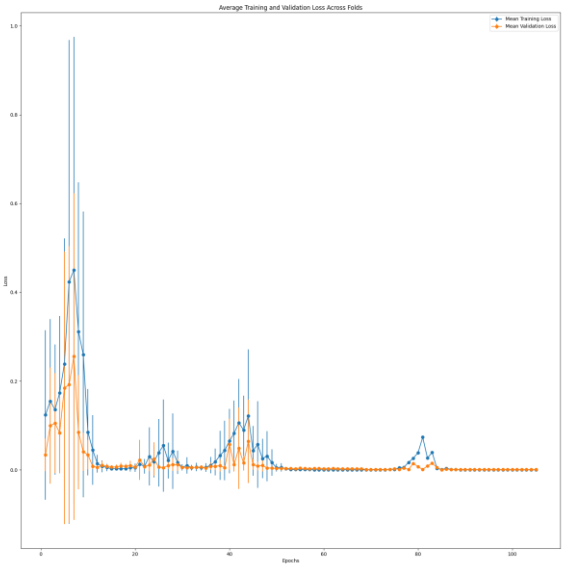
GOOG:

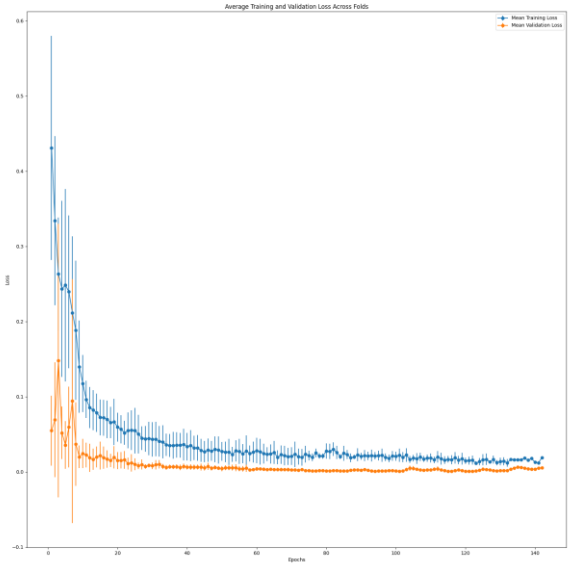

Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epoche

GOOGL:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epochs

MSFT:


Average Training and Validation Loss Across Folds
Mean Training Loss
Mean Validation Loss
Loss
Epochs



TSLA:

ANEXO C: PREDICCIONES

A continuación, se muestran las predicciones del último split de series temporales para la empresa Tesla, realizadas por todos los modelos. Es importante señalar que esta predicción representa una décima parte de todas las utilizadas para evaluar el rendimiento de las arquitecturas con el conjunto de datos de TSLA. Por esta razón, las GRUs exhiben un excelente desempeño en esta predicción específica, a pesar de eso las LSTMs muestran una mayor consistencia en todas las predicciones de los splits.

LSTM:


real
val_pred
test_pred
27.5
25.0
22.5
20.0
17.5
15.0
12.5
900
950
1000
1050
1100
1150
1200
1250

BiLSTM:


real
val_pred
test_pred
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
900
950
1000
1050
1100
1150
1200
1250

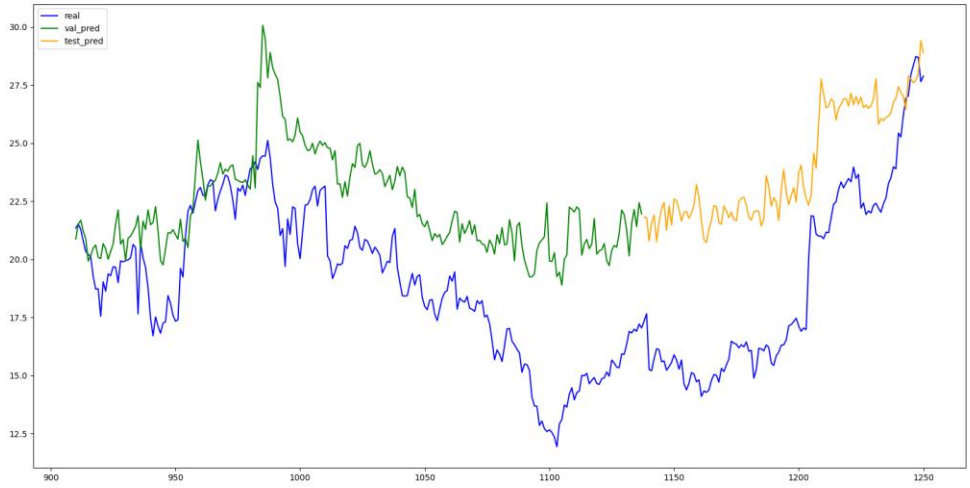
GRU:


real
val_pred
test_pred
27.5
25.0
22.5
20.0
17.5
15.0
12.5
900
950
1000
1050
1100
1150
1200
1250

BiGRU:


real
val_pred
test_pred
27.5
25.0
22.5
20.0
17.5
15.0
12.5
900
950
1000
1050
1100
1150
1200
1250

TCN:


real
val_pred
test_pred
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
900
950
1000
1050
1100
1150
1200
1250

BiTCN:


real
val_pred
test_pred
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
900
950
1000
1050
1100
1150
1200
1250