Ing. Jose Luis Maciel Jacobo

Predicción de excedencias de partículas contaminantes en el ambiente mediante técnicas de inteligencia artificial

2024

Universidad Autónoma de Querétaro
Facultad de Ingeniería

Predicción de excedencias de partículas contaminantes en el ambiente mediante técnicas de inteligencia artificial

Tesis

Que como parte de los requisitos para

obtener el Grado de

Maestro en Ciencias en Inteligencia Artificial

Presenta:

Ing. Jose Luis Maciel Jacobo

Dirigido por:

Dr. Marco Antonio Aceves Fernández

Co-Director:

Dr. Jesús Carlos Pedraza Ortega

Querétaro, Qro. a 28 de enero de 2025

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestro en Ciencias en Inteligencia Artificial

Predicción de excedencias de partículas contaminantes en el ambiente mediante técnicas de inteligencia artificial

Tesis

Que como parte de los requisitos para obtener el Grado de

Maestro en Ciencias en Inteligencia Artificial

Presenta
Ing. Jose Luis Maciel Jacobo

Dirigido por:
Dr. Marco Antonio Aceves Fernández

Co-dirigido por:
Dr. Jesús Carlos Pedraza Ortega

Dr. Marco Antonio Aceves Fernández
Presidente

Dr. Jesús Carlos Pedraza Ortega
Secretario

Dr. Juan Manuel Ramos Arreguín
Vocal

Dr. Sebastián Salazar Colores
Suplente

Dr. Saul Tovar Arriaga
Suplente

Centro Universitario, Querétaro, Qro.
Fecha de aprobación por el Consejo Universitario (mes y año)
México

Dedicatorias

Para toda mi familia.

Agradecimientos

Le agradezco a mis padres, Rossana y Jose Luis, todo su apoyo, dedicación y amor, sin ustedes nada de esto hubiera sido posible, los admiro y los amo, de igual forma agradezo a mis padrinos Carla y Pedro, que me hicieron parte de su hogar una vez más, como siempre lo han hecho, son mis segundos padres.

A mis hermanos, Carla, Jeshua y Sebastián, que siempre me han brindado cobijo, risas y amor, son mis compañeros de vida incondicionales, también agradezco a mi abuela Guadalupe y a mis tíos Guadalupe, Malena, Gabriela y Gustavo por su amor, enseñanzas y apoyo.

Agradezco a todos los amigos que estos últimos 2 años me han brindado empatía, consejos y palabras de aliento, a mis compañeros de la maestría que sin su colaboración y mentoría esto hubiera sido una labor imposible.

Le agradezo al Dr. Marco Antonio Aceves Fernández, por todas las enseñanzas y tiempo dedicado estos últimos 2 años, es un gran profesor, que genuinamente le interesa el bienestar y aprendizaje de los alumnos.

Por último, quisera agradecer a la Universidad Autónoma de Querétaro por el espacio y la oportunidad de ser parte de esta maestría.

Índice

Índice de Tablas

Índice de Figuras

Abreviaturas y Siglas

AJM - Ajusco Medio

AJU - Ajusco

BJU - Benito Juárez

CAM - Camarones

CCA - Centro de Ciencias de la Atmósfera

CDMX - Ciudad de México

CO - Carbon Monoxide (Monóxido de Carbono)

COVs - Volatile Organic Compounds (Compuestos Orgánicos Volátiles)

COY - Coyoacán

CPU - Central Processing Unit (Unidad de Procesamiento Central)

DDR6 - Double Data Rate Type 6 (Doble Tasa de Datos Tipo 6)

DL - Deep Learning (Aprendizaje Profundo)

EPA - Environmental Protection Authority (Autoridad de Protección Ambiental)

FAR - FES Aragón

GAM - Gustavo A. Madero

GB - Gigabytes (Gigabytes)

GHz - Gigahertz (Gigahercios)

GPU - Graphics Processing Unit (Unidad de Procesamiento Gráfico)

GRU - Gated Recurrent Unit (Unidad de Recurrencia Compuerta)

HGM - Hospital General de México

IA - Inteligencia Artificial

INECC - Instituto Nacional de Ecología y Cambio Climático

INN - Investigaciones Nucleares

LSTM - Long Short-Term Memory (Memoria a Corto y Largo Plazo)

MAE - Mean Absolute Error (Error Absoluto Medio)

Mb - Megabits (Megabits)

MER - Merced

MGH - Miguel Hidalgo

MICE - Multiple Imputation by Chained Equations (Imputación Múltiple por Ecuaciones Encadenadas)

ML - Machine Learning (Aprendizaje Automático)

MLP - Multi-Layer Perceptron (Perceptrón Multicapa)

MON - Montecillo

MPA - Milpa Alta

MSE - Mean Squared Error (Error Cuadrático Medio)

MCP - McCulloch-Pitts Neuron (Neurona McCulloch-Pitts)

NEZ - Nezahualcóyotl

NOM - Norma Oficial Mexicana

NOx - Nitrogen Oxides (Óxidos de Nitrógeno)

PED - Pedregal

PM2.5 - Particulate Matter 2.5 (Material particulado de 2.5 Micrómetros)

RAMA - Red Automática de Monitoreo Atmosférico

SAC - Santiago Acahualtepec

SAG - San Agustín

SFE - Santa Fe

SJA - San Juan de Aragón

SOx - Sulfur Oxides (Óxidos de Azufre)

SSA - Secretaría de Salud de México

TLA - Tlalnepantla

UIZ - UAM Iztapalapa

UAX - UAM Xochimilco

XAL - Xalostoc

ZMVM - Zona Metropolitana del Valle de México

Resumen

En los últimos años, la predicción de la calidad del aire se ha convertido en un área de investigación crítica, particularmente en entornos urbanos donde la contaminación del aire representa riesgos significativos para la salud. Este estudio presenta una investigación sobre la predicción de excedencias de partículas PM2.5 en la Ciudad de México utilizando una arquitectura innovadora de red neuronal conocida como Informer, una variante del modelo Transformer. El modelo Informer se compara con una red LSTM, una elección popular para tareas de predicción de datos secuenciales.

A través de una extensa experimentación y rigurosa evaluación, nuestros hallazgos destacan la efectividad del modelo Informer para capturar dependencias temporales y predecir con precisión las excedencias de PM2.5. Al aprovechar su mecanismo de autoatención y capacidades de procesamiento paralelo, el modelo Informer supera consistentemente a las redes LSTM tradicionales. Esta superioridad es particularmente evidente en escenarios caracterizados por patrones temporales complejos y dependencias a largo plazo. A lo largo de varios experimentos, en los que variamos tanto el tipo de predicción (univariada y multivariada) como los intervalos de predicción (24, 48 y 72 horas), el modelo Informer ofreció consistentemente mejores resultados en comparación con el LSTM. Específicamente, logramos mejoras del 29.59 % en MAE y del 22.97 % en MSE.

Esta investigación contribuye al avance de las metodologías de predicción de la calidad del aire, ofreciendo valiosas ideas sobre el potencial de las arquitecturas de redes neuronales de última generación para el monitoreo ambiental y la gestión de la salud pública en áreas urbanas. Los resultados subrayan la importancia de aprovechar técnicas avanzadas de apren-

dizaje profundo para realizar predicciones más precisas y confiables de los parámetros de calidad del aire, facilitando así medidas proactivas para mitigar los efectos adversos de la contaminación del aire en la salud humana.

Abstract

In recent years, air quality forecasting has become a critical area of research, particularly in urban environments where air pollution poses significant health risks. This study presents an investigation into the prediction of PM2.5 particle exceedances in Mexico City using an innovative neural network architecture known as the Informer, a variant of the Transformer model. The Informer model is compared against a Long Short-Term Memory (LSTM) network, a popular choice for sequential data prediction tasks.

Through extensive experimentation and rigorous evaluation, our findings underscore the effectiveness of the Informer model in capturing temporal dependencies and accurately forecasting PM2.5 exceedances. By leveraging its self-attention mechanism and parallel processing capabilities, the Informer consistently outperforms traditional LSTM networks. This superiority is particularly evident in scenarios characterized by complex temporal patterns and long-range dependencies. Across various experiments, where we varied both the type of prediction (univariate and multivariate) and the prediction intervals (24, 48, and 72 hours), the Informer model consistently delivered superior results compared to the LSTM. Specifically, we achieved improvements of 29.59 % in MAE and 22.97 % in MSE.

This research contributes to the advancement of air quality prediction methodologies, offering valuable insights into the potential of state-of-the-art neural network architectures for environmental monitoring and public health management in urban areas. The results underscore the importance of leveraging advanced deep learning techniques for more accurate and reliable predictions of air quality parameters, thereby facilitating proactive measures to mitigate the adverse effects of air pollution on human health.

Capítulo 1

Introducción

La calidad del aire en la Ciudad de México ha sido un tema de creciente preocupación en las últimas décadas. Cada año, los niveles de contaminación del aire, particularmente de partículas PM2.5, han mostrado una tendencia alarmante al alza, afectando la salud de millones de habitantes y contribuyendo a problemas ambientales graves. Las partículas PM2.5, debido a su tamaño diminuto, pueden penetrar profundamente en los pulmones y entrar en el torrente sanguíneo, causando enfermedades respiratorias y cardiovasculares. Este deterioro continuo de la calidad del aire exige métodos innovadores y precisos para la predicción de estos contaminantes, lo cual es esencial para implementar medidas preventivas y de mitigación.

Este trabajo es de vital importancia por la urgente necesidad de mejorar las predicciones de la calidad del aire en la Ciudad de México. Aunque existen diversos modelos y técnicas para la predicción de contaminantes atmosféricos, muchos de estos modelos no han logrado captar con precisión las complejas dependencias temporales y espaciales que caracterizan la dinámica de los contaminantes como las partículas PM2.5.

En este contexto, la investigación se centra en la aplicación del modelo Informer, una variante avanzada del Transformer, conocida por su capacidad para manejar secuencias largas y complejas de datos. Esta arquitectura de inteligencia artificial ha mostrado resultados prometedores en diversas áreas, pero su aplicación en la predicción de parámetros climáticos y medioambientales sigue siendo limitada. Este trabajo busca llenar este vacío, explorando el

potencial del Informer para mejorar las predicciones de PM2.5 en un entorno tan desafiante como el de la Ciudad de México.

Capítulo 2

Antecedentes

2.1. Contaminación del aire

La contaminación del aire se puede definir como condiciones atmosféricas en las que ciertas sustancias están presentes en concentraciones que pueden llevar a efectos no deseados tanto en los seres humanos como en el medio ambiente. Estas sustancias incluyen gases como SOx, NOx, CO y compuestos orgánicos volátiles (COVs), así como material particulado como humo, polvo, gases y aerosoles, junto con otros elementos, incluidos materiales radiactivos. Si bien muchas de estas sustancias existen naturalmente en la atmósfera a concentraciones bajas (niveles de fondo) y generalmente se consideran inofensivas, una sustancia se considera un contaminante del aire solo cuando su concentración supera significativamente el valor de fondo, lo que produce efectos adversos. Es decir, para que una sustancia en particular sea considerada contaminación, debe superar los niveles de fondo y causar efectos adversos (Admassu & Wubeshet, 2011).

La contaminación del aire urbano es una preocupación global con implicaciones significativas. El aumento en la utilización de combustibles, la creciente demanda de electricidad y las actividades mineras intensificadas desde la Revolución Industrial han surgido como los principales contribuyentes a la contaminación atmosférica. Las partículas ultrafinas pueden viajar al torrente sanguíneo y depositarse en órganos como el hígado, el bazo o el cerebro,

con la posibilidad de penetrar a través de mecanismos transinápticos (Falcon-Rodriguez et al., 2016).

2.2. Material particulado

La materia particulada consiste en partículas de diversos orígenes, que varían en tamaño y composición. Estas partículas generalmente se clasifican en tres grupos principales según su tamaño:

- Partículas gruesas: Estas tienen un diámetro que va desde los 10 micrómetros hasta menos de 2.5 micrómetros.
- Partículas finas: Estas tienen un tamaño entre 2.5 micrómetros y 0.1 micrómetros.
- Partículas ultrafinas: Estas son más pequeñas que 0.1 micrómetros.

La mayoría de los sistemas de monitoreo miden las partículas por su concentración de masa, centrándose en aquellas más pequeñas que 2.5 micrómetros (conocidas como PM2.5) y aquellas más pequeñas que 10 micrómetros (conocidas como PM10). PM10 incluye tanto partículas finas como gruesas, siendo que PM2.5 representa generalmente alrededor del 50-70 % de la masa total de PM10. Las partículas ultrafinas también se incluyen en las mediciones de PM2.5 y PM10. Las principales fuentes de PM2.5 son el tráfico vehicular, la generación de energía y la quema industrial y doméstica de petróleo, carbón o madera. Estas partículas finas están compuestas por carbono elemental, metales de transición, compuestos orgánicos complejos, sulfatos y nitratos (Newby et al., 2015).

Como se mencionó antes, las partículas PM2.5 consisten en partículas inhalables que son lo suficientemente pequeñas como para ingresar al sistema respiratorio, como se puede observar en la figura 2.1. Tanto la exposición a corto plazo como a largo plazo pueden resultar en diversos efectos en la salud, incluyendo:

- Exacerbación del asma

- Aumento de la mortalidad por enfermedades cardiovasculares
- Tasas elevadas de mortalidad por enfermedades respiratorias y cáncer de pulmón

La exposición a PM2.5 está asociada con una reducción promedio en la esperanza de vida de la población de la región de aproximadamente 8.6 meses. No hay evidencia que respalde un nivel de exposición seguro o un umbral por debajo del cual no ocurran efectos adversos para la salud (World Health Organization, 2013).

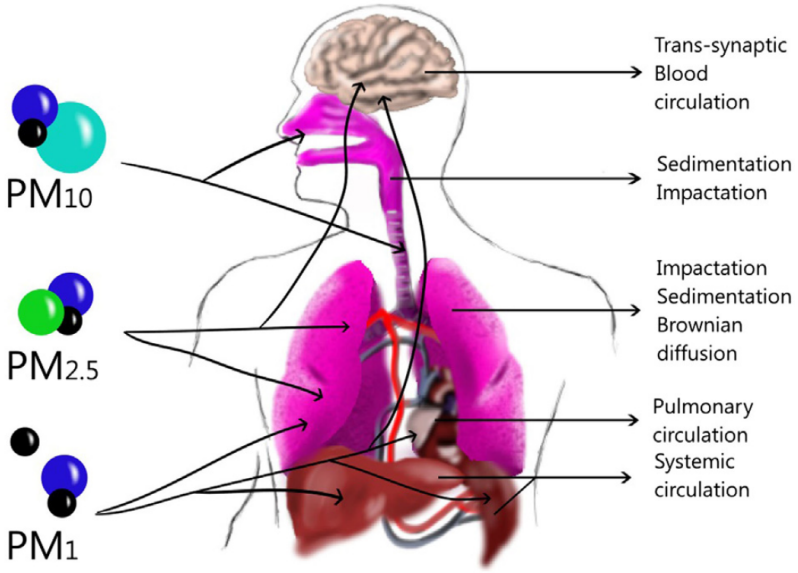

Figura 2.1: Tamaño de las partículas en comparación con el sistema respiratorio, tomado de (Falcon-Rodriguez et al., 2016).

2.3. Inteligencia artificial

La inteligencia artificial o IA, como se le conoce más ampliamente, engloba una amplia gama de áreas, desde aspectos generales como el aprendizaje y la percepción, hasta aplicaciones específicas como la demostración de teoremas matemáticos, el pronóstico de cambios climáticos, el diagnóstico médico y más recientemente la creación de arte. Y esta puede ser descrita como el área de la informática que se enfoca en automatizar acciones inteligentes.

Pero, esta definición tiene limitaciones ya que nos hace falta una definición más precisa y completa de lo que realmente constituye 'ser inteligente'. Aunque para nosotros los humanos es bastante fácil identificar un comportamiento inteligente cuando estamos frente a él, es bastante complicado llegar a una definición concreta de inteligencia que sea eficiente para comprobar la inteligencia de un algoritmo y que además abarque complejidad y profundidad del cerebro humano. Se puede decir con seguridad que la IA tiene relevancia en prácticamente cualquier tarea intelectual y se considera como uno de los campos más versátiles de la ciencia en la actualidad (Khorasani, 2008; Russell, 2010).

Como mencionamos anteriormente, definir 'inteligencia' no es una tarea sencilla, y para complicarla aún más, debemos tener en cuenta que la inteligencia humana no aparece de una forma única, por ejemplo: hay personas que son bastante hábiles para las artes mientras que otras muestran una inclinación más clara por las matemáticas, política o deportes. Esto particularmente se hace aparente cuando hay personas que sobresalen demasiado en el ámbito académico, mientras que otras destacan más en inteligencia emocional. Pese a la complejidad de esta tarea, como humanos siempre hemos intentado establecer un único umbral para medir la inteligencia, tal es el caso del coeficiente intelectual (CI), pero muchas veces dichas pruebas o umbrales no están libres de subjetividad, por ejemplo, en una prueba de CI se evalúa mayoritariamente la memoria a corto plazo, el pensamiento analítico y la habilidad matemática, teniendo en cuenta lo antes mencionado, es bastante obvio que como no tenemos un estándar fiable y sin sesgos para medir la inteligencia de una persona, mucho menos tenemos un estándar fiable para hacerlo con un algoritmo o computadora (Rose, 2020).

Los inicios del reconocimiento de la IA como disciplina se remontan a los años cuarenta, al trabajo pionero realizado por Warren McCulloch y Walter Pitts en el año de 1943. Para lograr esto, se basaron en tres fuentes clave: su conocimiento sobre la fisiología y funcionamiento básico de las neuronas en el cerebro, un análisis formal de la lógica proposicional desarrollada por Russell y Whitehead, y la teoría de la computación propuesta por Alan Turing. Su propuesta describió modelos matemáticos llamados perceptrones que representaban las neuronas del cerebro (esto lo lograron haciendo un análisis exhaustivo de las neuronas biológicas originales) en el que cada neurona puede estar 'activada' o 'desactivada' (de ma-

nera binaria), dependiendo si la estimulación de neuronas vecinas era suficiente. Concebían el estado de una neurona como 'equivalente a una proposición que planteaba su estímulo adecuado'. Demostraron que cualquier función podía ser computada por una red de neuronas conectadas, y que se podían recrear las compuertas lógicas como 'AND', 'OR' y 'NOT' con estructuras bastante sencillas. Además, McCulloch y Pitts sugirieron que estas redes definidas de manera adecuada podrían aprender y, por lo tanto, modificar su comportamiento con el tiempo (Russell, 2010: Warwick, 2013).

2.4. Aprendizaje automático

El aprendizaje automático se refiere a un conjunto de algoritmos para adquirir conocimiento a partir de datos y hacer predicciones, como en tareas de pronósticos, diagnóstico de enfermedades, o la detección de anomalías. Utilizando el aprendizaje automático, obtenemos conocimiento del conjunto de datos, básicamente la computadora distingue patrones de estos, se basa en la 'programación indirecta' que se logra mediante la provisión de los datos a nuestro algoritmo, en contraste, la programación 'tradicional' necesita explícitamente todas las reglas e instrucciones para poder funcionar de manera óptima, por lo antes mencionado, se puede decir que el aprendizaje automático se encuentra en el cruce de la informática, la ingeniería y la estadística, y se aplica en otros campos académicos y profesionales. Un algoritmo de aprendizaje automático es el software utilizado para aprender un modelo de aprendizaje automático a partir de los datos. En cambio, un modelo de aprendizaje automático es el programa resultante de aplicar un algoritmo de machine learning, que mapea las entradas a predicciones, como por ejemplo, Chat GPT, que en este caso es el modelo y el algoritmo es el transformer (Molnar, 2020; Harrington, 2012).

Por eso en la actualidad los datos son tan valiosos, ya que en tiempos pasados, la única manera de desarrollar software era creando algoritmos meticulosamente ajustados. Como mencionamos anteriormente, la programación convencional asume que para cada entrada hay una salida predecible. Sin embargo, el aprendizaje automático aborda problemas para los cuales las relaciones entre las entradas y salidas no son completamente comprendidas. De tal

forma que en aprendizaje automático adquirimos conocimiento a partir de experiencias previas, y con 'experiencias previas' nos referimos a los datos que ingresamos al modelo. Estos datos pueden incluir bases de datos que han sido etiquetadas por seres humanos (por ejemplo, imágenes con la etiqueta apropiada) o bases de datos obtenidas a través de la interacción con el entorno (por ejemplo, los datos obtenidos de las condiciones climáticas). Los modelos de aprendizaje automático mejoran su desempeño a medida que aumentamos tanto la calidad y cantidad de los datos. La expectativa en general de esta subrama de la informática, es que al proporcionar suficientes datos a este sistema, aprenderá patrones y será capaz de generar resultados inteligentes para nuevas entradas (Shukla & Fricklas, 2018; Mohri et al., 2018).

2.5. Aprendizaje profundo

Como se puede observar en la figura 2.2, el aprendizaje profundo está dentro del ámbito del aprendizaje automático, el aprendizaje profundo se emplea para abordar tareas prácticas en diversos campos, emplea grandes conjuntos de datos para modelar funciones complejas, cuyas entradas y salidas se encuentran considerablemente distantes, como la visión por computadora (imágenes), el procesamiento del lenguaje natural (texto) y el reconocimiento automático del habla (audio). Por ejemplo, puede asociar una imagen de entrada con una línea de texto que describe la misma. Incluso de manera más simple, puede establecer una conexión entre una imagen de un gato y una etiqueta que señala 'Sí, hay un gato en la imagen' (normalmente señalado con un número, por ejemplo, 1). En resumen, el aprendizaje profundo representa un conjunto de métodos dentro del repertorio del aprendizaje automático, utilizando principalmente redes neuronales artificiales, que se inspiran de manera general en la estructura del cerebro humano, esto nos posibilita desarrollar programas con funcionalidades que, hasta hace poco, eran exclusivas de los seres humanos (Trask, 2019; Stevens et al., 2020).

Los significativos avances en la tecnología de la computación han permitido que los sistemas actuales de inteligencia artificial (IA) se doten de billones de componentes básicos. Después del correcto entrenamiento, estos componentes básicos, permiten ejecutar tareas an-

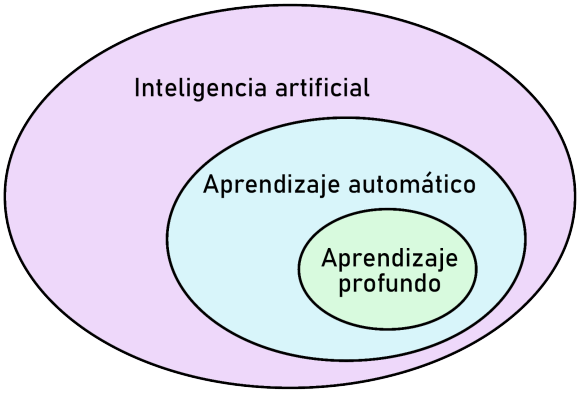

Figura 2.2: Diagrama de Venn de inteligencia artificial, aprendizaje automático y aprendizaje profundo, adaptado de (Stevens et al., 2020).

teriormente vistas como extraordinariamente complejas de programar, a tal grado que como mencionamos anteriormente se pensaba que solo seres con inteligencia natural, como los humanos, podían realizarlas. Un factor clave en muchos de estos avances en IA es el aprendizaje profundo, más específicamente las redes neuronales artificiales: las cuales permiten aprender diferentes formas de representar los datos a través de múltiples capas, estas capas facilitan la extracción automática de características relevantes de los datos. De manera simplificada, el enfoque del aprendizaje profundo busca desarrollar algoritmos de inteligencia artificial estructurando los datos en una jerarquía de conceptos organizados en capas, donde cada nivel se construye sobre la base de capas más simples. La estructura en capas es un elemento fundamental en los algoritmos de aprendizaje profundo, si bien las redes neuronales artificiales están inspiradas en cierta medida en las redes neuronales biológicas del cerebro humano, en realidad pueden considerarse mejor como un método sofisticado para definir una amplia gama de funciones flexibles, formadas por numerosos componentes computacionales básicos conocidos como neuronas. El aprendizaje profundo destaca por su alto rendimiento y capacidad de escalar eficientemente con grandes volúmenes de datos, superando a los algoritmos de aprendizaje automático tradicionales, esto quiere decir que mientras más datos tengamos a nuestro alcance mejor funcionará nuestro algoritmo de aprendizaje profundo en comparación con el aprendizaje automático, para ilustrar esto más claramente, se muestra una gráfica en

la Figura 2.3, con la cantidad de datos en el eje horizontal y el desempeño en el eje vertical (Roberts et al., 2022; Sarkar et al., 2018).

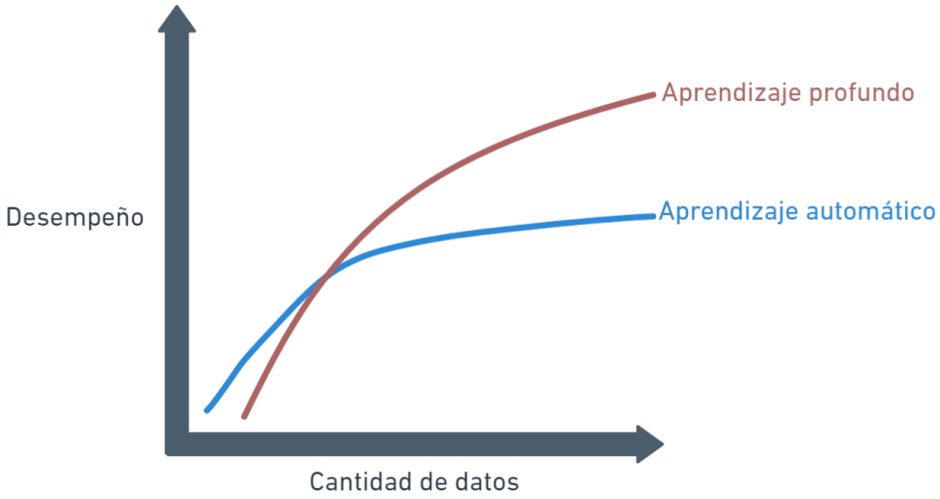

Figura 2.3: Comparación de desempeño con respecto a cantidad de datos entre aprendizaje automático y aprendizaje profundo, adaptado de (Sarkar et al., 2018).

2.6. Redes neuronales

Como mencionamos anteriormente, en 1943 Warren McCulloch y Walter Pitts introdujeron el concepto pionero de una versión simplista de la célula cerebral, conocida como la neurona McCulloch-Pitts (MCP) (McCulloch & Pitts, 1943). Este concepto plantea que las neuronas biológicas, pueden ser representadas de manera simplificada. Estas neuronas biológicas son la pieza más importante del cerebro humano, se estima que en una sección del cerebro comparable a un grano de arroz caben más de 10,000 de estas células y como cada neurona tiene alrededor de 6,000 conexiones con otras neuronas, se podría decir que en esta diminuta parte del cerebro hay más 60 millones de conexiones, las cuales forman una red que nos permite percibir y entender nuestro entorno. Estas neuronas están diseñadas para aceptar datos de otras células similares, procesarlos y transmitir los resultados procesados a otras células. Según McCulloch y Pitts, estas neuronas funcionan como compuertas lógicas binarias. Las señales ingresan a través de las dendritas, se suman en el cuerpo celular y, en

caso de que la suma supere un umbral específico, se emite una señal de salida a través del axón, como se muestra en la figura 2.4. Como sabemos el cerebro humano es muy bueno para procesar información, está caracterizado por su complejidad, capacidad de procesamiento no lineal y en paralelo, es capaz de organizar las neuronas, para ejecutar diversas operaciones, como el reconocimiento de patrones y el control de movimientos, que supera fácilmente a cualquier computadora de la actualidad. En este órgano extraordinario, las neuronas biológicas están dispuestas en distintas capas, específicamente, la corteza cerebral, se compone de seis capas. En este sistema, la información se transmite progresivamente de una capa a otra, transformando los datos sensoriales en entendimiento conceptual. Tomando como referencia el concepto de estos 2 científicos y las células biológicas, se ha desarrollado el concepto de las redes neuronales artificiales. Estas redes se forman al interconectar neuronas artificiales entre sí, así como con los datos de entrada y los nodos de salida, que representan las respuestas de la red a problemas específicos de aprendizaje. En la Figura 2.5 se presenta un modelo básico de cómo se estructura una red neuronal artificial (Raschka & Mirjalili, 2019; Buduma et al., 2022; Haykin, 2009).

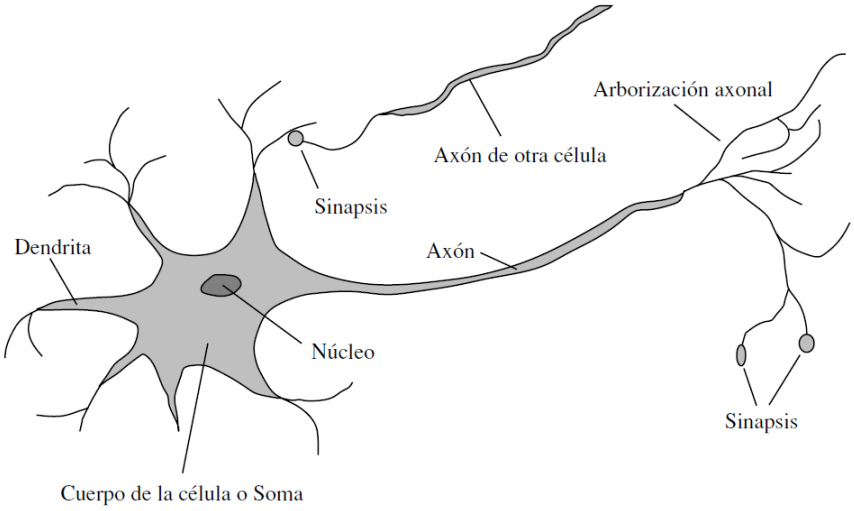

Figura 2.4: Partes de una neurona biológica (Russell, 2010).

En términos prácticos, cada algoritmo de aprendizaje profundo se basa en el uso de redes neuronales, con algunos conceptos básicas en común, como estar formadas de neuronas que

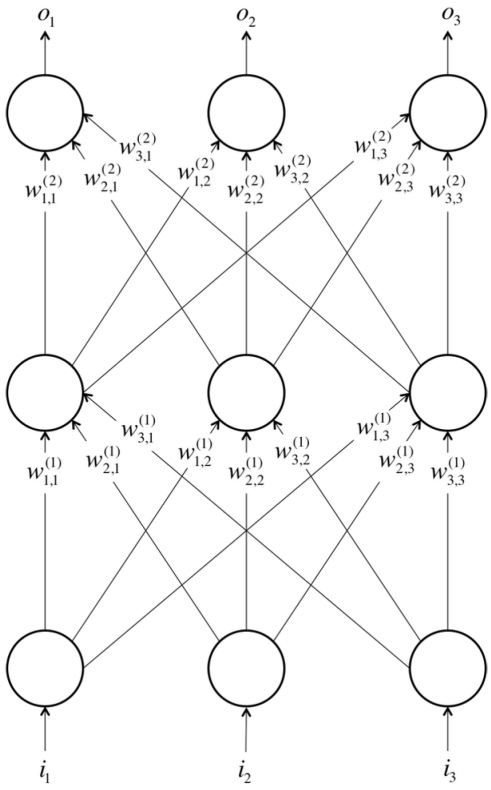

Figura 2.5: Ejemplo simple de una red neuronal de 3 capas (Buduma et al., 2022).

se conectan entre sí, distribuidas en capas, pero con diferentes formas entrenamiento, diversas configuraciones de datos de entrada (incluyendo vectores y matrices de distintas dimensiones) y cómo están acomodadas las neuronas dentro de la red, más adelante veremos distintas arquitecturas de estas redes, como lo son las LSTM (Vasilev et al., 2019).

2.7. Métricas de evaluación

Para medir el desempeño de un sistema de inteligencia artificial y verificar qué tan cerca se está del objetivo propuesto, es necesario utilizar una función que evalúe el resultado. Habitualmente, se emplean distintas funciones de evaluación para manejar problemas de clasificación binaria, clasificación multi etiqueta, regresión o clustering, a veces, incluso se desarrollan métricas específicas para resolver problemas empresariales particulares. Para

problemas de clasificación, las métricas más frecuentemente utilizadas incluyen:

- Exactitud (Accuracy)
- Precisión (P)
- Recuperación (R) o Recall
- Puntaje F1 (F1)

En tareas que implican la predicción de números reales o regresión, muchas métricas de error se derivan del álgebra euclidiana:

- Error absoluto medio (MAE): Representa la media de la norma L1 del vector de diferencias entre los valores predichos y los valores reales.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.1}$$

- MAE: Error Absoluto Medio.
- n: Número total de observaciones.
- y_i: Valor real de la i-ésima observación.
- $\hat{y}_i$: Valor predicho de la i-ésima observación.
- $|y_i - \hat{y}_i|$: Error absoluto de la i-ésima observación.
- Error cuadrático medio (MSE): Representa la media de la norma L2 del vector de diferencias entre los valores predichos y los valores reales

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.2}$$

- MSE: Error Cuadrático Medio.

- n: Número total de observaciones.
- y_i: Valor real de la i-ésima observación.
- $\hat{y}_i$: Valor predicho de la i-ésima observación.
- $(y_i - \hat{y}_i)^2$: Error cuadrático de la i-ésima observación (Boschetti & Massaron, 2015; Thakur, 2020).

2.8. Sobreentrenamiento

El problema del sobreentrenamiento se refiere al hecho de que ajustar un modelo a un conjunto de datos de entrenamiento particular no garantiza que proporcionará un buen rendimiento de predicción en datos de prueba no vistos, incluso si el modelo predice perfectamente los objetivos en los datos de entrenamiento. En otras palabras, siempre existe una brecha entre el rendimiento en los datos de entrenamiento y en los datos de prueba, que es particularmente grande cuando los modelos son complejos y el conjunto de datos es pequeño. Aumentar el número de instancias de entrenamiento mejora la capacidad de generalización del modelo, mientras que aumentar la complejidad del modelo reduce su capacidad de generalización. Al desarrollar un modelo muy complejo, es bastante sencillo ajustar perfectamente nuestro conjunto de datos de entrenamiento, ya que le otorgamos al modelo suficientes grados de libertad para adaptarse a las observaciones en dicho conjunto. Sin embargo, al evaluar un modelo tan complejo con nuevos datos, su rendimiento suele ser deficiente. En otras palabras, el modelo no generaliza adecuadamente. Este fenómeno, conocido como sobreentrenamiento, es uno de los mayores retos que enfrenta un ingeniero de aprendizaje automático. Este problema es aún más grave en el aprendizaje profundo, donde las redes neuronales cuentan con un gran número de capas y numerosos neuronas. La cantidad de conexiones en estos modelos es astronómica, alcanzando millones. Como resultado, el sobreentrenamiento es algo común. Al mismo tiempo, cuando se dispone de una gran cantidad de datos de entrenamiento, es poco probable que un modelo excesivamente simple capture relaciones complejas entre las características y el objetivo. Una buena regla general es que el número total de puntos de datos

de entrenamiento debe ser al menos 2 o 3 veces mayor que el número de parámetros en la red neuronal, aunque el número preciso de instancias de datos depende del modelo específico en cuestión. En general, se dice que los modelos con un mayor número de parámetros tienen una alta capacidad y requieren una mayor cantidad de datos para adquirir capacidad de generalización en datos de prueba no vistos. Como mencionamos anteriormente la mayor disponibilidad de datos ha revelado las ventajas de las redes neuronales sobre el aprendizaje automático tradicional. En general, las redes neuronales requieren un diseño cuidadoso para minimizar los efectos perjudiciales del sobreentrenamiento, incluso cuando se dispone de una gran cantidad de datos (Aggarwal, 2018; Buduma et al., 2022).

Una de las formas más comunes para evitar el sobreentrenamiento es dividir nuestros datos , se fragmentan en tres segmentos: entrenamiento, validación y prueba. Durante el entrenamiento, el conjunto de entrenamiento se emplea para calcular gradientes y determinar las actualizaciones de pesos. El conjunto de validación se utiliza para interrumpir el entrenamiento antes de que se produzca el sobreajuste. El conjunto de prueba se emplea para predecir el rendimiento futuro de la red, siendo el indicador de calidad de la misma. Si, después del entrenamiento de la red, el rendimiento en el conjunto de prueba no es satisfactorio, generalmente hay cuatro posibles escenarios:

- La red ha alcanzado un mínimo local.
- La red no cuenta con suficientes neuronas para adaptarse a los datos.
- La red está experimentando sobreajuste.
- La red está extrapolando (Demuth et al., 2014; Thakur, 2020).

El problema del mínimo local puede arreglarse mediante el reentrenamiento de la red con conjuntos aleatorios de pesos iniciales. La red con el mínimo error de entrenamiento suele representar un mínimo global. Los otros tres problemas se pueden identificar analizando los errores en los conjuntos de entrenamiento, validación y prueba. Por ejemplo, si el error de validación es considerablemente mayor que el error de entrenamiento, es probable que se haya

producido sobreajuste, incluso con el uso de la detención temprana. En tal caso, se puede aplicar un algoritmo de entrenamiento más lento para volver a entrenar la red. Si los errores en los conjuntos de validación, entrenamiento y prueba son similares en magnitud pero demasiado grandes, es probable que la red no sea lo suficientemente potente para ajustarse a los datos. En esta situación, se debería aumentar el número de neuronas en la capa oculta y proceder al reentrenamiento de la red. Cuando los errores en los conjuntos de validación y entrenamiento son comparables, pero los errores en el conjunto de prueba son considerablemente mayores, es posible que la red esté extrapolando. Esto indica que los datos de prueba están fuera del rango de los datos de entrenamiento y validación. En este caso, se requiere la obtención de más datos, pudiendo fusionar los datos de prueba con los datos de entrenamiento/validación y recopilar nuevos datos de prueba. Deberá continuar con este proceso hasta que los resultados en los tres conjuntos de datos sean similares. Si los errores en los conjuntos de entrenamiento, validación y prueba son semejantes y los errores son suficientemente pequeños, la red multicapa puede implementarse. No obstante, se debe tener precaución con la posibilidad de extrapolación, especialmente si las entradas de la red multicapa están fuera del rango de los datos con los que fue entrenada. Es difícil garantizar que los datos de entrenamiento cubrirán todas las aplicaciones futuras de la red neuronal. Cada vez que entrenamos una red neuronal, es crucial supervisar la pérdida tanto en el conjunto de entrenamiento como en el conjunto de prueba. Si el modelo es muy grande en comparación con el conjunto de datos (es decir, con muy pocas muestras), veremos que la pérdida en ambos conjuntos disminuye mientras continuamos entrenando. No obstante, llegará un punto en el que la pérdida en el conjunto de prueba alcanzará su mínimo y luego comenzará a aumentar, incluso si la pérdida en el conjunto de entrenamiento sigue disminuyendo, como se muestra en la figura 2.6. Es importante detener el entrenamiento en el momento en que la pérdida de validación alcanza su valor más bajo (Demuth et al., 2014; Thakur, 2020).

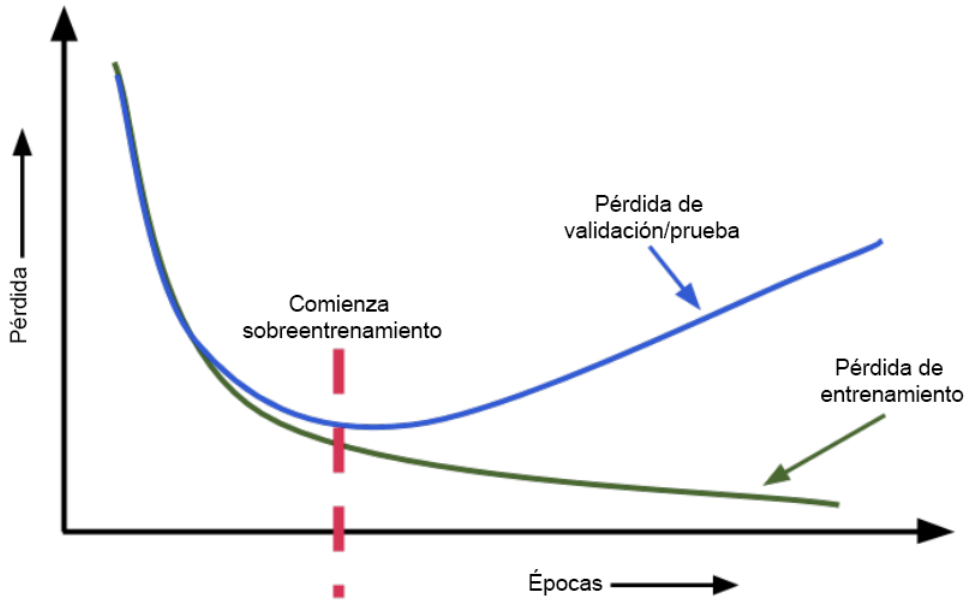

Figura 2.6: Ejemplificación de sobreentrenamiento, adaptado de (Thakur, 2020).

2.9. Hiperparámetros

Los parámetros que son ajustables pero no se actualizan durante el entrenamiento, se denominan hiperparámetros, y son diferentes de los parámetros fundamentales que representan los pesos de las conexiones en la red neuronal. la mayoría de los algoritmos de aprendizaje automático ofrecen múltiples hiperparámetros para ajustar, como la tasa de aprendizaje, la longitud de secuencia, las épocas, entre otros. Ajustar los hiperparámetros es el proceso de seleccionarlos. Es importante no ajustar los hiperparámetros utilizando los mismos datos que se usan para el entrenamiento del modelo. En su lugar, se reserva una parte de los datos como conjunto de validación, y se evalúa el rendimiento del modelo en este conjunto con diferentes opciones de hiperparámetros. Este enfoque evita el sobreajuste al conjunto de entrenamiento y proporciona una evaluación más realista del rendimiento del modelo. La técnica más común para ajustar los hiperparámetros es la búsqueda en cuadrícula, donde se selecciona un conjunto de valores para cada hiperparámetro. En la búsqueda en cuadrícula, se prueban todas las combinaciones posibles de valores de hiperparámetros para determinar la mejor elección. Sin embargo, este enfoque puede ser computacionalmente costoso, especialmente cuando el

número de hiperparámetros es grande. Para mitigar esto, se suelen utilizar cuadrículas más gruesas en una primera etapa y luego refinar la búsqueda en regiones específicas de interés. Es importante tener cuidado cuando el valor óptimo de un hiperparámetro está en el límite de un rango de búsqueda, ya que puede ser necesario explorar más allá de ese rango para encontrar valores aún mejores (Goodfellow et al., 2016; Zhang et al., 2023; Aggarwal, 2018)

2.10. Series de tiempo

Las series de tiempo están presentes en diversos ámbitos, abarcando meteorología, finanzas, econometría y marketing. A través del registro y análisis de datos, podemos sumergirnos en series de tiempo para analizar procesos industriales o monitorear métricas comerciales como las ventas o el compromiso. Además, con la abundancia de datos, los científicos de datos pueden aprovechar sus conocimientos en la aplicación de técnicas para la predicción de series temporales. El primer paso para comprender y ejecutar la predicción de este tipo de datos implica entender su naturaleza. Básicamente, una serie temporal son datos medidos a intervalos de tiempo regulares, conocidos como el intervalo de muestreo. En pocas palabras, los datos podrían registrarse por hora, mensualmente o anualmente. Ejemplos de series temporales incluyen el valor de cierre de una acción específica, el consumo de electricidad de un hogar o la temperatura exterior. Se puede representar una serie temporal de longitud n como:

$$\{x_t : t = 1, \ldots, n\} = \{x_1, x_2, \ldots, x_n\} \tag{2.3}$$

Donde:

- n es el número de valores muestreados en momentos discretos.
- x_t son los valores medidos (Peixeiro, 2022; Cowpertwait & Metcalfe, 2009).

Una característica clave de los datos de series temporales es que las observaciones vecinas a menudo dependen entre sí. Descifrar cómo están relacionadas estas observaciones en una serie temporal es realmente importante para usos prácticos. El análisis de series temporales

se centra en métodos para analizar esta interdependencia. Hacerlo bien significa que necesitamos desarrollar y utilizar modelos estocásticos y dinámicos para datos de series temporales en áreas donde realmente importan (Box et al., 2015).

Antes de dividir nuestras series temporales en subconjuntos de entrenamiento, validación y prueba, debemos realizar la segmentación de datos o ventaneo de datos, que es un procedimiento donde establecemos una serie de puntos de datos dentro de nuestra serie temporal, designando ciertos puntos como entradas (también llamadas X) y otros como etiquetas (también llamadas Y). Esto permite que el modelo de aprendizaje profundo se entrene en las entradas, genere predicciones, las compare con las etiquetas y repita este ciclo hasta que no sea posible lograr una mejora adicional en la precisión de la predicción. También nos permite mezclar nuestros subconjuntos sin perder el orden original, se puede observar cómo funciona el ventaneo de datos en la figura 2.7 (Peixeiro, 2022).

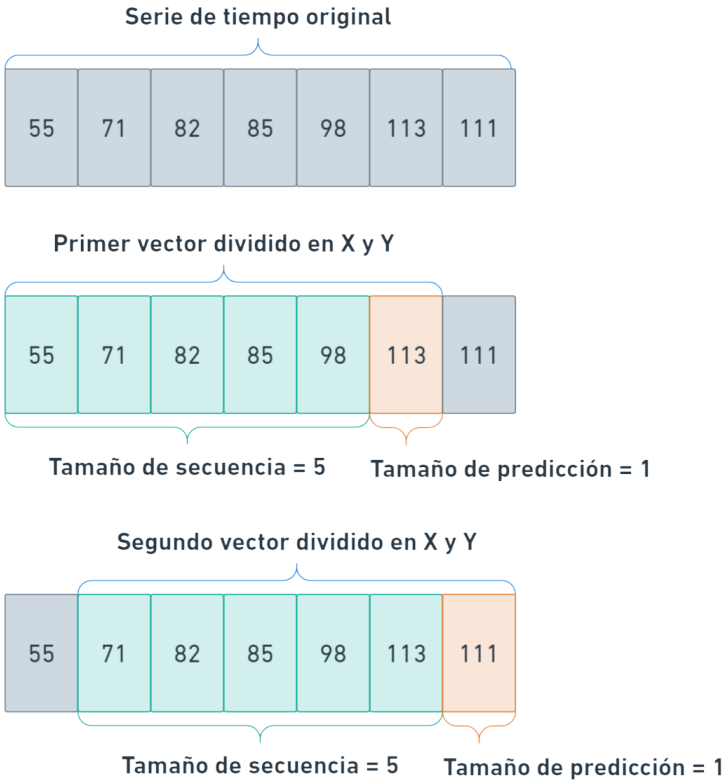

Figura 2.7: Ejemplificación gráfica de segmentación de datos temporales (Autoría propia).

2.11. Transformer

Desde 2017, una nueva arquitectura llamada ‘Transformer’ ha revolucionado la inteligencia artificial, este modelo ha estado superando a las redes neuronales recurrentes en la mayoría de las tareas de procesamiento del lenguaje natural, como la traducción de textos. Este modelo fue presentado en el artículo innovador ’Attention is all you need’ por Vaswani et al. La esencia del artículo se resume en el título: resultó que un mecanismo simple llamado ’atención neuronal’ podría ser utilizado para construir modelos de secuencia poderosos sin necesidad de usar las arquitecturas principales en ese momento, como capas recurrentes o de convolución. El objetivo del transformer es reemplazar completamente los componentes recurrentes y convolucionales con atención, esta arquitectura se ha convertido rápidamente en una de las ideas más influyentes en el aprendizaje profundo, se muestra el diagrama original de la arquitectura de este modelo en la figura 2.8 (Chollet, 2021; Azunre, 2021).

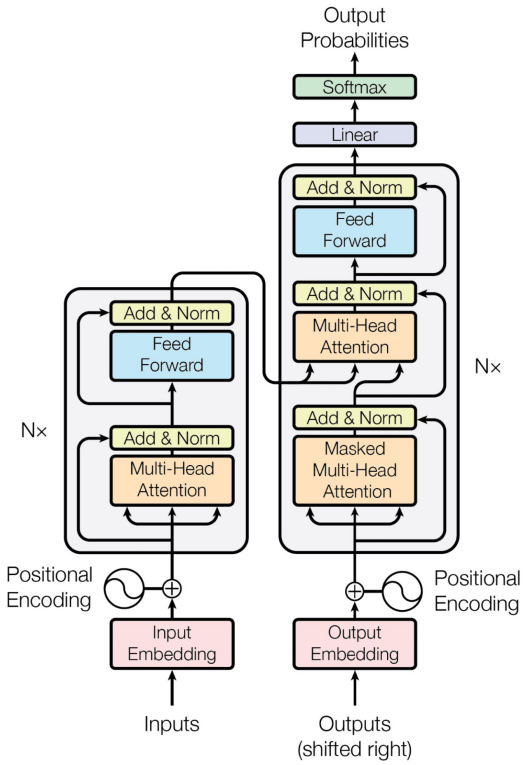

Figura 2.8: Arquitectura original del transformer (Vaswani et al., 2017).

Aunque inicialmente concebidos para el aprendizaje secuencia a secuencia en datos de texto, los transformers se han vuelto omnipresentes en una amplia variedad de aplicaciones modernas de aprendizaje profundo, abarcando áreas como el procesamiento del lenguaje natural, la visión por computadora y la predicción de series de tiempo. Esta arquitectura fue desarrollada en Google, se basó en el hecho de que los mejores modelos de traducción hasta ese momento utilizaban componentes convolucionales y recurrentes junto con un mecanismo de atención. Estos modelos emplean una arquitectura codificador-decodificador, donde el codificador convierte el texto de entrada en una representación vectorial numérica intermedia, llamada vector de contexto, y un decodificador que convierte este vector en texto de salida. El mecanismo de atención habilita a que el decodificador examine cualquier parte del historial de salidas del codificador y utilice esa información para producir la salida también. Sin embargo, los modelos LSTM y GRU siguen siendo bastante restrictivos, ya que solo pueden ver una salida en la secuencia en un momento dado y necesitan depender de un vector de estado limitado (es decir, memoria) para recordar lo que han visto. La atención permite un mejor rendimiento al modelar las dependencias entre partes de la salida y la entrada. Anteriormente, la atención se asociaba principalmente con componentes recurrentes, pero el transformer reemplaza toda la funcionalidad con atención, específicamente con una variante llamada autoatención. La autoatención se aplica a la misma secuencia tanto como entrada como salida, lo que le permite aprender las dependencias entre cada parte de la secuencia y todas las demás partes de la misma secuencia. Los embeddings de las secuencias de entrada (fuente) y salida (objetivo) se combinan con codificación posicional antes de ser introducidos en el codificador y el decodificador, que apilan módulos basados en la auto-atención. En contraste con los modelos LSTM, que tienen que mirar un paso de tiempo a la vez, los modelos Transformer pueden ver toda la secuencia simultáneamente. Esto les permite comprender el lenguaje de manera más efectiva que otros modelos. Además, los modelos Transformer disfrutan de una alta paralelización debido a la minimización de cálculos longitudinales (es decir, temporales) que requieren procesamiento secuencial de texto (Zhang et al., 2023; Azunre, 2021; Ganegedara, 2022).

En la figura 2.9, se observa que los codificadores idénticos están apilados en el lado

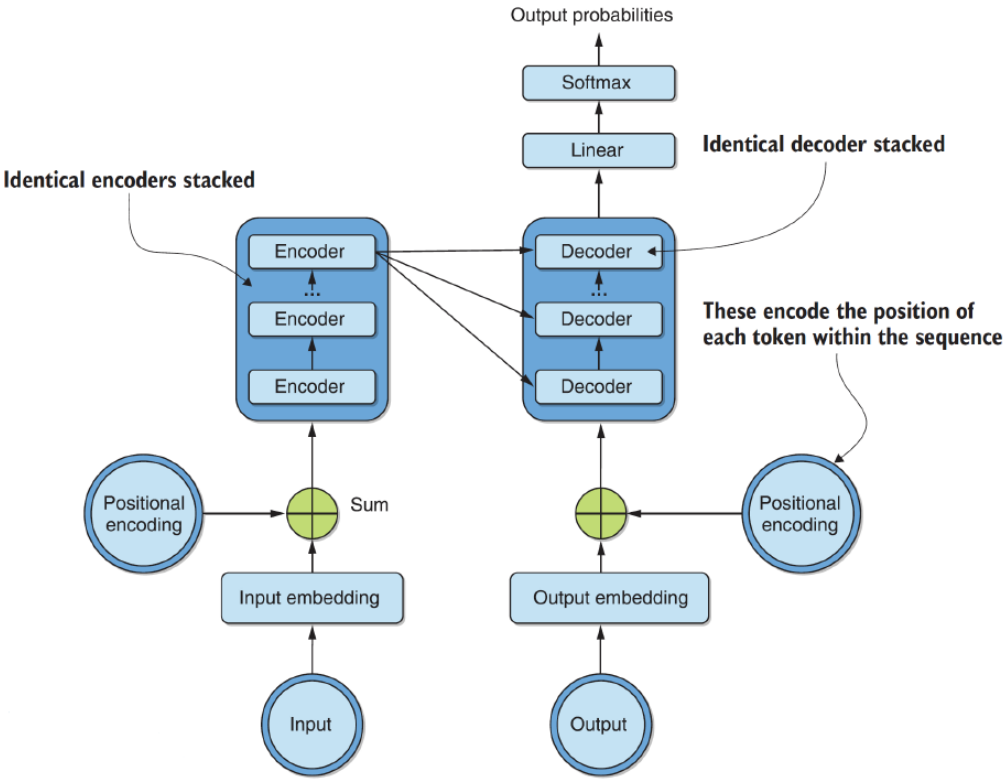

Figura 2.9: Arquitectura del transformer, mostrando las apilaciones de codificadores y decodificadores (Azunre, 2021).

de codificación de la arquitectura, y en el lado de decodificación, se apilan decodificadores idénticos. Además, tanto la entrada como la salida se convierten en vectores utilizando un algoritmo de incrustación. Se utilizan codificaciones posicionales para mantener la conciencia secuencial, lo que permite descartar los componentes recurrentes. En la figura 2.10 se puede observar como el codificador del transformer se compone de múltiples capas idénticas, cada una con dos subcapas. La primera subcapa es una agrupación de autoatención multi-cabeza que genera una representación latente para cada token de entrada en la secuencia, examinando toda la secuencia de entrada y seleccionando otros tokens que enriquecen la semántica de la salida oculta generada para ese token. y la segunda es una red neuronal feed-forward basada en la posición que genera una representación oculta elemento a elemento de la representación atendida. Específicamente, en la auto-atención del codificador, las consultas, claves y valores provienen de las salidas de la capa de codificador anterior y cada decodificador se descompo-

ne de manera similar, pero con la adición de una capa de atención codificador-decodificador. En la autoatención del decodificador, los tokens futuros están 'enmascarados' (lo que quiere decir, que se examinan todos los tokens a la izquierda de cada token de entrada y enmascara las palabras a la derecha para evitar que el modelo vea palabras en el futuro) asegurando que la predicción dependa únicamente de los tokens de salida generados hasta ese momento. La atención codificador-decodificador aprende dependencias similares entre las entradas al codificador y al decodificador (Zhang et al., 2023; Azunre, 2021; Ganegedara, 2022).

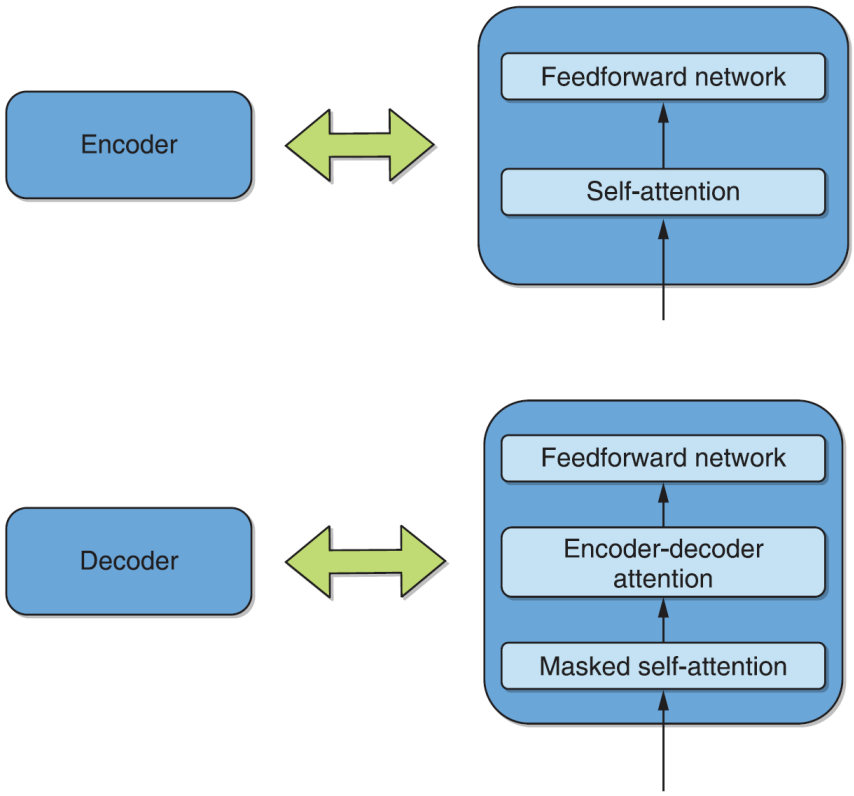

Figura 2.10: Arquitectura del transformer, mostrando la estructura interna de los codificadores y decodificadores (Azunre, 2021).

2.12. Informer

La arquitectura Informer, introducida por Zhou et al. (2020), es un modelo de aprendizaje profundo basado en el Transformer original desarrollado por Vaswani et al. (2017), donde las principales modificaciones respecto al original incluyen el mecanismo de autoatención ProbSparse. Este mecanismo tiene como objetivo mejorar la eficiencia computacional y reducir el consumo de memoria en comparación con la arquitectura estándar del Transformer. Además, Zhou et al. (2020) también integraron un proceso de destilación de autoatención que reduce significativamente la complejidad espacial total del modelo (Ahmed et al., 2023).

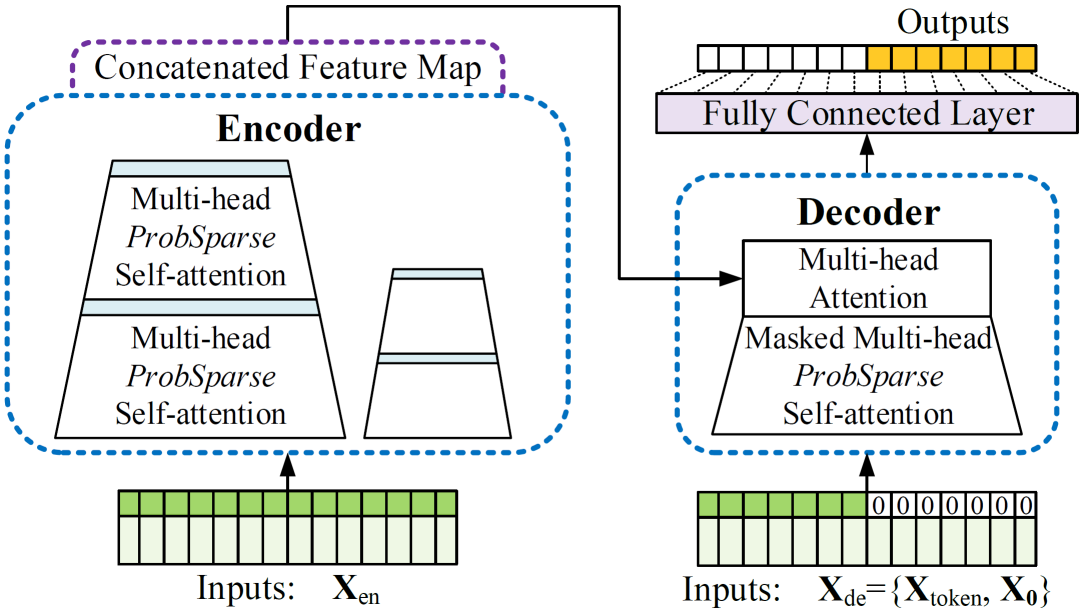

Figura 2.11: Arquitectura del Informer (Zhou et al., 2020).

El modelo del algoritmo Informer mejora la arquitectura del Transformer mediante el uso de una estructura multicapa similar compuesta por bloques Informer. Estos módulos cuentan con un mecanismo de autoatención multi-cabezal ProbSparse único dentro de una configuración codificador-decodificador. La figura 2.11 ilustra el diseño fundamental del modelo Informer. En el lado izquierdo de la figura, el codificador procesa un gran número de entradas

de secuencia extendida utilizando la autoatención ProbSparse especializada, reemplazando el método estándar de autoatención. La robustez del modelo se ve reforzada por las múltiples capas de estos bloques. En el lado derecho de la figura, el decodificador gestiona la entrada de secuencia larga, anula el elemento objetivo, calcula la mezcla ponderada de atención del mapa de características y luego genera directamente el elemento de salida. Este modelo muestra una mejora en las capacidades predictivas para problemas de Pronóstico de Series Temporales de Secuencia Larga (LSTF), destacando la capacidad de la familia de modelos transformadores para capturar las intrincadas dependencias a largo plazo entre la entrada y la salida en datos de series temporales extensas (Zhu et al., 2023).

Capítulo 3

Hipótesis

Es posible predecir con éxito las excedencias de partículas PM2.5 con varias horas de anticipación utilizando mecanismos de atención en modelos de aprendizaje automático.

Capítulo 4

Objetivos

4.1. Objetivo General

Generar un modelo de aprendizaje profundo que sea capaz de predecir satisfactoriamente las excedencias de contaminantes PM2.5 en el medio ambiente con horas de anticipación.

4.2. Objetivos Específicos

- Obtener y procesar los datos de manera adecuada para el entrenamiento de nuestro modelo
- Implementar un modelo de inteligencia artificial con módulos de atención capaz de predecir las excedencias de partículas
- Predecir con dicho modelo las excedencias de partículas con horas de anticipación
- Comparar resultados con el estado del arte

Capítulo 5

Metodología

5.1. Software y hardware

En esta investigación se utilizó una estación de trabajo con las siguientes características:

- **CPU**: AMD Ryzen 5 2600
  - 3.4 GHz de frecuencia
  - 6 núcleos de procesamiento
  - 12 hilos de procesamiento
- **GPU**: NVIDIA GeForce GTX 1660 SUPER
  - 1530 MHz de frecuencia
  - 6 GB de memoria gráfica GDDR6
  - 1408 núcleos CUDA
- **RAM**: DDR4
  - 1331 MHz de frecuencia
  - 64 GB de memoria
- **Sistema Operativo**: Windows 10 Pro

Se utilizaron distintas herramientas de software para la ejecución adecuada de cada una de las etapas de la investigación, en todos los siguientes programas siempre se utilizó Python como lenguaje de programación:

- IntelliJ IDEA
- PyCharm
- Visual Studio Code
- Google Colab
- Jupyter Notebook

5.2. Metodología general

Para asegurar la consistencia y la comparabilidad entre los experimentos realizados con el modelo LSTM y el modelo Informer, se adoptó una metodología estandarizada que guió el desarrollo de la experimentación en ambos modelos de manera idéntica, para así eliminar cualquier variación y poder comparar directamente las 2 arquitecturas. La metodología se detalla en la figura 5.1 ofreciendo una representación visual del proceso seguido.

De manera más específica, la metodología consta de 6 etapas principales:

1. Adquisición de datos
2. Preparación de datos
3. Experimentación inicial
4. Comparación de resultados
5. Experimentación final
6. Análisis de resultados

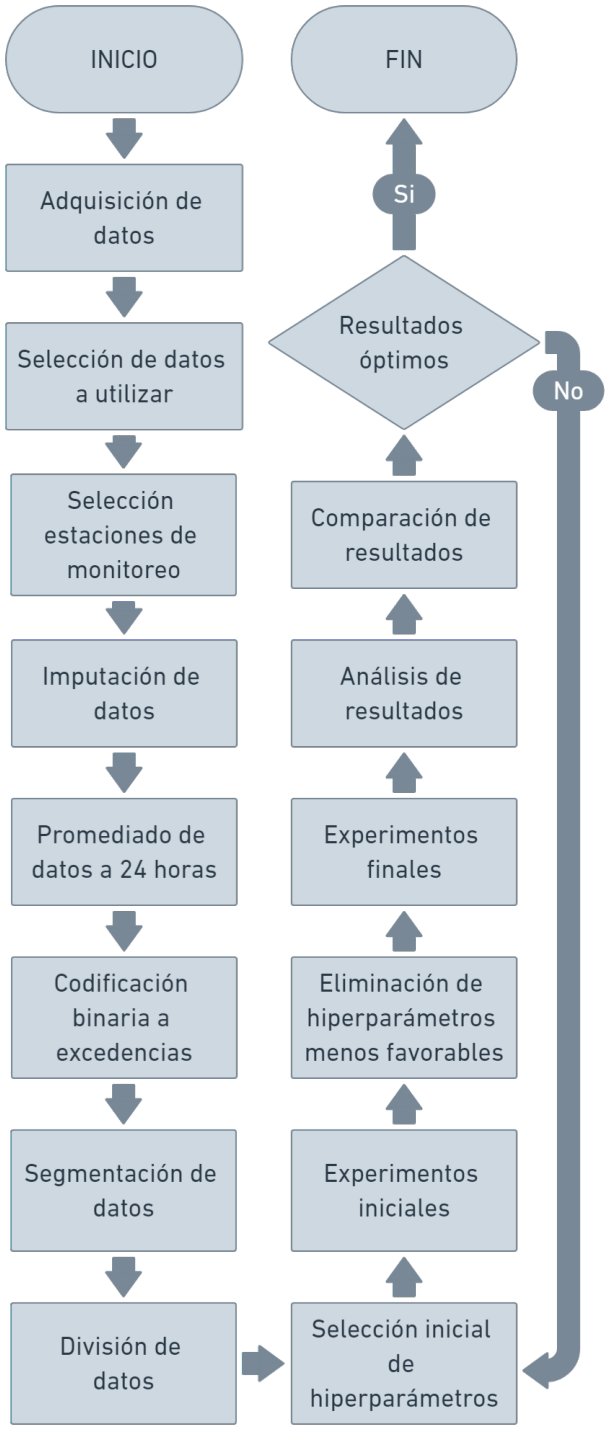

Figura 5.1: Diagrama de metodología general (Autoría propia).

5.3. Adquisición de datos

Naturalmente, el primer paso fue adquirir los datos, la base de datos que se utilizó en esta investigación es de la Red Automática de Monitoreo Atmosférico (RAMA) de la Ciudad de México, que se encuentra disponible de manera gratuita en la página oficial de RAMA: http://www.aire.cdmx.gob.mx/

Dicha base de datos se ha obtenido monitoreando el aire en 24 puntos estratégicos de la CDMX desde el año 2003 (para PM2.5), como se puede observar en la tabla 5.1 la base de datos está estructurada de la siguiente manera:

- Primera columna: Fecha de monitoreo (día/mes/año).
- Segunda columna: Hora de monitoreo (1 a 24 horas).
- A partir de la tercera columna: Concentración del contaminante por estación de monitoreo (representado en microgramos/metro cúbico, $\mu g/m^3$). Las estaciones se identifican con la clave de la estación, tal como se muestra en la tabla 5.2.

Tabla 5.1: Muestra de la base de datos

FECHA	HORA	BJU	UAX	MER	TLA
01-01-22	1	55	-99	59	48
01-01-22	2	71	-99	67	58
01-01-22	3	82	-99	84	69
01-01-22	4	85	-99	82	53
01-01-22	5	98	-99	101	41
01-01-22	6	113	-99	103	47
01-01-22	7	111	-99	114	36
01-01-22	8	110	-99	132	33
01-01-22	9	125	-99	167	43
01-01-22	10	141	-99	186	89
01-01-22	11	140	-99	158	93
01-01-22	12	121	-99	111	96
01-01-22	13	77	-99	79	92
01-01-22	14	65	-99	69	26
01-01-22	15	60	-99	57	20

Tabla 5.2: Los 24 puntos estratégicos de la ZMVM monitoreadas para detección de partículas PM2.5

Clave	Nombre	Alcaldía o Municipio	Entidad
AJU	Ajusco	Tlalpan	CDMX
AJM	Ajusco Medio	Tlalpan	CDMX
BJU	Benito Juárez	Benito Juárez	CDMX
CAM	Camarones	Azcapotzalco	CDMX
CCA	Centro de Ciencias de la Atmósfera	Coyoacán	CDMX
COY	Coyoacán	Coyoacán	CDMX
FAR	FES Aragón	Nezahualcóyotl	Estado de México
GAM	Gustavo A. Madero	Gustavo A. Madero	CDMX
HGM	Hospital General de México	Cuauhtémoc	CDMX
INN	Investigaciones Nucleares	Ocoyoacac	Estado de México
MER	Merced	Venustiano Carranza	CDMX
MGH	Miguel Hidalgo	Miguel Hidalgo	CDMX
MPA	Milpa Alta	Milpa Alta	CDMX
MON	Montecillo	Texcoco	Estado de México
NEZ	Nezahualcóyotl	Nezahualcóyotl	Estado de México
PED	Pedregal	Álvaro Obregón	CDMX
SAG	San Agustín	Ecatepec de Morelos	Estado de México
SFE	Santa Fe	Cuajimalpa de Morelos	CDMX
SAC	Santiago Acahualtepec	Iztapalapa	CDMX
SJA	San Juan de Aragón	Gustavo A. Madero	CDMX
TLA	Tlalnepantla	Tlalnepantla de Baz	Estado de México
UIZ	UAM Iztapalapa	Iztapalapa	CDMX
UAX	UAM Xochimilco	Coyoacán	CDMX
XAL	Xalostoc	Ecatepec de Morelos	Estado de México

5.4. Selección de datos a utilizar

La base de datos original está dividida en años y dado que se tienen datos disponibles desde el año de 2003, se definieron varios criterios para seleccionar de manera óptima los años a utilizar para esta investigación.

- Años sin ningún mes faltante
- Años más recientes
- Años sin cuarentena total por la pandemia
- Años con menos de el 30 % de datos faltantes

Considerando estos parámetros y la disponibilidad parcial de los datos del año 2023 al momento de la experimentación, se optó por excluir dicho año del análisis para preservar la integridad de los datos utilizados en el estudio, los años seleccionados fueron 2021 y 2022, ya que son los años más recientes, no se vieron afectados o sesgados por la cuarentena total de la pandemia de 2020 y estaban disponibles en su totalidad al momento de hacer la experimentación, en la tabla 5.3 se puede observar el razonamiento detrás del proceso de selección de años.

Tabla 5.3: Años seleccionados para la investigación con la explicación pertinente

Año	Veredicto	Razón
2019	Omitido	Puede sesgar nuestros experimentos por su antiguedad
2020	Omitido	Puede sesgar nuestros experimentos por la pandemia
2021	Incluido	Menos del 30 % de datos faltantes y es reciente
2022	Incluido	Menos del 30 % de datos faltantes y es reciente
2023	Omitido	Disponible sólo de manera parcial

5.5. Selección estaciones de monitoreo

Siendo los años elegidos 2021 y 2022 para esta investigación, el siguiente paso fue concatenar estas dos bases de datos para así poder tener una visión clara del porcentaje de datos faltantes para cada estación en estos 2 años, para seleccionar nuestras estaciones se descartaron todas las que tienen más del 30 % de datos faltantes, esto dejó un total de 13 estaciones utilizables. El análisis detallado se presenta en la Tabla 5.4.

Teniendo nuestras estaciones seleccionadas y sus porcentajes exactos de valores faltantes, el siguiente paso fue elegir una única estación a usar para ambos tipos de predicción (univariada y univarada), para así dar uniformidad a nuestros resultados, naturalmente se eligió la estación con menos datos faltantes, como se puede observar en la tabla 5.5 y en la figura 5.2, tanto para la predicción univariada como multivariada se utilizó la estación BJU.

Tabla 5.4: Porcentaje de valores faltantes para cada estación en los años 2021 y 2022

Estación	Porcentaje de valores faltantes	Veredicto
BJU	4.19 %	Incluida
UAX	8.34 %	Incluida
MER	7.52 %	Incluida
TLA	13.91 %	Incluida
FAR	14.89 %	Incluida
PED	20.10 %	Incluida
UIZ	22.11 %	Incluida
SAG	22.71 %	Incluida
NEZ	23.34 %	Incluida
INN	25.78 %	Incluida
SFE	26.39 %	Incluida
SAC	27.45 %	Incluida
MON	29.87 %	Incluida
CCA	34.14 %	Omitida
CAM	36.14 %	Omitida
MPA	42.90 %	Omitida
AJU	47.08 %	Omitida
GAM	74.87 %	Omitida
SJA	100 %	Omitida
COY	100 %	Omitida
XAL	100 %	Omitida
HGM	100 %	Omitida
AJM	100 %	Omitida
MGH	100 %	Omitida

Figura 5.2: Estaciones a utilizar en cada tipo de experimento (Autoría propia).

Tabla 5.5: Estaciones utilizadas para cada experimento

Predicción	Estaciones que entran al modelo como contexto	Estaciones a predecir
Univariada	BJU	BJU
Multivariada	BJU, UAX, MER, TLA, FAR, PED, UIZ, SAG, NEZ, INN, SFE, SAC, MON	BJU

5.6. Imputación de datos

Como pasa de manera usual en cualquier base de datos, se tenían datos faltantes, esto puede ocurrir por varias razones, como malfuncionamiento del equipo, fallas en los servidores o error humano, estos datos faltantes están representados con el número -99, para poder utilizar la base de datos de manera efectiva con nuestros modelos de inteligencia artificial se tuvo que efectuar imputación de datos, para nuestra imputación de datos se utilizó el algoritmo de Imputación Múltiple con Ecuaciones Encadenadas (o MICE por sus siglas en Inglés: Multiple Imputation by Chained Equations), esto se puede visualizar en las figuras 5.3 y 5.4.

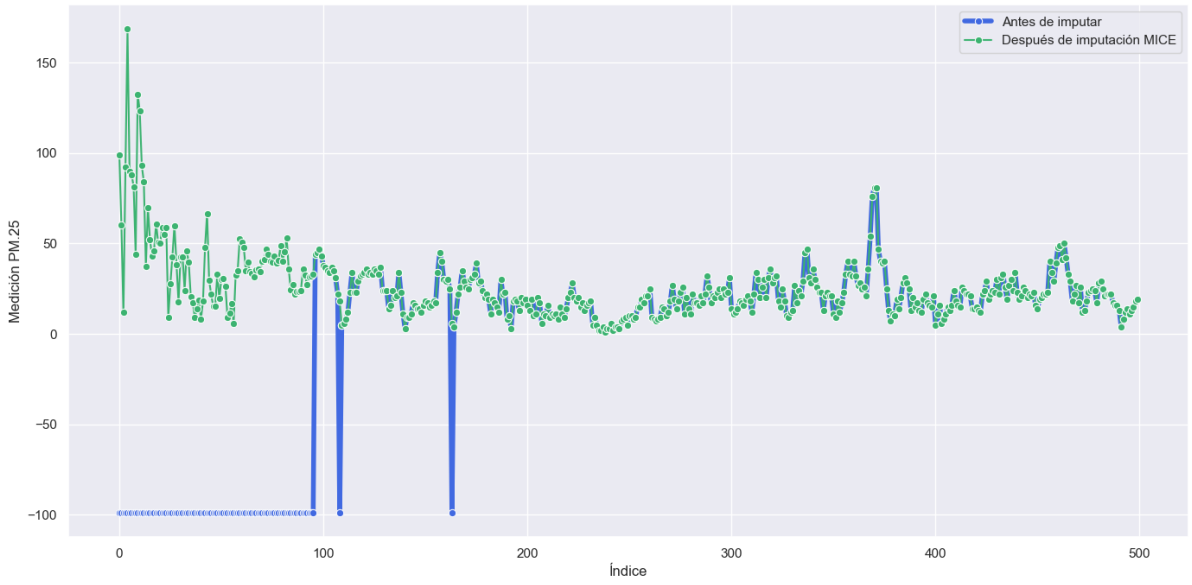

Figura 5.3: Muestra de la base de datos antes y después de imputar (Autoría propia).

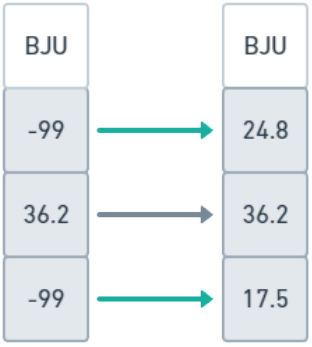

Figura 5.4: Muestra de la base de datos antes y después de imputar (Autoría propia).

5.7. Promediado de datos

Teniendo los datos imputados el siguiente paso es sacar el promedio por día de concentración de PM2.5, la base de datos original contiene una medición de concentración de PM2.5 cada hora, por lo tanto, se sacó el promedio a intervalos de 24 valores, esto se hizo, con la finalidad de determinar si nuestra medición promedio del día esta considerada como una excedencia o no de acuerdo a la normativa NOM-025-SSA1-2021.

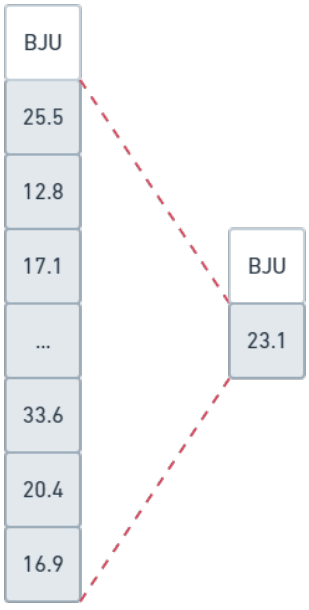

Figura 5.5: Muestra de la base de datos antes y después de promediado por día (Autoría propia).

5.8. Codificación a excedencias

Teniendo los datos promediados cada 24 horas, se convirtieron a ceros y unos, dependiendo si la concentración de PM2.5 es igual o está por encima a lo permitido por la normativa mexicana NOM-025-SSA1-2021 que establece que el promedio por día (24 horas) no debe exceder los 41 $\mu g/m^3$.

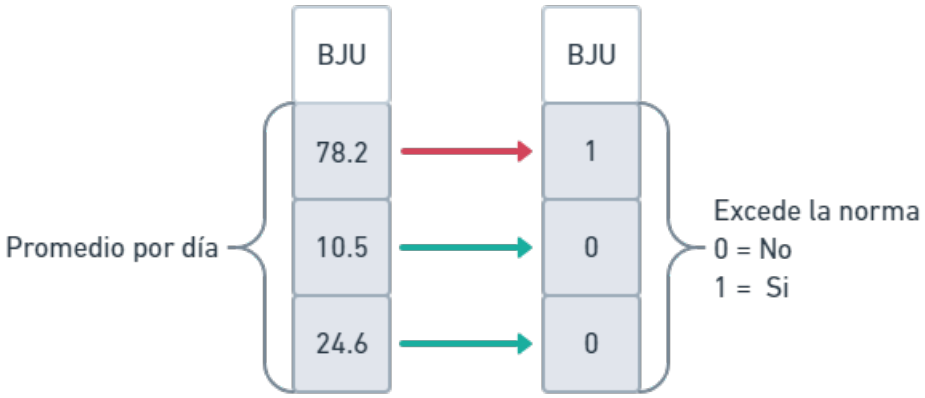

Figura 5.6: Ejemplo de valores codificados a excedencias (Autoría propia).

Tabla 5.6: Base de datos después de codificación

Date	BJU	UAX	MER	TLA	FAR
2021-01-01	1	1	1	1	1
2021-01-02	0	0	1	0	0
2021-01-03	0	0	0	0	0
2021-01-04	0	1	1	0	0
2021-01-05	0	0	0	0	1
2021-01-06	0	0	1	0	0
2021-01-07	0	0	1	1	0
2021-01-08	0	0	1	1	0
2021-01-09	0	0	0	0	0
2021-01-10	0	0	0	0	0
2021-01-11	0	0	0	0	0
2021-01-12	0	0	0	0	0
2021-01-13	0	0	1	0	0
2021-01-14	0	0	0	0	0
2021-01-15	0	0	1	1	1

5.9. División de datos

Para los experimentos, se dividieron los datos en tres subconjuntos: entrenamiento, validación y prueba, con proporciones del 70 %, 10 % y 20 % como se muestra en la figura 5.7. Esta división se realiza con el propósito de evitar el sobreajuste, un problema común en el entrenamiento de redes neuronales.

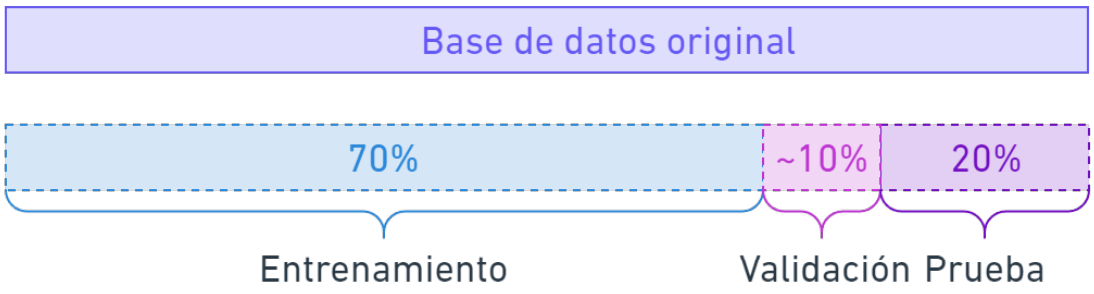

Figura 5.7: División de datos (Autoría propia).

Capítulo 6

Resultados y discusión

6.1. Experimentación

Los experimentos principales se clasifican en dos categorías: LSTM e Informer. Dentro de cada una de estas categorías, se distinguen dos subcategorías: experimentos multivariados y experimentos univariados. Además, cada subcategoría incluye experimentos realizados con horizontes de predicción de 24, 48 y 72 horas. En total, se llevaron a cabo 12 experimentos diferentes, distribuidos equitativamente entre las dos categorías principales y sus respectivas subcategorías, como se muestra en la tabla 6.1.

Tabla 6.1: Experimentos realizados de Informer y LSTM

Informer		**LSTM**	
Univariada	Multivariada	Univariada	Multivariada
24 h	24 h	24 h	24 h
48 h	48 h	48 h	48 h
72 h	72 h	72 h	72 h

Para cada una de estas 12 variantes se hicieron 1000 experimentos, dando un total de 6,000 experimentos por modelo, es decir, un total 12,000 experimentos para el análisis final de resultados para este trabajo. Dentro de estos 12,000 experimentos, se fueron variando diferentes hiperparámetros tanto en el modelo del Informer como en el modelo LSTM, a

continuación se muestran todos los hiperparámetros que se probaron de manera heurística en cada modelo.

Tabla 6.2: Hiperparámetros usados en modelo LSTM

Hiperparámetro	**Valores**
Tamaño del batch	10, 15, 20, 25, 30, 35, 40, 45, 50
Longitud de secuencia	2, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60
Función de activación	elu, gelu, relu, sigmoid, swish, tanh
Arquitectura	[10], [10, 10, 10], [100, 100, 100], [15, 15, 15], [20], [20, 20, 20], [30]

Tabla 6.3: Hiperparámetros usados en modelo Informer

Hiperparámetro	**Valores**
Tamaño del batch	8, 16, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50
Longitud de secuencia	20, 30, 40, 50, 60, 70
Función de activación	sigmoid, gelu, swish, tanh
Dimensión red totalmente conectada	128, 256, 512, 1024
Capas decodificador	1, 2, 4
Dimensión modelo	256, 512
Dropout	0.05, 0.1, 0.15, 0.2
Capas codificador	1, 2, 4
Longitud etiquetas	5, 10
Número de cabezas de atención	4, 8, 10, 12, 14, 18, 20
Taza de aprendizaje	0.01, 0.001, 0.0001

Como se puede observar en el diagrama de la metodología general (figura 5.1), el proceso comenzó con la realización de experimentos iniciales utilizando una amplia gama de hiperparámetros propuestos. En total, se llevaron a cabo 500 pruebas iniciales para evaluar el desempeño de los modelos con diferentes configuraciones de hiperparámetros. Estos experimentos fueron fundamentales para entender cómo cada hiperparámetro afectaba el rendimiento de los modelos en términos de errores MAE y MSE en el conjunto de prueba.

Al finalizar estas pruebas iniciales, se llevó a cabo un análisis exhaustivo de los resultados. Este análisis permitió identificar cuáles hiperparámetros arrojaban los resultados menos favorables, es decir, aquellos que resultaron en los mayores errores MAE y MSE. La identificación de estos hiperparámetros fue crucial para mejorar la eficiencia de los modelos, ya que su eliminación permitió centrarse en las configuraciones más prometedoras.

En consecuencia, los hiperparámetros que no contribuyeron positivamente al rendimiento de los modelos fueron eliminados del conjunto de parámetros considerados para los experimentos posteriores. Este enfoque iterativo y basado en la evidencia garantizó que los modelos finales se construyeran utilizando únicamente los hiperparámetros más efectivos.

A continuación, en las tablas 6.4 y 6.5, se presentan los hiperparámetros que proporcionaron los mejores resultados para nuestros modelos. Los hiperparámetros mostrados en estas tablas son el producto de un riguroso proceso de selección y optimización, destacando los valores de hiperparámetros que minimizaron los errores MAE y MSE, mejorando así la precisión y la eficiencia de los modelos propuestos.

Tabla 6.4: Hiperparámetros con mejores resultados en modelo LSTM

Hiperparámetro	**Valores**
Tamaño del batch	10, 15, 20, 25, 30, 35, 40, 45, 50
Longitud de secuencia	2, 5, 10, 15, 20, 25, 35
Función de activación	elu, gelu, relu, sigmoid, swish, tanh
Arquitectura	[10], [15, 15, 15], [20], [20, 20, 20], [30]

Tabla 6.5: Hiperparámetros con mejores resultados en modelo Informer

Hiperparámetro	**Valores**
Tamaño del batch	32, 34
Longitud de secuencia	40, 50, 60, 70
Función de activación	gelu, swish, tanh
Dimensión red totalmente conectada	128, 256, 512, 1024
Capas decodificador	1, 2, 4
Dimensión modelo	256, 512
Dropout	0.05, 0.1
Capas codificador	1, 2, 4
Longitud etiquetas	5, 10
Número de cabezas de atención	12, 14, 18, 20
Taza de aprendizaje	0.0001

En las tablas 6.6 y 6.7 se puede observar de manera detallada los resultados obtenidos en ambos modelos con las métricas de MAE y MSE respectivamente, estos resultados son producto de la utilización de los hiperparámetros que no fueron eliminados dentro de las primeras 500 pruebas, ya que mostraban los errores menores, tanto para MAE como para

MSE, para cada modelo, estas tablas están ordenadas de menor a mayor por el promedio de los errores, cabe destacar que el informer siempre está en las primeras posiciones, lo cual hace claro que este modelo tiene un mejor desempeño.

Tabla 6.6: Resultados de los modelos propuestos (métrica MAE)

Modelo	**Tipo**	**Horas**	**MAE min**	**MAE max**	**MAE avg**	**MAE std**
Informer	MS	24	0.1730	0.3862	**0.2487**	0.0312
	S	24	0.1721	0.4031	**0.2532**	0.0339
	MS	48	0.1976	0.3974	**0.2705**	0.0309
	S	48	0.2004	0.3823	**0.2713**	0.0326
	MS	72	0.2127	0.3863	**0.2816**	0.0305
	S	72	0.2079	0.3886	**0.2829**	0.0314
LSTM	MS	24	0.2738	0.5386	**0.3641**	0.0419
	MS	48	0.2927	0.4583	**0.3733**	0.0378
	MS	72	0.3063	0.4607	**0.3807**	0.0356
	S	24	0.3035	0.4605	**0.3823**	0.0345
	S	48	0.3074	0.5254	**0.3879**	0.0336
	S	72	0.3171	0.4745	**0.3958**	0.0302

Tabla 6.7: Resultados de los modelos propuestos (métrica MSE)

Modelo	**Tipo**	**Horas**	**MSE min**	**MSE max**	**MSE avg**	**MSE std**
Informer	S	24	0.1019	0.2485	**0.1458**	0.0222
	MS	24	0.1025	0.2466	**0.1516**	0.0200
	S	48	0.1113	0.2368	**0.1550**	0.0215
	S	72	0.1178	0.2399	**0.1618**	0.0199
	MS	48	0.1222	0.2526	**0.1715**	0.0210
	MS	72	0.1274	0.2975	**0.1829**	0.0218
LSTM	S	24	0.1674	0.2723	**0.2019**	0.0216
	MS	24	0.1588	0.3435	**0.2023**	0.0245
	S	48	0.1707	0.2855	**0.2066**	0.0209
	S	72	0.1803	0.2865	**0.2130**	0.0194
	MS	48	0.1717	0.3527	**0.2130**	0.0279
	MS	72	0.1788	0.3694	**0.2190**	0.0285

Para facilitar una comparación más rigurosa y detallada entre los modelos, se añadieron las tablas 6.8 y 6.9. En estas tablas, se presentan en la misma fila los resultados de los experimentos equivalentes realizados con ambos modelos, lo que permite una evaluación directa de su desempeño. Esta disposición permite calcular la mejora porcentual en cada experimento,

proporcionando una visión más clara y precisa de las ventajas del modelo Informer sobre el modelo LSTM.

Es importante destacar que los resultados muestran consistentemente que el modelo Informer supera al modelo LSTM en los 12 experimentos realizados en este estudio. Estos experimentos abarcan tanto análisis multivariados como univariados y consideran periodos de predicción de 24, 48 y 72 horas. La comparación detallada y sistemática en las tablas 6.8 y 6.9 demuestra de manera inequívoca que el Informer ofrece un mejor rendimiento en términos de precisión y eficiencia en comparación con el LSTM, independientemente del tipo y la duración de la predicción.

Tabla 6.8: Comparación de promedio de MAE entre Informer y LSTM

Tipo	**Horas**	**MAE AVG LSTM**	**MAE AVG Informer**	**Mejora (%)**
MS	24	0.3641	**0.2487**	31.69
S	24	0.3823	**0.2532**	33.77
MS	48	0.3733	**0.2705**	27.54
S	48	0.3879	**0.2713**	30.06
MS	72	0.3807	**0.2816**	26.03
S	72	0.3958	**0.2829**	28.52
Promedio total		0.3807	**0.2680**	29.59

Tabla 6.9: Comparación de promedio de MSE entre Informer y LSTM

Tipo	**Horas**	**MSE AVG LSTM**	**MSE AVG Informer**	**Mejora (%)**
MS	24	0.2023	**0.1516**	25.06
S	24	0.2019	**0.1458**	27.79
MS	48	0.2130	**0.1715**	19.48
S	48	0.2066	**0.1550**	24.98
MS	72	0.2190	**0.1829**	16.48
S	72	0.2130	**0.1618**	24.04
Promedio total		0.2093	**0.1614**	22.97

En la figura 6.1, se presenta una visualización gráfica que ilustra la mejora porcentual obtenida con el Informer en comparación con la LSTM de cada experimento realizado, tal como se detalla en las tablas anteriores. En esta figura, la métrica MAE se representa con barras de color azul ubicadas a la izquierda, mientras que la métrica MSE se representa con barras de

color verde situadas a la derecha. Es notable que las barras se mantienen consistentemente por arriba del 20 % en la mayoría de experimentos.

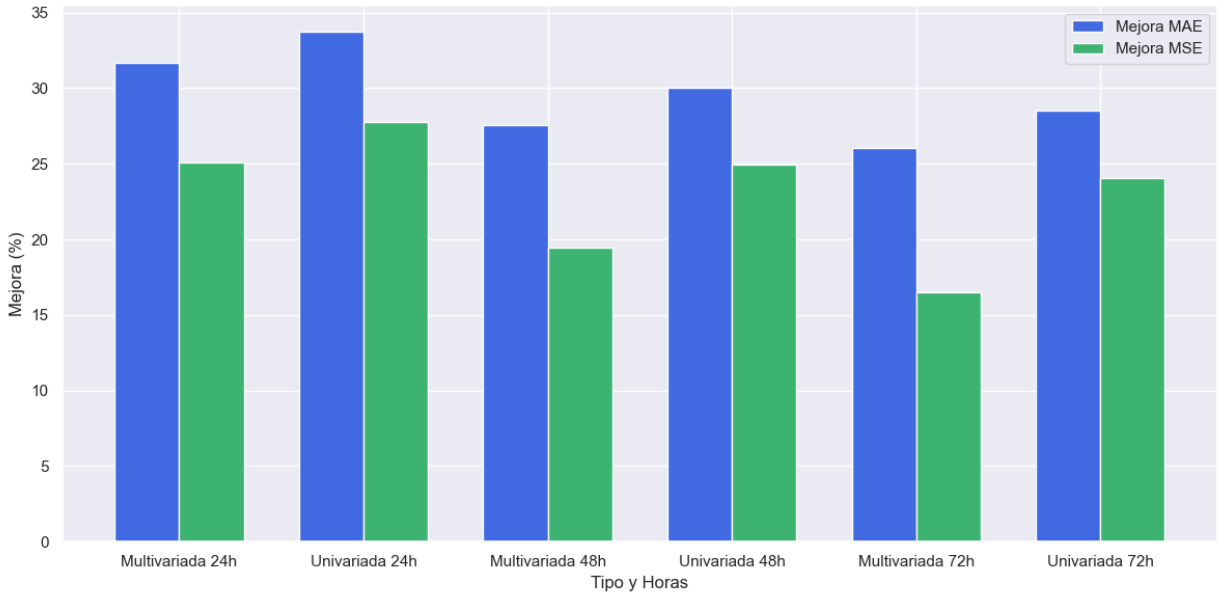

Figura 6.1: Comparación de mejora del error entre Informer y LSTM

De manera más específica, en las figuras 6.2, 6.3, 6.4 y 6.5 se presentan gráficos de caja que proporcionan una visualización detallada de los resultados de los experimentos. La figura 6.2 muestra los gráficos de caja para el MAE univariado, mientras que la figura 6.3 presenta los gráficos de caja para el MSE univariado. Por otro lado, la figura 6.4 ilustra los gráficos de caja para el MAE multivariado y la figura 6.5 se enfoca en los gráficos de caja para el MSE multivariado.

Estos gráficos permiten observar la distribución de los datos, destacando la mediana, los cuartiles y la presencia de posibles valores atípicos (outliers). La inclusión de estos gráficos de caja facilita la identificación de la variabilidad y la dispersión de los datos para cada modelo, proporcionando una comprensión más profunda de su desempeño en los distintos experimentos.

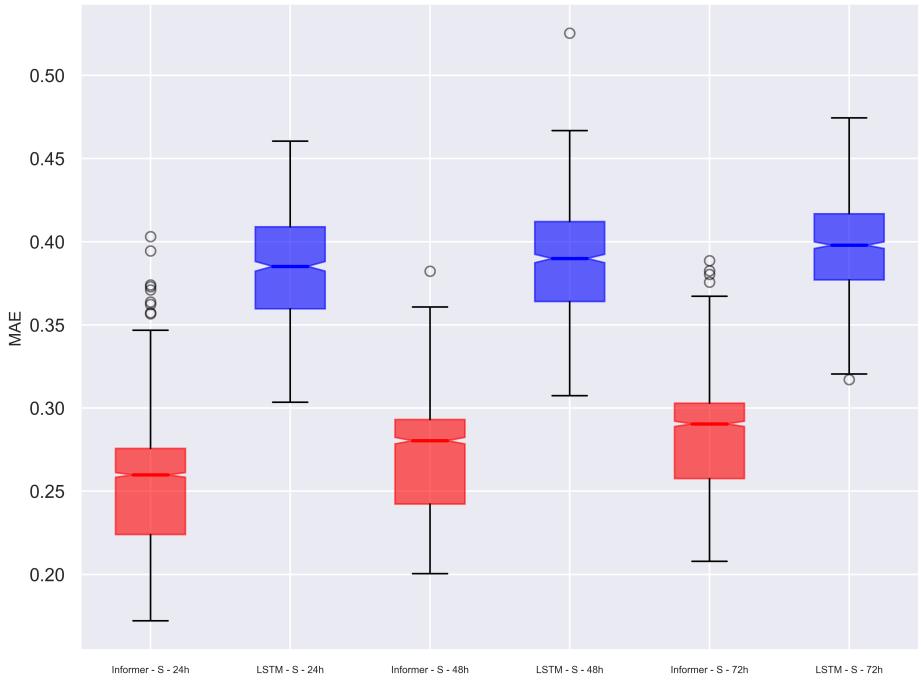

Figura 6.2: Diagrama de caja para MAE en predicción univariada

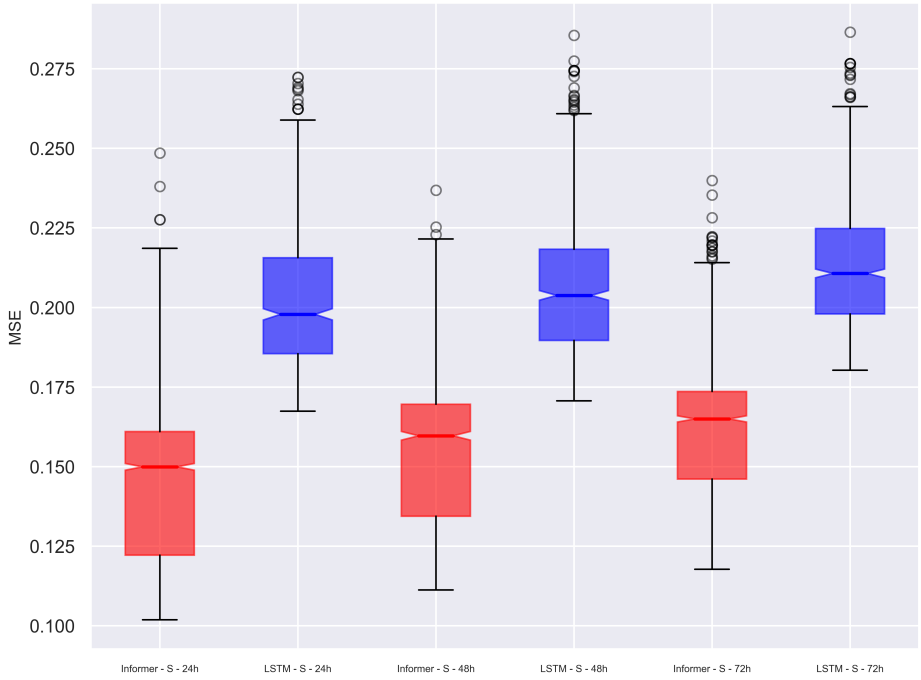

Figura 6.3: Diagrama de caja para MSE en predicción univariada

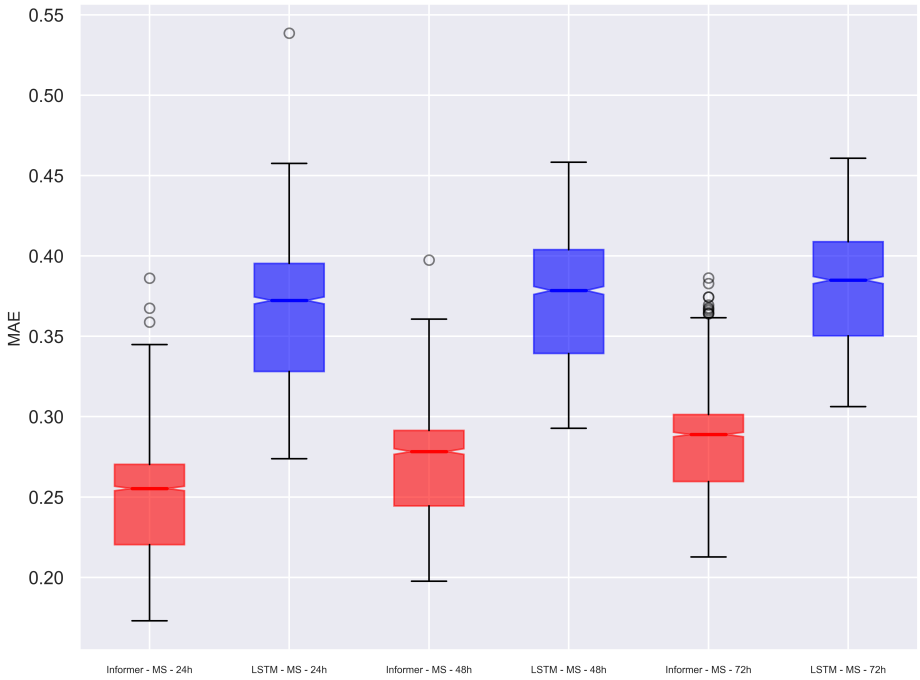

Figura 6.4: Diagrama de caja para MAE en predicción multivariada

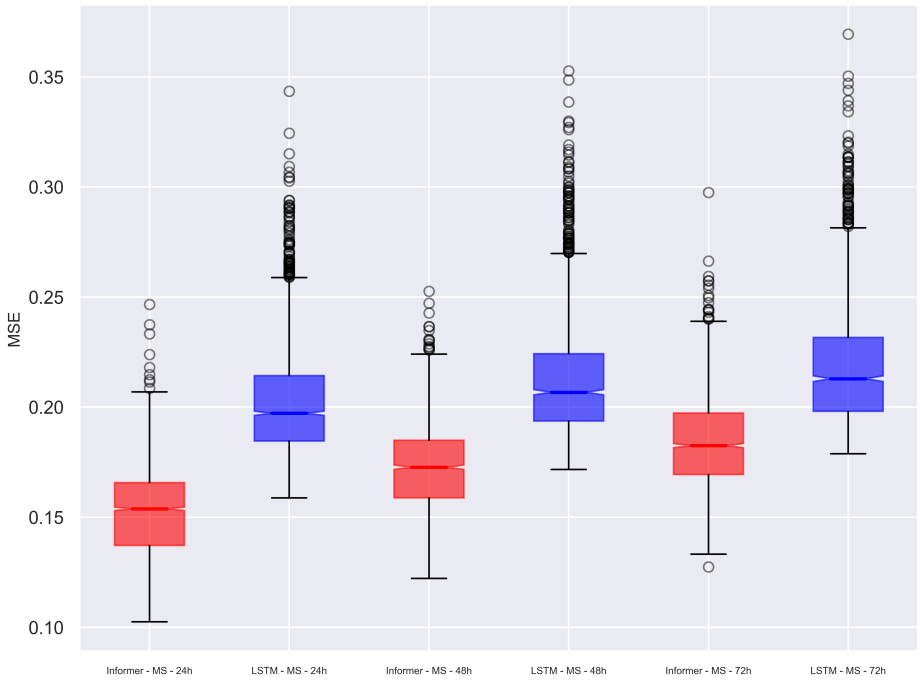

Figura 6.5: Diagrama de caja para MSE en predicción multivariada

6.2. Discusión

El modelo Informer supera consistentemente a la red LSTM en términos de desempeño, tanto en términos de Error Absoluto Medio (MAE) como de Error Cuadrático Medio (MSE), en todos los experimentos, incluidas las predicciones multivariadas y univariadas. La mejora promedio en los errores fue del 29.59 % para MAE y del 22.97 % para MSE. Esto se evidencia claramente en la Figura 6.1, donde se muestra la comparación directa de la mejora del error, tanto en MAE (en azul) como en MSE (en verde), para cada uno de los experimentos realizados. Además, se observa que la mejora del error es siempre más significativa en el caso de las predicciones univariadas.

Es evidente que el modelo Informer supera a la red LSTM en ambos tipos de predicciones, multivariadas y univariadas. Específicamente, el Informer se beneficia de tener más información disponible, lo que se refleja en los resultados obtenidos para los valores de longitud de secuencia de ambos modelos, esto se puede observar en las Tablas 6.4 y 6.5 donde los valores más pequeños de longitud de secuencia, como 20 y 30, no proporcionaron los mejores resultados para el modelo Informer, a diferencia de la LSTM, que obtuvo buenos resultados con longitudes más cortas, como 2, 5 y 10.

Asimismo, las Tablas 6.8 y 6.9 evidencian una mejora significativa en las métricas de predicción univariada al utilizar el modelo Informer. Estos resultados sugieren que la efectividad del modelo Informer no está limitado a escenarios de predicción multivariada, donde se dispone de un contexto más amplio, por la información extra que se ingresa al modelo, sino que también se extiende a predicciones univariadas basadas únicamente en los datos de la columna objetivo. Este hallazgo respalda la versatilidad y robustez del modelo Informer en diversas aplicaciones de predicción.

Capítulo 7

Conclusiones

Se puede concluir de manera definitiva que el modelo Informer muestra una ventaja significativa en comparación con la arquitectura LSTM para predicciones de series de tiempo, tanto univariadas como multivariadas. Esta investigación demostró que esta innovadora arquitectura, basada en transformers y modelos de atención, es capaz de mejorar el error de manera consistente, incluso en predicciones con varias horas de anticipación, en este caso específico de 24, 48 y 72 horas.

Referencias

Admassu, M., & Wubeshet, M. (2011). For Environmental Health Science Students.

Aggarwal, C. C. (2018). Neural networks and deep learning. Springer, 10(978), 3.

Ahmed, S., Nielsen, I. E., Tripathi, A., Siddiqui, S., Ramachandran, R. P., & Rasool, G. (2023). Transformers in time-series analysis: A tutorial. Circuits, Systems, and Signal Processing, 42(12), 7433-7466.

Azunre, P. (2021). Transfer learning for natural language processing. Simon and Schuster.

Boschetti, A., & Massaron, L. (2015). Python data science essentials. Packt Publishing Ltd.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: forecasting and control. John Wiley & Sons.

Buduma, N., Buduma, N., & Papa, J. (2022). Fundamentals of deep learning. .º'Reilly Media, Inc.".

Chollet, F. (2021). Deep learning with Python. Simon and Schuster.

Cowpertwait, P. S., & Metcalfe, A. V. (2009). Introductory time series with R. Springer Science & Business Media.

Demuth, H. B., Beale, M. H., De Jess, O., & Hagan, M. T. (2014). Neural network design. Martin Hagan.

Falcon-Rodriguez, C. I., Osornio-Vargas, A. R., Sada-Ovalle, I., & Segura-Medina, P. (2016). Aeroparticles, composition, and lung diseases.

Ganegedara, T. (2022). TensorFlow in Action. Simon and Schuster.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Harrington, P. (2012). Machine learning in action. Simon and Schuster.

Haykin, S. (2009). Neural networks and learning machines, 3/E. Pearson Education India.

Khorasani, E. S. (2008). Artificial intelligence: Structures and strategies for complex problem solving. Scalable Computing: Practice and Experience, 9(3).

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). Foundations of machine learning. MIT press.

Molnar, C. (2020). Interpretable machine learning. Lulu. com.

Newby, D. E., Mannucci, P. M., Tell, G. S., Baccarelli, A. A., Brook, R. D., Donaldson, K., ... & Storey, R. F. (2015). Expert position paper on air pollution and cardiovascular disease. European heart journal, 36(2), 83-93

Peixeiro, M. (2022). Time series forecasting in python. Simon and Schuster.

Raschka, S., & Mirjalili, V. (2019). Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing Ltd.

Roberts, D. A., Yaida, S., & Hanin, B. (2022). The principles of deep learning theory. Cambridge, MA, USA: Cambridge University Press.

Rose, D. (2020). Artificial Intelligence for Business. FT Press.

Russell, S. J., & Norvig, P. (2010). Artificial intelligence a modern approach. London.

Sarkar, D., Bali, R., & Sharma, T. (2018). Practical machine learning with Python. Book"Practical Machine Learning with Python, 25-30.

Shukla, N., & Fricklas, K. (2018). Machine learning with TensorFlow. Greenwich: Manning.

Stevens, E., Antiga, L., & Viehmann, T. (2020). Deep learning with PyTorch. Manning Publications.

Thakur, A. (2020). Approaching (almost) any machine learning problem. Abhishek Thakur.

Trask, A. W. (2019). Grokking deep learning. Simon and Schuster.

Vasilev, I., Slater, D., Spacagna, G., Roelants, P., & Zocca, V. (2019). Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow. Packt Publishing Ltd.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need.

Warwick, K. (2013). Artificial intelligence: the basics. Routledge.

World Health Organization. (2013). Health Effects of Particulate Matter: Policy implications for countries in eastern Europe, Caucasus and central Asia. In Health effects of particulate matter: policy implications for countries in eastern Europe, Caucasus and central Asia.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). Dive into deep learning. Cambridge University Press.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021, May). Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 12, pp. 11106-11115).

Zhu, Q., Han, J., Chai, K., & Zhao, C. (2023). Time Series Analysis Based on Informer Algorithms: A Survey. Symmetry, 15(4), 951.