Universidad Nacional de Misiones

Facultad de Ciencias Exactas Químicas y Naturales

Tesis de grado Licenciatura en Sistemas de Información

Pronóstico de Series de Tiempo de Tráfico web. Estudio de Caso Universidad Nacional de Misiones

Autor: Enrique Gauto Sand

Tutor: Alice Raquel Rambo

Año 2025

Dedicatoria

A mi padre Roberto Gauto por recomendarme estudiar.

A mi mamá Delia Sand por su apoyo incondicional.

Resumen

El tópico de pronosticar tráfico web, es hoy en día un campo ampliamente estudiado, esto ayuda a saber anticipadamente la posible demanda de la cantidad de visitas que pueda tener un sitio web, esto ayuda a los administradores de las páginas web a poder tomar medidas a tiempo, de esta manera poder escalar recursos según consideren necesario. Evitando así problemas que puede traer la falta de antelación ante grandes demandas, como ser, la caída de los servidores, carga más lenta de las páginas web, usuarios que se van porque las páginas web no responden a tiempo, entre otros.

Esta tesis aborda el caso de estudio del pronóstico de tráfico web de las páginas Institucionales de la Universidad Nacional de Misiones, para ello se utilizaron los datos de Google Analytics y se utilizaron redes neuronales recurrentes, en específico, las redes de unidad de puerta recurrente con la arquitectura secuencia a secuencia, como resultado se obtuvieron modelos de las páginas web institucionales que tienen un buen rendimiento con un error medio absoluto por debajo de 30.

Palabras Claves: *tráfico web, pronóstico de series de tiempo, GRU*

Abstract

Nowadays the topic of web traffic forecasting is a well-studied field. Web traffic forecasting helps to know earlier the possible demand of the number of visits that a web site could have. Web traffic forecasting helps web sites' administrators to take action in time, hence web sites' administrators can scale resources if they need that. Web sites' administrators could avoid problems that can be caused by a lack of anticipation when a situation of large demands appears, such as server crashes, slower

loading of web pages, users leaving website because web pages do not respond in time, among others.

This thesis addresses the case study of time series web traffic forecasting of institutional websites of Universidad Nacional de Misiones, to achieve that I used datasets from Google Analytics, I used recurrent neural networks, specifically Gate Recurrent Unit GRU, and the sequence to sequence architecture. As a result, we got models that on the institutional websites datasets have a good performance with an absolute mean error below 30.

Keywords: *web traffic, time series forecasting, GRU*

Índice

Lista de Tablas

Lista de Figuras

Capítulo 1

Introducción

1.1. Motivación

Las páginas institucionales de la U.Na.M. registran desde 2018 sus datos del tráfico web a través de la herramienta de Google Analytics, específicamente por la herramienta Universal Analitics y desde 2022 también se registran los datos del tráfico web a través de la herramienta Google Analitics 4. Estos datos si bien son registrados prácticamente no se usan, actualmente con las tecnologías que disponemos podemos hacer un buen uso de los mismos, como ser, utilizarlos como entrada a redes neuronales que pronostiquen el tráfico web y nos ayuden a saber las posibles demandas de tráfico web que puedan tener las páginas institucionales.

Dado que tener información sobre el comportamiento del usuario como ser la cantidad de visitas que podría recibir una página web en un periodo de tiempo determinado, ayuda al personal del Departamento de Gestión de Recursos de Redes y Comunicaciones de las páginas institucionales a llevar a cabo su trabajo y según consideren poder reaccionar a la demanda de los usuarios. Por lo tanto, es importante realizar el pronóstico de series de tiempo de tráfico web ya que será de ayuda para el personal del Departamento de Gestión de Recursos de Redes y Comunicaciones.

1.2. Objetivos

1.3. Estructura del documento

La tesis se divide en 5 capítulos y 1 anexo. A continuación se describe brevemente el contenido de los mismos.

En el Capítulo 2: se brinda un marco teórico para el trabajo de investigación, en principio se introduce los conceptos de tráfico web, series de tiempo y el estado del arte respecto pronóstico de series de tiempo de tráfico web, finalizando con las métricas de evaluación de desempeño de los modelos.

El Capítulo 3 describe el problema y la estructura organizacional.

El Capítulo 4 describe la metodología y las herramientas utilizadas, el pre procesado de los datos, la selección de la arquitectura para los modelos, después se describe el desarrollo del plan de pruebas y explicaciones de las pruebas realizadas, como así el rendimiento de los modelos por cada prueba realizada.

Finalmente en el Capítulo 5 se realizan las conclusiones obtenidas a partir del trabajo realizado y las posibles futuras líneas de investigación.

En el Anexo 1 se presentan las tablas e imágenes que detallan los modelos de las pruebas realizadas.

Capítulo 2

Marco Teórico

En este capítulo se presenta el marco teórico del trabajo realizado, comenzando con la definición de tráfico web y series de tiempo, luego con variados ejemplos de la literatura, distintas tecnologías que realizan el pronóstico de series de tiempo de tráfico web finalizando con las formas de medir el desempeño de los modelos.

2.1.Tráfico web

El tráfico web es generado por los usuarios de una página, el tráfico son los datos que se envían y se reciben cuando los usuarios visitan la página web[1].

2.2.Series de Tiempo

Las series de tiempo son un conjunto de valores medidos en orden secuencial en un rango determinado [2]; cuando se miden los datos para generar una serie de tiempo, generalmente se toman los valores con la misma separación entre cada valor [3].

2.3. Pronóstico de series de tiempo de tráfico web

El pronóstico de series de tiempo se basa en las observaciones pasadas de la serie de tiempo a pronosticar y otras entradas, siendo el proceso de predecir valores futuros de las mismas[2].

Entre las formas del pronóstico de series de tiempo de tráfico web los autores en[3] proponen redes Modelo Generativo Adversario - GAN (*Generative Adversarial model*)[3] con reders de Memoria a Corto y Largo Plazo – LSTM (*Long Short Term Memory*)[4] y un perceptrón multicapa - MLP (*Multilayer Perceptron*)[5], donde LSTM y GAN actuarían como generador y el MLP como discriminador, para generar series de tiempo dado el conjunto de datos real, finalmente se realizaría el pronóstico con la librería *Prophet*[6] comparando la combinación de tecnologías anteriores con

métodos estadísticos, finalmente llegando a la conclusión de que los autores no obtuvieron una diferencia notable[3].

Entre las formas del pronóstico de series de tiempo de tráfico web que existen, los autores en [7] proponen la técnica de redes neuronales LSTM[4] con entrenamiento asíncrono distribuido, cuya métrica de desempeño que se utilizó según el autor en[7] es el Error Medio Absoluto - MAE (*mean absolute error*)[8] y la función de pérdida de Huber[9] para probar la precisión del modelo, han logrado un buen grado de asertividad con 200 épocas, en el documento científico[7] mencionan que obtuvieron una MAE en promedio menor que 30, lo que consideran un buen resultado[7].

Hay otro artículo[1] en el cual se menciona el uso de Redes Neuronales Recurrentes - RNN (*Recurrent Neural Networks*) seq2seq (*sequence to sequence*) [1] con la ayuda de la arquitectura Codificador –Decodificador (*encoder/decoder)*[10], el *encoder* es cuDNN-GRU.

Es decir una Unidad Recurrente con Puerta (*Gate Recurrent Unit*)[11] con *backend* de CUDA, ya que realiza la tarea con mayor velocidad en comparación con los tensores regulares, el *decoder* es TensorFlow GRUBlockCell[11], en este documento científico[1] menciona que realizaron algunos cambios al modelo ganador de *Kaggle*[11] cómo agregar la mediana de 7,30,90 y 180 días, usaron para medir el desempeño del modelo la métrica Error Absoluto Medio Porcentual Simétrico - SMAPE (*Symmetric Mean Absolute Percentage Error*)[12] donde comentan los autores que obtuvieron un SMAPE de 0.349[1].

En otro artículo[13], proponen utilizar un enfoque híbrido de optimización de enjambre de partículas (*particle swarm optimization)* e inspiración cuántica (*quantum-inspired)* resultando en QPSO[13] y redes Exógenas Autorregresivas No Lineales – NARX(*Nonlinear autoregressive exogenous*)[14] resultando en NARX - QPSO[13] en dicho artículo se menciona que el modelo NARX –QPSO tiene un rendimiento sobresaliente en los resultados comparados con otros modelos[13].

En otro artículo[15] propone un diseño de un sistema Aprendizaje Automático Automatizado - *Automated Machine Learning* (Auto ML) [15], una arquitectura neuronal nueva denominada Auto-PyTorch-TS[15] donde lo comparan con otros modelos y demuestran que su modelo tiene un mejor rendimiento[15].

También se propone en el artículo[16] el uso de un modelo combinado *Prophet* y Máquina de Empuje de Gradiente de Luz - LGBM (*light gradient boosting machine*)[16] donde se menciona que tiene mejores resultados comparado con modelos individuales[16].

Otra tecnología son las redes bidireccionales LSTM (BI-LSTM)[17] las cuales en este artículo[17] fueron probadas en el conjunto de datos (*dataframe*) *M3-Competition*[18] el cual se usa para probar modelos de pronóstico de series de tiempo dado que posee distintas categorías, en el artículo[17] mencionan que probaron las redes BI-LSTM sobre el conjunto de datos M3 en el periodo trimestral comparándolo con el modelo Media móvil integrada autorregresiva ARIMA (*Auto-Regressive Integrated Moving Average*)[2] y otros, donde BI-LSTM tuvo un mejor rendimiento.

2.4.Evaluación de desempeño de los modelos predictivos

El desempeño es la forma de evaluar el modelo, se utiliza para comparar los valores predichos del modelo entrenado, con los valores observados[19]

La distancia media cuadrática mínima - RMSE (*Root Mean Square Errors*): el error cuadrático medio es una medida de las diferencias entre los valores predichos por un modelo y los valores observados.

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}(r'_n - r_n)^2}{N}} \quad (1)$$

En la fórmula (1) r'_n es el valor predicho y r_n es el valor real[8]

El error medio absoluto - MAE (*mean absolute error*)[8] se calcula de la siguiente forma:

$$MAE = \frac{\sum_{n=1}^{N}|r'_n - r_n|}{N} \quad (2)$$

En la fórmula (2) r'_n es el valor predicho y r_n es el valor real[8],

La métrica Error Absoluto Medio Porcentual Simétrico - SMAPE (*Symmetric Mean Absolute Percentage Error*)[12] se calcula de la siguiente forma:

$$SMAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|A_t| + |F_t|)/2} \quad (3)$$

En la fórmula (3) F_t el valor pronosticado, A_t es el valor observado y n es el tamaño de la muestra.

Capítulo 3

Descripción del problema

En este capítulo se describe la problemática a abarcar como así también la estructura organizacional.

3.1 El problema

El presente trabajo pretende hacer análisis de tráfico web para poder estimar y predecir futuras demandas en las páginas institucionales de la Universidad Nacional de Misiones – U.Na.M. Para lo cual se consideraran indicadores como volumen de tráfico, cantidad de conexiones, cantidad de visitantes, velocidad de carga de las paginas, páginas vistas, promedio de páginas por vista, porcentaje de rebote, entre otros indicadores según la disponibilidad de los mismos. Siendo provisto por parte del personal técnico referente institucional acceso a las herramientas de captura de datos estadístico de los siguientes enlaces:

- Sitio Institucional de la U.Na.M. disponible en https://unam.edu.ar/
- Sitio de la Editorial Universitaria disponible en https://editorial.unam.edu.ar/
- Portal de acceso al contenido generado por la radio y la televisión de la Universidad disponible en https://transmedia.unam.edu.ar/

De los sitios mencionados anteriormente se extraerán los datos para utilizar en este proyecto con la finalidad de realizar el pronóstico de series de tiempo de tráfico web. Inicialmente como herramienta de acceso a los datos se dispone *Google Analytics* con permisos cedidos por los administradores.

3.2 Estructura Organizacional

Dentro de la institución se encuentra el Departamento de Gestión de Recursos de Redes y Comunicaciones, este se encuentra dependiente de la Secretaría General de Extensión Universitaria, la cual le responde directamente al Rector, con el fin de entender el funcionamiento del departamento se procede a describir la estructura organizacional del Departamento de Gestión de Recursos de Redes y Comunicaciones, como así también los roles de los trabajadores.

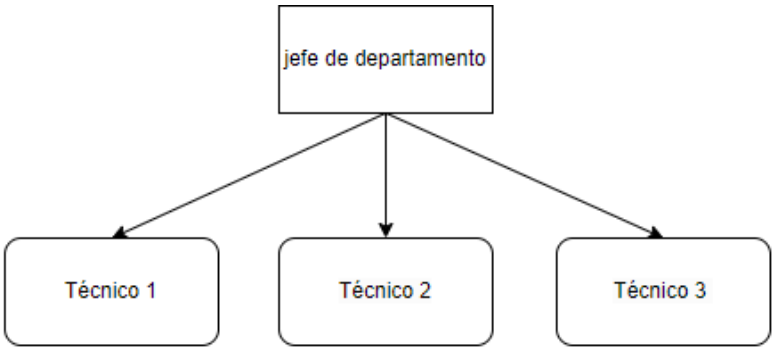

Figura 1: Organigrama del Departamento de Gestión de Recursos de Redes y Comunicaciones

El Departamento de Gestión de Recursos de Redes y Comunicaciones, gestiona la infraestructura, además gestiona las herramientas para la puesta en servicio de las páginas web y servicios añadidos. Pero no controla los contenidos, diseño o función de los sitios.

- **Jefe de departamento**: Realiza las tareas propias del jefe de departamento, también se encarga de mantener las plataformas de los sitios web, por ejemplo Moodle para Aulas Virtuales.
- **Técnico 1:** Se encarga principalmente de la gestión y mantenimiento de los servicios de voz sobre protocolo de Internet - VoIP, redes WiFi, gestión de las Redes de Área Local Virtual - VLAN (*Virtual Local Area Network*) y redes de

la unidad central de Rectorado, gestión de usuarios en la plataforma de *Google Workspace for education*, entre otros.

- **Técnico 2 y Técnico 3:** Estos dos técnicos se encargan de la gestión y ruteo de las redes de la universidad y con los proveedores de servicio, gestión y mantenimiento de los Servidores físicos y virtuales del Centro de Procesamiento de Datos - CPD, plataformas de respaldo de datos, sistemas de monitoreo de las plataformas y servicios esenciales de la infraestructura de la universidad como Sistema de Nombres de Dominio – DNS (*Domain Name System*), Protocolo de Puerta de Enlace Fronteriza – BGP, Camino Más Corto Primero – OSPF (*Open Shortest Path First*), Red Privada Virtual – VPN (*Virtual Private Network*), etc.

Capítulo 4

Solución Propuesta

En este capítulo se describe la metodología seleccionada para la solución, como así también las herramientas utilizadas, las pruebas realizadas y los modelos generados.

4.1 Materiales y Métodos

Para el desarrollo del presente trabajo se utilizó la metodología CRISP-DM[20] como guía, ya que hoy en día es un estándar de facto para los proyectos de ciencia de datos y minado de datos[21].

Las fases generales de la metodología CRISP-DM son las siguientes:

4.1.1 Comprensión del negocio:

En esta fase inicial se comprenden los objetivos y los requisitos para cumplir el proyecto de Pronóstico de Series de Tiempo de Tráfico web, donde también se realizaron entrevistas a los trabajadores del área de Departamento de Gestión de Recursos de Redes y Comunicaciones.

4.1.2 Comprensión de los datos:

En esta fase se realiza la extracción de los datos de *Google analytics*, se analizan los formatos de los datos a ser descargados, se comprende el proceso de recolección de los datos de *Google analytics* de las páginas de la Universidad y qué significa cada uno de ellos.

4.1.3 Preparación de los datos:

En esta fase se procedió a descargar los datos de *Google analytics,* esto a través de la herramienta de *Looker Studio,* también se limpiaron los datos nulos, y se hizo una eliminación de los datos anómalos del conjunto de datos (*dataset).*

4.1.4 Modelado de los datos:

En esta fase se seleccionó la arquitectura *encoder/decoder*[10] y la red neuronal GRU a utilizar, como así también se utilizó un algoritmo de afinamiento de los modelos a valores óptimos.

4.1.5 Evaluación:

En esta fase se diseñaron y realizaron distintas pruebas a los modelos, de ser necesario se volvía a la fase de modelado a ejecutar nuevamente los algoritmos de calibración de los modelos.

4.1.6 Despliegue:

Los últimos modelos obtenidos son funcionales con los datos de GA4, es decir, se tienen modelos que funcionan con los datos del entorno real más actual.

En esta última fase se realizan las conclusiones del proyecto de Pronóstico de Series de Tiempo de Tráfico web y se analizan las futuras líneas de investigación.

4.2 Herramientas

4.2.1 Python

Python según dice en su página oficial es un lenguaje de programación interpretado, soporta múltiples paradigmas como pudiera ser programación funcional o programación orientada a objetos, Python es un lenguaje multipropósito, es decir, sirve para solucionar distintos tipos de problemas[22], para este trabajo se utilizaron las versiones de Python 3.7.x y 3.10.x.

4.2.2 Google Drive

Google ofrece una plataforma que permite almacenar archivos para uso personal a sus usuarios en la nube, este servicio además permite compartir archivos entre usuarios de Google Drive, como así también la edición de archivos en tiempo real con las aplicaciones que nos permite conectar[23].

4.2.3 Universal Analytics (UA)

Esta es una herramienta de Analytics, básicamente según el soporte de Google cuando se registran vistas u otro tipo de datos del tráfico web se recopilan en la herramienta; Universal Analytics es la generación anterior de Analytics, desde el 1 Julio del 2023 dejó de procesar datos[24].

4.2.4 Google Analytics 4 (GA4)

GA4 es una herramienta de Analytics, según el soporte de Google, GA4 recoge datos y los procesa de las páginas web y aplicaciones, mientras que UA no va a funcionar más GA4 es la nueva generación, GA4 por su parte recompila datos desde antes de la fecha en la que UA dejo de funcionar[24].

4.2.5 Looker Studio

Según la documentación de Google es una herramienta para realizar informes y visualizar los datos, esta herramienta permite la conexión de fuente de datos propios de los usuarios, como pudiera ser Google Analytics[24], también se pueden exportar

datos de tablas de informes de Looker Studio en archivos de valores separados por coma CSV (*Comma Separated Values*).

4.2.6 Microsoft Word

Microsoft Word es un procesador de texto, este software es de Microsoft, sirve para crear, editar, leer y escribir documentos de texto[25].

4.2.7 Microsoft Excel

Microsoft Excel es un software para poder crear, leer y editar planillas de cálculos, este software es de Microsoft[25].

4.2.8 Numpy

Según su documentación oficial es una biblioteca, es un estándar para trabajar con datos numéricos en Python, la librería posee estructuras de datos como Arrays, además posee funciones para realizar cálculos matemáticos con dichos Arrays [26].

4.2.9 Tensorflow

Es una biblioteca según su documentación oficial es de código abierto para el aprendizaje automático[27].

4.2.10 Keras

Es una biblioteca que sirve de Interfaz de Programación de Aplicaciones-API (*Aplication Programing Interface*) de alto nivel de Tensorflow, actualmente Keras se encuentra dentro de la biblioteca Tensorflow y es una API de alto nivel oficial de Tensorflow[27].

4.2.11 Pandas

Es una librería de Python para el análisis y manipulación de datos, posee el objeto *dataframe*, también funciones para manipular los datos del *dataframe* y funciones para leer y escribir archivos CSV [28].

4.2.12 Matplotlib

Matplotlib según su página oficial es una librería para la creación de gráficas, animaciones y distintas formas de visualización de datos en Python[29].

4.2.13 Google Colaboratory

Según su página oficial Colaboratory es un producto de Google Research que permite a cualquier usuario escribir y ejecutar código de Python en la nube. Este producto se puede conectar con Google Drive, para de esta manera tener acceso y poder leer y escribir los archivos en Google Drive[30].

4.2.14 Git

Git es un sistema de control de versiones, sirve para controlar las versiones de un proyecto, un proyecto es un repositorio de Git, un proyecto comprende todo el conjunto de carpetas y archivos al que se le realiza el control de versiones, de manera que se guardan los cambios y también se pueden recuperar las versiones anteriores del proyecto[31].

4.2.15 Hiperparametros (*Hyperparameters*)

Son las variables que se utilizaron durante el entrenamiento del modelo, como así también pueden definir la topología del modelo, estas variables tienen un impacto en el rendimiento del modelo[32].

4.2.16 Época (*Epoch)*

Según la documentación oficial de Keras una época *(Epoch)*, es cuando el modelo pasa por todo el conjunto de datos[33], en Keras, se permite agregar funciones al final de una época para guardar o imprimir información, como por ejemplo guardar el modelo, de forma que si se ejecutan x cantidad de épocas, se pueda recuperar el modelo con mejor rendimiento de la mejor época.

4.2.17 Trial

Según su documentación oficial, un trial es una prueba, donde cada trial tiene x cantidad de épocas, en un trial se evalúa un conjunto de valores de hyperparámetros, es decir, en cada trial se evalúa una configuración diferente del modelo[34].

4.2.18 Unidad Recurrente con Puerta – GRU (*Gate Recurrent Unit*)

Es una red neuronal creada por K. Cho en 2014, se describe en detalle en [10].

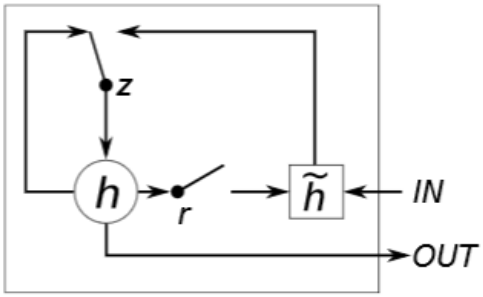

Figura 2: Red Neuronal GRU[10].

Donde r es la compuerta de reinicio (*reset gate)* y z es la compuerta de actualización (*update gate*) y h' es el candidato a activación y h es la activación[10].

La siguiente fórmula (4) es para cada j-esimo valor de la compuerta de reset.

$$r_j = \sigma([W_r x]_j + [U_r h_{(t-1)}]_j) \quad (4)$$

En la fórmula (4) σ es la función sigmoidea, W y U son matrices que aprenden, $h_{(t-1)}$ es el estado oculto anterior y x es la entrada

La siguiente fórmula (5) es para cada j-ésimo valor de la compuerta de actualización.

$$z_j = \sigma([W_z x]_j + [U_z h_{(t-1)}]_j) \quad (5)$$

En la fórmula (5) σ es la función sigmoidea W y U son matrices que aprenden, $h_{(t-1)}$ es el estado oculto anterior y x es la entrada

La siguiente fórmula (6) es para cada j-ésimo valor de activación, o también llamado estado oculto.

$$h_j^{(t)} = z_j h_j^{(t-1)} + \left(1 - z_j\right) h'^{(t)}_j \tag{6}$$

En la fórmula (6) $h'^{(t)}_j$ es el candidato a activación y se define en la siguiente fórmula para cada j-esimo valor.

$$h'^{(t)}_j = \phi\left([Wx]_j + \left[U\left(r \odot h_{(t-1)}\right)\right]_j\right) \tag{7}$$

En la fórmula (7) ϕ es la función de tangente hiperbólica y $\odot$ es el producto de Hadamard, W y U son matrices que aprenden, $h_{(t-1)}$ es el estado oculto anterior y x es la entrada[10].

4.2.19 Arquitectura Codificador-Decodificador (*Encoder-Decoder*)

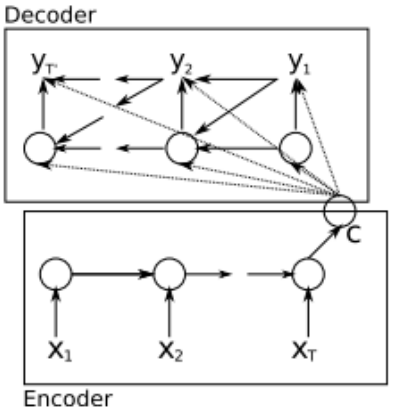

Figura 3: Arquitectura *encoder-decoder* GRU[10].

La arquitectura Codificador-Decodificador funciona de la siguiente manera, previo a utilizar la arquitectura, se aplica un preprocesado a los datos de Google Analytics que se explica en 4.3.2 Preparación de los datos, a continuación, en el codificador se ingresan los datos, el codificador es una capa de GRU que lee los datos en orden, por cada valor representado en la figura 3 como Xi, aprende actualizando su estado oculto, por otro lado el decodificador también es una capa de GRU, esta capa se encarga de aprender a predecir el siguiente valor de la secuencia, en el caso de este proyecto pronosticar las vistas de los sitios institucionales, en la figura 3 se puede observar el vector c, el cual contiene los estados ocultos del codificador y se ingresa dicho vector en la entrada del decodificador[10].

4.2.20 Algoritmo *Hyperband*

Es un algoritmo que se usa para el afinamiento de los hiperparametros, este algoritmo ejecuta el algoritmo de *Successive Halving*[35] variando como entrada n y r, siendo n el número de configuraciones y r el número de recursos, como ser número de épocas, para de esta manera encontrar la mejor configuración de hiperparametros[35], a

diferencia de la búsqueda aleatoria[36] y la Optimización bayesiana[37] el algoritmo Hyperband no tiene como dato de entrada un número máximo de trials, solo los valores de R (épocas) y n el cual por defecto es 3, también se menciona que Hyperband es 5 a 30 veces más rápido que los algoritmos de Optimización Bayesiana[35].

4.2.21 Detención Temprana (*Early Stopping)*

La detención temprana es una configuración que sirve para cortar el entrenamiento de manera temprana[38], esto es, se configura de manera que si el rendimiento no mejora en un número x de épocas se detiene el trial. Por ejemplo si se configura un modelo para que entrene 150 épocas pero la detención temprana está configurada en 20, y al entrenar el modelo no mejora el rendimiento desde la época 40 a la 60, este se detiene y no continúa el entrenamiento.

En el documento científico[39] se menciona el uso de una detención temprana de 10, además en este otro documento científico[38] se menciona el uso de la detención temprana para evitar el sobre entrenamiento.

4.2.22 Github

Github según su página oficial es una plataforma que hospeda repositorios de Git, ofreciendo además el servicio de que los equipos de desarrollo puedan trabajar en grupo[31].

4.3 Comprensión de los datos

Esta fase inicia con recolectar los datos y continúa con actividades para familiarizarse con los datos, identificar problemas de calidad de los datos, descubrir los primeros conocimientos de los datos[40].

4.3.1 Recolección de datos iniciales

Se tienen acceso a datos históricos de las siguientes 3 páginas webs:

- Sitio Institucional de la U.Na.M. disponible en https://unam.edu.ar/
- Sitio de la Editorial Universitaria disponible en https://editorial.unam.edu.ar/
- Portal de acceso al contenido generado por la radio y la televisión de la Universidad disponible en https://transmedia.unam.edu.ar/

De las cuales tenemos una mayor cantidad de datos registrados por UA y una menor cantidad de datos expresados en días registrados en GA4, las siguientes tablas explican los datos correspondientes por fecha que se tienen de dichas páginas web.

Página Web	Desde	Hasta
https://unam.edu.ar/	28-05-2018	04-07-2023
https://editorial.unam.edu.ar/	08-10-2018	04-10-2023
https://transmedia.unam.edu.ar/	22-08-2018	01-06-2023

Tabla 1 de datos de *Universal Analytics*

Página Web	Desde	Hasta
https://unam.edu.ar/	14-06-2022	23-11-2023
https://editorial.unam.edu.ar/	14-06-2022	24-01-2024
https://transmedia.unam.edu.ar/	14-06-2022	01-01-2024

Tabla 2 de datos de *Google Analytics* 4

Para poder descargar los datos se procedió a conectar *Google Analytics* como fuente de datos de *Looker Studio*, de esta manera se pueden pasar los datos a tablas en *Looker Studio* y posteriormente descargar los datos en archivos CSV.

4.3.2 Preparación de los datos

Identificando algunos problemas de calidad de los datos la página web https://unam.edu.ar/ desde noviembre del 2023 no está capturando el tráfico correctamente en GA4, por lo que se procede a descartar este conjunto de datos.

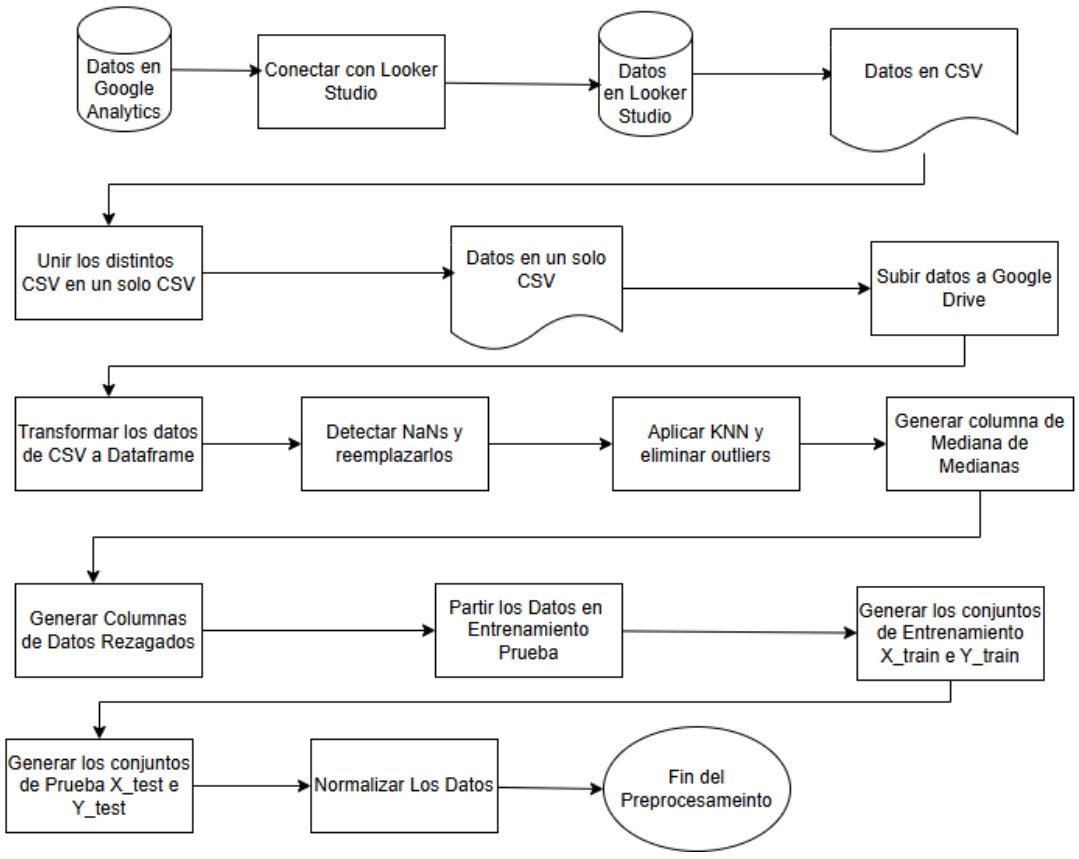

Figura 4: Diagrama de Flujo del Pre procesamiento

Lo siguiente que se procede a realizar, comenzando por la página web https://transmedia.unam.edu.ar/ se descargan los datos en archivos CSV del *Looker*

Studio de UA, son 5 archivos CSV que se procede a unificar en un solo archivo a través de código en Python.

Se descargan los siguientes datos en 5 archivos CSV:

Vistas por Sistemas Operativos, Vistas por Agrupación de canales predeterminados, número de vistas de página, Usuarios Nuevos, Usuarios, Numero de Sesiones Por usuario, Sesiones, Vistas por Categoría de dispositivo, Vistas por País

A continuación se transforman los datos obtenidos de los archivos CSV de Looker Studio para obtener las columnas del *dataframe*.

De vistas por sistema operativo salieron las siguientes columnas:

Windows, Android, Linux, Macintosh, Sistema operativo (not set), iOS, Windows Phone, Chrome OS, Tizen, Sistema operativo promedio, Sistema operativo std,

mediana Sistema operativo.

Siendo Sistema operativo (not set) donde se registran las vistas los sistemas operativos que UA no pudo detectar, Sistema operativo promedio es el promedio de las otras columnas de sistemas operativos, Sistema operativo std el desvió estándar de las columnas de Sistema operativo y el resto son vistas de sistemas operativos conocidos, por ejemplo la columna Windows tendrá en cada fila datos de las vistas de los usuarios que accedieron a la página web a través de un dispositivo cuyo sistema operativo es Windows.

De vistas por Agrupación de canales predeterminados salieron las siguientes columnas:

Organic Search, Social, Direct**,** Referral, Agrupación de canales predeterminada promedio, Agrupación de canales predeterminada std, mediana Agrupación de canales predeterminada

Donde Organic Search, Social, Direct y Referral son los canales predeterminados, es decir, las fuentes de tráficos más comunes[41], Agrupación de canales predeterminada promedio es el promedio de los canales predeterminados, Agrupación de canales predeterminada std es el desvío estándar y mediana Agrupación de canales predeterminada es la mediana de las columnas de agrupación de canales predeterminados.

De vistas por Categoría de dispositivo salieron las siguientes columnas:

Mobile, Desktop, Tablet, Categoría de dispositivo promedio, Categoría de dispositivo std., mediana Dispositivos.

Donde Mobile, Desktop y Tablet son la Categoría de dispositivo, es decir, el tipo de dispositivo que tenía el usuario cuando se registró la visita en la página web[42], además Categoría de dispositivo promedio es como su nombre indica el promedio de vistas por Categoría de dispositivo, Categoría de dispositivo std es el desvió estándar y mediana Dispositivos es la mediana de las columnas de dispositivos.

Las siguientes son tal cual del CSV no se necesitó mayor trabajo de transformación:

Número de vistas de página, Usuarios Nuevos, Usuarios, Número de Sesiones Por usuario, Sesiones.

Es decir una columna de Usuarios Nuevos del CSV se pasó al *dataframe* como una columna de Usuarios Nuevos sin mayores complicaciones. De vistas por País se procedió a dividir en 2 columnas, las vistas que son de Argentina y las que son de otros países. El proceso se continuó usando Google Colaboratory en Google Drive.

4.3.4 Detección de Anomalías

Finalmente se realizó una limpieza de los datos anómalos (*outliers)* detectados usando los k vecinos más cercanos - KNN (*K Nearest Neighbors*), donde se tomaron los 7 vecinos más cercanos, se consideran más probable que sean *outliers* aquellos datos del *dataset* que estén más alejados de sus vecinos más cercanos[43].

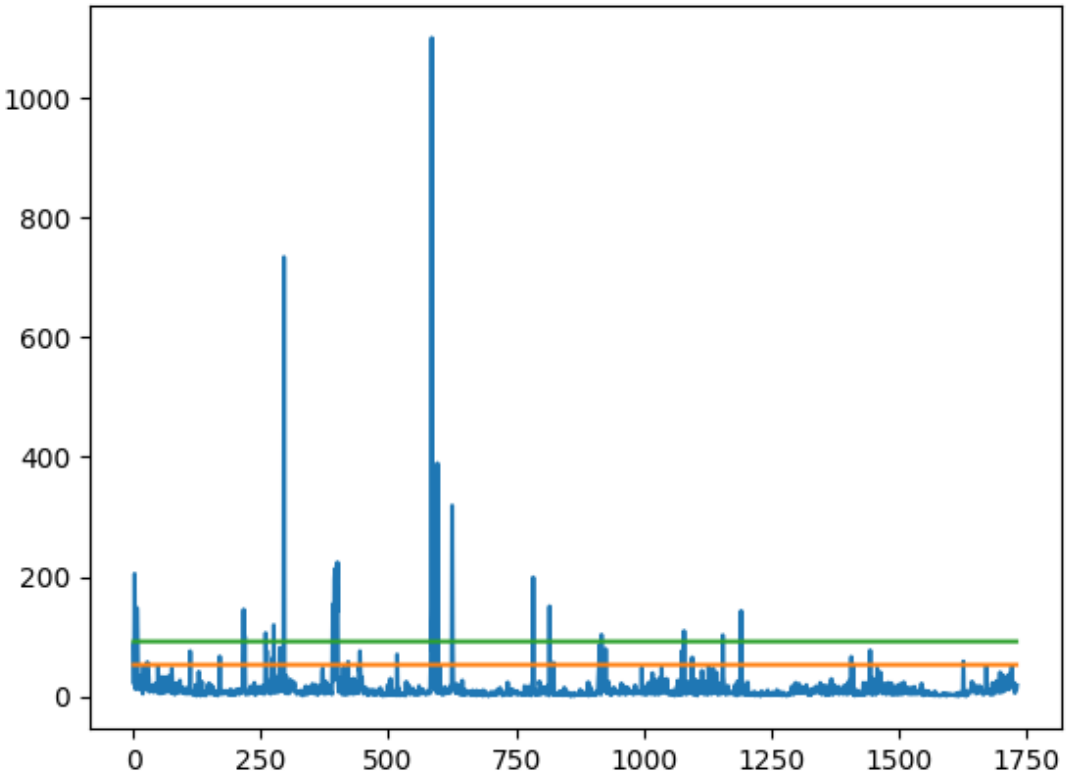

Figura 5: Distancia Promedio de los 7 Vecinos más cercanos para cada fila del *dataframe* de TUM Transmedia.

En la figura anterior muestra la distancia Promedio de los 7 vecinos más cercanos para cada fila del conjunto de datos, de esta manera podemos considerar los que se encuentren más lejanos de sus 7 vecinos más cercanos como un *outlier*, en este caso se quitaron las filas del conjunto de datos que estén por encima de 500, finalizando así el proceso de limpieza de los datos de TUM Transmedia. En la figura 5 se puede observar una línea naranja y una línea verde, estas son, la media de las distancias sumado un desvío estándar y la media de las distancias sumada con 2 desvíos estándar.

En el documento científico [44] que trata sobre *outliers* en series temporales se menciona que a los *outliers* normalmente se los elimina o se los reemplaza con algún valor, en este caso se decidió eliminarlos del conjunto de datos.

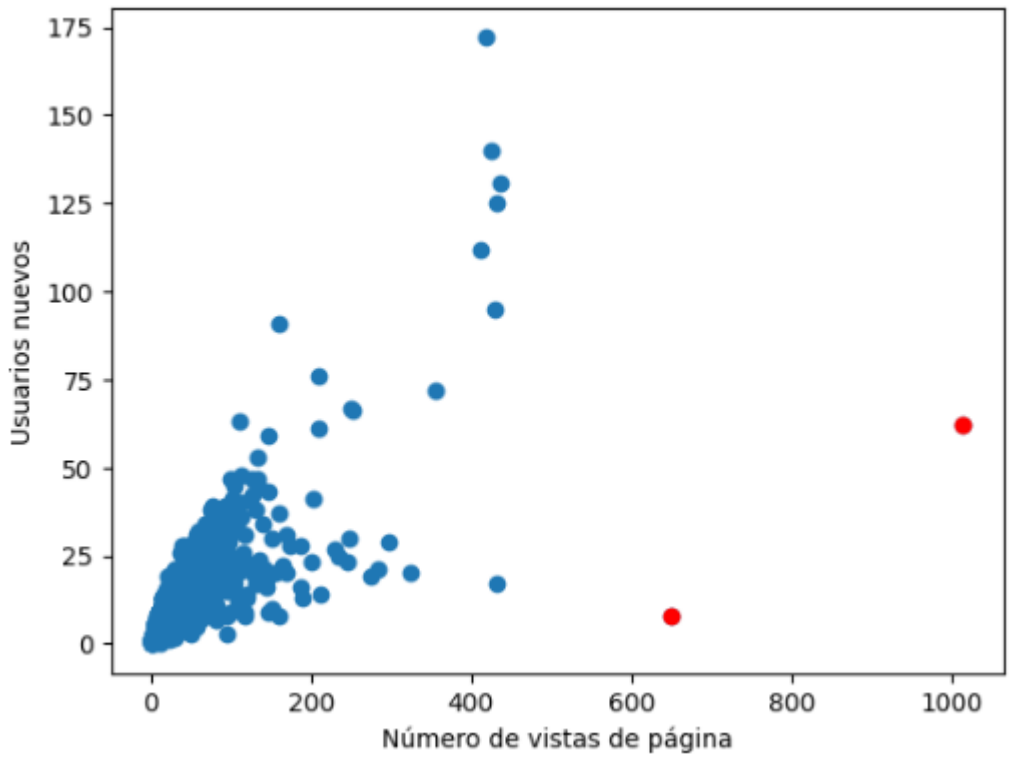

Figura 6: Gráfico de dispersión de Número de vistas de página y Usuarios nuevos del *dataframe* de TUM Transmedia.

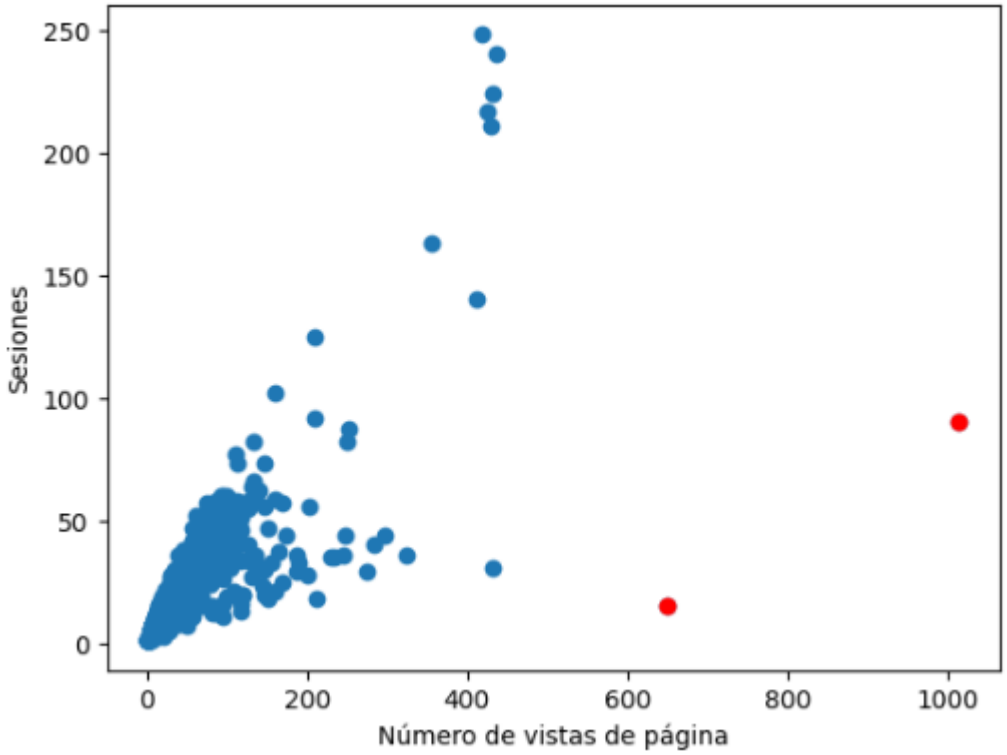

Figura 7: Gráfico de dispersión de Número de vistas de página y Sesiones del *dataframe* de TUM Transmedia.

En las figuras 6 y 7 en rojo aparecen los *outliers*, se puede ver que de esta manera se considera que el valor probado con los 7 vecinos más cercanos para el KNN es el correcto.

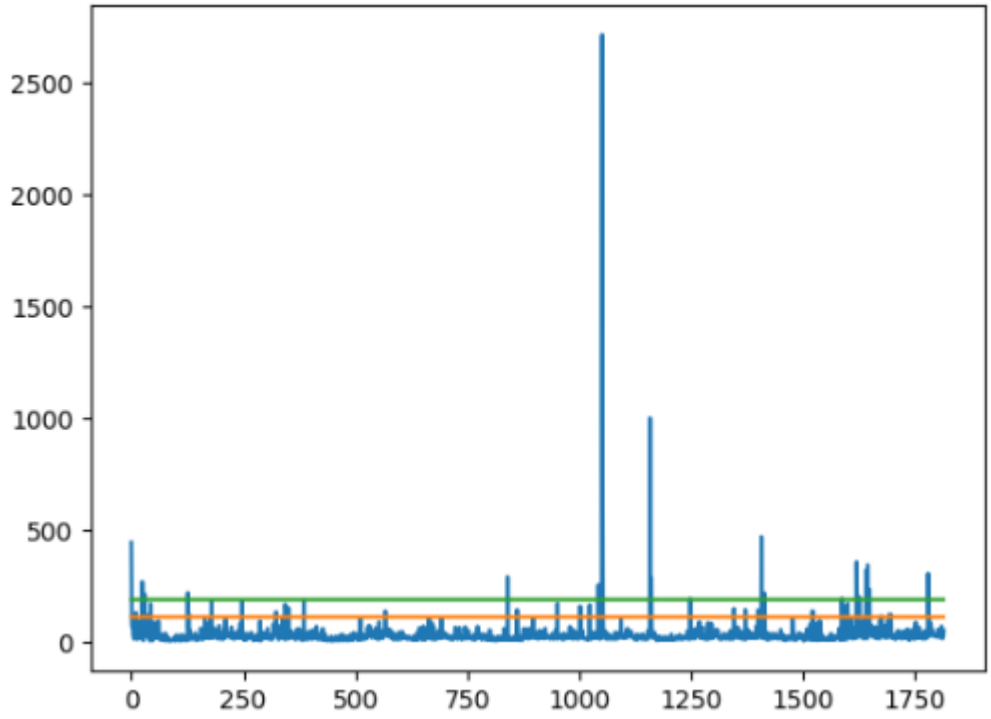

Figura 8: Distancia Promedio de los 7 Vecinos más cercanos para cada fila del *dataframe* de Editorial Universitaria.

Mediante la observación de la Figura 8 se tomaron como *Outliers* de Editorial Universitaria a partir de una distancia 500 mayor que 500 de media de sus 7 vecinos más cercanos.

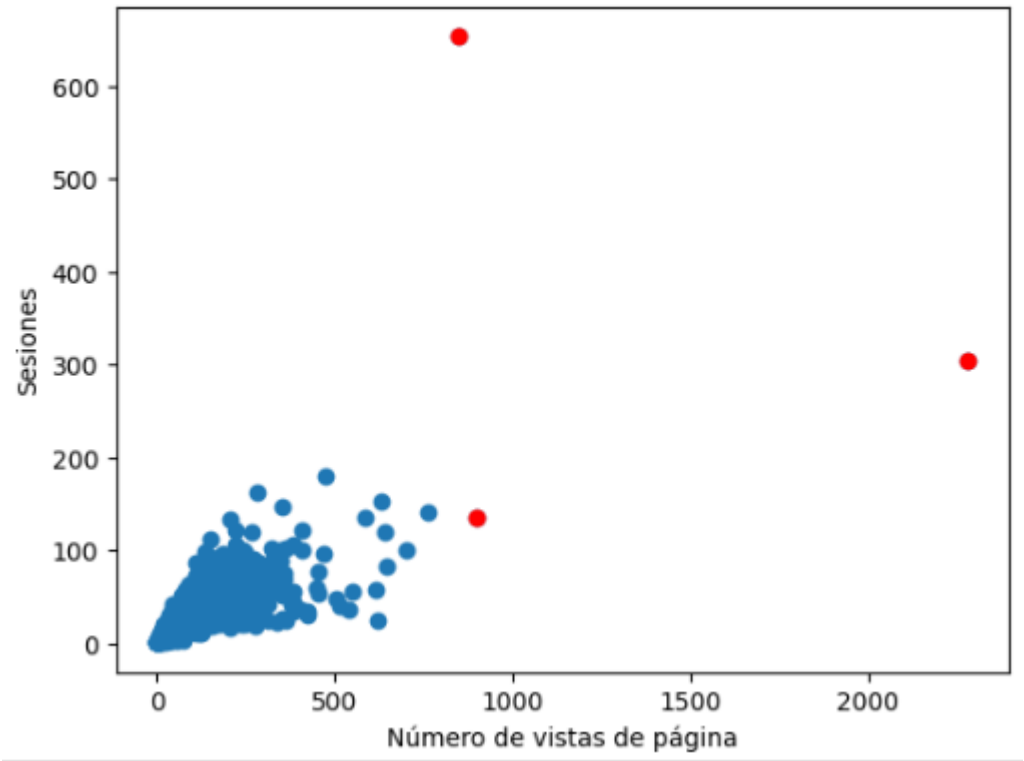

Figura 9: Gráfico de dispersión de Número de vistas de página y Sesiones del *dataframe* de Editorial Universitaria.

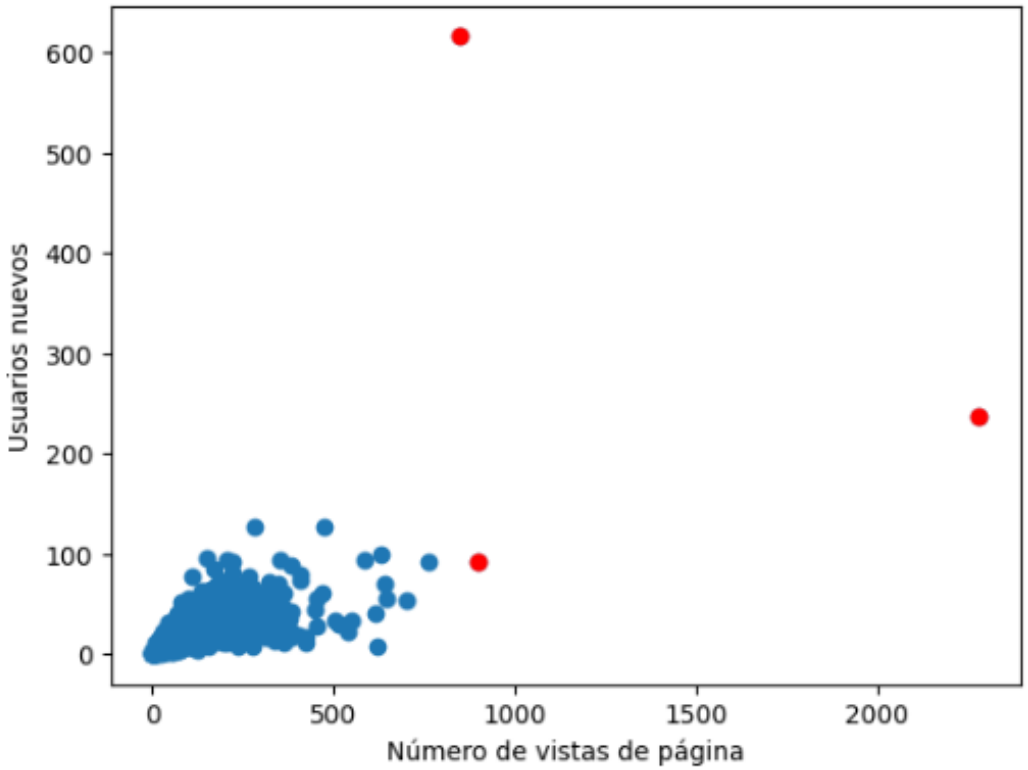

Figura 10: Gráfico de dispersión de Número de vistas de página y Usuarios nuevos del dataframe de Editorial Universitaria.

En las figuras 9 y 10 se pueden observarla detección de outliers en el conjunto de datos de Editorial Universitaria. Luego, se procede a agregar las columnas mediana de medianas de acuerdo al documento [1] donde toman las medianas rodantes de 6, 12, 18, 30, 48, 78, 126, 203, 329 días de la columna Número de vistas de página y de ahí se procede a calcular la mediana de medianas, según lo explicado en [1].

También dado que en el código original [45] agrega valores rezagados, es decir valores anteriores los agrega como columnas, entonces se agregan valores de la columna a predecir Número de vistas de página, se agregaron valores rezagados de 3, 6, 9,12 meses como las columnas:

lag_90, lag_180, lag_270, lag_360

Los nombres de las columnas vienen del inglés *lag* que se utiliza para referirse al rezago, o también se puede traducir como retraso, el número en días correspondiente al mes de retraso tomando 1 mes como 30 días aplicando esa regla para hacer las columnas de valores retrasados. A continuación se procedió a confirmar la existencia de NaN (*Not an number*) de las columnas de datos rezagados, y reemplazarlos por su promedio.

Quedando como resultado del preprocesado las siguientes columnas de la Television Universitaria Misionera - TUM transmedia, que fueron las mismas columnas que se tomaron en cuenta para las otras páginas web de la institución.

Listado de columnas del *Dataframe*:

Número de vistas de página, Usuarios nuevos, Usuarios ,

Número de sesiones por usuario, Sesiones, Argentina,

Otros Países , mobile , desktop , tablet ,

Categoría de dispositivo promedio, Categoría de dispositivo std ,

Windows , Android , Linux , Macintosh ,

Sistema operativo (not set) , iOS , Windows Phone , Chrome OS ,

Tizen , Sistema operativo promedio , Sistema operativo std ,

Organic Search , Social , Direct , Referral ,

Agrupación de canales predeterminada promedio ,

Agrupación de canales predeterminada std , mediana_de_medianas ,

lag_90,lag_180, lag_270, lag_360, mediana Dispositivos ,

mediana Sistema operativo,

mediana Agrupación de canales predeterminada.

Siendo un total de 37 columnas, por supuesto que desde *Google Analytics* se registran más datos, sin embargo debido al cambio de UA a GA4[46] solo se

eligieron los datos que a nivel tanto conceptual como de medición, fueran más parecidos entre sí.

4.3.5 Partición de Los datos

Los datos se parten en conjuntos de entrenamiento y prueba, al igual que en [7] partimos los datos en 80-20, es decir, donde el 80% de los datos se utiliza para entrenar el modelo, y el 20% para probar el rendimiento del mismo.

En el caso de los datos de GA4 dado que hay una menor cantidad y se necesita al menos 120 días de datos en el conjunto de prueba, se parte en 70-30, 70% para entrenamiento y 30% para prueba. A su vez los datos se parten en muestras de a 120, 60 días de datos de entrada del modelo y 60 días de datos que se comparan con la salida del modelo. Las muestras de 60 días de entrada usados en entrenamiento son llamadas X_train mientras que para prueba son llamadas X_test. Las muestras de 60 días de salida usadas en entrenamiento son llamadas Y_train mientras que para prueba son llamadas Y_test[47].

4.3.6 Normalización

Para poder ingresar los datos a la red neuronal se normalizan los datos, primero se aplica log1p junto con la normalización z-score [47].

$$x1 = \log(1 + x) \tag{8}$$

En la fórmula 8 se ve como se aplica log1p donde x es el dato y x1 es el dato aplicado log1p.

$$z = \frac{(x-\mu)}{\sigma} \tag{9}$$

Luego se aplica la normalización z-score esto para entrenamiento, de acuerdo al código original[45], como se ve en la fórmula 9 el resultado normalizado es el valor del dato x menos la media μ dividido el desvió estándar σ, otro tipo de normalización que se

aplica es solo z-score esto para otras pruebas en los que el modelo tuviera como función de pérdida Huber, en el plan de pruebas se explica con más detalle.

4.4 Modelos

Tecnologías	**Red N.**	**Resultados según Artículos científicos**	**Finalidad**	**Aplica**
MLP	SI	-	Clasificación, predicción	No
LSTM	SI	Combinándolo con otras tecnologías se tiene mejor resultado que usarlo individualmente.	Predictivo, hallar patrones	No
ARIMA	No	Buenos	No es redes neuronales, es estadística predictiva (predicción de series temporales)	No
GRU	SI	El modelo ganador de Kaggle utiliza GRU.	Sistemas predictivos basados en información secuencial.	No
Bi-LSTM	SI	Mejor rendimiento comparándolo en el M3-Competition	Reconocimiento y clasificación	No
GAN	SI	Regular, no hay ganancia comparado con ARIMA	Creación artificial	No
RNN seq2seq	SI	Buenos	Modelo ganador del concurso de Kaggle de tráfico web 2017	Si
NARX -QPSO	SI	Buenos	Predecir el siguiente valor de la señal de entrada	Si

Tecnologías	**Red N.**	**Resultados según Artículos científicos**	**Finalidad**	**Aplica**
Prophet-LGBM	SI	Buenos Mejores resultados que modelos individuales	Combinación ligthGBM y Prophet para predecir tráfico web	Si

Tabla 3 Comparación de tecnologías

4.4.1 Selección de Modelos

En la Tabla 3 se presenta una comparación de las tecnologías, de ahí se procede a seleccionar las tecnologías que mejor se adaptan al problema, son las consideradas a ser usadas para la solución del problema; el criterio de selección es el siguiente, descartar las tecnologías que no usan redes neuronales, otras son tecnologías cuya finalidad no es la del pronóstico de series de tiempo, también se toma en cuenta la disponibilidad de los datos y código usado en los documentos científicos y por último, de acuerdo a los artículos científicos presentados en el marco teórico, algunas tecnologías no tienen buen rendimiento o el rendimiento suficiente para ser tomado en cuenta.

Entre los modelos que quedan seleccionados están RNN seq2seq que usa GRU, NARX –QPSO y Prophet-LGBM de estos modelos Prophet-LGBM si bien se mencionan las fechas de los datos que utilizaron no está disponible el *dataset* para compararlo con los datos de la problemática de esta tesis por lo tanto Prophet-LGBM se descarta como posible tecnología para la solución de la problemática de esta tesis, mientras que por otro lado tanto RNN seq2seq y NARX –QPSO ambos utilizan el mismo conjunto de datos el cual está disponible online para que cualquiera lo pueda descargar, de esta manera y dado que RNN seq2seq es el ganador del concurso de Kaggle[11] y por lo tanto se encuentra más información de la misma en la página de Kaggle se opta por utilizar esta tecnología.

En cuanto a la métrica seleccionada para la medición del desempeño de los modelos, se eligió la métrica MAE que se utilizó en [7], también la métrica SMAPE debido a que es la misma con la que se evaluó el modelo RNN seq2seq, que es el ganador del concurso de Kaggle[11].

4.5 Desarrollo

4.5.1 Plan de pruebas

Durante el desarrollo de la tesis sucedió que desde el primero de Julio de 2023 la herramienta Universal Analytics (UA) dejó de registrar datos de las páginas web, siendo Google Analytics 4 (GA4) la nueva generación que continuará registrando los datos del tráfico web de las páginas de la Universidad, por lo tanto en el plan de pruebas se tendrán en cuenta estos cambios, teniendo como prioridad los datos de UA ya que es de donde se pueden obtener mayor cantidad de datos históricos para entrenar los modelos, sin embargo, se plantearon pruebas con los datos de GA4, a modo de poder ver el rendimiento de los modelos con los datos de la nueva generación.

El plan de prueba consiste en generar varios modelos GRU con la arquitectura seq2seq, y se procederá a medir usando la métricas MAE,RMSE y SMAPE, en primera instancia se utilizarán los datos de TUM Transmedia de UA, posteriormente se utilizará el mejor modelo resultante en los datos de TUM Transmedia de GA4, también se probará en los datos de Editorial Universitaria UA y el mejor modelo resultante de Editorial Universitaria UA se utilizará para generar un modelo y probarlo con los datos de Editorial Universitaria GA4.

Como función de pérdida de los modelos se utilizó la función de pérdida de Huber[9] utilizado en este documento científico [7] y también se utilizó MAE como función de pérdida, esto a nivel entrenamiento y prueba, ya una vez los modelos están entrenados

se midió el rendimiento usando las métricas MAE,RMSE y SMAPE sobre los datos desnormalizados.

Un SMAPE aceptable de acuerdo a [1] es alrededor o por debajo de 0.349, mientras que un MAE aceptable de acuerdo a [7] es por debajo de 30, estos fueron los valores a tener en cuenta a la hora de medir el rendimiento de los modelos.

4.5.2 Construcción de los Modelos

En esta sección se detallan las características de los modelos generados, básicamente en primera instancia se generaron modelos utilizando el algoritmo de ajuste de hiperparametros, siendo el algoritmo utilizado el algoritmo Hyperband. En las pruebas realizadas se busca obtener los mejores valores de los parámetros e hiperparametros del modelo.

Se utilizó la arquitectura Secuencia a Secuencia con Redes Neuronales - RNN seq2seq[48] *"Sequence to sequence learning with neural networks"*, donde el *encoder* se encarga de leer la secuencia de entrada y actualizar sus estados ocultos, el ultimo estado oculto es la salida del *encoder* también llamado vector de contexto, se usa una capa *RepeatVector* de Keras[33], esta capa se encarga de repetir el vector de contexto la cantidad de días a futuro que vamos a pronosticar (en mi caso es 60), la salida del *RepeatVector* se utiliza como entrada del *decoder*. El último estado oculto del *encoder* aparte de ser usado en el *RepeatVector* también es usado como estado inicial del *decoder*. Finalmente se utilizó la capa *TimeDistributed* de Keras[33], la cual se le paso una capa densa[33], esta capa actúa aplicando una capa densa para cada paso del pronóstico. Por ejemplo si la salida del GRU del *decoder* tiene forma de (número de lotes, pasos a futuro, cantidad de neuronas) el *TimeDistributed* con la capa densa aplicaría una capa densa a cada uno de los pasos a futuro así obteniendo la salida con forma de (número de lotes, pasos a futuro).

4.5.3 Cantidad de Trials por Modelo

El algoritmo Hyperband se configuró con un factor de 3 (el cual es el valor por defecto) y con objetivo de 200 épocas ya que es el número utilizado en este documento científico [7], se utilizó un *early stopping* de 10 que se lo retiró en las pruebas que no fueran correspondientes al algoritmo Hyperband.

4.5.4 Hiperparametros que se Afinaron

Nombre	Valores posibles	Definición
Unidades	128-512 con un paso de 32	Es la cantidad de unidades de GRU por capa.
Ratio de Aprendizaje	Entre 1e-4 y 1e-2	Es el ratio de aprendizaje, *Learning Rate* en inglés Se utiliza el Optimizador Adam[49].
Batch size	64 a 256 con un paso de 32	Tamaño de lote a tomar, normalmente, esto es para dividir la cantidad de muestras a tomar, ya que no se puede correr todo el conjunto de datos entero de una sola vez.
Capas	1-2	Es el número de capas de GRU de la red neuronal.

Tabla 4 Hiperparametros que se afinaron.

En la Tabla 4 se puede ver los hiperparametros que se calcularon utilizando el algoritmo *Hiperband*, estos rangos de valores que se toman para hacer el afinamiento de hiperparametros, muchos de ellos son elegidos semejantes a los que se utilizaron en el código original del ganador del concurso de Kaggle[45].

Un concepto muy importante que no se puede determinar el cálculo con la librería Keras es el N-Días el cual es básicamente la cantidad de días hacia atrás que toma como datos de entrada el *encoder*, ya que este valor no se puede precisar por la librería Keras, se toma como entrada 60 días, ya que según lo mencionado por el ganador del concurso de Kaggle [11] es un numero con el cual se puede obtener resultados aceptables con el objetivo de pronosticar 60 días a futuro.

Otra variable muy importante que se puede determinar con la librería Keras, pero se hizo manualmente fue el número de capas, dado que el algoritmo de cálculo de hiperparametros genera cientos de modelos y al aumentar el número de capas también aumenta el tamaño de almacenamiento requerido para guardar los modelos, como también se requiere más memoria RAM y GPU, por lo que rápidamente me quedaría sin recursos suficientes de Google Colab y Google Drive, debido a esa limitante se afinó manualmente el número de capas.

El código utilizado se puede encontrar en:

https://github.com/EnriqueGautoSand/Tesis-Enrique-Gauto-Sand

La primera prueba que se realiza es la de correr el algoritmo Hyperband con los datos normalizados mediante log1p y z-score, con función de pérdida MAE.

#	Unidades	Capas	lr	Batch Size
1	224	1	0.0012	64

Tabla 5 Prueba 1 Parámetros encontrados por el Algoritmo de Afinamiento de Hiperparametros.

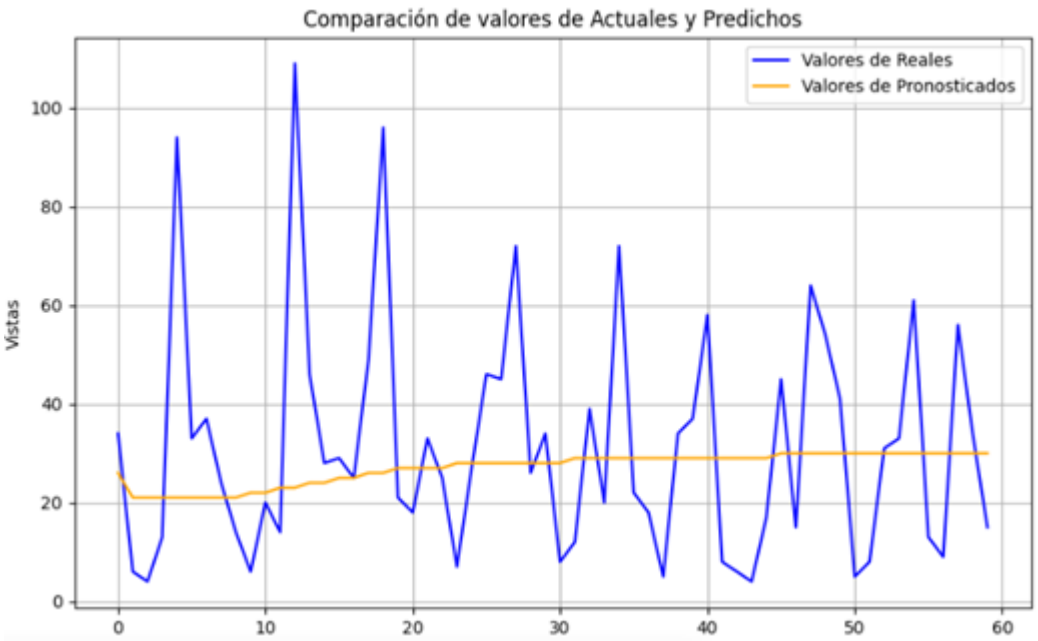

Figura 11: Gráfico de Los Valores Reales y los Valores Pronosticados de la salida del modelo de una parte del conjunto Y_test, de la prueba 1.

El modelo de la primera prueba como se puede ver tiende a pronosticar una media.

#	Unidades	Capas	lr	Batch Size
2	192	1	0.0018	128

Tabla 6 Prueba 2 Parámetros encontrados por el Algoritmo de Afinamiento de Hiperparametros.

La segunda prueba que se realizó fue la de correr el algoritmo Hyperband con los datos normalizados mediante z-score, con función de pérdida Huber. Se puede observar en la tabla 6 los parámetros encontrados como resultado de la prueba 2.

Comparaciones de rendimientos las pruebas 1 y 2

#	MAE	SMAPE	RMSE
1	11.126461988304094	0.5977127357014901	16.491247900324023
2	13.274267093758834	0.6972292246352332	17.523084885859184

Tabla 7 Comparaciones de Rendimientos entre Pruebas.

En la tabla 7 se puede ver que la prueba 1 tiene mejores rendimientos en todas las métricas, por lo tanto en las pruebas siguientes se seguirán utilizando los parámetros de la prueba 1.

4.5.5 Prueba de capas

Esta prueba consiste en agregar más capas al mejor modelo obtenido, se entrenó por 500 épocas con 2 capas de *encoder* y 2 capas de *decoder*.

Modelo	MAE	SMAPE	RMSE
1 Capa	11.126461988304094	0.5977127357014901	16.491247900324023
2 Capas	12.152266081871344	0.6348007961755141	17.790148975902234

Tabla 8 Comparaciones de Rendimientos entre Pruebas.

Como se puede ver en la tabla 8 el primer modelo sigue siendo el mejor modelo, en el código de GitHub del ganador del concurso de Kaggle[45], se hacen pruebas de 1 y 2 capas como máximo, por lo tanto no continuamos con las pruebas de capas, siendo la prueba de 2 capas la tercera prueba.

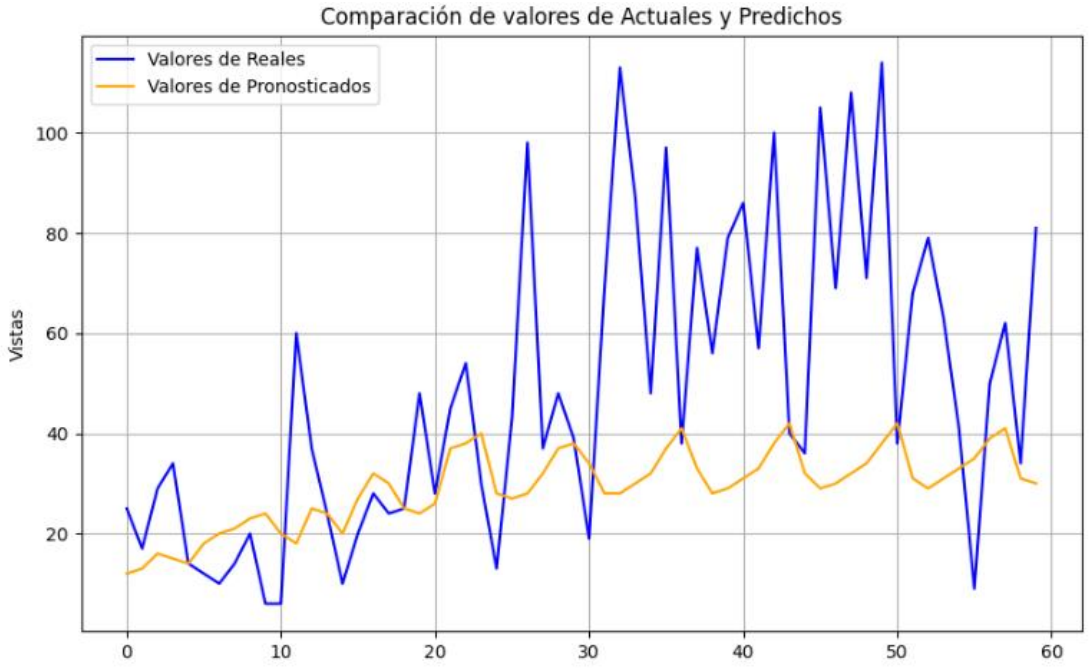

Figura 12: Gráfico de Los Valores Reales y los Valores Pronosticados de la salida del modelo de una parte de conjunto de prueba Y_test, de la prueba 3.

En el caso de la figura 12 se obtiene un modelo que al principio predice bien la tendencia y luego se aleja de la misma terminando en una media.

4.5.6 Pruebas Editorial Universitaria UA

Concluidas las pruebas en TUM Transmedia UA se seleccionó el modelo con mejor rendimiento y se pasaron a probar con los datos de la página web de Editorial U.Na.M.

Para utilizar el mejor modelo de TUM en Editorial UA se construyó el *dataset* con las mismas columnas que TUM Transmedia, y se utilizó ese *dataset* para pruebas posteriores de Editorial Universitaria UA.

#	Modelo	MAE	SMAPE	RMSE
4	El mejor modelo de TUM UA	258.66926229	1.1840436456	364.641053760

Tabla 9 Prueba 4

En la tabla 9 se ve el rendimiento del mejor modelo de TUM Transmedia UA aplicado y reentrenado por 500 épocas con los datos de Editorial Universitaria UA.

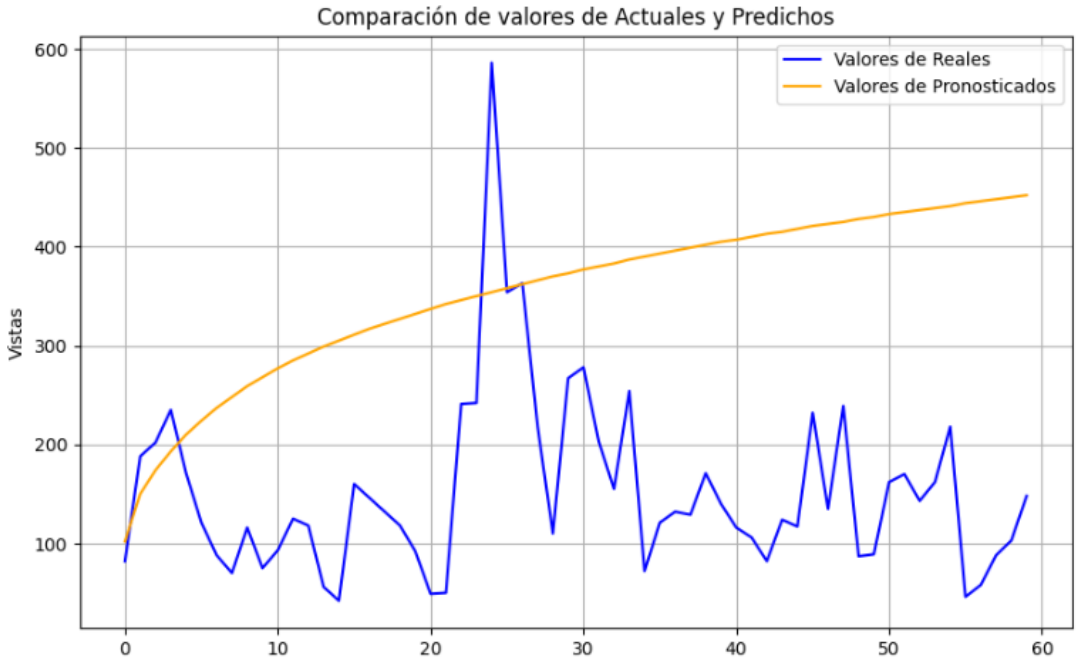

Figura 13: Gráfico de Los Valores Reales y los Valores Pronosticados de la salida del modelo de una parte del conjunto de prueba Y_test, de la prueba 4.

En la figura 13 se puede ver que los valores pronosticados por el modelo de la prueba 4 se alejan de los valores reales.

#	Unidades	Capas	lr	Batch Size
5	160	1	0.000133802 8856513705	192

Tabla 10 Prueba 5 de Editorial Universitaria de UA

Dado que la prueba 4 no tuvo un MAE por debajo de 30, decidí hacer una búsqueda de hiperparametros usando el algoritmo Hyperband, en la tabla 10 se puede ver los mejores parámetros obtenidos.

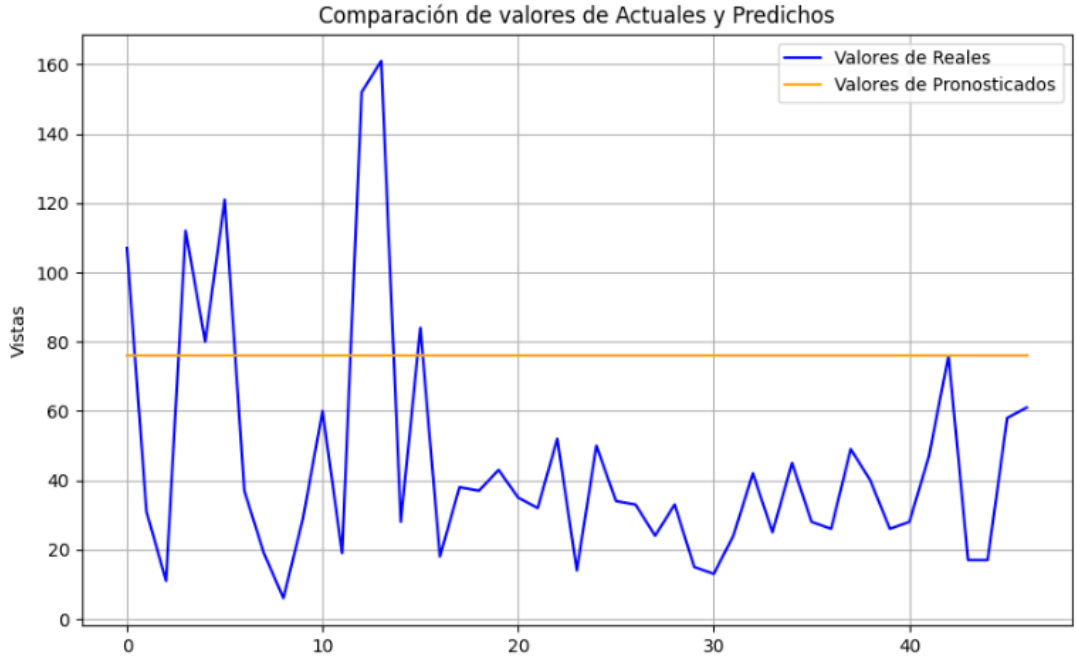

Figura 14: Gráfico de Los Valores Reales y los Valores Pronosticados de la salida del modelo de una parte del conjunto de prueba Y_test, de la prueba 5.

Como se puede observar en la figura 14 el modelo obtenido es capaz de pronosticar una media, de manera que se pueda estimar la media de las vistas a futuro.

#	MAE	SMAPE	RMSE
4	258.66926229	1.1840436456	364.641053760
5	81.53907103825136	0.609703658254883	130.59741930437121

Tabla 11 Comparaciones de Rendimientos entre Pruebas 4 y 5

Como se puede observar en la tabla 11 se pudo obtener un MAE de 81.53 no es un resultado esperado, ya que un buen resultado lo consideraríamos por debajo de 30.

4.5.7 Conclusiones Pruebas con datos de UA

Una vez finalizadas las pruebas tanto para Editorial Universitaria UA como para TUM Transmedia UA, procedemos a resumir los mejores resultados en la siguiente tabla:

Prueba #	Conjunto de Datos	MAE
1	TUM Transmedia UA	11.126461988304094
5	Editorial Universitaria UA	81.53907103825136

Tabla 12 Resumen Pruebas con Mejores Resultados con datos de UA.

En la tabla 12 se puede ver que el resultado en TUM Transmedia UA comparado con [7] fue bastante mejor, mientras que el resultado de Editorial Universitaria UA es un resultado intermedio ya que el mejor rendimiento está entre su mejor rendimiento [7] que fue un MAE de 27 y su peor mejor rendimiento que fue un MAE de 132.26, esto ya que en [7] tienen varios resultados, uno por cada idioma, aun así no lo considero un buen resultado en Editorial Universitaria UA debido a que esperaríamos obtener un MAE por debajo de 30.

4.5.8 Conversión de Modelos de UA a TUM Transmedia GA4

Se procedió a reentrenar por 500 épocas el mejor modelo de TUM Transmedia UA con los datos de TUM Transmedia GA4. Para utilizar el mejor modelo de TUM Transmedia UA en TUM Transmedia GA4 se construyó el dataset con las mismas columnas que TUM Transmedia UA, y se utilizó ese dataset para las pruebas posteriores en TUM Transmedia GA4.

#	Modelo	MAE	SMAPE	RMSE
6	El mejor modelo de TUM UA	26.5404255319	0.57887830853	34.6621485053

Tabla 13 Prueba Número 6

En la tabla 13 se puede ver el rendimiento del mejor modelo de TUM UA aplicado a TUM GA4, como se puede observar tiene un MAE por debajo de 30 con lo que se concluye que es un buen resultado y se decide no continuar realizando más pruebas en TUM GA4.

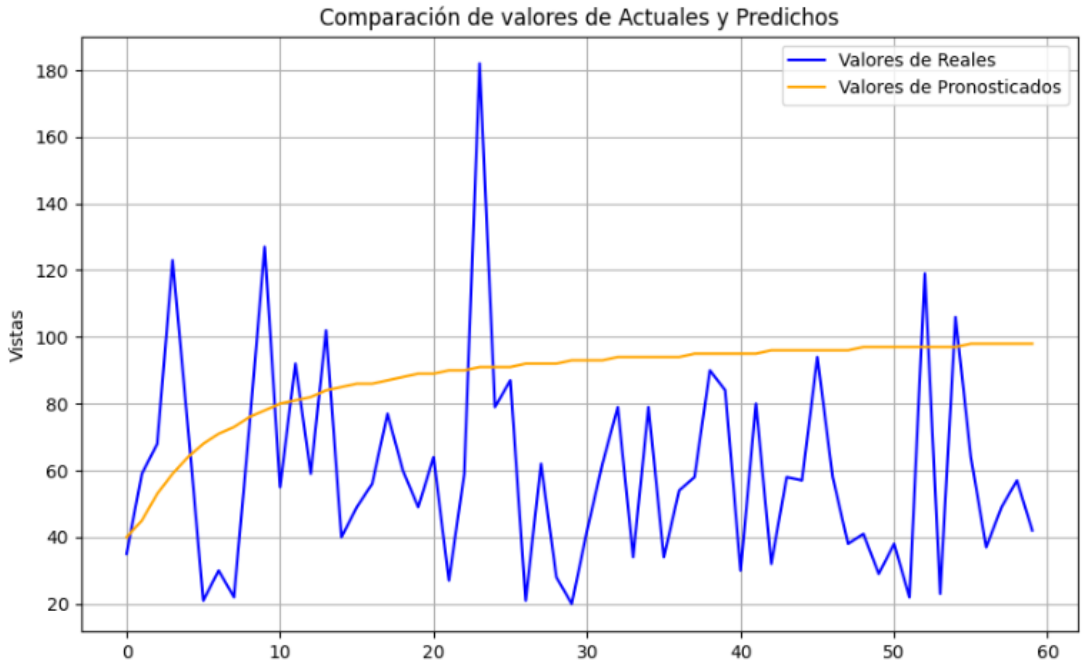

Figura 15: Gráfico de Los Valores Reales y los Valores Pronosticados de la salida del modelo de una parte del conjunto de prueba Y_test, de la prueba 6.

En la figura 15 se puede observar que el modelo es capaz de pronosticar tendencia, comenzando en valores bajos y subiendo hasta los valores medios de los picos.

4.5.9 Pruebas Modelos de Editorial Universitaria GA4

Se procedió a reentrenar por 500 épocas el mejor modelo de TUM Transmedia UA con los datos de Editorial Universitaria GA4 y también se reentrenó por 500 épocas el mejor modelo de Editorial Universitaria UA con los datos de Editorial Universitaria GA4.

#	Modelo	MAE	SMAPE	RMSE
7	El mejor modelo de TUM UA	89.944642857	0.9568072360	129.76339273
8	El mejor modelo de Editorial UA	71.077976190	0.6042800485	104.79610588

Tabla 14 Rendimientos de los Modelos de las Pruebas Número 7 y 8

La tabla 14 resume las 2 pruebas realizadas con los datos de Editorial Universitaria GA4, donde el mejor modelo es el que corresponde a la prueba 8, porque con las 3 métricas se puede ver un mejor rendimiento en la prueba 8.

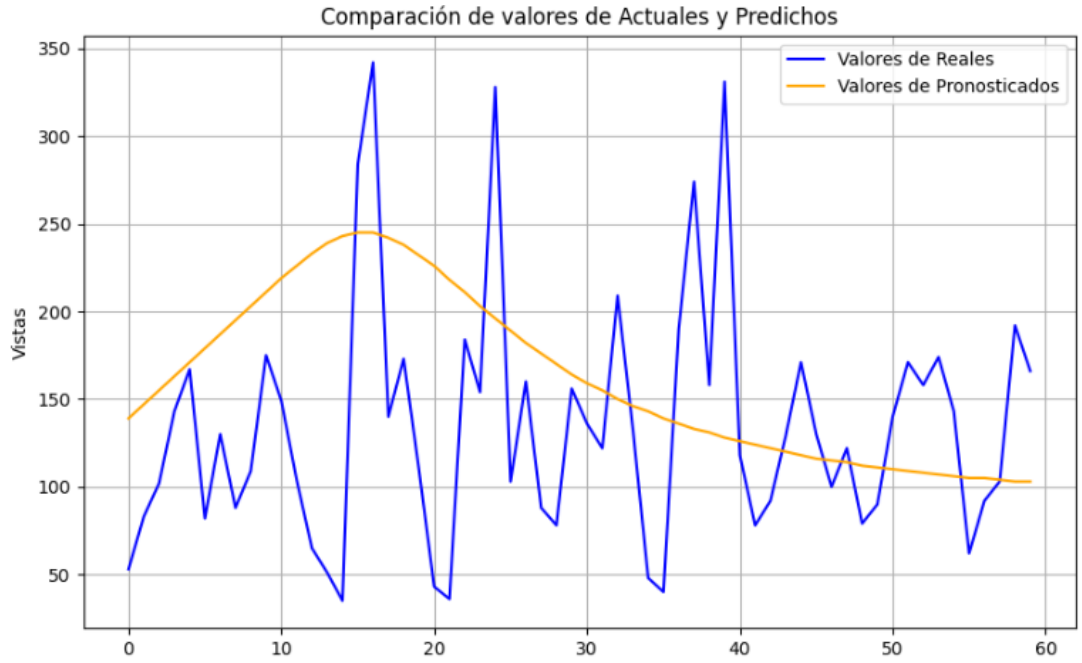

Figura 16: Gráfico de Los Valores Reales y los Valores Pronosticados de la salida del modelo de una parte del conjunto de prueba Y_test de la prueba 7.

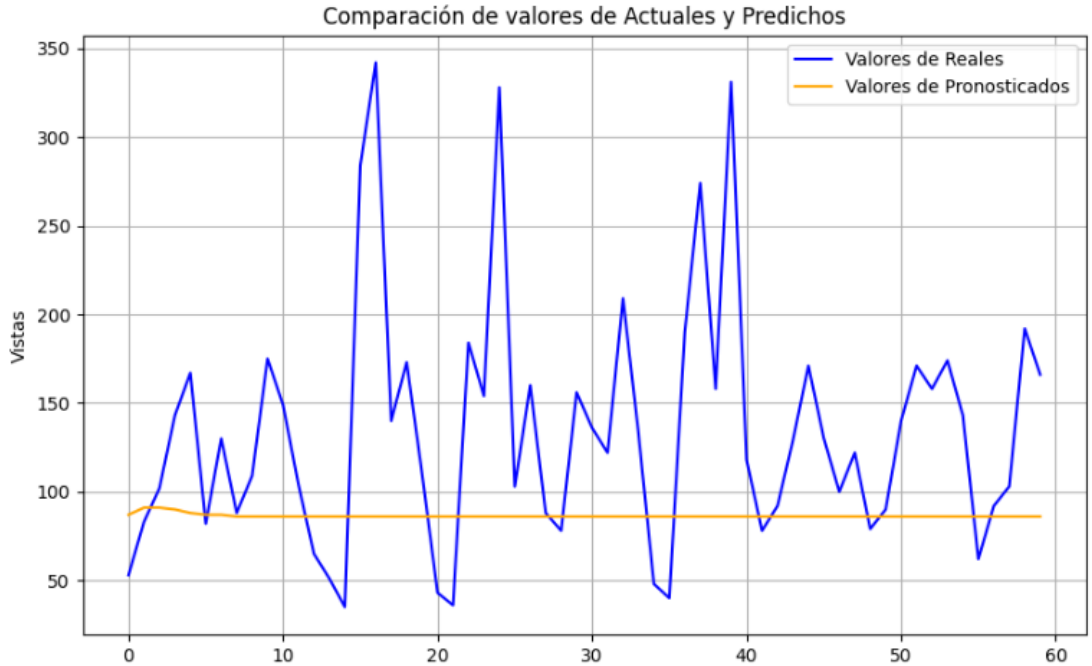

Figura 17: Gráfico de Los Valores Reales y los Valores Pronosticados de la salida del modelo de una parte del conjunto de prueba Y_test, de la prueba 8.

Visualmente se puede ver en la figura 16 que el modelo de la prueba 7 se adapta mejor al acertar el pico más potente, es decir este modelo es capaz de pronosticar tendencia, sin embargo, el modelo de la prueba 8 como se puede ver en la figura 17 tiene una salida que está más cerca de una mayor cantidad de puntos de datos, esto de acuerdo al error medio absoluto y las otras métricas, esto se explica en [7] donde menciona que los modelos no con capaces de predecir picos, esto debido a que es un comportamiento impredecible proveniente de que los visitantes de las páginas web son humanos y tienen sus formas propias de actuar [7], en este caso el modelo de la prueba 8 es más un modelo de pronóstico de media, que de tendencia, y el modelo de la prueba 7 es más un modelo de pronóstico de tendencia.

4.5.10 Resumen Pruebas de Modelos

Para la página web de TUM Transmedia, el mejor modelo con los datos de UA tiene un MAE de 11, mientras que el mejor modelo en el documento científico [7] donde se pronostican páginas web de Wikipedia, tiene un MAE de 17 sobre el *dataset* de idioma chino de Wikipedia, dado que menores números de MAE se traducen en mejores resultados, se puede concluir que es un buen resultado el rendimiento del modelo obtenido para TUM Transmedia UA.

Para la página web de la Editorial Universitaria UA tenemos el mejor modelo generado, este modelo tiene un MAE de 81, en el documento científico [7] se menciona que un buen rendimiento es un MAE por debajo de 30, lo cual para este conjunto de datos no se ha podido obtener.

Durante el desarrollo de la tesis hubo un suceso inesperado, y este fue el cambio de la forma de registrar el tráfico web desde la propiedad Universal Analytics a Google Analytics 4 [46] siendo este último el único en continuar funcionando. Se realizaron pruebas de pasar los modelos entrenados con los datos de UA a GA4.

Los mejores modelos de TUM Transmedia UA y Editorial Universitaria UA se probaron con los datos de Editorial Universitaria GA4 el cual mejoró levemente el rendimiento pasando de un MAE de 81 a 71, aun así, es un rendimiento que está por debajo de lo esperado, por otro lado, el mejor modelo de TUM Transmedia UA, se probó con los datos de TUM Transmedia GA4 manteniendo un buen rendimiento con un MAE por debajo de 30.

Dado que los modelos de GA4 tuvieron un conjunto reducido de datos para poder reentrenarse, está la posibilidad de que si contáramos con una mayor cantidad de datos, pudiéramos tener un mejor rendimiento en el pronóstico.

Hay modelos resultantes de las pruebas que se alejan de la media y predicen tendencia mientras que otros dan como resultado algo semejante a una media.

La siguiente tabla muestra cuales modelos son útiles para predecir tendencia o media.

#Prueba	Predice	MAE	SMAPE
1	Media	11.126461988304094	0.5977127357014901
2	Media	13.274267093758834	0.6972292246352332
3	Tendencia	12.152266081871344	0.6348007961755141
4	Tendencia	258.66926229	1.1840436456
5	Media	81.53907103825136	0.609703658254883
6	Tendencia	26.5404255319	0.57887830853
7	Tendencia	89.944642857	0.9568072360
8	Media	71.077976190	0.6042800485

Tabla 15 Resumen de las pruebas realizada

Capítulo 5

Conclusión

5.1 Conclusiones

El desarrollo de este trabajo aporta a la U.Na.M modelos de Pronóstico de Tráfico Web para las páginas institucionales TUM Transmedia y Editorial Universitaria.

En el apartado 4.4 Modelos, del capítulo 4 en la tabla 3 se puede observar una comparación de las distintas tecnologías de los modelos propuestos en el marco teórico, para determinar cuál se adecua más a la problemática, se utilizaron los criterios mencionados en el apartado 4.4.1.

Como se puede observar en el capítulo 4, a partir del apartado 4.5.10, se detalla que se ha logrado encontrar modelos que pueden servir para pronosticar series de tiempo de tráfico web y el desempeño de los modelos es determinado usando datos extraídos de las páginas institucionales de la U.Na.M.

En la tabla 15 del capítulo 4, en el apartado 4.5.10, se puede ver un resumen de las pruebas realizadas, se han encontrado modelos que al medir su rendimiento con la métrica del error medio absoluto, estos obtienen un valor por debajo de 30, tanto para el conjunto de datos de Universal Analytics, como el conjunto de datos de Google Analytics 4, correspondiente a la página web de TUM Transmedia, lo que es considerado un buen resultado.

Respecto al punto de detectar patrones de estacionalidad y tendencia, referente al tráfico web de las páginas institucionales de la U.Na.M. Como se puede ver en el capítulo 4, en el apartado 4.5.10 en el resumen de pruebas, en la tabla 15 se detalla que se han encontrado modelos que pueden servir para pronosticar las páginas web de la institución algunos más enfocados en tendencia y otros en media.

5.2 Futuras líneas de investigación

A partir del desarrollo de esta Tesis surgen varias líneas de trabajo que son enumeradas a continuación:

- Desarrollar un sistema que realice el pronóstico de series de tiempo de tráfico de manera automatizada cada x cantidad de tiempo.

- Realizar un estudio de caso comparativo de diferentes IAs con los datos de las páginas web de la U.Na.M.

- Generar nuevos modelos agregando nuevos datos que solo se miden en GA4 que no se medían en UA.

- Utilizar los resultados del Pronóstico de Series de Tiempo de Tráfico Web para pronosticar el uso de recursos de hardware.

- Entrenar modelos de pronóstico de series de tiempo de tráfico web cuando se tengan una mayor cantidad de datos de GA4.

- Detectar e informar indicadores que permitan predecir el comportamiento de los servidores para permitir la toma de decisiones que posibilite el mayor rendimiento de los mismos.

- Entender el comportamiento de los usuarios de las páginas institucionales.

Anexos

Anexo 1

En este anexo estarán las tablas resúmenes de los modelos de las pruebas, esto es las capas, el orden de las capas, el tamaño de los tensores de salida y la cantidad de parámetros entrenables. Este anexo también contará con el gráfico en formato dot de los modelos de las pruebas. Dado que las pruebas correspondientes al algoritmo Hyperband genera cientos de modelos en este anexo sólo se incluirá lo correspondiente al mejor modelo de cada prueba.

Resumen modelo de la Prueba 1

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 60, 37)]	0	[]
gru (GRU)	[(None, 224), (None, 224)]	176736	['input_1[0][0]']
repeat_vector (RepeatVector)	(None, 60, 224)	0	['gru[0][0]']
gru_1 (GRU)	(None, 60, 224)	302400	['repeat_vector[0][0]', 'gru[0][1]']
time_distributed (TimeDistributed)	(None, 60, 1)	225	['gru_1[0][0]']

Total params: 479361 (1.83 MB)
Trainable params: 479361 (1.83 MB)
Non-trainable params: 0 (0.00 Byte)

Tabla 16 Resumen del Modelo Obtenido de la Prueba Número 1

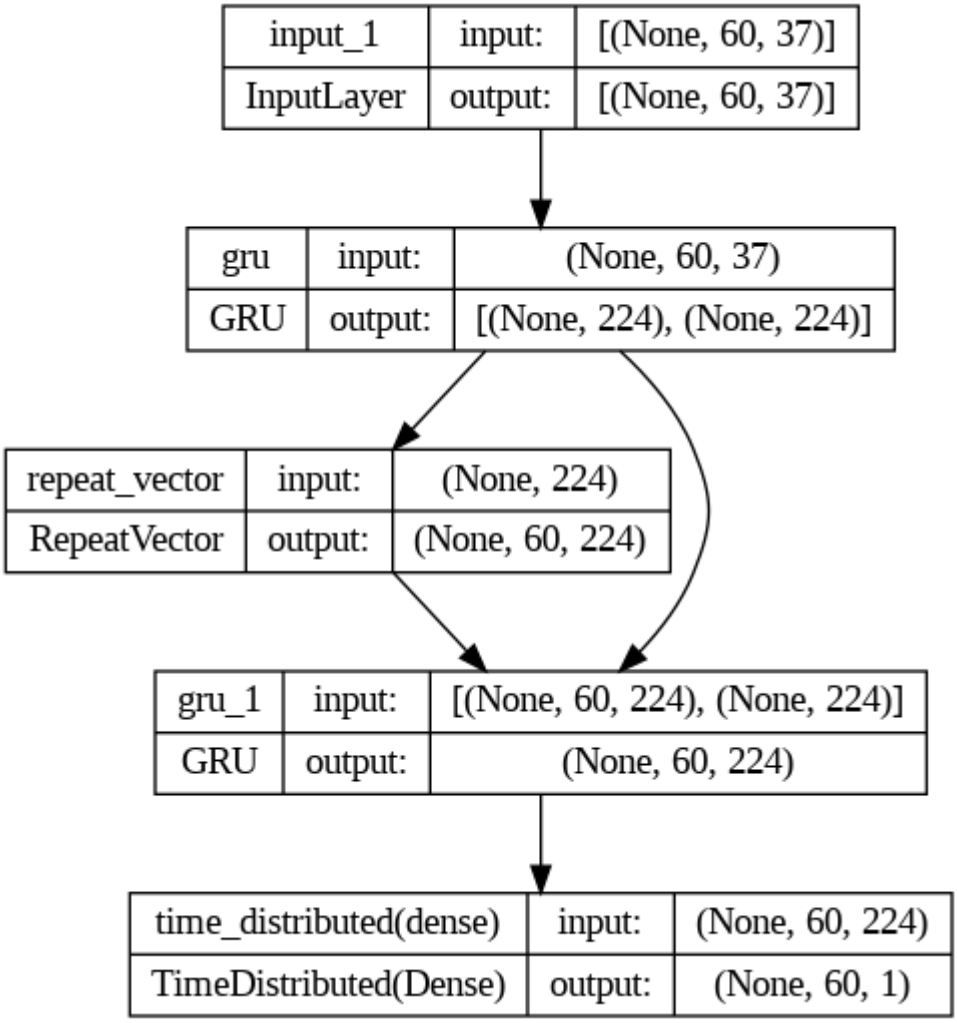

Figura 18: Gráfico Formato Dot de la primera prueba

Resumen modelo de la Prueba 2

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_1 (InputLayer)        [(None, 60, 37)]          0         []

 gru (GRU)                   [(None, 192),             133056    ['input_1[0][0]']
                              (None, 192)]

 repeat_vector (RepeatVecto  (None, 60, 192)           0         ['gru[0][0]']
 r)

 gru_1 (GRU)                 (None, 60, 192)           222336    ['repeat_vector[0][0]',
                                                                  'gru[0][1]']

 time_distributed (TimeDist  (None, 60, 1)             193       ['gru_1[0][0]']
 ributed)

=================================================================================================
Total params: 355585 (1.36 MB)
Trainable params: 355585 (1.36 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________________________________________
```

Tabla 17 Resumen del Modelo Obtenido de la Prueba Número 2

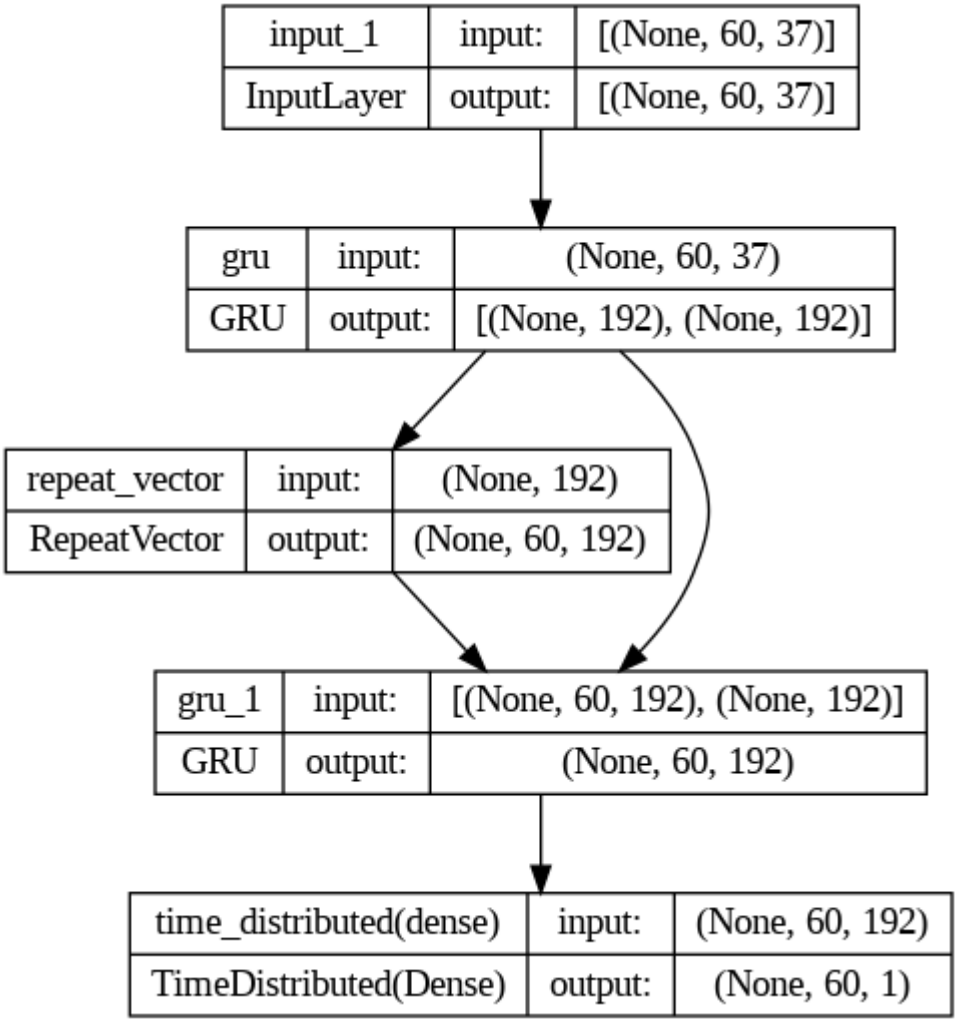

Figura 19: Gráfico Formato Dot de la segunda prueba

Resumen modelo de la Prueba 3

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================
 input_1 (InputLayer)        [(None, 60, 37)]          0         []

 gru (GRU)                   [(None, 60, 224),         176736    ['input_1[0][0]']
                              (None, 224)]

 gru_1 (GRU)                 [(None, 224),             302400    ['gru[0][0]']
                              (None, 224)]

 repeat_vector (RepeatVecto  (None, 60, 224)           0         ['gru_1[0][0]']
 r)

 gru_2 (GRU)                 (None, 60, 224)           302400    ['repeat_vector[0][0]',
                                                                 'gru[0][1]']

 gru_3 (GRU)                 (None, 60, 224)           302400    ['gru_2[0][0]',
                                                                 'gru_1[0][1]']

 time_distributed (TimeDist  (None, 60, 1)             225       ['gru_3[0][0]']
 ributed)

=================================================================
Total params: 1084161 (4.14 MB)
Trainable params: 1084161 (4.14 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Tabla 18 Resumen del Modelo Obtenido de la Prueba Número 3

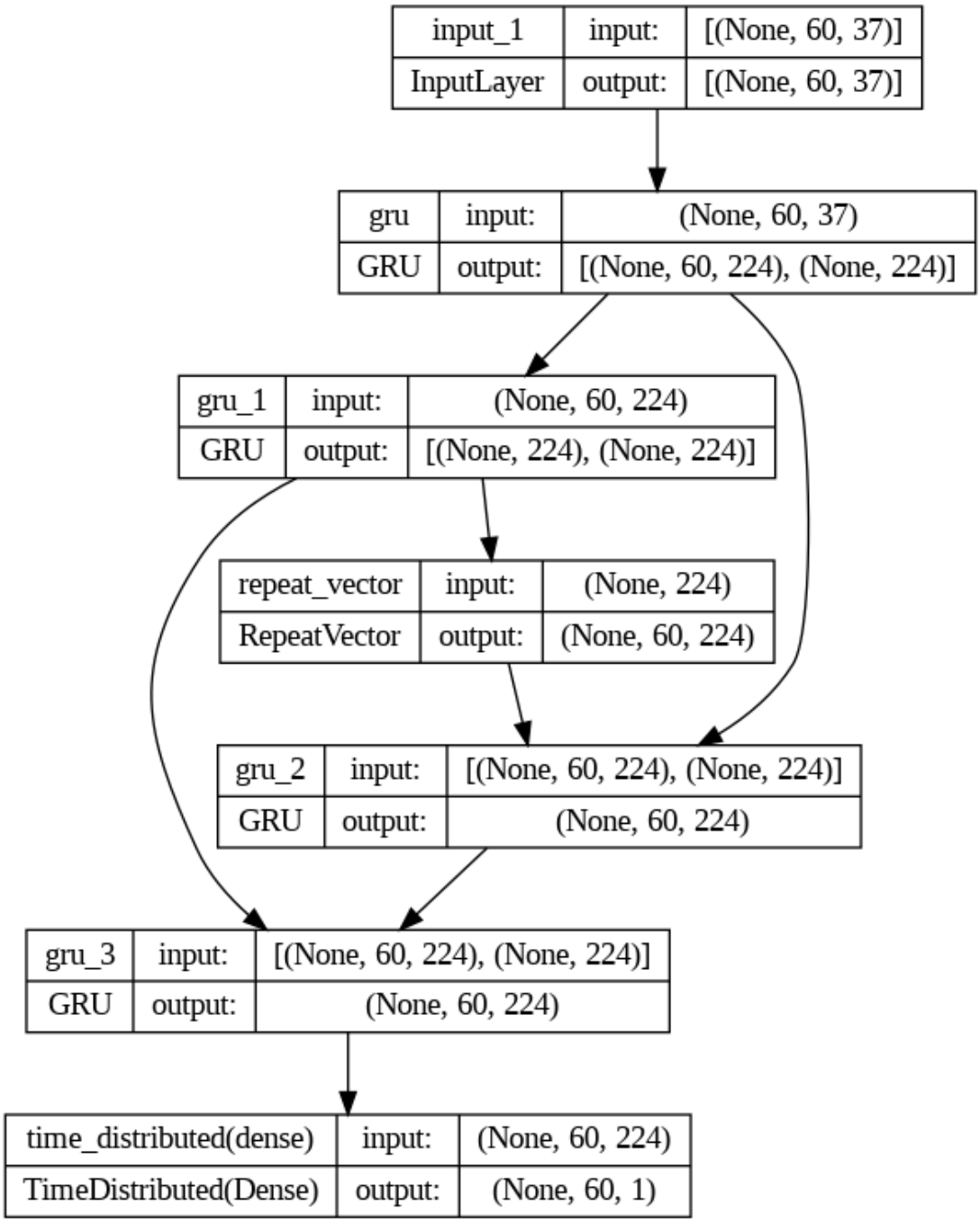

Figura 20: Gráfico Formato Dot de la tercera prueba

Resumen modelo de la Prueba 4

La arquitectura de la prueba 4 es la misma que la de la prueba 1.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================
 input_1 (InputLayer)        [(None, 60, 37)]          0         []

 gru (GRU)                   [(None, 224),             176736    ['input_1[0][0]']
                              (None, 224)]

 repeat_vector (RepeatVecto  (None, 60, 224)           0         ['gru[0][0]']
 r)

 gru_1 (GRU)                 (None, 60, 224)           302400    ['repeat_vector[0][0]',
                                                                 'gru[0][1]']

 time_distributed (TimeDist  (None, 60, 1)             225       ['gru_1[0][0]']
 ributed)

=================================================================
Total params: 479361 (1.83 MB)
Trainable params: 479361 (1.83 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Tabla 19 Resumen del Modelo Obtenido de la Prueba Número 4

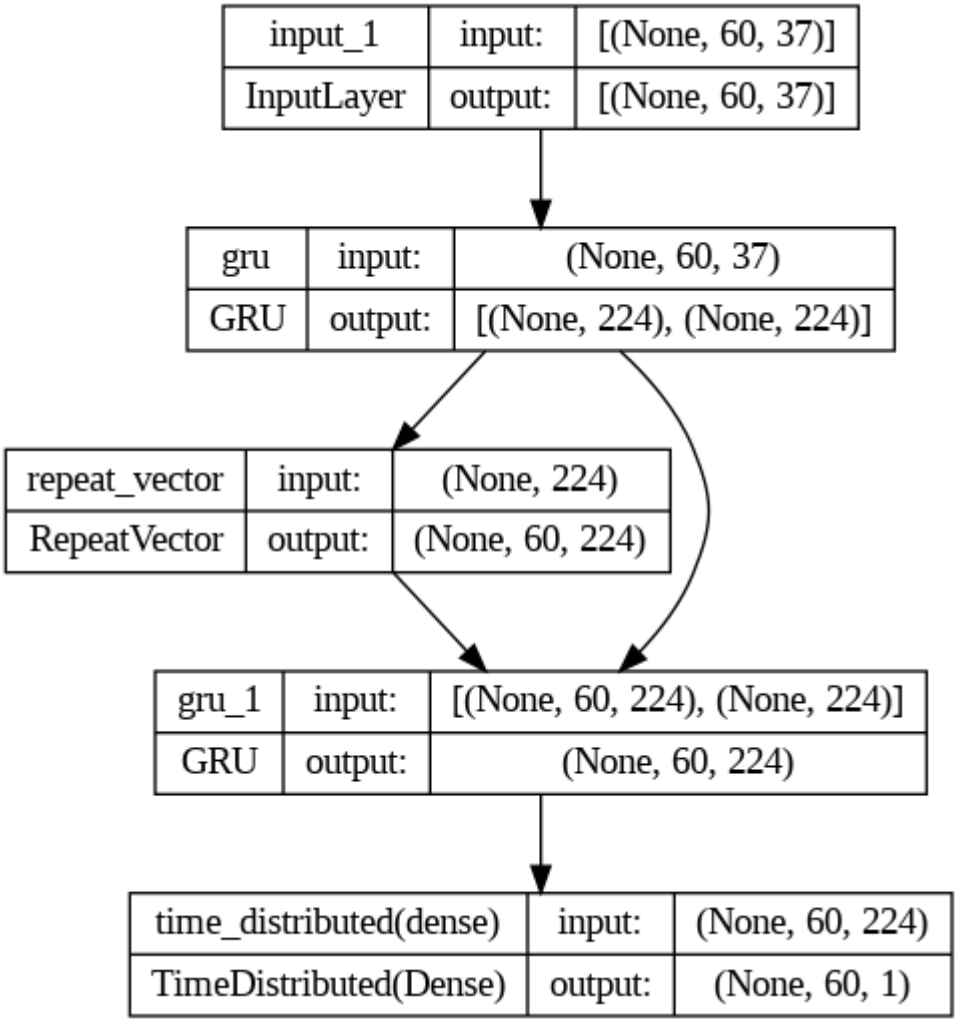

Figura 21: Gráfico Formato Dot de la cuarta prueba

Resumen modelo de la Prueba 5

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 60, 37)]          0

 gru (GRU)                   [(None, 160),             95520
                              (None, 160)]

 repeat_vector (RepeatVecto  (None, 60, 160)           0
 r)

 gru_1 (GRU)                 (None, 60, 160)           154560

 time_distributed (TimeDist  (None, 60, 1)             161
 ributed)

=================================================================
Total params: 250241 (977.50 KB)
Trainable params: 250241 (977.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Tabla 20 Resumen del Modelo Obtenido de la Prueba Número 5

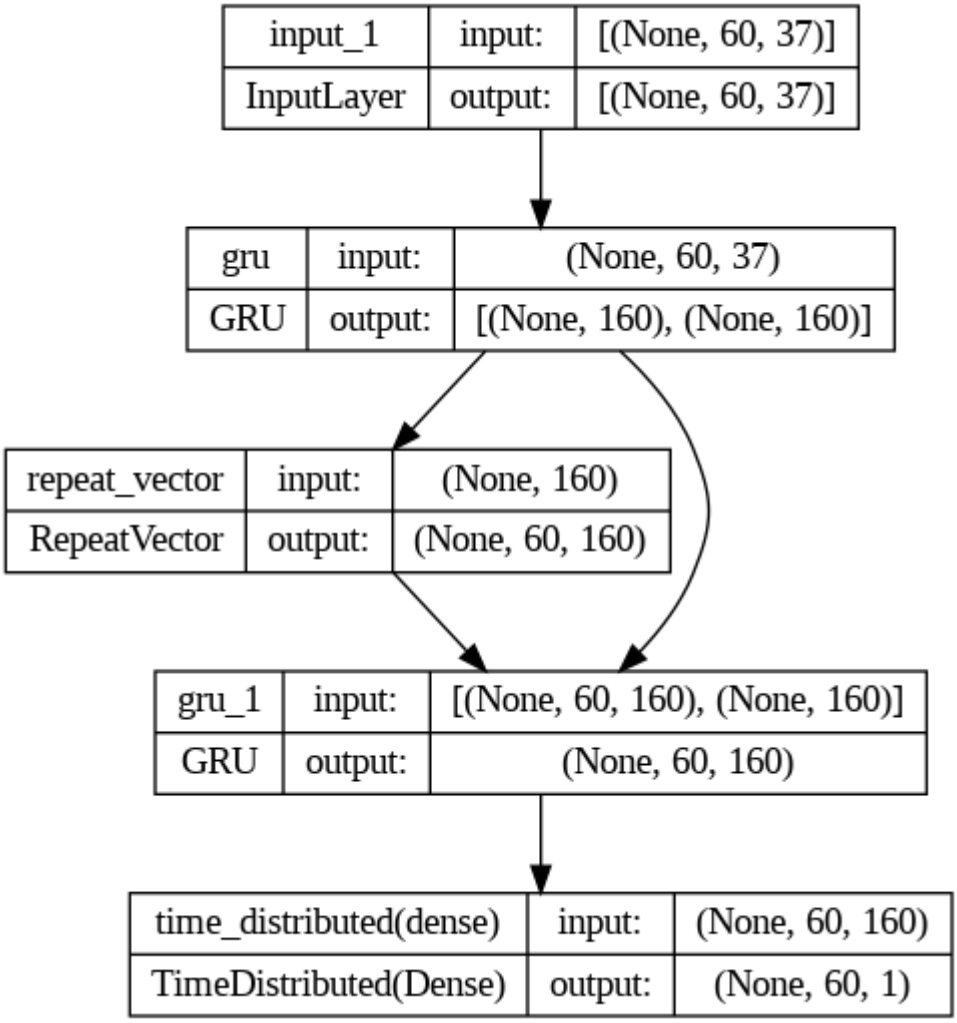

Figura 22: Gráfico Formato Dot de la quinta prueba

Resumen modelo de la Prueba 6

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 60, 37)]	0	[]
gru (GRU)	[(None, 224), (None, 224)]	176736	['input_1[0][0]']
repeat_vector (RepeatVector)	(None, 60, 224)	0	['gru[0][0]']
gru_1 (GRU)	(None, 60, 224)	302400	['repeat_vector[0][0]', 'gru[0][1]']
time_distributed (TimeDistributed)	(None, 60, 1)	225	['gru_1[0][0]']

Total params: 479361 (1.83 MB)
Trainable params: 479361 (1.83 MB)
Non-trainable params: 0 (0.00 Byte)

Tabla 21 Resumen del Modelo Obtenido de la Prueba Número 6

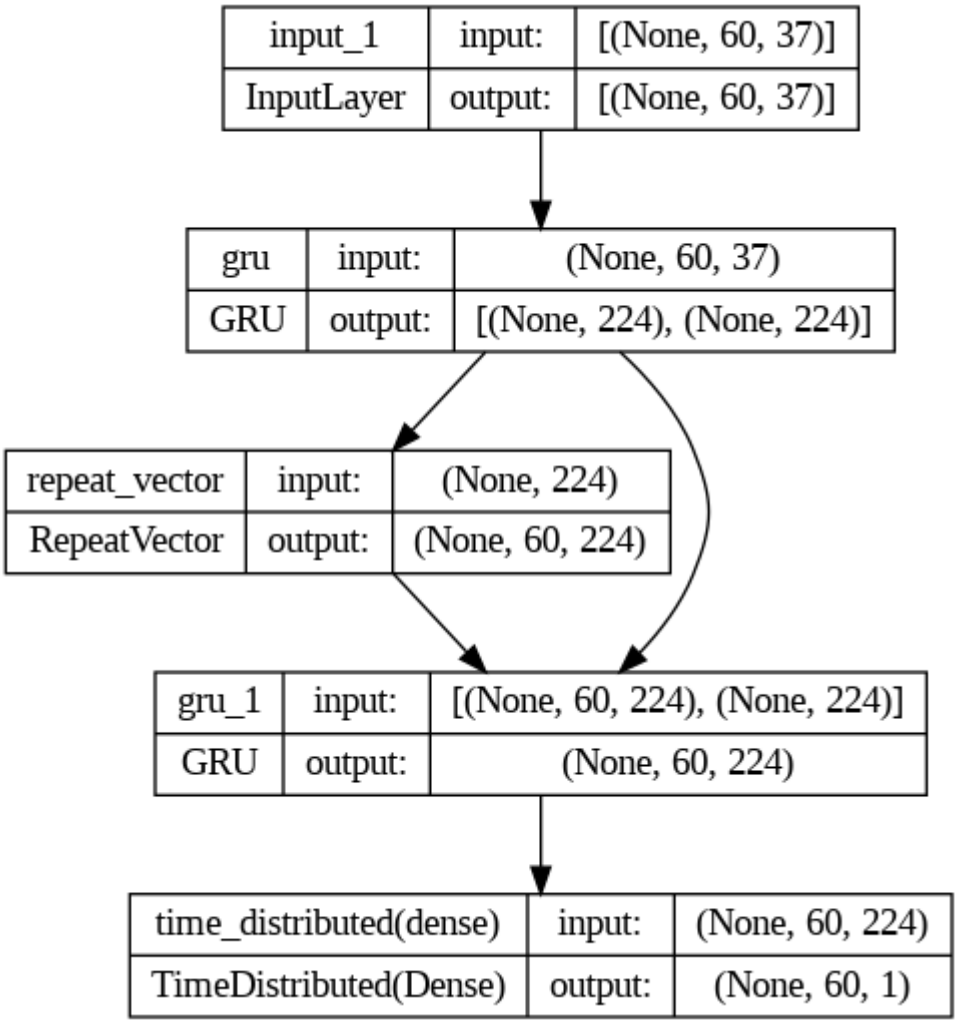

Figura 23: Gráfico Formato Dot de la sexta prueba

Resumen modelo de la Prueba 7

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_1 (InputLayer)        [(None, 60, 37)]          0         []

 gru (GRU)                   [(None, 224),             176736    ['input_1[0][0]']
                              (None, 224)]

 repeat_vector (RepeatVecto  (None, 60, 224)           0         ['gru[0][0]']
 r)

 gru_1 (GRU)                 (None, 60, 224)           302400    ['repeat_vector[0][0]',
                                                                  'gru[0][1]']

 time_distributed (TimeDist  (None, 60, 1)             225       ['gru_1[0][0]']
 ributed)

=================================================================================================
Total params: 479361 (1.83 MB)
Trainable params: 479361 (1.83 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________________________________________
```

Tabla 22 Resumen del Modelo Obtenido de la Prueba Número 7

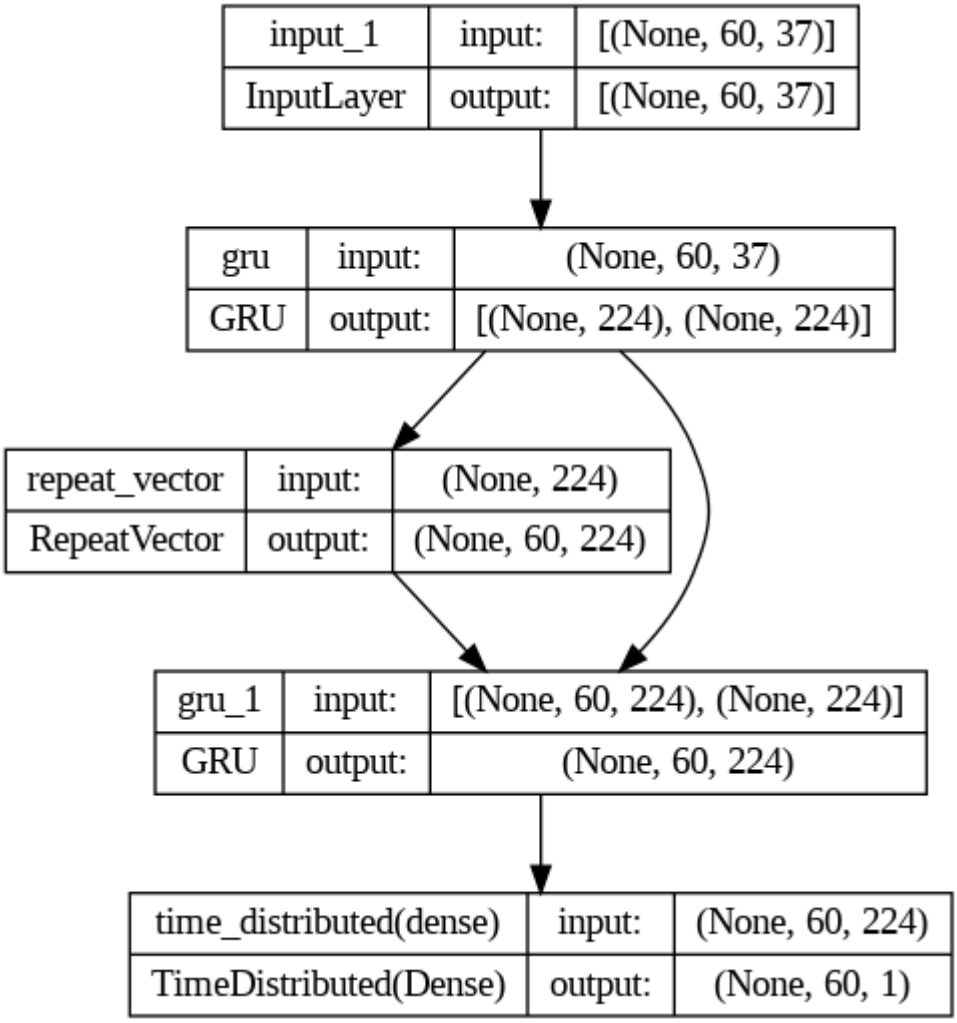

Figura 24: Gráfico Formato Dot de la séptima prueba

Resumen modelo de la Prueba 8

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================
 input_1 (InputLayer)        [(None, 60, 37)]          0         []

 gru (GRU)                   [(None, 160),             95520     ['input_1[0][0]']
                              (None, 160)]

 repeat_vector (RepeatVecto  (None, 60, 160)           0         ['gru[0][0]']
 r)

 gru_1 (GRU)                 (None, 60, 160)           154560    ['repeat_vector[0][0]',
                                                                 'gru[0][1]']

 time_distributed (TimeDist  (None, 60, 1)             161       ['gru_1[0][0]']
 ributed)

=================================================================
Total params: 250241 (977.50 KB)
Trainable params: 250241 (977.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Tabla 23 Resumen del Modelo Obtenido de la Prueba Número 8

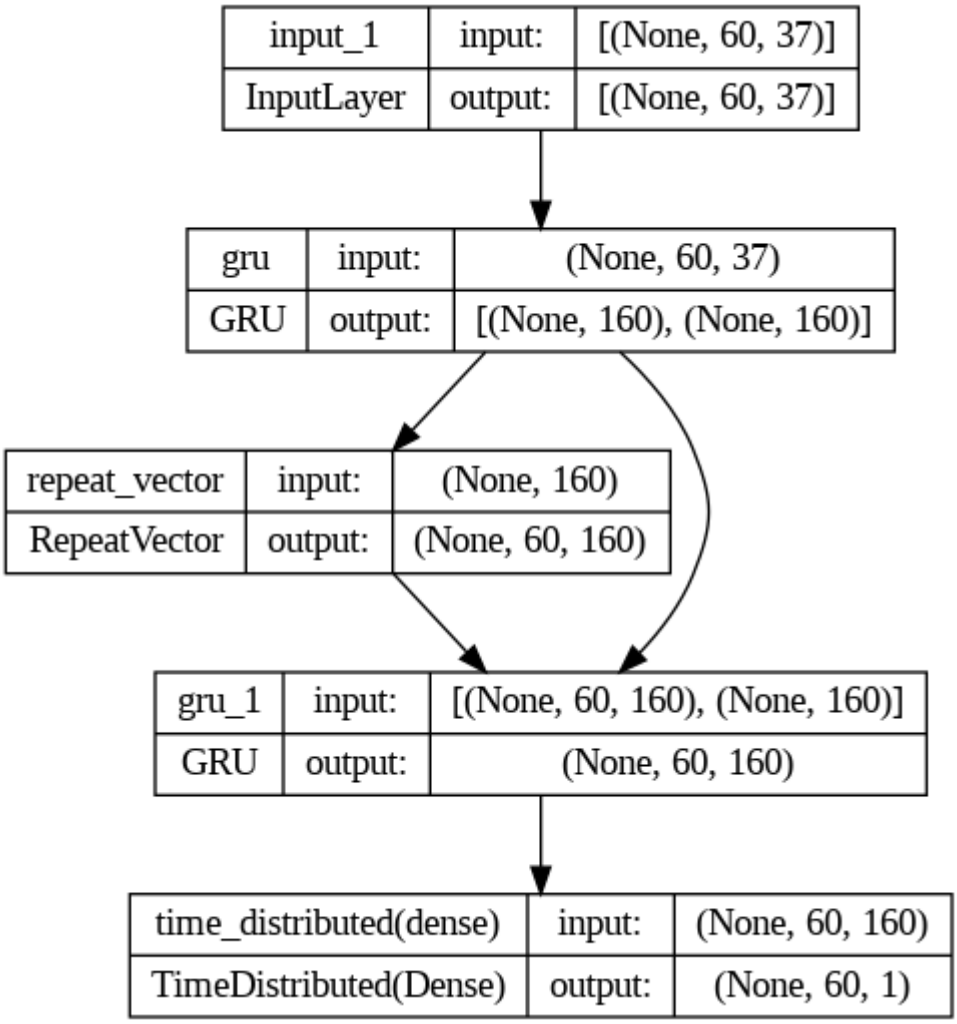

Figura 25: Gráfico Formato Dot de la octava prueba

Bibliografía

[1] N. Petluri and E. Al-Masri, "Web Traffic Prediction of Wikipedia Pages," *Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018*, pp. 5427–5429, 2019.

[2] V. Kotu and B. Deshpande, "Chapter 12 - Time Series Forecasting," V. Kotu and B. B. T.-D. S. (Second E. Deshpande, Eds. Morgan Kaufmann, 2019, pp. 395–445.

[3] K. Zhou, W. Wang, L. Huang, and B. Liu, "Comparative study on the time series forecasting of web traffic based on statistical model and Generative Adversarial model," *Knowledge-Based Syst.*, vol. 213, p. 106467, Feb. 2021.

[4] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[5] K. Zhou, W. Wang, L. Huang, and B. Liu, "Comparative study on the time series forecasting of web traffic based on statistical model and Generative Adversarial model," *Knowledge-Based Syst.*, vol. 213, Feb. 2021.

[6] "Prophet: forecasting at scale." [Online]. Available: https://research.facebook.com/blog/2017/2/prophet-forecasting-at-scale/. [Accessed: 24-Oct-2020].

[7] R. Casado-Vara, A. M. del Rey, D. Pérez-Palau, L. De-La-fuente-valentín, and J. M. Corchado, "Web Traffic Time Series Forecasting Using LSTM Neural Networks with Distributed Asynchronous Training," *Math. 2021, Vol. 9, Page 421*, vol. 9, no. 4, p. 421, Feb. 2021.

[8] W. Wang and Y. Lu, "Analysis of the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) in Assessing Rounding Model," in *IOP Conference Series: Materials Science and Engineering*, 2018, vol. 324, no. 1.

[9] P. J. Huber, "Robust Estimation of a Location Parameter," in *Breakthroughs in Statistics: Methodology and Distribution*, S. Kotz and N. L. Johnson, Eds. New York, NY: Springer New York, 1992, pp. 492–518.

[10] K. Cho *et al.*, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014, pp. 1724–1734.

[11] “Web Traffic Time Series Forecasting,” 2017. [Online]. Available: https://www.kaggle.com/c/web-traffic-time-series-forecasting/discussion/39367. [Accessed: 12-Oct-2022].

[12] “Choosing the correct error metric: MAPE vs. sMAPE,” 2020. [Online]. Available: https://towardsdatascience.com/choosing-the-correct-error-metric-mape-vs-smape-5328dec53fac. [Accessed: 12-Oct-2022].

[13] C. Kuranga and N. Pillay, “A comparative study of nonlinear regression and autoregressive techniques in hybrid with particle swarm optimization for time-series forecasting,” *Expert Syst. Appl.*, vol. 190, Mar. 2022.

[14] P. Khanarsa, A. Luangsodsai, K. Sinapiromsaran, I. F. Astachova, K. A. Makoviy, and Y. V Khitskova, “Possibilities for predicting the state of usability web resources,” *J. Phys. Conf. Ser.*, vol. 1902, no. 1, p. 012029, May 2021.

[15] D. Deng, F. Karl, F. Hutter, B. Bischl, and M. Lindauer, “Efficient Automated Deep Learning for Time Series Forecasting,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2023, vol. 13715 LNAI, pp. 664–680.

[16] S. Xu, C. Han, and C. Ran, “A Time Series Combined Forecasting Model Based on Prophet-LGBM,” in *ACM International Conference Proceeding Series*, 2021.

[17] D. Quoc Nguyen, M. Nguyet Phan, and I. Zelinka, “Periodic Time Series Forecasting with Bidirectional Long Short-Term Memory: Periodic Time Series Forecasting with Bidirectional LSTM,” *ACM Int. Conf. Proceeding Ser.*, pp. 60–64, 2021.

[18] “The M3-Competition Database.” [Online]. Available:

https://forecasters.org/resources/time-series-data/m3-competition/. [Accessed: 18-Oct-2022].

[19] A. Botchkarev, “Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology,” 2018.

[20] P. Chapman *et al.*, “CRISP-DM 1.0: Step-by-step data mining guide,” *SPSS inc*, vol. 9, no. 13, pp. 1–73, 2000.

[21] C. Schröer, F. Kruse, and J. M. Gómez, “A systematic literature review on applying CRISP-DM process model,” in *Procedia Computer Science*, 2021, vol. 181, pp. 526–534.

[22] Python Software Foundation, “Python 3.12.1 documentation.” [Online]. Available: https://docs.python.org/3/. [Accessed: 02-Feb-2024].

[23] Google, “Plataforma de archivos compartidos y almacenamiento personal en la nube - Google.” [Online]. Available: https://www.google.com/intl/es-419_ar/drive/. [Accessed: 02-Feb-2024].

[24] Google and Google Lcc, “Ayuda de Google,” *Ayuda de Google*, 2023. [Online]. Available: https://support.google.com/. [Accessed: 02-Feb-2024].

[25] Microsoft corporation, “Microsoft: Nube, aplicaciones y juegos,” *Microsoft: Nube, aplicaciones y juegos*, 2023. [Online]. Available: https://www.microsoft.com/es-ar/. [Accessed: 16-Apr-2024].

[26] Numpy.org, “NumPy Documentation.” [Online]. Available: https://numpy.org/doc/. [Accessed: 05-Feb-2024].

[27] Google Lcc, “TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático.” [Online]. Available: https://www.tensorflow.org/. [Accessed: 05-Feb-2024].

[28] NumFOCUS I, “About pandas.” [Online]. Available: https://pandas.pydata.org/about/. [Accessed: 06-Feb-2024].

[29] The Matplotlib development team, “Matplotlib: Visualization with Python.” [Online]. Available: https://matplotlib.org/. [Accessed: 06-Feb-2024].

[30] Google, “Google Colab.” [Online]. Available: https://research.google.com/colaboratory. [Accessed: 02-Feb-2024].

[31] GitHub, “Acerca de Git.” [Online]. Available: https://docs.github.com/es/get-started/using-git/about-git. [Accessed: 06-Feb-2024].

[32] Google Lcc, “Introduction to the Keras Tuner.” [Online]. Available: https://www.tensorflow.org/tutorials/keras/keras_tuner. [Accessed: 15-Feb-2024].

[33] F. Chollet and Others, “Keras,” 2015. [Online]. Available: https://keras.io. [Accessed: 15-Feb-2024].

[34] T. O’Malley *et al.*, “KerasTuner,” 2019. [Online]. Available: https://keras.io/api/keras_tuner/tuners/base_tuner/. [Accessed: 15-Feb-2024].

[35] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *J. Mach. Learn. Res.*, vol. 18, pp. 1–52, 2018.

[36] J. Bergstra and B. Yoshua, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, no. 10, pp. 281–305, 2012.

[37] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian Optimization of Machine Learning Algorithms,” *Adv. Neural Inf. Process. Syst.*, vol. 25, 2012.

[38] A. Baghbanpourasl, E. Lughofer, P. Meyer-Heye, H. Zorrer, and C. Eitzinger, “Virtual quality control using bidirectional lstm networks and gradient boosting,” in *IEEE International Conference on Industrial Informatics (INDIN)*, 2019, vol. 2019-July, pp. 1638–1643.

[39] J. Tourille, O. Ferret, X. Tannier, and A. L. Nvol, “Neural architecture for temporal relation extraction: A Bi-LSTM approach for detecting narrative containers,” in *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 2017, vol. 2, pp. 224–230.

[40] W. Rüdiger and J. Hipp, “CRISP-DM : Towards a Standard Process Model for

Data Mining," *Proc. Fourth Int. Conf. Pract. Appl. Knowl. Discov. Data Min.*, no. 24959, pp. 29–39, 2000.

[41] Google Lcc, "[UA] Información sobre las agrupaciones de canales." [Online]. Available: https://support.google.com/analytics/answer/6010097. [Accessed: 07-Feb-2024].

[42] Google Lcc, "[GA4] Informe 'Detalles de la tecnología.'" [Online]. Available: https://support.google.com/analytics/answer/12980150. [Accessed: 07-Feb-2023].

[43] T. T. Dang, H. Y. T. Ngan, and W. Liu, "Distance-based k-nearest neighbors outlier detection method in large-scale traffic data," *Int. Conf. Digit. Signal Process. DSP*, vol. 2015-Septe, pp. 507–510, 2015.

[44] A. Blázquez-Garc\'\ia, A. Conde, U. Mori, and J. A. Lozano, "A Review on Outlier/Anomaly Detection in Time Series Data," *ACM Comput. Surv.*, vol. 54, no. 3, 2021.

[45] A. Suilin, "kaggle-web-traffic." [Online]. Available: https://github.com/Arturus/kaggle-web-traffic. [Accessed: 07-Nov-2024].

[46] Google Lcc, "[UA→GA4] Diferencias entre los datos de Universal Analytics y Google Analytics 4." [Online]. Available: https://support.google.com/analytics/answer/9964640. [Accessed: 10-Feb-2024].

[47] Y.-F. Lim, C. Koon Ng, U. Vaitesswar, and K. Hippalgaonkar, "Extrapolative Bayesian Optimization with Gaussian Process and Neural Network Ensemble Surrogate Models," 2021.

[48] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, pp. 3104–3112, 2014.

[49] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv Prepr. arXiv1412.6980*, 2014.