Optimización de Problemas de Ingeniería Basada en Datos e IA

Jordy Gabriel Garzón Gaona
Maestría de Ciencia de Datos
Universidad San Francisco de Quito

Quito, Ecuador
jggarzong@stud.usfq.edu.ec
https://lightning.ai/jgarzong/studios/proyectofinal.

Resumen*— El siguiente documento detalla la propuesta del proyecto que propone la integración de inteligencia artificial (IA) y análisis de big data para optimizar problemas complejos en el campo de la ingeniería. Utilizando modelos de aprendizaje automático y aprendizaje profundo, se analizarán grandes volúmenes de datos históricos y en tiempo real para identificar patrones, predecir resultados y recomendar soluciones óptimas. El objetivo es mejorar la eficiencia, sostenibilidad y rentabilidad en diferentes sectores de la ingeniería.

Palabras clave— Ingeniería, inteligencia artificial, Big Data, aprendizaje automático

I. Introducción

En la era de la digitalización y la automatización, la integración de la inteligencia artificial (IA) y el análisis de Big Data ha emergido como una herramienta poderosa para abordar desafíos complejos en diversos campos [1]. Este proyecto se centra en la aplicación de estas tecnologías avanzadas para optimizar problemas de ingeniería, con el objetivo de mejorar la eficiencia, la sostenibilidad y la rentabilidad en sectores clave como la fabricación, el diseño estructural y la distribución de energía. La capacidad de analizar grandes volúmenes de datos históricos y en tiempo real, combinada con algoritmos de aprendizaje automático, permite descubrir patrones ocultos, predecir resultados y recomendar soluciones óptimas. Este enfoque no solo mejora la toma de decisiones, sino que también impulsa la innovación y la competitividad en la ingeniería moderna.

II. Objetivos

A. Objetivos del Proyecto

El principal objetivo de este proyecto es desarrollar un sistema de optimización basado en IA y Big Data que pueda aplicarse a diferentes dominios de la ingeniería. Los objetivos específicos incluyen:

- Analizar grandes conjuntos de datos históricos y en tiempo real para identificar patrones y tendencias.
- Desarrollar modelos predictivos utilizando algoritmos de aprendizaje automático y aprendizaje profundo.
- Implementar soluciones de optimización que mejoren la eficiencia, la sostenibilidad y la rentabilidad en procesos de ingeniería.
- Validar y perfeccionar continuamente los modelos de IA con nuevos datos para asegurar su relevancia y precisión.

III. Aplicaciones

El proyecto tiene como objetivo optimizar los procesos en varios dominios de ingeniería como son:

A. Fabricación

Los algoritmos de IA pueden predecir fallas en las máquinas antes de que ocurran, minimizando el tiempo de inactividad y los costos de mantenimiento. Además, pueden optimizar los procesos de producción para mejorar la eficiencia y reducir el desperdicio.

B. Ingeniería Estructural

En la ingeniería estructural, el proyecto puede optimizar el uso de materiales y los diseños para mejorar la resistencia y la economía. Esto resultará en construcciones más seguras y asequibles, con un menor impacto ambiental.

C. Distribución de Energía

En la distribución de energía, los modelos de IA pueden optimizar la gestión de la red eléctrica, mejorando la eficiencia y reduciendo las pérdidas. Esto contribuirá a una distribución de energía más sostenible y rentable.

D. Servicios

En el campo empresarial operador - suscriptor, la IA y Big Data optimizarían los procesos de retención y adhesión de clientes antiguos/nuevos, así como mejorar sus o cambiar la matriz productiva de sus servicios prestados mejorando métricas como el QoS [2].

IV. Desarrollo

Como parte de una solución preliminar al problema planteado para el desarrollo del proyecto de Fundamentos de Ciencia de datos, se ha determinado el usos de un dataset de origen publico conocido como *Nasa Turbofan Dataset*[3]

A través de modelos de aprendizaje automático y análisis de datos históricos, se pueden identificar patrones y predecir comportamientos para optimizar procesos.

A. Definición del Problema

El objetivo es determinar el tiempo de vida útil restante de los motores turbofan, de acuerdo al dataset seleccionado, este tiempo de vida útil restante (RUL por sus siglas en inglés, *Remaing Useful Life*)[4], permitirá realizar recomendaciones sobre cuando realizar un mantenimiento adecuado o a su vez

un reemplazo del equipamiento antes de que el equipo deje de funcionar.

B. Selección del Dataset

El dataset *Nasa Turbofan Dataset* [3]**,**, es ideal para este caso de uso. Aunque se publicó hace más de una década, el conjunto de datos de simulación de degradación de motores de turbofán (CMAPSS) de la NASA sigue siendo popular y relevante en la actualidad. Hasta el momento, se han publicado más de 90 artículos de investigación nuevos en 2020 [5].

La última versión de este dataset fue actualizada en noviembre del 2020, se lo ha seleccionado porque se alinea adecuadamente con el proyecto y como parte de una de las aplicaciones en este sector mencionadas anteriormente.

C. Características del dataset

El *NASA Turbofan Engine Degradation Simulation Dataset* es un conjunto de datos utilizado comúnmente para problemas de predicción de la vida útil remanente (RUL) de motores. Contiene datos de simulaciones de la degradación de motores turbofan bajo condiciones de funcionamiento, con 4 subconjuntos etiquetados como FD001, FD002, FD003 y FD004, como se observa en la Fig 1.

Donde se entiende que para cada subconjunto se tienen diferentes modos de operación que van desde 1 y 6 definidos por tres características en cada subconjunto de datos, además de 2 tipos de fallas definidas en una sola variable o característica del conjunto de datos y para terminar cada subset de datos vienen con datos de entrenamiento, con el numero de motores definido, donde para cada motor describe la información hasta que el motor falla es decir su RUL llega a cero, en cambio que para los datos de prueba lleva la información de cada motor antes de que produzca la falla.

Dataset	Operating conditions	Fault modes	Train size (nr. of engines)	Test size (nr. of engines)
FD001	1	1	100	100
FD002	6	1	260	259
FD003	1	2	100	100
FD004	6	2	248	249

Figura 1 Subconjuntos Dataset TurboFan

El dataset describe los siguiente:

- Ciclos de vuelo**:** Registros de datos de diferentes motores a lo largo de varios ciclos de operación.
- 21 variables**:** Incluyen parámetros como presión, temperatura y flujo de aire.
- Subconjuntos FD**:** Varían en complejidad y número de condiciones operativas.

Los conjuntos de datos incluyen simulaciones de múltiples motores de turbofán a lo largo del tiempo; cada fila de datos contiene la siguiente información:

- Número de unidad del motor
- Tiempo, en ciclos
- Tres configuraciones operativas
- 21 lecturas de sensores

Este dataset es ideal para análisis de series temporales y problemas de mantenimiento predictivo. Como punto final de esta subsección cabe señalar que se uso el subconjunto de datos FD001 para realizar este trabajo.

D. Análisis de Datos y Preprocesamiento

Antes que nada, cabe aclarar que la idea en este escrito no es ir describiendo todo el código realizado sino más bien ir revisando sus resultados, esto debido a que el código lo puede encontrar en [6].

El conjunto de datos fue descargado de la página oficial de los dataset de la Nasa[7], una vez leído los datos observamos el conjunto de datos como se ve en Fig 2.

	unit_nr	time_cycles	setting_1	setting_2	setting_3	s_1	s_2	s_3	s_4	s_5	...
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...

Figura 2 Dataset Cargado

FD001 debe contener datos de cien motores, inspeccionemos el número de unidad para verificar que sea así. Elegí usar la función de descripción de Pandas para que también podamos comprender la distribución. Mientras estamos en esto, inspeccionemos también los ciclos de tiempo para ver qué podemos aprender sobre la cantidad de ciclos que los motores funcionaron en promedio antes de averiarse.

	unit_nr		time_cycles
count	20631.000000	count	100.000000
mean	51.506568	mean	206.310000
std	29.227633	std	46.342749
min	1.000000	min	128.000000
25%	26.000000	25%	177.000000
50%	52.000000	50%	199.000000
75%	77.000000	75%	229.250000
max	100.000000	max	362.000000

Figura 3 Descripción estadística de unit_nr y time_cycles

En Fig 3. cuando inspeccionamos las estadísticas descriptivas de unit_nr podemos ver que el conjunto de datos tiene un total de 20631 filas, los números de unidad comienzan en 1 y terminan en 100 como se esperaba. Lo que es interesante es que la media y los cuantiles no se alinean perfectamente con las estadísticas descriptivas de un vector de 1 a 100, esto se puede explicar debido a que cada unidad tiene diferentes time_cycles máximos y, por lo tanto, una cantidad diferente de filas. Al inspeccionar los time_cycles máximos, puede ver que el motor que falló primero lo hizo después de 128 ciclos, mientras que el motor que funcionó durante más tiempo se averió después de 362 ciclos. El motor promedio se avería entre 199 y 206 ciclos, sin embargo, la desviación estándar de 46 ciclos es bastante grande. Visualizaremos esto más abajo para comprenderlo aún mejor.

La descripción del conjunto de datos también indica que los turbofán funcionan en una única condición de funcionamiento. Verifiquemos la configuración para verificarla.

	setting_1	setting_2	setting_3
count	20631.000000	20631.000000	20631.0
mean	-0.000009	0.000002	100.0
std	0.002187	0.000293	0.0
min	-0.008700	-0.000600	100.0
25%	-0.001500	-0.000200	100.0
50%	0.000000	0.000000	100.0
75%	0.001500	0.000300	100.0
max	0.008700	0.000600	100.0

Figura 4 Descripción de los settings, variables que describen el modo de operación del motor

En Fig. 4 se observan las desviaciones estándar de los ajustes 1 y 2, no son completamente estables. Sin embargo, las fluctuaciones son tan pequeñas que no se pueden identificar otras condiciones de funcionamiento. Por último, inspeccionaremos las estadísticas descriptivas de los datos del sensor, buscando indicadores de fluctuación de la señal (o ausencia de ella).

	count	mean	std	min	25%	50%	75%	max
s_1	20631.0	518.670000	0.000000e+00	518.6700	518.6700	518.6700	518.6700	518.6700
s_2	20631.0	642.680934	5.000533e-01	641.2100	642.3250	642.6400	643.0000	644.5300
s_3	20631.0	1590.523119	6.131150e+00	1571.0400	1586.2600	1590.1000	1594.3800	1616.9100
s_4	20631.0	1408.933782	9.000605e+00	1382.2500	1402.3600	1408.0400	1414.5550	1441.4900
s_5	20631.0	14.620000	1.776400e-15	14.6200	14.6200	14.6200	14.6200	14.6200
s_6	20631.0	21.609803	1.388985e-03	21.6000	21.6100	21.6100	21.6100	21.6100
s_7	20631.0	553.367711	8.850923e-01	549.8500	552.8100	553.4400	554.0100	556.0600
s_8	20631.0	2388.096652	7.098548e-02	2387.9000	2388.0500	2388.0900	2388.1400	2388.5600
s_9	20631.0	9065.242941	2.208288e+01	9021.7300	9053.1000	9060.6600	9069.4200	9244.5900
s_10	20631.0	1.300000	0.000000e+00	1.3000	1.3000	1.3000	1.3000	1.3000
s_11	20631.0	47.541168	2.670874e-01	46.8500	47.3500	47.5100	47.7000	48.5300
s_12	20631.0	521.413470	7.375534e-01	518.6900	520.9600	521.4800	521.9500	523.3800
s_13	20631.0	2388.096152	7.191892e-02	2387.8800	2388.0400	2388.0900	2388.1400	2388.5600
s_14	20631.0	8143.752722	1.907618e+01	8099.9400	8133.2450	8140.5400	8148.3100	8293.7200
s_15	20631.0	8.442146	3.750504e-02	8.3249	8.4149	8.4389	8.4656	8.5848
s_16	20631.0	0.030000	1.387812e-17	0.0300	0.0300	0.0300	0.0300	0.0300
s_17	20631.0	393.210654	1.548763e+00	388.0000	392.0000	393.0000	394.0000	400.0000
s_18	20631.0	2388.000000	0.000000e+00	2388.0000	2388.0000	2388.0000	2388.0000	2388.0000
s_19	20631.0	100.000000	0.000000e+00	100.0000	100.0000	100.0000	100.0000	100.0000
s_20	20631.0	38.816271	1.807464e-01	38.1400	38.7000	38.8300	38.9500	39.4300
s_21	20631.0	23.289705	1.082509e-01	22.8942	23.2218	23.2979	23.3668	23.6184

Figura 5 Descripción estadística de los sensores

Al observar la desviación estándar, queda claro que los sensores 1, 10, 18 y 19 no fluctúan en absoluto; se pueden descartar sin problemas, ya que no contienen información útil. Al inspeccionar los cuantiles, se indica que los sensores 5, 6 y 16 tienen poca fluctuación y requieren una inspección más exhaustiva. Los sensores 9 y 14 tienen la fluctuación más alta; sin embargo, esto no significa que los demás sensores no puedan contener información valiosa.

1) Cálculo de la vida útil restante

Antes de comenzar a representar gráficamente nuestros datos para continuar con nuestro análisis de vida útil restante, calcularemos una variable objetivo para la vida útil restante (RUL). La variable objetivo tendrá dos propósitos:

a) Servirá como nuestro eje X mientras trazamos las señales de los sensores, lo que nos permitirá interpretar fácilmente los cambios en las señales de los sensores a medida que los motores se acercan a la avería.
b) Servirá como variable objetivo para nuestros modelos de aprendizaje automático supervisado. Sin más información sobre el RUL de los motores en el conjunto de entrenamiento, tendremos que elaborar nuestras propias estimaciones.

Supondremos que el RUL disminuye linealmente con el tiempo y tiene un valor de 0 en el último ciclo de tiempo del motor. Esta suposición implica que el RUL sería 10 a los 10 ciclos antes de la avería, 50 a los 50 ciclos antes de la avería, etc

2) Representación gráfica

La representación gráfica siempre es una buena idea para comprender mejor el conjunto de datos. Comencemos por representar gráficamente el histograma de RUL máximo para comprender su distribución como se ve en Fig. 6.

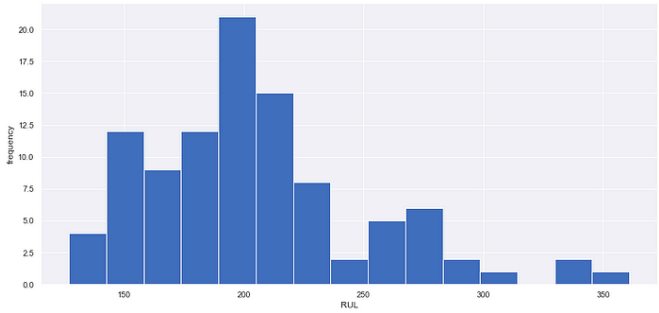


Figura 6 Histograma del RUL

El histograma confirma que la mayoría de los motores se averían alrededor de los 200 ciclos. Además, la distribución está sesgada hacia la derecha, y pocos motores duran más de 300 ciclos.

Debido a la gran cantidad de motores, no es posible representar gráficamente cada motor para cada sensor. Los gráficos ya no serían interpretables con tantas líneas en un gráfico. Por lo tanto, elegí representar gráficamente cada motor cuyo número de unidad es divisible por 10 con un resto de 0. Invertimos el eje X para que el RUL disminuya a lo largo del eje, y un RUL de cero indica una falla del motor. Debido a la gran cantidad de sensores, analizaré algunos gráficos que son representativos de todo el conjunto. Recuerde que, en función de nuestras estadísticas descriptivas, definitivamente deberíamos inspeccionar los gráficos de los sensores 5, 6 y 16.

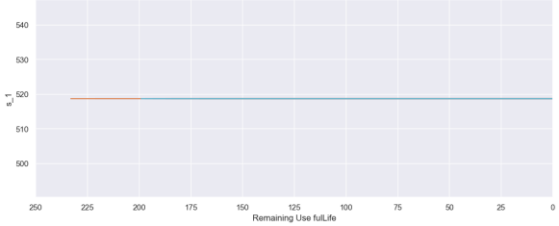


Figura 7 RUL vs S_1

El gráfico que se observa en Fig 7 se ve igual en los sensores 1, 10, 18 y 19 parece similar; la línea plana indica que los sensores no contienen información útil, lo que confirma nuestra conclusión a partir de las estadísticas descriptivas. Los sensores 5 y 16 también muestran una línea plana; se pueden agregar a la lista de sensores para excluir.

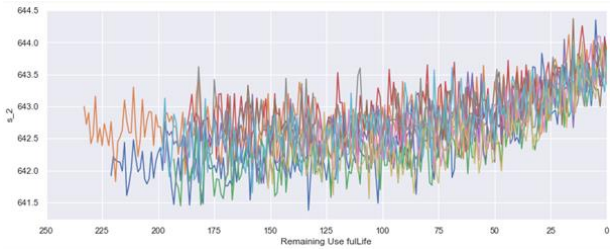


Figura 8 RUL VS S_2

El sensor 2 muestra una tendencia ascendente, un patrón similar se puede observar para los sensores 3, 4, 8, 11, 13, 15 y 17.

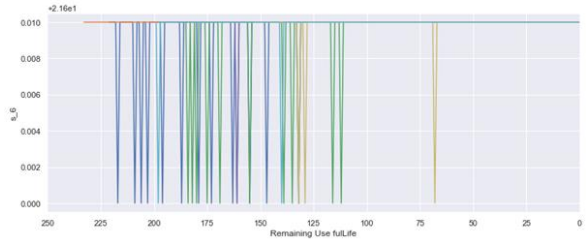


Figura 9 RUL vs S_6

Las lecturas del sensor 6 a veces alcanzan un pico descendente, pero no parece haber una relación clara con la disminución del RUL.

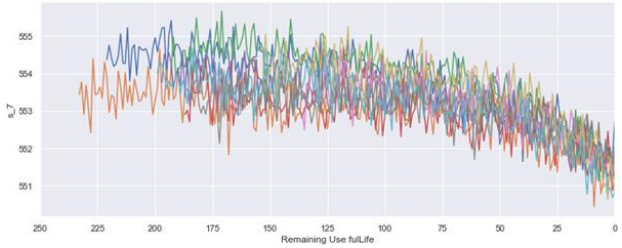


Figura 10 RUL vs S_7

El sensor 7 muestra una tendencia descendente, que también se puede observar en los sensores 12, 20 y 21.

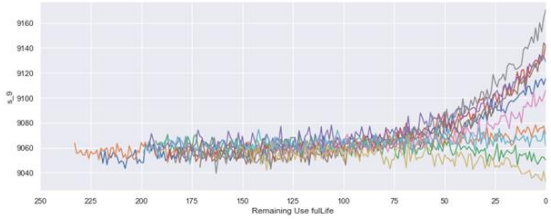


Figura 11 RUL vs S_9

En la Fig. 11 se observa que el sensor 9 tiene un similar patrón que el sensor 14.

Con base en nuestro análisis exploratorio de datos, podemos determinar que los sensores 1, 5, 6, 10, 16, 18 y 19 no contienen información relacionada con el RUL, ya que los valores de los sensores permanecen constantes a lo largo del tiempo. Comencemos el desarrollo de nuestro modelo con un modelo de regresión lineal de referencia. El modelo utilizará los sensores restantes como predictores.

E. Desarrollo del Modelo de Aprendizaje Automático

Durante el Desarrollo del Proyecto trabajé con varios modelos que se ajuste al escenario planteado y que permitan predecir el RUL con mayor precisión sin embargo en esta subsección hablaremos de tres modelos es el primero una regresión lineal básica que servirá de punto de partida para ver cómo funciona nuestras selección de características del dataset un segundo modelo llamado DLM nos enfocaremos tanto en la descripción del código si no únicamente en sus resultados y por el último el algoritmo que me dio mejores resultados LSTM.

1) Regresión lineal de referencia

Primero, definiremos una pequeña función para evaluar nuestros modelos. Elegí incluir el error cuadrático medio (RMSE), ya que dará una indicación de cuántos ciclos de tiempo se desvían las predicciones en promedio, y la varianza explicada (o puntuación R^2) para indicar qué proporción de nuestra variable dependiente puede explicarse por las variables independientes que usamos.

Del código droppearemos las características unit_nr, time_cycle, settings and sensors que no tienen información. La columna RUL del conjunto de entrenamiento se almacena en su propia variable. Para nuestro conjunto de pruebas dejemos las mismas columnas. Además, sólo nos interesa el último ciclo de cada motor en el conjunto de pruebas, ya que sólo tenemos valores de True RUL para esos registros.

```
# create and fit model
lm = LinearRegression()
lm.fit(X_train, y_train)

# predict and evaluate
y_hat_train = lm.predict(X_train)
evaluate(y_train, y_hat_train, 'train')

y_hat_test = lm.predict(X_test)
evaluate(y_test, y_hat_test)
✓ 0.0s
```

Figura 12 Modelo de regresión lineal de referencia

De la Fig 12 observamos que configurar la regresión lineal es bastante sencillo. Instalamos el modelo simplemente llamando al método y asignándolo a la variable "lm". A continuación, encajaremos con el modelo pasando nuestro "X-train". Finalmente, predecimos tanto en el tren como en el conjunto de pruebas para obtener la imagen completa de cómo nuestro modelo se está comportando con los datos que se le presentaron.

```
# returns
# train set RMSE:44.66819159545453, R2:0.5794486527796716
# test set RMSE:31.952633027741815, R2:0.40877368076574083
```

Figura 13 RMSE R2 Modelo de regresión básico

Tenga en cuenta que el RMSE es menor en el conjunto de prueba, lo que es contraintuitivo, ya que, por lo general, un modelo funciona mejor con los datos que ha visto durante el entrenamiento. Estos resultados es un punto de partida para ir mejorándolos, a continuación, describiré modelos más complejos que apliqué en este set de datos.

2) DLM

El Modelo de Rezago Distribuido (Distributed Lag Model, DLM) [8] es una técnica de regresión utilizada para analizar el impacto de una variable explicativa sobre una variable dependiente a lo largo del tiempo. Es particularmente útil en situaciones donde los efectos de un cambio en una variable no se sienten inmediatamente, sino que se distribuyen a lo largo de varios periodos de tiempo.

En un DLM, la relación entre la variable dependiente y la variable independiente se extiende a lo largo de múltiples periodos de tiempo pasados. En lugar de suponer que un cambio en una variable independiente tiene un impacto inmediato en la variable dependiente, el modelo captura cómo este impacto se distribuye en el tiempo.

$$Yt = \alpha + \beta 0Xt + \beta 1Xt - 1 + \beta 2Xt - 2 + \cdots + \beta kXt - k + \epsilon t \qquad (1)$$

Donde:

- Yt: Variable dependiente en el tiempo t.
- Xt: Variable independiente en el tiempo t.
- β0,β1,…,βk : Coeficientes del rezago (miden el impacto en el tiempo actual y en los periodos previos).
- k: Número de rezagos.
- ϵt: Término de error

Para esta etapa solo mencionare que se utilizó un rezago de 9, para un modelo muy estacionario es decir sus variables como la media y la varianza, no cambian con el tiempo, además la covarianza (extensión de las series de tiempo) no debe depender del tiempo. Adicional cabe indicar que el numero de rezagos usados se baso mejorando metricas como AIC y BIC [9], una vez aplicado el modelo los resultados fueron:

```
train set RMSE:20.742950740267165, R2:0.7542608298904978
test set RMSE:20.852234864407475, R2:0.7482058292992066
```

Figura 14 Metricas DLM

Como se observa en la Fig 14. Un resultado para el set de datos de entrenamiento de 20.85 no está mal representa una mejoría respecto al modelo de referencia sin embargo mejoraremos estos resultados en el siguiente modelo LSTM.

3) LSTM

Los **LSTM (**Long Short-Term Memory**) [4]** son una versión avanzada de las RNN que puede recordar información a largo plazo y superar el problema del "desvanecimiento del gradiente", lo que las hace especialmente útiles para datos con dependencias de largo plazo.

Continuando con el código realice lo mismo que para la regresión lineal y el modelo anterior que es extraer las características a no utilizar como los sensores y las configuraciones del motor.

Para el desarrollo del algoritmo se los datos se escalan para mejorar el rendimiento del modelo, y se crean secuencias temporales de una longitud de predefinida (en este caso,30). Para organizar los datos en forma de ventanas deslizantes para cada unidad como se observa en la Fig 15.

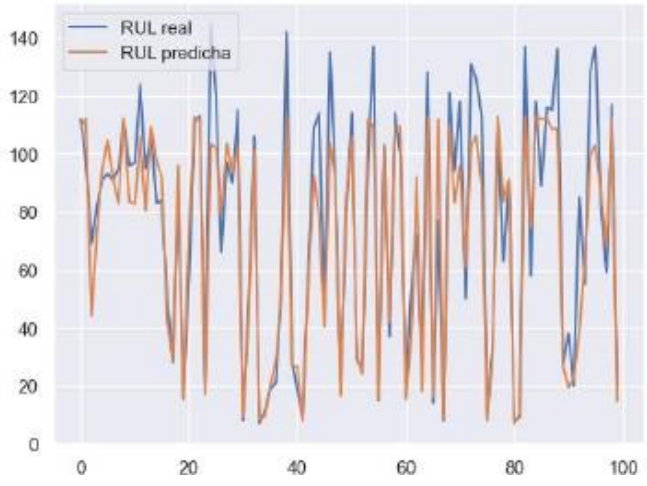
```python
# Crear secuencias de datos (ventanas temporales)
def create_sequences(data, seq_length, features):
    X = []
    y = []
    for unit in data['unit'].unique():
        unit_data = data[data['unit'] == unit]
        for start in range(len(unit_data) - seq_length):
            X.append(unit_data[features].iloc[start:start + seq_length].values)
            y.append(unit_data['RUL'].iloc[start + seq_length - 1])
    return np.array(X), np.array(y)

# Definir características y longitud de secuencia
sequence_length = 30
features = train_data_scaled.columns[:-2]  # Excluyendo 'unit', 'time', 'RUL'

# Crear las secuencias
X_train, y_train = create_sequences(train_data_scaled, sequence_length, features)
```

✓ 5.5s

Figura 15 creador se secuencias LSTM

Luego de preparar los datos y generar las secuencias entrenamos el modelo LSTM el cual fue generado mediante dos capas de preprocesamiento la primera con 100 neuronas y la segunda con 50 para un optimizador Adam para el modelo tomando en cuenta que el costo computacional en este modelo también fue mayor, y en la etapa de entrenamiento se ejecutó con 30 épocas, eso, en resumen.

En fin, se obtuvieron los siguientes resultados para este modelo.



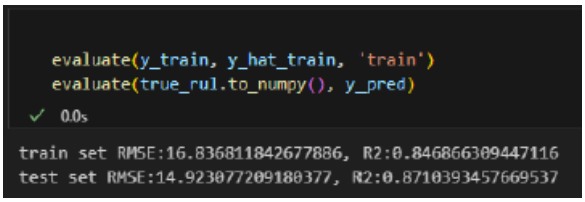


Figura 16 Resultados LSTM

Como se pudo observar en la Fig. 16 obtenemos la grafica de la RUL real que ya viene en el dataset con respecto a la RUL predicha con el modelo dando como resultado un RMSE de 14.92 que ya es una mejora con respecto a los anteriores modelos.

F. Resultados

- **Análisis de Resultados:** Los resultados hemos vistos que al aplicar modelos mas complejos las métricas como el RMSE y R2 mejoraron, con el modelo que mejor resultados vimos fue con LSTM sin embargo tenemos mucho por mejorar algo que se ira realizando a medida que avancemos en el proyecto

V. CONCLUSIÓN

Con el modelo LSTM Se obtuvieron mejores predicciones con valores de RMSE:14.92, R2:0.871 que no es lo ideal pero se acerca a lo buscado.

VI. RECOMENDACIONES Y MEJORAS

Finalmente, para garantizar el éxito a largo plazo del modelo, es recomendable seguir estos pasos adicionales:

Mantener una documentación detallada de todos los procesos, modelos y resultados. Comunica los avances y resultados del proyecto de manera clara y regular a todas las partes interesadas.

Probar la configuración de hiperparámetros para el modelo que mejor resultados ha dado tomando en cuenta también conceptos como la ingeniería característica para mejorar los resultados del modelo.

Explorar o Investigar otros modelos que se pueden adaptar al la necesidad del problema planteado como son los tranformers que actualmente están dando mucho que hablar para análisis secuencial o de series de tiempo.

VII. QR LIGHTNING

El siguiente código QR tiene el link del código en lightning una vez escaneado accederás a la pagina

Figura 17 https://lightning.ai/jgarzong/studios/proyectofinal

VIII. REFERENCIAS

[1] R. Salgotra, U Singh and N. Mitta, “Self-adaptive salp swarm algorithm for engineering optimization problems” Code Ocean, Aug. 2020.[Online]. Available:https://www.sciencedirect.com/science/article/abs/pii/S0307904X20304431

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475, Dec. 1998. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2475.txt

[3] Kaggle, "Nassa Turbofan Dataset", available at: https://www.kaggle.com/datasets/behrad3d/nasa-cmaps.

[4] Y. Zhang, R. Xiong, H. He and M. G. Pecht, "Long Short-Term Memory Recurrent Neural Network for Remaining Useful Life Prediction of Lithium-Ion Batteries," in IEEE Transactions on Vehicular Technology, vol. 67, no. 7, pp. 5695-5705, July 2018, doi: 10.1109/TVT.2018.2805189.

[5] Papers published on NASA’s CMAPSS data in 2020 so far: Google scholar search, accessed on 2020–08–08.

[6] J. Garzón, "Proyecto final," Lightning AI, [Online]. Available: https://lightning.ai/jgarzong/studios/proyectofinal. [Accessed: 12-Sep-2024].

[7] NASA, "Intelligent Systems Division," [Online]. Available: https://data.nasa.gov/dataset/C-MAPSS-Aircraft-Engine-Simulator-Data/xaut-bemq/about_data [Accessed: 12-Sep-2024]

[8] S. W. Carter, D. M. Higdon, J. A. Gattiker, and M. A. Williams, "Bayesian Melding for Predicting Multivariate Time Series with Missing Values," *Biometrics*, vol. 69, no. 2, pp. 537-545, Jun. 2013. [Online]. Available: https://academic.oup.com/biometrics

[9] A. Krishnakumar, "Time series analysis for predictive maintenance of turbofan engines," Towards Data Science, [Online]. Available: https://towardsdatascience.com/time-series-analysis-for-predictive-maintenance-of-turbofan-engines-1b3864991da4. [Accessed: 12-Sep-2024].