Capítulo 7

Aplicaciones de redes neuronales recurrentes en el análisis cuantitativo de series temporales para la gestión sostenible de recursos naturales y medioambientales

Gilberto Bojórquez-Delgado
Jesús Bojórquez-Delgado
Adalid Graciano-Obeso

[8] https://doi.org/10.61728/AE24250085

Introducción

La gestión eficiente y sostenible de los recursos naturales y medioambientales es uno de los mayores retos que enfrenta la humanidad en el siglo XXI (Rodríguez Morales et al., 2011). Este desafío se ve exacerbado por la complejidad inherente a los sistemas naturales, los patrones climáticos en constante cambio y la necesidad urgente de adaptar nuestras prácticas a una realidad ambiental que evoluciona rápidamente (Jaquenod De Zsögön, 2019). En este panorama, el análisis de series temporales emerge como una herramienta crítica, esencial para entender, predecir y manejar fenómenos ambientales con una precisión sin precedentes. Aquí, la justificación para centrarse en las Redes Neuronales Recurrentes (RNR) se hace evidente, ya que estas poseen una capacidad única para procesar y aprender de datos temporales, ofreciendo un enfoque innovador y más efectivo para enfrentar estos desafíos ambientales complejos.

Los antecedentes del análisis cuantitativo en la ciencia medioambiental tradicionalmente han dependido de modelos estadísticos y matemáticos que, aunque útiles, a menudo se ven limitados por supuestos de linealidad y estacionariedad (Quiguiri Daquilema, 2023). Estas limitaciones son particularmente problemáticas en el contexto de fenómenos naturales, donde la no linealidad y los patrones temporales complejos son la norma (Alonso, 2022). Las RNR, que se benefician de la capacidad de las redes neuronales para aprender de los datos sin la necesidad de supuestos previos, representan una evolución natural de estas metodologías, proporcionando un marco más flexible y robusto para el análisis medioambiental (Vega Moreno, 2021).

Dentro del espectro de las RNR, las Redes de Memoria a Largo Plazo (LSTM por sus siglas en inglés) se destacan por su capacidad para evitar el problema del desvanecimiento del gradiente (Guamán Pachacama y Segura Muñoz, 2021), lo que les permite no solo capturar relaciones a corto plazo, sino también aprender dependencias de largo plazo (Torres, 2021). Esta característica las hace particularmente adecuadas para modelar y predecir series temporales ambientales que requieren la comprensión de efectos acumulativos y tendencias a largo plazo (Salazar Molina, 2023)por lo que los gobiernos del mundo entero se han comprometido a disminuir estas emisiones para lograr el objetivo 7 del desarrollo sostenible 2030 de las Naciones Unidas: Energía

asequible y no contaminante. [Resumen truncado para la entrada .bib]",”genre”:”Doctoral thesis”,”publisher”:”Universidad Internacional Iberoamericana México”,”title”:”Metodología para la Evaluación en Proyectos de Energía Solar Fotovoltaica”,”URL”:”https://repositorio.unini.edu.mx/id/eprint/6010”,”author”:[{“family”:”Salazar Molina”,”given”:”Erick Alonso”}],”issued”:{“date-parts”:[[“2023”,2]]}}],”schema”:”https://github.com/citation-style-language/schema/raw/master/csl-citation.json”} .

La arquitectura única de las LSTM les permite retener información por periodos prolongados, haciendo uso de estructuras llamadas “celdas de memoria” (Lindemann et al., 2021). Estas celdas regulan el flujo de información permitiendo que la red recuerde u olvide información de manera selectiva. Este mecanismo es fundamental cuando se trata de datos ambientales, donde la relevancia de la información puede variar drásticamente a lo largo del tiempo (Nguyen et al., 2021).

Dado el complejo entramado de datos que caracteriza al medioambiente, se hace imprescindible recurrir a herramientas computacionales avanzadas que puedan manejar y procesar eficientemente la información. Python emerge como una opción preeminente (Bobadilla, 2021), avalado por su rica colección de bibliotecas científicas y de aprendizaje automático como TensorFlow y Keras, que complementan y potencian la estructura de las LSTM (Pang et al., 2020). La magnitud y la complejidad de los datos medioambientales exigen un enfoque automatizado, dado que la capacidad de procesamiento manual es insuficiente. La implementación de Python en este contexto no solo facilita la gestión de grandes volúmenes de datos, sino que también optimiza la capacidad de las redes neuronales para aprender de patrones temporales extensos y variables (Joseph et al., 2021).

Además, la necesidad de iterar modelos, ajustar parámetros y validar resultados requiere de una plataforma que ofrezca no solo flexibilidad y potencia computacional sino también una comunidad activa y recursos de aprendizaje.

La elección de Python como herramienta para el análisis de RNR responde a estos requisitos, proporcionando un entorno propicio para el desarrollo rápido y eficiente de modelos predictivos robustos (Wang y Dowling, 2022) calibrating, and validating said science-based models often remains an art in practice. Model-based design of experiments (MBDoE. La simplicidad sintáctica de Python, junto con su capacidad para manejar operaciones de alto nivel, permite a los investigadores centrarse en la solución de problemas complejos sin la carga de detalles de programación de bajo nivel. Este enfoque permite trasladar los esfuerzos del análisis manual a uno automatizado, lo cual es esencial para manejar la escala y complejidad de los desafíos que presenta la

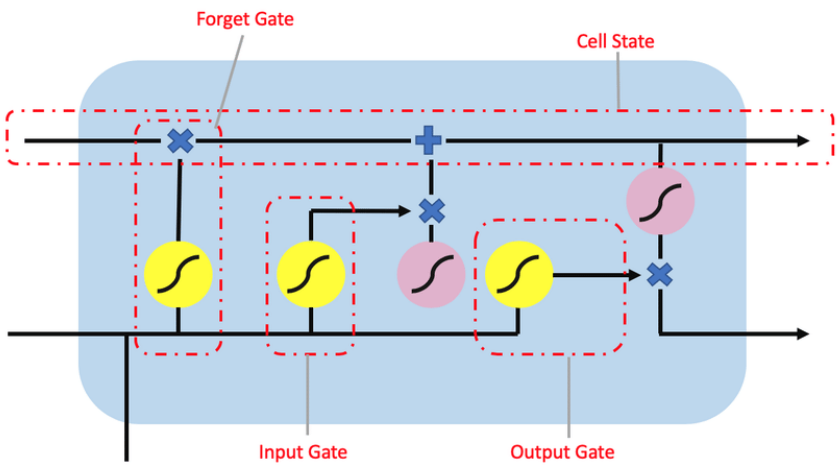
gestión sostenible de recursos naturales y medioambientales (Landázuri Páez et al., 2023). El objetivo de este capítulo es, por lo tanto, doble: en primer lugar, demostrar cómo las RNR y las LSTM pueden ser aplicadas para interpretar y predecir con precisión los patrones en los datos medioambientales. Esto se logra a través de una exploración detallada de su teoría subyacente, arquitectura y aplicabilidad práctica en escenarios reales. En segundo lugar, el capítulo busca inspirar y equipar a los profesionales medioambientales, científicos de datos y responsables políticos con el conocimiento y las herramientas necesarias para integrar estas tecnologías en la toma de decisiones relativas a la sostenibilidad.

Metodología

Fundamentos teóricos

La arquitectura de las RNR, específicamente las LSTM, es fundamental para modelar series temporales, gracias a su capacidad para aprender y recordar información a lo largo de periodos extensos (Veléz et al., 2021). La Figura 1 ilustra la arquitectura interna de una celda LSTM, la cual es esencial para la comprensión profunda del funcionamiento de las RNR aplicadas al análisis ambiental (Rengasamy et al., 2020).

Figura 1. Arquitectura de una Celda LSTM.



Fuente: (Rengasamy et al., 2020).

Puertas de Olvido (Forget Gates): La función de estas puertas es seleccionar la información del estado de la celda anterior que ya no es relevante para la secuencia actual y debe ser descartada. Esto es crítico en datos ambientales donde algunos factores, como los datos estacionales, pueden dejar de ser influyentes después de cierto tiempo.

Puertas de Entrada (Input Gates) y Estado de la Celda (Cell State): Las puertas de entrada deciden qué nueva información es relevante y debe ser añadida al estado de la celda. El estado de la celda actúa como un transportador de información relevante a lo largo de la secuencia de tiempo, siendo actualizado en cada paso.

Puertas de Salida (Output Gates): Estas puertas determinan qué información del estado de la celda contribuirá a la salida y será utilizada para las predicciones o decisiones del modelo.

Esta dinámica de puertas y estados permite a las LSTM manejar y ajustarse a la naturaleza no estacionaria de los datos ambientales, en los cuales los patrones y relaciones pueden cambiar o evolucionar con el tiempo. Al aplicar esta metodología, se puede capturar tanto la influencia de eventos recientes como la de aquellos acontecimientos más antiguos que todavía tienen un impacto en el estado actual del sistema ambiental estudiado.

Ecuación de Actualización del Estado de la Celda:

El estado de la celda en las LSTM es actualizado en cada paso de tiempo para mantener la información relevante a lo largo de la secuencia. La actualización se realiza mediante la siguiente ecuación:

$$C_t = f_t \cdot C_{\{t-1\}} + i_t \cdot \sim C_t$$

Donde:

C_t es el estado de la celda en el tiempo actual.

f_t es la activación de la puerta de olvido, determinando qué parte del estado anterior $C_{\{t-1\}}$ debe conservarse.

$C_{\{t-1\}}$ es el estado de la celda en el paso de tiempo anterior.

i_t es la activación de la puerta de entrada, que decide qué nueva información se añadirá al estado de la celda.

$\sim C_t$ es el estado de la celda candidato, generado a partir de la información actual, que podría añadirse al estado de la celda si la puerta de entrada así lo permite.

Implementación con Python y Google Colab:

La implementación de las LSTM para el análisis de series temporales ambientales se realiza mediante Python, aprovechando bibliotecas de aprendizaje automático como TensorFlow y Keras. Google Colab ofrece un entorno de computación en la nube que facilita la ejecución de modelos complejos de LSTM. Los investigadores pueden colaborar en tiempo real, compartiendo y ejecutando código en un entorno estandarizado que asegura la reproducibilidad de los resultados.

En el contexto del análisis ambiental, las LSTM se entrenan utilizando grandes conjuntos de datos históricos. Estos datos pueden incluir mediciones de temperatura, concentraciones de gases, niveles de precipitación, entre otros factores relevantes. La formación de estos modelos se orienta a predecir futuros estados ambientales y fenómenos con implicaciones críticas para la gestión de recursos y la planificación sostenible.

Preparación de datos

Recolección de datos temporales ambientales:

La recolección de datos precisos y relevantes es vital para el análisis efectivo y significativo de series temporales, especialmente en estudios ambientales. Estos datos, a menudo heterogéneos y voluminosos, pueden provenir de múltiples fuentes, como sensores remotos, estaciones meteorológicas, observaciones satelitales, y registros históricos.

- Sensores remotos: Utilizados para recoger datos a gran escala, como imágenes satelitales y mediciones de radar, que pueden proporcionar información sobre la cobertura terrestre, la temperatura superficial y los patrones climáticos.
- Registros históricos: Datos acumulados a lo largo del tiempo, a menudo disponibles a través de bases de datos gubernamentales o investigaciones científicas, que pueden incluir temperaturas pasadas, niveles de precipitación y otras mediciones meteorológicas.
- Mediciones In Situ: Mediciones directas tomadas en el campo, como niveles de contaminantes en un lago o concentraciones de CO2 en un bosque, que proporcionan datos detallados a nivel local.

Preprocesamiento de datos:

El preprocesamiento es un paso crítico que prepara los datos brutos para su análisis, asegurando que sean limpios, normalizados y estructurados adecuadamente para la entrada en el modelo LSTM. Esto implica técnicas como la imputación de datos faltantes, la detección y corrección de anomalías, y la transformación de datos para facilitar la extracción de características temporales relevantes para la predicción (García et al., 2016).

1. Limpieza de datos:

- Se identifican y manejan valores faltantes, que pueden ser rellenados utilizando métodos como la imputación media o la interpolación, dependiendo del contexto y la naturaleza de los datos.
- Se eliminan outliers o valores atípicos que podrían distorsionar el entrenamiento del modelo.
- Se corrigen errores de entrada y se verifica la consistencia de los datos.

2. Normalización de datos:

- Se aplican técnicas de normalización para escalar los datos dentro de un rango adecuado, generalmente entre 0 y 1, utilizando métodos como la normalización min-max o la estandarización Z-score.
- Se asegura de que todas las características (variables de entrada) estén en la misma escala para que el modelo no se sesgue hacia características con valores numéricos más altos.

3. Transformación de datos:

- Se convierten datos no numéricos en formatos numéricos mediante codificación, si es necesario, como la codificación one-hot para variables categóricas.
- Los datos se estructuran en secuencias que las LSTM pueden procesar. Por ejemplo, si se están prediciendo niveles diarios de contaminación, se organizan en ventanas de tiempo, como secuencias de 7 días para predecir el octavo día.

4. División de datos:

- Se dividen los datos en conjuntos de entrenamiento, validación y prueba. Un enfoque común para esta división es el temporal, donde los datos más antiguos se utilizan para el entrenamiento, y los más recientes para la prueba. Este método asegura que el modelo se entrene con datos históricos y sea evaluado en un contexto más actual, lo que puede ser crucial en aplicaciones donde las tendencias y patrones cambian con el tiempo.
- Se realiza la validación del modelo utilizando una porción del conjunto de entrenamiento o aplicando técnicas de validación cruzada. Este proceso es crucial para evaluar la capacidad de generalización del modelo, asegurando que su desempeño sea robusto y confiable frente a datos nuevos y no vistos durante el entrenamiento.

Diseño del modelo

Configuración de la Arquitectura LSTM

La configuración de una red LSTM es un paso crítico que requiere un enfoque reflexivo y experimental para encontrar la arquitectura óptima que se adapte a la complejidad de los datos y la problemática específica. Este proceso se desglosa en dos componentes esenciales que se describen a continuación:

1. Estructuración de la Red LSTM

- Número de capas y unidades: Se recomienda iniciar con una sola capa LSTM para establecer una línea de base, aumentando progresivamente el número de capas y unidades si es necesario para capturar complejidades adicionales en los datos.
  - Comando Python: keras.layers.LSTM(units=50, return_sequences=True) para una capa LSTM con 50 unidades y salida secuencial.
- Funciones de activación: La función tanh es comúnmente usada para las activaciones internas de la celda LSTM, mientras que la función sigmoid es típica para las puertas de olvido, entrada y salida.
  - Comando Python: keras.layers.LSTM(activation='tanh', recurrent_activation='sigmoid').

2. Configuración de parámetros

- Tasa de aprendizaje (Learning rate): Una tasa de aprendizaje más pequeña puede ser necesaria para entrenamientos más estables, aunque puede hacer que el entrenamiento sea más lento.
  - Comando Python: keras.optimizers.Adam(learning_rate=0.001) para configurar el optimizador Adam con una tasa de aprendizaje de 0.001.
- Número de épocas: El número de épocas debe ser lo suficientemente grande para permitir que la red aprenda de los datos, pero no tan grande que cause sobreajuste.
  - Comando Python: model.fit(X, y, epochs=100) para entrenar el modelo durante 100 épocas.
- Tamaño de los Lotes (Batch Size): Un tamaño de lote más pequeño a menudo proporciona una estimación del gradiente más ruidoso, lo que puede ayudar a evitar mínimos locales, pero también puede hacer que el entrenamiento sea más variable.
  - Comando Python: model.fit(X, y, batch_size=32) para entrenar el modelo con un tamaño de lote de 32.

Consideraciones adicionales:

Además de la estructuración básica y la configuración de parámetros en una red LSTM, hay varias consideraciones adicionales que pueden influir significativamente en el rendimiento y la eficiencia del modelo. Estas prácticas avanzadas son esenciales para refinar aún más la red y asegurar su robustez frente a desafíos comunes como el sobreajuste. A continuación, se detallan algunas de estas consideraciones clave:

- Regularización: Para evitar el sobreajuste, se debe considerar usar técnicas como dropout o regularización L1/L2 en tu red.
  - Comando Python: keras.layers.LSTM(units=50, dropout=0.2) para aplicar un dropout del 20 % en la capa LSTM.
- Inicialización de Pesos: Se considera una práctica recomendada la inicialización adecuada de los pesos de la red para mejorar el entrenamiento.
  - Comando Python: keras.layers.LSTM(units=50, kernel_initializer='glorot_uniform') para usar la inicialización de Glorot/Xavier.

Al diseñar el modelo LSTM, es esencial realizar una búsqueda de hiperparámetros, ya sea manualmente o mediante técnicas automatizadas como la

búsqueda en cuadrícula (grid search) o la optimización bayesiana. Además, es esencial asegurar el monitoreo de métricas de rendimiento clave y la visualización del entrenamiento para realizar ajustes informados a la arquitectura y configuración del modelo. Este proceso involucra ajustar aspectos como el número de capas LSTM, la tasa de aprendizaje, y el número de épocas, para maximizar la precisión de las predicciones. La interpretación de los resultados y la retroalimentación iterativa entre los pasos de preprocesamiento y modelado son fundamentales para refinar el modelo y alcanzar resultados óptimos.

Entrenamiento y validación

El proceso de entrenamiento es fundamental para la construcción de un modelo preciso y fiable. La manera en que se entrena una red LSTM es crucial para su capacidad de generalización y rendimiento en datos no vistos previamente.

Una vez establecido el marco general del proceso de entrenamiento, es crucial abordar aspectos específicos como la división de datos y las estrategias para prevenir el sobreajuste, que son fundamentales para garantizar la efectividad y precisión del modelo LSTM.

1. División de datos

- Antes del entrenamiento, es importante dividir el conjunto de datos en tres partes: entrenamiento, validación y prueba.
  - ○ Comando Python: train_test_split() de la biblioteca sklearn.model_selection para realizar la división.
- La porción de entrenamiento se utiliza para ajustar los pesos del modelo, la validación para afinar los hiperparámetros y evitar el sobreajuste, y la prueba para evaluar la capacidad predictiva del modelo en datos nuevos.

2. Prevención del sobreajuste

- Técnicas como la regularización y el dropout son esenciales para prevenir el sobreajuste.
  - ○ Comando Python: keras.layers.LSTM(units=50, dropout=0.2) para aplicar dropout a la capa LSTM.
  - ○ Comando Python: keras.regularizers.l1(0.01) o keras.regularizers.l2(0.01) para aplicar regularización L1 o L2 a los pesos de la red.
- Es vital monitorear la pérdida de entrenamiento y validación durante el entrenamiento para detectar señales de sobreajuste. Si la pérdida de validación

aumenta mientras la de entrenamiento disminuye, se puede estar ante un caso de sobreajuste.

Evaluación del modelo

1. Métricas de rendimiento
- Una vez entrenado el modelo, se utiliza el conjunto de prueba para evaluar su rendimiento mediante métricas estándar.
  - Comando Python: model.evaluate() para obtener la pérdida y las métricas del modelo.
- El Error Cuadrático Medio (MSE) mide la media de los cuadrados de los errores, siendo una métrica común para problemas de regresión.
- El Coeficiente de Determinación (R^2) indica la proporción de la varianza de la variable dependiente que es predecible a partir de las variables independientes.
  - Comando Python: sklearn.metrics.mean_squared_error() y sklearn.metrics.r2_score() para calcular MSE y R^2, respectivamente.

2. Validación cruzada
- Para garantizar la robustez del modelo, se puede emplear la validación cruzada, especialmente útil en conjuntos de datos más pequeños o cuando se requiere una evaluación exhaustiva del modelo.
  - Comando Python: sklearn.model_selection.KFold() o sklearn.model_selection.TimeSeriesSplit() para la validación cruzada, dependiendo de si los datos pueden ser aleatorizados o no (TimeSeriesSplit es para series temporales donde el orden es importante).

Implementación práctica con Python y Colab

Python se ha establecido como uno de los lenguajes de programación más versátiles y ampliamente utilizados en el campo del aprendizaje automático y el análisis de datos. Su simplicidad sintáctica, junto con un extenso ecosistema de bibliotecas y herramientas, lo hacen ideal para implementar y experimentar con modelos de aprendizaje profundo como las redes LSTM.

Google Colab es una plataforma en la nube que facilita la escritura y ejecución de código Python, especialmente útil para el aprendizaje profundo. Se puede configurar rápidamente y es compatible con bibliotecas como TensorFlow y Keras. A continuación, se presentan los pasos clave para iniciar con Colab y la programación en Python.

1. Uso de Google Colab

Google Colab es un servicio de nube basado en Jupyter Notebooks que facilita la escritura y ejecución de código Python. Ofrece un entorno colaborativo con acceso gratuito a recursos computacionales, incluyendo GPUs y TPUs, lo que lo hace ideal para entrenar modelos de aprendizaje profundo.
Configuración de Colab:

- Debe iniciarse un nuevo notebook en Colab y cargar las bibliotecas necesarias con el comando !pip install tensorflow keras.
- El comando from google.colab import drive, seguido de drive.mount('/content/drive'), se utiliza para acceder y cargar datos desde Google Drive si es necesario.
- Es importante asegurarse de seleccionar la configuración de hardware adecuada en Colab, como la GPU, para acelerar el entrenamiento del modelo.

2. Codificación en Python

El desarrollo de modelos LSTM se realiza con TensorFlow y Keras, dos de las bibliotecas más populares para el aprendizaje profundo, que proporcionan abstracciones de alto nivel para diseñar y entrenar redes neuronales.

- Implementación de un Modelo LSTM:
  - El modelo se define utilizando la clase Sequential de Keras, y para agregar una capa LSTM, se utiliza model.add(LSTM(units=50, return_sequences=-True)), ajustando las unidades según la complejidad del problema.
  - El modelo se compila con el comando model.compile(optimizer='adam', loss='mean_squared_error'), seleccionando un optimizador y una función de pérdida adecuada para tu tarea.
- Visualización de resultados:
  - Se emplea matplotlib o seaborn para visualizar los resultados, como la pérdida durante el entrenamiento, utilizando plt.plot(history.history['loss']) y plt.plot(history.history['val_loss']).
  - Para la interpretación de las salidas del modelo se examinan las predicciones en comparación con los datos reales y ajusta el modelo si es necesario. Utiliza técnicas de posprocesamiento para analizar errores, ajustar hiperparámetros y probar diferentes arquitecturas de red.

La implementación práctica de las LSTM mediante Python y Colab facilita un avance significativo en la comprensión y predicción de series temporales

complejas. El proceso meticuloso de recolección y preprocesamiento de datos, seguido por una cuidadosa configuración y entrenamiento del modelo, es esencial en la capacidad de predecir dinámicas temporales con gran relevancia para el sector del campo de estudio.

Ejercicio práctico:

Aplicación de redes neuronales recurrentes para el análisis de series temporales en la gestión sostenible de recursos energéticos renovables.

En esta sección, se aborda el desafío de analizar y predecir el consumo de energía renovable, un componente clave en la gestión sostenible de los recursos naturales. Para ello, se utiliza el conjunto de datos *WorldSustainabilityDataset.* csv, obtenido del reconocido repositorio en línea Kaggle. Este amplio conjunto de datos encapsula métricas de sostenibilidad de 173 países a lo largo de 19 años, reflejando la diversidad y complejidad de los desafíos a nivel mundial en sostenibilidad. Compilado originalmente para el TrueCue Women+Data Hackathon, el dataset incluye 54 variables variadas, proporcionando un espejo de la riqueza y la heterogeneidad de los datos ambientales globales.

El análisis se centra en un subconjunto específico: el consumo de energía renovable en México. La elección de esta área focal representa una oportunidad valiosa para demostrar cómo las metodologías rigurosas de análisis de series temporales, con especial énfasis en las RNR, pueden ser aplicadas para obtener insights significativos en el contexto de la sostenibilidad medioambiental. Los datos analizados presentan desafíos típicos encontrados en entornos profesionales, tales como valores faltantes y registros incompletos. Estos desafíos no solo ponen a prueba la robustez de las herramientas analíticas empleadas, sino que también refuerzan la necesidad de enfoques de análisis cuidadosos y considerados. A través de este caso de estudio, se ilustrará la aplicación práctica de las RNR, destacando su utilidad en el procesamiento y modelado de datos complejos para la toma de decisiones informadas en la gestión de recursos naturales. A continuación, se presentan los pasos para realizar el estudio y predicción del comportamiento del consumo de energía renovable en México. Tomando como base los datos históricos desde el año 2000 hasta 2015, se genera una ventana de tres años para predecir el consumo hasta 2018 y compararlo con los datos reales de esa fecha.

1. Importación de librerías y definición de funciones

En Google Colab, se comienza importando todas las librerías necesarias y se definen las funciones que se utilizarán. La función *split_sequence* convierte los datos de series temporales en un formato que las redes LSTM pueden procesar, tal como se muestra en el siguiente fragmento de código:

```
import pandas as pd
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dropout, Dense
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        end_ix = i + n_steps
        if end_ix > len(sequence)-1:
            break
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return np.array(X), np.array(y)
```

En el código se muestra:

- pandas y numpy son librerías estándar para la manipulación de datos y operaciones matemáticas.
- De tensorflow.keras, se debe importar *Sequential, LSTM, Dropout, y Dense* para construir nuestra red neuronal, y *EarlyStopping* para evitar el sobreajuste.
- matplotlib.pyplot se utiliza para la visualización de datos.

La función *split_sequence* es crucial para preparar los datos. Convierte una secuencia de valores en un conjunto de pares de entrada/salida que la red puede aprender.

2. Carga y filtrado de datos

A continuación, se carga el conjunto de datos, los registros específicos para México son seleccionados, enfocándose en el consumo de energía renovable, en el siguiente código se muestra la carga de los datos.

```
data = pd.read_csv('WorldSustainabilityDataset.csv')
mexico_data = data[(data['Country'] == 'Mexico') & (data['Year'] >= 2000)]
mexico_data = mexico_data[['Year', 'RenewableEnergyConsumption']].dropna()
```

Se utiliza pd.read_csv() para cargar los datos desde un archivo CSV. Posteriormente, se filtran los datos para México utilizando operaciones de indexación booleana en Pandas. La función .dropna() se emplea para eliminar cualquier fila con valores faltantes, asegurando así la integridad de los datos.

3. Preparación de los datos

Se utiliza pd.read_csv() para cargar los datos desde un archivo CSV. Luego, los datos se filtran para México empleando operaciones de indexación booleana en Pandas. La función dropna() se aplica para eliminar cualquier fila con valores faltantes, con el fin de asegurar la integridad de los datos. A continuación, se transforma la serie temporal en un conjunto de datos adecuado para el entrenamiento de la red LSTM, dividiendo los datos en conjuntos de entrenamiento y prueba, como se muestra en el siguiente fragmento de código:

```
years = mexico_data['Year'].values
consumption = mexico_data['RenewableEnergyConsumption'].values
n_steps = 3
X, y = split_sequence(consumption, n_steps)
n_features = 1
X = X.reshape((X.shape[0], X.shape[1], n_features))
```

Aquí, se preparan las secuencias de datos para el entrenamiento, n_steps define el número de pasos temporales que se considerarán para predecir el siguiente valor, n_features representa el número de atributos utilizados para la predicción en cada paso de tiempo, que en este caso es solo uno: el consumo de energía renovable. La operación reshape modifica la forma de X para ajustarse a lo que la capa LSTM espera: un tensor tridimensional de la forma [*samples*, *timesteps*, *features*].

4. Construcción del modelo LSTM

El modelo LSTM es creado para ser ajustado a los datos. Se añaden capas de LSTM y Dropout para modelar la secuencia y prevenir el sobreajuste, como se ilustra en el siguiente fragmento de código:

```
    model = Sequential()
    model.add(LSTM(100, activation='relu', return_sequences=True, input_shape=(n_steps,
n_features)))
    model.add(Dropout(0.2))
    model.add(LSTM(100, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse', metrics=['mse', 'mae', 'mape', 'ac-
curacy'])
```

La red se inicia con una capa Sequential, que facilita el apilamiento de capas de red una tras otra. Cada capa LSTM contiene 100 unidades y utiliza la función de activación *relu*. La capa Dropout, con un valor de 0.2, descarta aleatoriamente el 20 % de las características durante el entrenamiento para combatir el sobreajuste. La capa Dense al final emite la predicción de un solo valor continuo. Finalmente, se compila el modelo utilizando el optimizador *adam* y la función de pérdida de MSE.

5. Entrenamiento y evaluación del modelo

El modelo es entrenado utilizando los datos, utilizando técnicas de validación y Early Stopping para optimizar el proceso de aprendizaje y prevenir el sobreajuste. Posteriormente, se evalúa su rendimiento, como se muestra en el siguiente fragmento de código:

```
    early_stopping = EarlyStopping(monitor='val_loss', patience=20, restore_best_
weights=True)
    history = model.fit(X_train, y_train, epochs=200, validation_split=0.2, callbacks=[ear-
ly_stopping], verbose=1)
    mse, mae, mape, acc = model.evaluate(X_test, y_test, verbose=0)
```

EarlyStopping, una función de *Keras*, se utiliza para interrumpir el entrenamiento cuando una métrica monitoreada, en este caso la pérdida de validación (*val_loss*), deja de mejorar. La propiedad *patience* establece el número de épocas que se esperan sin mejora antes de detener el entrenamiento. Se emplea *model.fit()* para entrenar el modelo, especificando los datos de entrenamiento y validación, así como el número de épocas. Después del entrenamiento, el modelo se evalúa con *model.evaluate()*, obteniendo métricas como el MSE, el error absoluto medio (MAE), el error porcentual absoluto medio (MAPE), y la precisión.

6. Visualización de resultados

Finalmente, se calcula y se muestra el rendimiento del modelo utilizando la biblioteca matplotlib, como se muestra en el siguiente fragmento de código:

```
predictions = model.predict(X_test, verbose=0)
test_years = years[n_train+n_steps:]
real_consumption = y_test
predicted_consumption = predictions.flatten()

plt.figure(figsize=(12,6))
plt.plot(years[:n_train], consumption[:n_train], label='Entrenamiento')
plt.plot(test_years, real_consumption, label='Real', color='orange')
plt.plot(test_years, predicted_consumption, label='Predicción', linestyle='--', color='green')
plt.title('Consumo de Energía Renovable en México')
plt.ylabel('Consumo de Energía Renovable')
plt.xlabel('Año')
plt.legend()
plt.show()
```

Aquí, se generan las predicciones del modelo y se preparan los datos de tiempo y consumo para la visualización. Se utiliza *plt.plot()* para trazar los datos de entrenamiento, los datos reales de prueba y las predicciones. Este gráfico ayudará a visualizar cómo las predicciones del modelo se alinean con los valores reales, proporcionando una representación intuitiva de la precisión del modelo.

El ejemplo ilustra meticulosamente la secuencia de pasos cruciales en la aplicación de las RNR para desentrañar el patrón del consumo de energía renovable en México. Se inicia con la configuración de un entorno de programación en Google Colab, seguido de la importación de bibliotecas de análisis de datos y aprendizaje automático. Posteriormente, se realiza la recolección de datos ambientales específicos de México, enfocándose en el consumo de energía renovable, y se prepara el conjunto de datos a través de métodos de limpieza, normalización y transformación para su posterior análisis con la red LSTM.

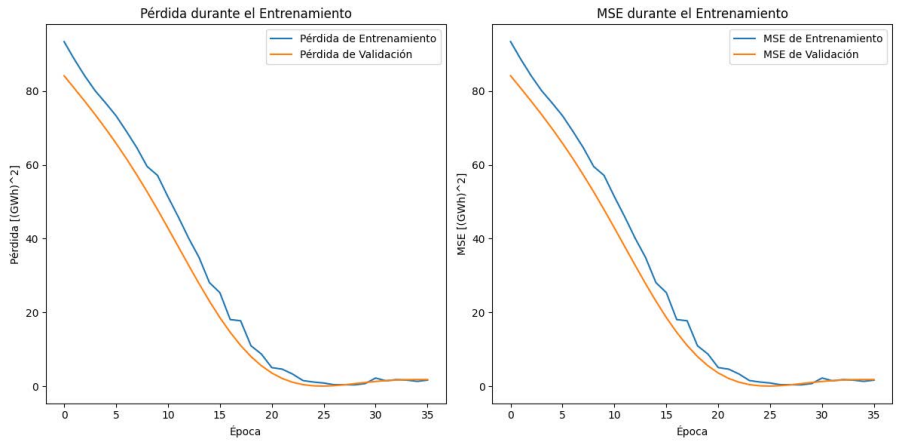
El proceso continúa con la estructuración de los datos en formatos adecuados para las LSTM y la construcción de un modelo de red neuronal que incorpora capas LSTM y Dropout, diseñado para capturar tanto las dependencias a corto como a largo plazo en los datos temporales. La configuración de los parámetros del modelo, la selección de funciones de activación y la inicialización de pesos, se realizan de manera reflexiva para adaptarse a la complejidad y naturaleza de los datos.

Resultados y discusión

Análisis de la evolución del aprendizaje del modelo

Esta sección presenta una evaluación exhaustiva de los resultados obtenidos a través del modelo propuesto. Se detalla la eficacia del algoritmo, sus patrones de aprendizaje y la relevancia de las métricas obtenidas en el marco del estudio. La figura 2 ilustra la evolución del aprendizaje del modelo a lo largo de las épocas de entrenamiento, proporcionando información crítica sobre la eficacia del proceso de aprendizaje automático y la capacidad de generalización del modelo. Las dos gráficas que componen la figura exhiben la dinámica de la pérdida y el error cuadrático medio (MSE), que son indicadores clave del progreso y la precisión del modelo en la tarea en cuestión.

Figura 2. Evaluación y evolución del rendimiento y aprendizaje del modelo.



Fuente: Elaboración propia.

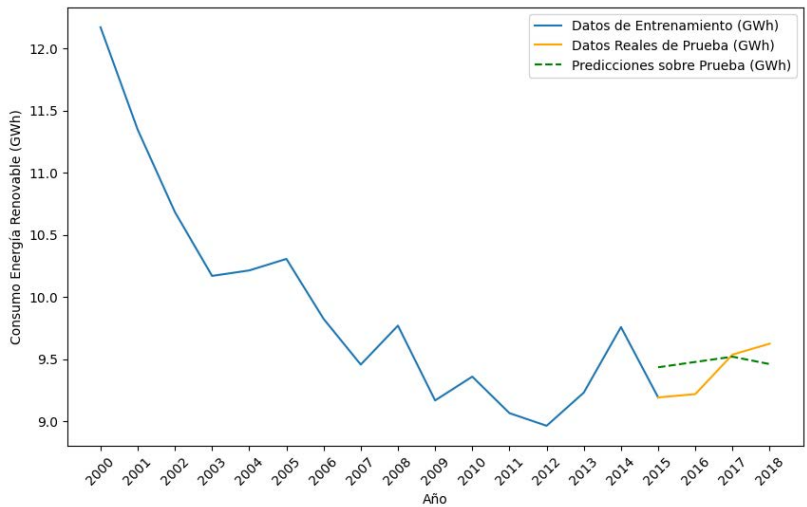
La gráfica izquierda de la figura 2 muestra la curva de pérdida durante el proceso de entrenamiento del modelo. La pérdida de entrenamiento y la pérdida de validación disminuyen conforme se suceden las épocas, una tendencia que es indicativa de un aprendizaje efectivo. La pérdida se mide generalmente como la discrepancia entre las predicciones del modelo y los valores verdaderos de los datos. Si las líneas comenzaran a divergir, con la pérdida de validación aumentando o estancándose mientras la de entrenamiento sigue disminuyendo,

sería indicativo de sobreajuste (overfitting). En la gráfica derecha, se muestra el MSE (Mean Squared Error) durante el entrenamiento, que es una medida cuantitativa del error del modelo. Al igual que con la gráfica de pérdida, una disminución en el MSE para los conjuntos de entrenamiento y de validación es deseable y sugiere un aprendizaje adecuado.

Comparación de predicciones frente a datos reales

El análisis detallado de las proyecciones del modelo se visualiza en la figura 3, la cual ofrece una comparativa entre las proyecciones del modelo y los datos reales de consumo de energía renovable, poniendo a prueba su precisión predictiva.

Figura 3. Análisis de tendencias de consumo real vs predicciones.



Fuente: Elaboración propia.

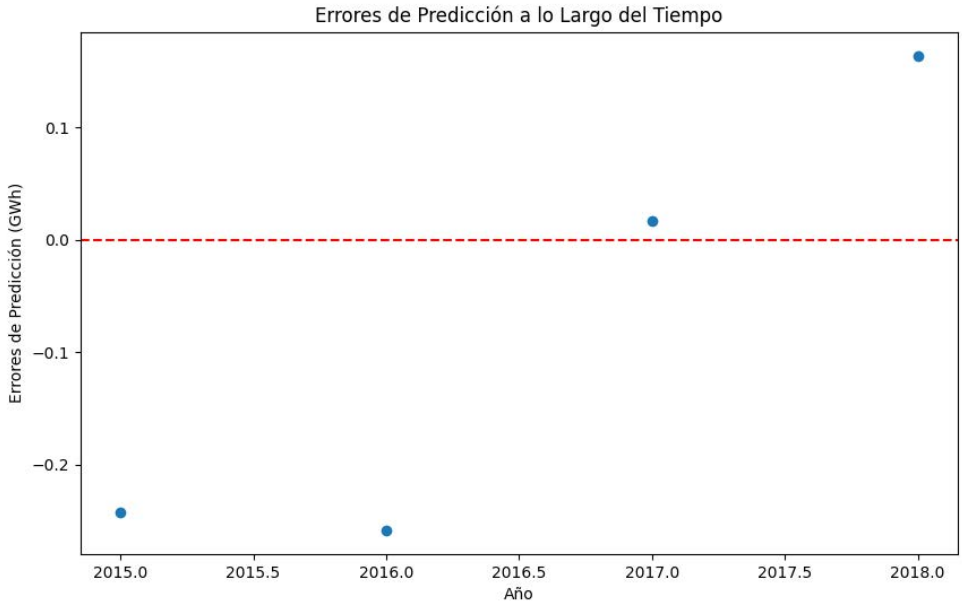
La correspondencia observada entre las predicciones y los valores reales durante el periodo de prueba no solo refleja la tendencia general sino también las variaciones específicas en el consumo a lo largo del tiempo, lo cual demuestra la capacidad del modelo para imitar el comportamiento histórico del consumo energético. Se observa que las predicciones replican un patrón de incremento seguido de un descenso, alineándose con las fluctuaciones inherentes a los datos históricos. Esta coherencia subraya la relevancia del modelo para anticipar tendencias futuras en la gestión de energía renovable. La línea de predicción

discontinua, que se mantiene cercana a los datos reales, sugiere un modelo adecuadamente ajustado. Sin embargo, la presencia de desviaciones en los puntos finales señala la necesidad de investigar la flexibilidad del modelo frente a variaciones inesperadas, lo que es fundamental para su aplicación en escenarios dinámicos de consumo energético.

Análisis de los errores de predicción

Finalmente, la figura 4 detalla los errores de predicción individuales a lo largo del tiempo, lo que nos permite evaluar en qué momentos y en qué medida el modelo se desvía de la realidad.

Figura 4. Errores de predicción del modelo.



Fuente: Elaboración propia.

Aunque hay errores en las predicciones, la mayoría de los puntos están cerca de la línea de error cero, lo que indica que las predicciones son generalmente precisas. Las desviaciones proporcionan una oportunidad para el análisis posterior, potencialmente señalando áreas donde el modelo podría ser refinado para mejorar la precisión.

Basado en las gráficas proporcionadas, el análisis del conjunto de datos *WorldSustainabilityDataset.csv* se centra en la evaluación del consumo de energía renovable en México y cómo este puede ser proyectado en el futuro utilizando un modelo LSTM. El resultado de este análisis ofrece varias interpretaciones significativas en el contexto de la sostenibilidad y la gestión de recursos energéticos:

Interpretación de tendencias históricas:

La trayectoria descendente en el consumo de energía renovable hasta 2015, evidenciada en los datos de entrenamiento, puede ser el reflejo de múltiples factores interrelacionados, incluyendo variaciones económicas que afectan la viabilidad de las inversiones en tecnologías limpias y la posible influencia de subsidios o incentivos gubernamentales. Además, este patrón podría estar ligado a la conciencia social y la demanda del mercado, que a su vez son impulsados por la percepción pública y la educación sobre las ventajas de las energías renovables.

Evaluación de la capacidad predictiva del modelo:

Las predicciones generadas por el modelo, aunque no se alinean perfectamente con los datos reales, capturan un cambio significativo en la tendencia que sugiere un aumento en el consumo de energía renovable seguido de una estabilización o disminución. Esto podría reflejar la volatilidad en el mercado de energías renovables o variaciones en la adopción de estas energías que podrían ser críticas para la planificación futura.

Implicaciones para la planificación y política energética:

Los resultados del análisis aportan una base sólida para la formulación de políticas y la planificación energética. Al comprender mejor las tendencias históricas y las proyecciones futuras, los responsables de la toma de decisiones pueden diseñar estrategias más efectivas que no solo apunten a mejorar la infraestructura existente sino también a incentivar la integración de energías renovables en el tejido energético nacional. Esto podría traducirse en el desarrollo de nuevos proyectos de energía renovable, la actualización de los marcos regulatorios para facilitar la adopción de tecnologías limpias y la implementación de programas de subsidios y créditos fiscales que estimulen tanto la producción como el consumo de energías renovables.

Influencia en la inversión en energías renovables:

Los inversores y empresas del sector energético podrían emplear los datos analizados para discernir tendencias emergentes y periodos críticos en la demanda de energía renovable, lo cual es vital para la toma de decisiones estratégicas. La capacidad de prever con anticipación los ciclos de auge o recesión en la adopción de energías renovables brinda una ventaja competitiva en el mercado, permitiendo a las empresas ajustar sus carteras de inversión y estrategias de crecimiento de manera que se alinee con las proyecciones de demanda y las metas de sostenibilidad. Además, las previsiones derivadas del modelo pueden ser cruciales para la optimización de operaciones logísticas, como la cadena de suministro y la gestión de inventarios, así como para la planificación estratégica de la capacidad productiva y de infraestructura. Esta optimización garantiza que las inversiones estén sincronizadas con los pronósticos de demanda futura, permitiendo una asignación de recursos más eficiente y una respuesta ágil a las fluctuaciones del mercado.

Conclusiones

El análisis minucioso que emplea las RNR con arquitectura de LSTM para desglosar el conjunto de datos "WorldSustainabilityDataset.csv" ha sido revelador en la identificación de patrones complejos del consumo de energía renovable en México. La metodología se centra en la predicción de series temporales y ha probado ser un enfoque robusto y fiable, destacando su capacidad para captar la dinámica subyacente y las tendencias a lo largo del tiempo. Esta capacidad analítica no solo detecta las tendencias existentes sino que también ofrece la posibilidad de proyectar futuros escenarios, permitiendo a los stakeholders anticipar y planificar en consecuencia. Además, este enfoque metodológico podría ser replicado en otros contextos geográficos y sectoriales, lo que sugiere su aplicabilidad y valor en un amplio espectro de desafíos relacionados con la sostenibilidad energética.

La evaluación del modelo reveló que las RNR pueden prever con una precisión razonable la evolución del consumo de energía renovable, una tarea crucial para la planificación de la sostenibilidad energética. Aunque se observaron desviaciones entre las predicciones y los datos reales, estos errores proporcionaron una visión crítica sobre los aspectos del modelo que podrían mejorarse, como la necesidad de refinar los hiperparámetros o incorporar datos adicionales que podrían influir en la precisión predictiva.

Los resultados obtenidos subrayan la utilidad del método LSTM en la interpretación de datos complejos y en la proyección de tendencias futuras en el sector de la energía renovable. El modelo no solo ofreció proyecciones de la demanda de energía renovable, sino que también destacó la posibilidad de cambios en las tendencias de consumo, lo que es esencial para adaptarse a futuras necesidades y para la toma de decisiones basadas en datos.

El estudio confirmó que las técnicas avanzadas de aprendizaje profundo, como las RNR y LSTM, son herramientas valiosas para los analistas de datos y responsables de la formulación de políticas en el ámbito de la sostenibilidad energética. Al aplicar estas técnicas al problema estudiado, se descubrieron patrones significativos y se generaron predicciones útiles que pueden informar estrategias futuras para la gestión sostenible de recursos energéticos renovables. Este enfoque analítico, por tanto, tiene el potencial de contribuir de manera significativa a la optimización de recursos y a la elaboración de políticas energéticas eficientes y sostenibles.

Referencias

Alonso, J. I. (2022). *Utilización de redes neuronales recurrentes en la predicción de tendencias del mercado de harina de soja*.

Bobadilla, J. (2021). *Machine learning y deep learning: Usando Python, Scikit y Keras* (1a ed.). Ediciones de la U; Editorial Ra-Ma.

García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M., & Herrera, F. (2016). Big data preprocessing: Methods and prospects. *Big Data Analytics*, 1(1), 9. https://doi.org/10.1186/s41044-016-0014-0

Guamán Pachacama, J. A., & Segura Muñoz, G. B. (2021). *Red neuronal Long Short-Term Memory (LSTM) aplicada a series temporales para pronosticar consumo energético en edificaciones*. [B.S. thesis]. Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas.

Jaquenod De Zsögön, S. (2019). *Antropología ambiental*. Dykinson.

Joseph, F. J. J., Nonsiri, S., & Monsakul, A. (2021). Correction to: Keras and TensorFlow: A Hands-On Experience. En K. B. Prakash, R. Kannan, S. A. Alexander, & G. R. Kanagachidambaresan (Eds.), *Advanced Deep Learning for Engineers and Scientists* (pp. C1–C1). Springer International Publishing. https://doi.org/10.1007/978-3-030-66519-7_12

Landázuri Páez, L. L., Gallo Perugachi, K. M., & Estrella Tapia, D. F. (2023). Sistema de Monitoreo/Control de Consumo de energía eléctrica en el hogar mediante Raspberry Pi y Python.: Home energy Consumption monitoring/

control System using Raspberry Pi and Python. *Revista Científica Multidisciplinar G-nerando*, 4(2). https://doi.org/10.60100/rcmg.v4i2.126

Lindemann, B., Maschler, B., Sahlab, N., & Weyrich, M. (2021). A survey on anomaly detection for technical systems using LSTM networks. *Computers in Industry*, 131, 103498. https://doi.org/10.1016/j.compind.2021.103498

Nguyen, H. D., Tran, K. P., Thomassey, S., & Hamad, M. (2021). Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*, 57, 102282. https://doi.org/10.1016/j.ijinfomgt.2020.102282

Pang, B., Nijkamp, E., & Wu, Y. N. (2020). Deep Learning With TensorFlow: A Review. *Journal of Educational and Behavioral Statistics*, 45(2), 227–248. https://doi.org/10.3102/1076998619872761

Quiguiri Daquilema, C. M. (2023). *Contrastando el Machine Learning y la Econometría en series temporales*. [B.S. thesis]. Quito: EPN, 2023.

Rengasamy, D., Jafari, M., Rothwell, B., Chen, X., & Figueredo, G. P. (2020). Deep Learning with Dynamically Weighted Loss Function for Sensor-Based Prognostics and Health Management. *Sensors*, 20(3), 723. https://doi.org/10.3390/s20030723

Rodríguez Morales, V., Bustamante Alfonso, L. M., & Mirabal Jean-Claude, M. (2011). La protección del medio ambiente y la salud, un desafío social y ético actual. *Revista Cubana de salud pública*, 37, 510–518.

Salazar Molina, E. A. (2023). *Metodología para la Evaluación en Proyectos de Energía Solar Fotovoltaica* [Doctoral thesis, Universidad Internacional Iberoamericana México]. https://repositorio.unini.edu.mx/id/eprint/6010

Torres, V. (2021, julio). *La Sustentabilidad Como Estrategia Para El Desarrollo De La Industria*. CONGRESO CIENTÍFICO MULTIDISCIPLINARIO LATAM 2021 Investigación en latinoamérica. http://cathi.uacj.mx/20.500.11961/19976

Vega Moreno, B. D. (2021). *Diseño y desarrollo de un sistema de recomendación basado en filtrado colaborativo utilizando datos secuenciales mediante redes neuronales recurrentes* [B.S. thesis].

Veléz, A., Mera, C. A., Orduz, S., & Branch, J. W. (2021). Generación de péptidos antimicrobianos mediante redes neuronales recurrentes. *Dyna*, 88(216), 210–219.

Wang, J., & Dowling, A. W. (2022). Pyomo.DOE: An open-source package for model-based design of experiments in Python. *AIChE Journal*, 68(12), e17813. https://doi.org/10.1002/aic.17813