Tabla de contenido

[Prompt 1 4](#_Toc210514930)

[Contexto 4](#_Toc210514931)

[Prompt 4](#_Toc210514932)

[Respuesta 4](#_Toc210514933)

[Prompt 2 5](#_Toc210514934)

[Contexto 5](#_Toc210514935)

[Prompt 5](#_Toc210514936)

[Prompt 3 6](#_Toc210514937)

[Contexto 6](#_Toc210514938)

[Prompt 6](#_Toc210514939)

[Respuesta 6](#_Toc210514940)

[Prompt 4 7](#_Toc210514941)

[Contexto 7](#_Toc210514942)

[Prompt 7](#_Toc210514943)

[Respuesta 7](#_Toc210514944)

[Prompt 5 9](#_Toc210514945)

[Contexto 9](#_Toc210514946)

[Prompt 9](#_Toc210514947)

[Prompt 6 9](#_Toc210514948)

[Contexto 9](#_Toc210514949)

[Prompt 9](#_Toc210514950)

[Prompt 7 10](#_Toc210514951)

[Contexto 10](#_Toc210514952)

[Prompt 10](#_Toc210514953)

[Prompt 7 14](#_Toc210514954)

[Contexto 14](#_Toc210514955)

[Prompt 14](#_Toc210514956)

[Respuesta 14](#_Toc210514957)

[Prompt 9 17](#_Toc210514958)

[Contexto 17](#_Toc210514959)

[Prompt 17](#_Toc210514960)

[Prompt 10 17](#_Toc210514961)

[Contexto 17](#_Toc210514962)

[Prompt 18](#_Toc210514963)

[Respuesta 18](#_Toc210514964)

[Prompt 11 19](#_Toc210514965)

[Contexto 19](#_Toc210514966)

[Prompt 19](#_Toc210514967)

[Respuesta 19](#_Toc210514968)

[Prompt 12 21](#_Toc210514969)

[Contexto 21](#_Toc210514970)

[Prompt 21](#_Toc210514971)

[Prompt 13 21](#_Toc210514972)

[Contexto 21](#_Toc210514973)

[Prompt 21](#_Toc210514974)

[Respuesta 21](#_Toc210514975)

[Prompt 14 21](#_Toc210514976)

[Contexto 21](#_Toc210514977)

[Prompt 21](#_Toc210514978)

[Respuesta 22](#_Toc210514979)

[Prompt 15 22](#_Toc210514980)

[Contexto 22](#_Toc210514981)

[Prompt 23](#_Toc210514982)

[Respuesta 23](#_Toc210514983)

[Prompt 16 24](#_Toc210514984)

[Contexto 24](#_Toc210514985)

[Prompt 24](#_Toc210514986)

[Respuesta 24](#_Toc210514987)

[Prompt 17 27](#_Toc210514988)

[Contexto 27](#_Toc210514989)

[Prompt 27](#_Toc210514990)

[Respuesta 27](#_Toc210514991)

[Prompt 18 29](#_Toc210514992)

[Contexto 29](#_Toc210514993)

[Prompt 29](#_Toc210514994)

[Respuesta 29](#_Toc210514995)

[Prompt 19 31](#_Toc210514996)

[Contexto 31](#_Toc210514997)

[Prompt 31](#_Toc210514998)

[Respuesta 31](#_Toc210514999)

[Prompt 20 33](#_Toc210515000)

[Contexto 33](#_Toc210515001)

[Prompt 33](#_Toc210515002)

[Respuesta 33](#_Toc210515003)

[Prompt 21 36](#_Toc210515004)

[Contexto 36](#_Toc210515005)

[Prompt 36](#_Toc210515006)

# Prompt 1

Contexto: En las fases iniciales me apoyé en ChatGPT para el aporte de ideas para la temática del TFM

Fecha: Mayo 2025

Prompt: Asume el rol de experto en contaminación atmosférica y ciencia de datos, proporcióname algunas ideas para un trabajo de fin de master sobre el uso de machine learning para predicción de contaminantes en Cádiz

Respuesta:

🔬 ENFOQUES GENERALES

1. Predicción de concentraciones de contaminantes a corto plazo

Usar modelos de ML para predecir niveles horarios o diarios de contaminantes como **NO₂, SO₂, PM10, PM2.5, O₃ o CO**.

**Modelos**: Random Forest, XGBoost, LSTM, Prophet, SVR.

**Input**: Variables meteorológicas (viento, temperatura, humedad), hora/día del año, tráfico o actividad industrial.

2. Modelos espacio-temporales de predicción

Predecir niveles de contaminantes no solo en el tiempo sino en el espacio (por ejemplo, interpolación de zonas sin estaciones de medida).

**Técnicas**: Spatio-temporal kriging + ML, LSTM con coordenadas, GNN (Graph Neural Networks).

**Aplicación**: Mejora de redes de monitorización, estimación en zonas rurales sin sensores.

3. Análisis comparativo entre modelos ML y métodos clásicos

Comparar la eficacia de modelos ML frente a modelos tradicionales como ARIMA, SARIMA, modelos de dispersión (CALPUFF, AERMOD), etc.

**Objetivo**: Validar si ML mejora la predicción sin perder interpretabilidad.

4. Predicción de episodios de alta contaminación (alertas)

Clasificación binaria o multiclase para predecir si se superará un umbral legal o crítico.

**Etiquetas**: Episodio sí/no (clasificación), concentración exacta (regresión).

**Modelos**: XGBoost, LSTM, redes neuronales, modelos híbridos.

**Aplicación**: Sistemas de alerta temprana o planificación municipal.

5. Modelos interpretables para toma de decisiones

Enfoque centrado en modelos **explainable AI (XAI)** como SHAP o LIME para explicar predicciones a responsables ambientales.

**Valor añadido**: Facilita la adopción en organismos públicos.

**Útil en**: Predicción de O₃ o PM2.5, donde intervienen múltiples factores.

6. Modelos específicos para contaminantes secundarios como el Ozono

Predicción de O₃ troposférico, que se forma a partir de NOₓ y COVs bajo condiciones específicas de radiación solar y temperatura.

Requiere análisis no lineal, lo cual encaja bien con modelos ML.

7. Modelos para imputación y corrección de datos faltantes

Diseñar modelos de ML que **completen series temporales de contaminantes**, usando datos meteorológicos o series correlacionadas.

También puede usarse para calibrar sensores de bajo coste.

8. Predicción de impactos por eventos extremos

Predecir aumentos de contaminantes debido a eventos como **incendios forestales, tormentas de polvo (Sáhara), episodios de calor extremo o inversión térmica**.

Datos satelitales como MODIS (FRP, AOD) + meteorología + ML.

📍IDEAS CON FOCO REGIONAL O LOCAL

9. Modelo ML para la provincia de Cádiz (o zona específica)

Cruzar con datos meteorológicos de AEMET y actividad portuaria/tráfico (si está disponible).

Posible enfoque multizona o comparación de zonas urbanas vs industriales.

💡 COMBINACIONES INTERESANTES

**ML + Teledetección**: Uso de AOD (Aerosol Optical Depth) para predecir PM10/PM2.5.

**ML + Series temporales multivariantes**: uso de modelos multivariantes LSTM o Transformer para capturar relaciones complejas.

**ML + Big Data Streaming**: análisis en tiempo real de sensores usando Spark MLlib.

**AutoML**: experimentación con AutoML (TPOT, H2O.ai, AutoSklearn) para facilitar la selección de modelos.

✅ CONSEJOS PARA ELEGIR LA IDEA

**Disponibilidad de datos**: asegúrate de que puedes acceder a series completas de contaminación y meteorología.

**Aplicabilidad real**: intenta que sea útil para un municipio, institución o región concreta.

**Viabilidad técnica**: evalúa si puedes implementar modelos complejos como LSTM o GNN según tu experiencia y tiempo.

**Enfoque CRISP-DM/KDD**: estructura el proyecto siguiendo fases claras de análisis, modelado y evaluación.

# Prompt 2

Contexto: En las fases iniciales del proyecto se buscó documentación sobre contaminantes y su medición y surgieron dudas con respecto a varios conceptos

Fecha: Mayo 2025

Prompt: Asume el rol de experto en contaminación atmosférica y su medición, ayúdale a entender los conceptos de AQI (air Quality index), Aerosol Optical Depth (AOD), Fire Radiative Power (FRP) y Aerosol Index (AI).

# Prompt 3

Contexto: En las fases iniciales del proyecto se buscó documentación sobre contaminantes y su medición y surgieron dudas con respecto a varios conceptos

Fecha: Mayo 2025

Prompt: Asume el rol de experto en contaminación atmosférica, ¿me explicas brevemente la dinámica del ozono troposférico?

Respuesta:

El **ozono troposférico (O₃)** no se emite directamente, sino que se forma en la **baja atmósfera** a partir de reacciones fotoquímicas. Su dinámica se entiende como un equilibrio entre **formación, transporte y destrucción**:

* **Formación:**  
  Se genera cuando los **óxidos de nitrógeno (NOₓ: NO y NO₂)** y los **compuestos orgánicos volátiles (COVs)** reaccionan en presencia de **radiación solar**. El NO₂ se fotodisocia y libera oxígeno atómico, que al unirse con O₂ da lugar a O₃. Este proceso es más intenso en verano y en días soleados.
* **Destrucción:**  
  El O₃ puede reaccionar con el **óxido nítrico (NO)**, con radicales (OH, HO₂), o depositarse en la superficie terrestre y vegetación. Por eso, en zonas con mucho tráfico (altas emisiones de NO), se observan niveles más bajos de ozono en el centro urbano que en áreas suburbanas.
* **Transporte:**  
  Las corrientes de aire lo desplazan a decenas o cientos de kilómetros, de modo que las zonas rurales y costeras pueden presentar picos altos incluso sin tener fuentes locales de precursores.

En resumen, el ozono troposférico es un **contaminante secundario** de comportamiento complejo: depende de la radiación solar, la meteorología (viento, temperatura, estabilidad atmosférica) y la proporción de precursores. Esto explica por qué su gestión es más difícil que la de contaminantes primarios como NO₂ o SO₂.

¿Quieres que te prepare también un **esquema visual sencillo** con el ciclo de formación y destrucción del ozono?

# Prompt 4

Contexto: Uso de IAG para la ayuda a la generación de ideas en las fases iniciales del proyecto.

Fecha: Mayo 2025

Prompt: Asume el rol de experto en contaminación atmosférica y su medición y colaboras con un científico de datos en un trabajo de investigación sobre el uso de machine learning para la predicción de contaminantes, plantea ideas de cómo enfocar un trabajo sobre la predicción del ozono troposférico en la provincia de Cádiz

Respuesta:

Perfecto. Como experto en contaminación atmosférica colaborando con un científico de datos, podemos estructurar un enfoque sólido, interdisciplinar y con aplicabilidad real sobre la predicción del ozono troposférico. A continuación, te presento una propuesta de cómo podríamos plantear el trabajo, combinando la experiencia ambiental con el análisis de datos y modelos predictivos:

⸻

🧠 Enfoque del trabajo de investigación: Predicción de la concentración de ozono troposférico mediante técnicas de machine learning

1. Justificación científica y social

• El ozono troposférico es un contaminante secundario cuya formación depende de precursores (NOₓ, COVs) y condiciones atmosféricas (radiación solar, temperatura, etc.).

• Niveles elevados de O₃ tienen impactos en la salud humana, especialmente respiratorios, así como en la vegetación y materiales.

• La predicción del O₃ es compleja debido a su carácter no lineal, lo que hace idóneo el uso de modelos de machine learning (ML).

• Un sistema de predicción automatizado puede ser una herramienta útil para autoridades ambientales y ciudadanía.

⸻

2. Zonificación y selección de áreas de estudio

Podemos enfocar el trabajo en áreas con características contrastadas:

• Zonas industriales (como el Campo de Gibraltar o Bahía de Algeciras)

• Zonas urbanas densas (como Jerez de la Frontera o Cádiz capital)

• Zonas rurales o de transición (como la Sierra de Grazalema o la Campiña de Jerez)

La comparación entre zonas permitirá:

• Evaluar la influencia de diferentes fuentes de emisión.

• Contrastar el comportamiento del O₃ según el entorno.

⸻

3. Datos a utilizar

Datos de entrada (features)

• Meteorológicos: temperatura, humedad, radiación solar, presión, velocidad y dirección del viento, punto de rocío, precipitación.

• Contaminantes: NO₂, SO₂, CO, PM₁₀, PM₂.₅ (precursores o co-ocurrentes).

• Variables temporales: hora del día, día de la semana, estación, feriados.

• Variables geoespaciales: altitud, distancia a la costa o a industrias.

Variable objetivo

• Concentración horaria de ozono troposférico (O₃).

⸻

4. Metodología de modelado

a. Preprocesamiento

• Limpieza e imputación de datos faltantes.

• Agrupación por estación o zona.

• Ingeniería de variables temporales.

• Análisis exploratorio y de estacionalidad.

b. Modelos a utilizar

• Modelos clásicos: Random Forest, XGBoost, Support Vector Regression.

• Modelos de deep learning: LSTM (por naturaleza secuencial), GRU, modelos híbridos.

• Comparación con modelos tradicionales de regresión (baseline).

c. Evaluación

• MAE, RMSE, R² para cuantificar precisión.

• Comparación por zona y por estación del año.

• Validación cruzada con división temporal (TimeSeriesSplit o walk-forward).

⸻

5. Análisis de resultados

• ¿Qué variables tienen más peso? → Feature importance.

• ¿Qué zonas presentan más dificultad de predicción?

• ¿Hay patrones meteorológicos que preceden niveles altos de O₃?

• Evaluar si los modelos podrían servir para alertas tempranas.

⸻

6. Visualización

• Mapas horarios con niveles predichos vs reales.

• Paneles interactivos (con Streamlit, Dash o QuickSight).

• Representación de episodios críticos o picos.

⸻

7. Aplicaciones reales

• Desarrollo de un sistema de alerta temprana.

• Incorporación en plataformas web o apps ciudadanas.

• Apoyo a estrategias de movilidad o control de emisiones en picos críticos.

# Prompt 5

Contexto: En las fases iniciales del trabajo me planteaba usar AWS y tenía dudas sobre los servicios a usar en caso de montar todo el pipeline en Amazon. Finalmente decidí usar PyCharm y Colab dado que Colab Pro me daba acceso a los recursos necesarios.

Fecha: Mayo 2025

Prompt: Asume el rol de un ingeniero de datos experto en los servicios de AWS, ¿me podrías orientar sobre los servicios de AWS que podría utilizar para cargar unos datos de series de tiempo de contaminantes y datos atmosféricos, transformarlos e integrarlos y aplicarles algoritmos de ML de forma que todo el pipeline esté ubicado en AWS?

# Prompt 6

Contexto: En las fases iniciales del trabajo me planteaba usar AWS e hice algunas pruebas subiendo ficheros a AWS y me apoyé de ChatGPT en la resolución de errores. Finalmente decidí usar PyCharm y Colab dado que Colab Pro me daba acceso a los recursos necesarios.

Fecha: Mayo 2025

Prompt: Ayúdame a identificar qué problema puedo tener aquí al tratar de subir a mi bucket de AWS ficheros

s3 = boto3.client(  's3',

    aws\_access\_key\_id= os.environ.get("AWS\_ACCESS\_KEY\_ID"),

    aws\_secret\_access\_key=os.environ.get("AWS\_SECRET\_ACCESS\_KEY"),

    region\_name=os.environ.get("AWS\_REGION\_NAME"),)

s3.put\_object( Bucket='airqualityinput/Input/datos\_atmosfericos',

    Key='datos\_atmosfericos.csv',

    Body=csv\_buffer.getvalue(),

# Prompt 7

Contexto: Ayuda en la resolución de errores llamando a las apis durante la obtención de datos

Fecha: junio 2025

Prompt: Ayúdame a identificar qué problema puedo tener en este código en el bucle while:

# Parámetros por ubicación

BahiaCadiz\_params = {

    "limit": 100,

    "page": 1,

    "coordinates": "36.52672,-6.2891",

    "radius": 15000,

}

Jerez\_params = {

    "limit": 100,

    "page": 1,

    "coordinates": "36.684,-6.13043",

    "radius": 15000,

}

BahiaAlgeciras\_params = {

    "limit": 100,

    "page": 1,

    "coordinates": "36.1821,-5.4114",

    "radius": 15000,

}

# Parámetros para mediciones horarias

params = {

    "limit": 1000,

    "page": 1,

    "datetime\_from": "2022-12-31T23:00:00",

    "datetime\_to": "2025-05-31T23:00:00",

}

# Obtener sensores

Bahia\_Cadiz = sensores.sensor(BahiaCadiz\_params)

Jerez = sensores.sensor(Jerez\_params)

Bahia\_Algeciras = sensores.sensor(BahiaAlgeciras\_params)

sensor = pd.concat([Bahia\_Cadiz, Jerez, Bahia\_Algeciras], axis=0, ignore\_index=True)

# Lista para acumular resultados

registros = []

# Iterar por cada sensor

for id in range(len(sensor)):

    sensor\_id = sensor['SensorID'].iloc[id]

    location\_id = sensor['LocationID'].iloc[id]

    location\_name = sensor['LocationName'].iloc[id]

    base\_url = f"https://api.openaq.org/v3/sensors/{sensor\_id}/measurements/hourly"

    all\_results = []

    local\_params = params.copy()

    local\_params["page"] = 1

    while True:

        response = requests.get(base\_url, headers=headers, params=local\_params)

        if response.status\_code == 429:

            print("Rate limit alcanzado. Esperando 10 segundos...")

            time.sleep(10)

            continue

        if response.status\_code != 200:

            print(f"Error en la solicitud del sensor {sensor\_id}: {response.status\_code}")

            break

        data = response.json()

        results = data.get("results", [])

        all\_results.extend(results)

        meta = data.get("meta", {})

        found = meta.get("found", 0)

        limit = meta.get("limit", 1000)

        page = meta.get("page", 1)

        if page \* limit >= found:

            break

        else:

            local\_params["page"] += 1

    # Convertir a DataFrame si hay datos

    df = pd.DataFrame(all\_results)

    if not df.empty:

        for \_, row in df.iterrows():

            parametro = row['parameter']['name']

            valor = row['value']

            fecha = row['coverage'].get('datetimeTo', {}).get('local')

            registros.append({

                'estacion': location\_id,

                'nombre': location\_name,

                'datetime': fecha,

                'parametro': parametro,

                'valor': valor })

# Crear DataFrame largo y pivotar

df\_largo = pd.DataFrame(registros)

df\_pivot = df\_largo.pivot\_table(

    index=['estacion', 'nombre', 'datetime'],

    columns='parametro',

    values='valor' ).reset\_index()

# Prompt 7

Contexto: Planteando la estructura de la memoria, no tenía claro cómo enfocar el marco teórico y el estado del arte. Me apoyé en ChatGPT para obtener sugerencia sobre subapartados y contenido.

Fecha: Junio 2025

Prompt: Asume el rol de experto en ciencia de datos y que tutorizas un TFM sobre modelos de machine learning y big data aplicados a la predicción de ozono, ¿me sugieres algunos puntos o ideas para la sección de estado del arte

Respuesta:

Perfecto. Como tu tutor en un TFM sobre modelos de Machine Learning y Big Data aplicados a la predicción de ozono troposférico, te propongo una estructura detallada y sugerencias clave para la sección de “Estado del Arte”, para que demuestres dominio técnico y conocimiento actualizado del área. Esta sección debe contextualizar tu trabajo dentro de las investigaciones actuales, mostrando por qué es relevante y qué aporta.

⸻

🧠 OBJETIVO DEL ESTADO DEL ARTE

• Mostrar cómo se ha abordado la predicción de ozono (O₃) en estudios anteriores.

• Analizar enfoques con Machine Learning, Big Data, y también enfoques clásicos para justificar tu propuesta.

• Identificar lagunas en la literatura (por ejemplo, falta de modelos explicables, baja resolución espacial, o escasez de estudios en tu zona geográfica).

⸻

🧩 ESTRUCTURA RECOMENDADA PARA EL ESTADO DEL ARTE

⸻

1. Introducción al ozono troposférico y su predicción

• Breve explicación del ozono como contaminante secundario (formación a partir de NOx y COVs con radiación solar).

• Importancia sanitaria y ambiental (efectos en salud, agricultura, cambio climático).

• Dificultad para predecirlo por su comportamiento no lineal y dependiente de múltiples factores.

• Mención a normativas relevantes (por ejemplo, límite horario de 180 μg/m³ según la normativa europea).

⸻

2. Modelos clásicos de predicción de ozono

• Modelos deterministas basados en química atmosférica (por ejemplo, CMAQ, WRF-Chem, etc.).

• Modelos estadísticos: regresión lineal, ARIMA/SARIMA, modelos autoregresivos.

• Limitaciones: alta necesidad computacional o baja capacidad de generalización/precisión.

⸻

3. Aplicaciones de Machine Learning en la predicción de ozono

📌 Modelos más usados:

• Árboles de decisión, Random Forest, Gradient Boosting (XGBoost, LightGBM).

• Redes neuronales profundas (DNN, MLP).

• Redes LSTM y GRU para series temporales.

• SVR (Support Vector Regression).

📌 Casos representativos:

• Estudios en zonas urbanas e industriales (por ejemplo, Madrid, Milán, Beijing, Nueva York).

• Predicción de concentraciones horarias o diarias.

• Comparación entre modelos (ML vs tradicionales).

• Algunos estudios aplican ensembles o modelos híbridos (ej. LSTM + RF).

📌 Variables más usadas:

• Meteorológicas: temperatura, humedad, viento, presión, radiación solar.

• Contaminantes precursores: NO₂, CO, SO₂, COVs.

• Variables temporales: día de la semana, estación del año, hora.

• En algunos casos: tráfico, teledetección, eventos extremos.

⸻

4. Big Data y arquitecturas para predicción en tiempo real

• Uso de plataformas tipo Apache Spark, Kafka, Hadoop, etc. para ingesta, procesamiento y modelado en tiempo real.

• Modelos entrenados offline y desplegados online.

• Integración con sensores IoT o redes de bajo coste.

Puedes citar experiencias reales en ciudades inteligentes donde se monitoriza y predice la calidad del aire en tiempo real.

⸻

5. Modelos explicables e interpretabilidad (XAI)

• Creciente interés por explicar predicciones: técnicas como SHAP, LIME, Permutation Importance.

• Útiles para generar confianza en stakeholders o autoridades.

• Falta de estudios que combinen buenos resultados con explicaciones claras.

⸻

6. Limitaciones actuales y oportunidades

• Déficit de estudios regionales (por ejemplo, en Andalucía o Cádiz).

• Pocas aplicaciones de modelos con alta resolución espacial (Interpolación + ML).

• Escasa incorporación de variables de teledetección como AOD (Aerosol Optical Depth) o Fire Radiative Power (FRP).

• Potencial en usar modelos de deep learning más avanzados (Transformers, attention-based).

• Pocas experiencias con integración ML + Big Data para flujos en streaming.

⸻

7. Tabla comparativa de estudios relevantes

Incluye una tabla con 6-8 estudios clave, mostrando:

• Zona geográfica

• Modelo usado

• Variables

• Resolución temporal

• Métricas obtenidas (RMSE, MAE, R²)

• Aplicación (alerta, predicción general, alta contaminación, etc.)

⸻

8. Resumen crítico

• ¿Qué se ha hecho?

• ¿Qué falta?

• ¿Cómo tu trabajo contribuirá al área?

⸻

📚 FUENTES BIBLIOGRÁFICAS RECOMENDADAS

• Scopus, Web of Science, Google Scholar con términos como:

• “Ozone concentration prediction machine learning”

• “Tropospheric ozone forecasting with deep learning”

• “Big Data air quality prediction architecture”

• “XGBoost ozone air pollution”

• Proyectos europeos como Copernicus CAMS, informes de la EEA, papers de Atmospheric Environment, Environmental Pollution, etc.

# Prompt 9

Contexto: Mejora de la redacción de párrafos / apartados específicos de forma que la redacción sea más académica y coherente. Usado en resumen, introducción y marco teórico

Fecha: Junio 2025

Prompt: toma el rol de experto en contaminación atmosférica, reformula los siguientes párrafos de forma que la redacción tenga un tono más académico y una estructura más ordenada.

# Prompt 10

Contexto: Quería tener una visión del contexto histórico/evolución de algunos de los algoritmos utilizados en el trabajo

Fecha: Julio 2025

Prompt: ¿Me proporcionas información resumida sobre el contexto histórico / evolución de los algoritmos Random Forest?

Respuesta:

**1. Antecedentes: Árboles de decisión**

* Antes de Random Forest, los **árboles de decisión** (como CART – *Classification and Regression Trees*) ya eran una técnica popular para clasificación y regresión.
* Inventados por **Breiman et al. (1986)**, los árboles ofrecían interpretabilidad, pero sufrían de **alta varianza**: un pequeño cambio en los datos podía generar un árbol completamente distinto.

**2. Bagging: Primer paso hacia el ensamblado**

* En **1996**, **Leo Breiman** propuso el método de **bagging (bootstrap aggregating)**.
* Consistía en entrenar múltiples modelos sobre subconjuntos del dataset creados por muestreo con reemplazo y promediar (o votar) los resultados.
* El bagging aplicado a árboles de decisión ya mejoraba la precisión y reducía la varianza.

**3. Random Forest: Nacimiento del algoritmo**

* En **2001**, **Leo Breiman** introduce **Random Forest** como una extensión del bagging.
* Clave innovadora: en cada división de un árbol, en lugar de considerar todas las variables, se selecciona un **subconjunto aleatorio de variables**.
  + Esto reduce la correlación entre árboles, mejorando el rendimiento del ensamblado.
* Breiman lo presentó como un **modelo robusto, rápido y preciso**, especialmente útil en clasificación de datos con muchas variables.

**4. Evolución y aplicaciones**

* A lo largo de los años 2000 y 2010, Random Forest se convirtió en un **modelo de referencia en Machine Learning**, especialmente antes del auge del deep learning.
* Se aplicó ampliamente en **bioinformática**, **finanzas**, **ecología**, **predicción meteorológica**, y más.
* Se desarrollaron variantes como:
  + **Extremely Randomized Trees (ExtraTrees)**: añade más aleatoriedad en la selección de umbrales de división.
  + **Random Survival Forests**: adaptado para análisis de supervivencia.

**5. En la actualidad**

* Aunque modelos como **XGBoost, LightGBM** y redes neuronales profundas han ganado protagonismo, **Random Forest sigue siendo muy valorado** por:
  + Su **robustez**.
  + Bajo riesgo de overfitting.
  + Buen rendimiento en datasets tabulares.
  + Facilidad de uso (pocos hiperparámetros).
  + Interpretabilidad relativa (importancia de variables, visualización de árboles).

# Prompt 11

Contexto: Quería tener una visión del contexto histórico/evolución de algunos de los algoritmos utilizados en el trabajo

Fecha: Julio 2025

Prompt: ¿Me proporcionas información resumida sobre el contexto histórico / evolución del modelo LSTM?

Respuesta:

**Contexto histórico y evolución de LSTM**

**1. Problema inicial: RNN y el desvanecimiento del gradiente**

* Las **redes neuronales recurrentes (RNN)** surgieron en los años 80 como una arquitectura capaz de **modelar secuencias** de datos, como texto, audio o series temporales.
* Sin embargo, las RNN tradicionales tenían dificultades para **capturar dependencias a largo plazo** debido al problema del **desvanecimiento (y explosión) del gradiente** durante el entrenamiento por *backpropagation through time* (BPTT).
* Esto limitaba su efectividad en tareas donde la información relevante estaba muy separada en el tiempo.

**2. Nacimiento de LSTM**

* En **1997**, **Sepp Hochreiter y Jürgen Schmidhuber** propusieron **Long Short-Term Memory (LSTM)** como solución al problema.
* LSTM introdujo una estructura con **puertas de control (input, forget, output)** y una **celda de memoria** que permite mantener y regular el flujo de información a lo largo del tiempo.
  + Esta arquitectura permite **capturar dependencias largas** sin que el gradiente desaparezca.

**3. Adopción y mejoras posteriores**

* Durante los años 2000, el uso de LSTM fue limitado debido al alto coste computacional y la escasa infraestructura.
* Sin embargo, con el auge del **deep learning y el incremento de capacidad computacional** (GPUs), LSTM ganó protagonismo en la década de 2010.
* Se empezaron a aplicar con gran éxito en tareas como:
  + **Reconocimiento de voz**
  + **Traducción automática**
  + **Modelado del lenguaje**
  + **Predicción de series temporales**

**4. Variantes y mejoras**

* **GRU (Gated Recurrent Unit)**: propuesta en 2014 por Cho et al., como una variante simplificada de LSTM con menos parámetros, pero resultados comparables.
* LSTM también se ha combinado con otras arquitecturas: redes bidireccionales (BiLSTM), redes convolucionales (ConvLSTM), y más.

**5. Situación actual**

* A partir de 2017, con el auge de **modelos basados en atención y transformers** (como BERT y GPT), LSTM ha perdido protagonismo en NLP.
* Sin embargo, **LSTM sigue siendo muy útil en modelado de series temporales, análisis de secuencias pequeñas o tabulares, y sistemas con restricciones de datos o recursos.**

# Prompt 12

Contexto: Generación de imágenes de apoyo

Fecha: Julio 2025

Prompt: Genera un mapa de la provincia de Cádiz marcando aproximadamente la bahía de Cádiz, la bahía de Algeciras y Jerez.

# Prompt 13

Contexto: Asistencia en la corrección de errores en el código

Fecha: julio 2025

Prompt: Obtengo este error en Google Colaboratory, me indicas cómo resolverlo? TypeError: cannot do indexing on Index with these indexers [2025-01-15 23:00:00] of type str

Respuesta: Ese error aparece cuando intentas indexar con una cadena (str) y tu índice no es un DatetimeIndex (o tiene zona horaria distinta). Deja la serie con un DatetimeIndex limpio y verás que el slicing funciona.

# Prompt 14

Contexto: Asistencia en la corrección de errores en el código

Fecha: julio 2025

Prompt: ¿me ayudas aquí?, no consigo cambiar el color

serie\_filtrada = o3\_alge["2025-01-15 23:00:00":"2025-01-31 23:00:00"]

decompose\_data = seasonal\_decompose(serie\_filtrada, model="additive", period=24)

# Graficar

fig = decompose\_data.plot()

fig.suptitle('Ozono Troposférico - Algeciras', fontsize=10)

# Formato de fecha y hora en el eje x

for ax in fig.axes:

    ax.xaxis.set\_major\_formatter(mdates.DateFormatter('%Y-%m-%d %H:%M'))

    ax.tick\_params(axis='x', rotation=45)  # rotar para mejor legibilidad

plt.tight\_layout()

plt.rcParams["lines.color"] =  '#e78ac3'

plt.show()

Respuesta:

Te está fallando porque:

* seasonal\_decompose(...).plot() ya crea las líneas con los colores del **ciclo de colores** activo en ese momento.
* Cambiar plt.rcParams["lines.color"] **después** de dibujar no afecta a las líneas ya creadas.
* Además, estilos como "fivethirtyeight" definen su propio axes.prop\_cycle y sobreescriben colores.

# Prompt 15

Contexto: Asistencia en la corrección de errores en el código o resolución de dudas.

Fecha: julio 2025

Prompt: en este código para crear la matriz de correlación, cómo puedo influenciar el estilo y colores para que esté en línea con mis otras graficas que usan Set2

# Seleccionar variables relevantes

variables = [

    "temperature\_2m", "direct\_radiation", "wind\_speed\_10m",

    "wind\_direction\_10m", "relative\_humidity\_2m", "o3", "no2", "co"]

# Filtrar y limpiar

df\_corr = df\_merge[variables]

# Matriz de correlación

corr\_matrix = df\_corr.corr(method="pearson")

# Visualizar

plt.figure(figsize=(8, 6))

sns.heatmap(corr\_matrix, annot=True, cmap="coolwarm", fmt=".2f", vmin=-1, vmax=1)

plt.title("Matriz de correlación de Pearson (con O₃)")

plt.tight\_layout()

plt.show()

Respuesta:

Para adaptar el estilo visual de tu **matriz de correlación** y que esté en línea con el resto de tus gráficos —que usan la paleta Set2 de Seaborn y un estilo suave y profesional— puedes seguir los siguientes ajustes en tu código:

Cambios clave que haremos:

* Aplicar el **estilo whitegrid** de Seaborn.
* Definir una **paleta de colores basada en Set2** para el mapa de calor.
* Usar una **paleta personalizada en forma de gradiente** basada en tonos cálidos de Set2.
* Ajustar la estética general (bordes, anotaciones, tipografía).

# Prompt 16

Contexto: Asistencia en la generación de código de ejemplos de aplicación de algoritmos

Fecha: agosto 2025

Prompt: ¿Me proporcionas un ejemplo sencillo en Python de aplicación del algoritmo de redes LSTM a series temporales?

Respuesta:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

# 1. Generar datos artificiales (senoide)

np.random.seed(42)

timesteps = np.arange(0, 100, 0.1)

data = np.sin(timesteps)

# 2. Escalar datos

scaler = MinMaxScaler()

data\_scaled = scaler.fit\_transform(data.reshape(-1, 1))

# 3. Crear secuencias para LSTM

def create\_sequences(data, seq\_length):

X, y = [], []

for i in range(len(data) - seq\_length):

X.append(data[i:i+seq\_length])

y.append(data[i+seq\_length])

return np.array(X), np.array(y)

seq\_length = 20

X, y = create\_sequences(data\_scaled, seq\_length)

# 4. Dividir en entrenamiento y test

split = int(0.8 \* len(X))

X\_train, X\_test = X[:split], X[split:]

y\_train, y\_test = y[:split], y[split:]

# 5. Construir modelo LSTM

model = Sequential([

LSTM(50, activation='tanh', input\_shape=(seq\_length, 1)),

Dense(1)])

model.compile(optimizer='adam', loss='mse')

# 6. Entrenar modelo

model.fit(X\_train, y\_train, epochs=10, batch\_size=16, verbose=1)

# 7. Hacer predicciones

y\_pred = model.predict(X\_test)

# 8. Desescalar predicciones

y\_test\_inv = scaler.inverse\_transform(y\_test)

y\_pred\_inv = scaler.inverse\_transform(y\_pred)

# 9. Graficar resultados

plt.figure(figsize=(10, 4))

plt.plot(y\_test\_inv, label='Real')

plt.plot(y\_pred\_inv, label='Predicho')

plt.legend()

plt.title("Predicción con LSTM")

plt.show()

¿Qué hace este script?

* Crea una serie seno artificial.
* Prepara los datos en secuencias para alimentar la red LSTM.
* Construye una red LSTM básica.
* Entrena el modelo.
* Predice los valores futuros.
* Compara los valores reales y predichos.

# Prompt 17

Contexto: Asistencia en la generación de código de ejemplos de aplicación de algoritmos

Fecha: agosto 2025

Prompt: Cómo puedo controlar el horizonte y otros hiperparámetros de LSTM en el modelo LSTM anterior

Respuesta:

En un modelo **LSTM básico como el anterior**, puedes controlar el **horizonte de predicción** y otros **hiperparámetros clave** modificando ciertos elementos del preprocesamiento y del modelo.

¿Qué significa “horizonte” en LSTM?

* **Horizonte de predicción** = cuántos pasos futuros quieres predecir.
* Por defecto, muchos ejemplos predicen **1 paso adelante** (horizonte = 1).
* Puedes ajustar el código para predecir **n pasos hacia adelante** (por ejemplo, 24 horas o 3 días).

Cómo controlar el  horizonte de predicción

Caso 1: Predecir 1 paso adelante (por defecto)

y.append(data[i+seq\_length]) # Predice t+1 usando t-seq\_length hasta t

Caso 2: Predecir n pasos hacia adelante (horizonte = n)

horizon = 6 # por ejemplo, 6 pasos adelante

for i in range(len(data) - seq\_length - horizon + 1):

X.append(data[i:i+seq\_length])

y.append(data[i+seq\_length:i+seq\_length+horizon]) # predice varios pasos

Luego, asegúrate de que tu modelo tenga una **capa Dense(horizon)** si y es multivariado (más de un paso):

model = Sequential([

LSTM(50, activation='tanh', input\_shape=(seq\_length, n\_features)),

Dense(horizon) # <-- salida con tantos pasos como horizonte])

Hiperparámetros importantes en LSTM

| **Hiperparámetro** | **Descripción** |
| --- | --- |
| seq\_length | Número de pasos pasados usados como entrada (ventana de entrada) |
| horizon | Número de pasos futuros que quieres predecir (salida del modelo) |
| n\_features | Número de variables por cada paso (dimensión multivariable) |
| units de la LSTM | Número de neuronas en la capa LSTM (ej. 50, 64, 128…) |
| batch\_size | Tamaño de lote para entrenamiento |
| epochs | Número de épocas (pasadas completas sobre el dataset de entrenamiento) |
| dropout (opcional) | Técnica para evitar overfitting |
| learning\_rate | Velocidad de aprendizaje del optimizador |
| loss function | Para regresión suele ser 'mse' |
| optimizer | Suele usarse 'adam', pero puedes probar con 'rmsprop', 'sgd', etc. |

Ejemplo con  seq\_length = 48, horizon = 6 (predecir 6 horas después de ver 48)

seq\_length = 48

horizon = 6

X, y = [], []

for i in range(len(data\_scaled) - seq\_length - horizon + 1):

X.append(data\_scaled[i:i+seq\_length, :-1]) # input: features sin O3

y.append(data\_scaled[i+seq\_length:i+seq\_length+horizon, -1]) # salida: solo O3

X = np.array(X)

y = np.array(y)

# Model: salida con 6 valores

model = Sequential([

LSTM(64, input\_shape=(X.shape[1], X.shape[2])),

Dense(horizon)])

model.compile(optimizer='adam', loss='mse')

# Prompt 18

Contexto: Asistencia en la generación de código de ejemplos de aplicación de algoritmos

Fecha: agosto 2025

Prompt: ¿me explicas los hiperparámetros más importantes de las redes LSTM?

## Respuesta

Claro, te explico los **hiperparámetros más importantes** en modelos **LSTM** para series temporales, centrándonos en su **significado y efecto** sin entrar en código.

⚙️ Hiperparámetros clave en LSTM

**1. seq\_length (longitud de la secuencia de entrada)**

Define **cuántos pasos temporales pasados** se usan como entrada para predecir el futuro.

En el contexto de series temporales, es el tamaño de la “ventana deslizante”.

🔸 **Impacto**:

Valores pequeños pueden perder patrones a largo plazo.

Valores grandes pueden aumentar la complejidad y el coste computacional.

2. horizon (horizonte de predicción)

Es el número de pasos temporales **hacia adelante** que el modelo debe predecir.

Puede ser 1 (una hora o día futuro) o más (por ejemplo, los próximos 3 días).

🔸 **Impacto**:

Horizontes cortos suelen ser más precisos.

Horizontes largos son más útiles pero más difíciles de predecir con precisión.

**3. n\_features (número de variables de entrada)**

Es el número de variables independientes (features) que se usan para cada paso del tiempo.

🔸 **Impacto**:

Más variables pueden mejorar el modelo si son informativas, pero también pueden introducir ruido si no están bien seleccionadas.

**4. units (número de neuronas en la capa LSTM)**

Define la **capacidad de memoria** del modelo.

Es el número de unidades (celdas) que la LSTM usa internamente.

🔸 **Impacto**:

Unidades pequeñas pueden no capturar bien patrones complejos.

Unidades grandes pueden aprender más, pero pueden sobreajustar y requerir más recursos.

**5. batch\_size (tamaño del lote)**

Es la cantidad de muestras que se procesan antes de actualizar los pesos del modelo durante el entrenamiento.

🔸 **Impacto**:

Tamaños pequeños permiten mayor ajuste fino, pero son más lentos.

Tamaños grandes son más rápidos pero menos precisos en el ajuste.

**6. epochs (número de épocas)**

Indica cuántas veces el modelo recorre **todo el conjunto de entrenamiento**.

🔸 **Impacto**:

Pocas épocas pueden dejar el modelo sin entrenar completamente.

Demasiadas épocas pueden causar **overfitting** (memoriza los datos).

**7. dropout (tasa de abandono)**

Es una técnica para evitar el sobreajuste.

Durante el entrenamiento, se “apagan” aleatoriamente algunas neuronas.

🔸 **Impacto**:

Reduce el riesgo de sobreajuste.

Tasas típicas: entre 0.1 y 0.5.

**8. learning\_rate (tasa de aprendizaje)**

Controla **cuánto se ajustan los pesos del modelo** en cada paso de entrenamiento.

🔸 **Impacto**:

Tasas altas pueden hacer que el modelo aprenda rápido pero de forma inestable.

Tasas bajas permiten ajustes finos pero hacen el entrenamiento más lento.

**9. loss function (función de pérdida)**

Mide el error entre las predicciones del modelo y los valores reales.

Para regresión, se usa comúnmente el **error cuadrático medio (MSE)**.

🔸 **Impacto**:

Determina cómo el modelo evalúa su rendimiento y se ajusta en consecuencia.

**10. optimizer (optimizador)**

Es el algoritmo que ajusta los pesos del modelo para minimizar la función de pérdida.

🔸 **Opciones comunes**:

**Adam**: rápido, robusto, funciona bien en la mayoría de los casos.

**RMSprop**: bueno para secuencias.

**SGD**: más simple, pero requiere ajuste fino del learning rate.

# Prompt 19

Contexto: Asistencia en la generación de código de ejemplos de aplicación de algoritmos

Fecha: agosto 2025

Prompt: Cómo puedo controlar el número de capas de la red LSTM

Respuesta:

Controlar el **número de capas** en un modelo LSTM significa decidir **cuántas capas LSTM (y capas adicionales como Dense)** vas a usar en la arquitectura del modelo.

Esto afecta directamente a la **profundidad del modelo** y, por tanto, a su **capacidad para aprender patrones complejos** en los datos temporales.

✅ ¿Cómo se controla el número de capas?

En términos conceptuales:

**1. Capas LSTM**

Puedes apilar **varias capas LSTM** unas sobre otras.

Cada capa intermedia debe tener return\_sequences=True para pasar una secuencia completa a la siguiente capa.

🔹 Ejemplo conceptual (sin código):

Input → LSTM (return\_sequences=True)

→ LSTM (return\_sequences=True)

→ LSTM (return\_sequences=False)

→ Dense (output)

**return\_sequences=True** indica que la capa **devuelve toda la secuencia** (necesario si hay otra capa LSTM después).

La **última LSTM** (antes de una capa Dense) suele tener return\_sequences=False para devolver sólo el último paso.

2. Capas Dense (completamente conectadas)

Se colocan al final del modelo para producir la salida.

Pueden ser:

Una sola capa si estás prediciendo un solo valor.

Múltiples capas Dense si necesitas un procesamiento más profundo antes de la salida final.

🎯 ¿Cuántas capas deberías usar?

| **Capas** | **Cuándo usarlas** | **Pros** | **Contras** |
| --- | --- | --- | --- |
| **1 LSTM + 1 Dense** | Problemas simples, predicción a corto plazo | Simple, rápido | Limitado en complejidad |
| **2-3 LSTM + Dense** | Series complejas, multivariadas, con patrones no lineales | Captura patrones profundos | Más lento, riesgo de sobreajuste |
| **>3 capas** | Casos avanzados, datos masivos y complejos | Potencia alta | Muy costoso, difícil de generalizar |

⚠️ Consideraciones clave

Más capas **≠** mejor rendimiento.

Aumentar capas puede llevar a:

**Sobreajuste** si no hay suficientes datos.

**Mayor tiempo de entrenamiento**.

**Dificultad para encontrar la arquitectura óptima** sin validación cuidadosa.

🛠️ ¿Cómo encontrar el número óptimo de capas?

**Empieza simple** (1 LSTM + 1 Dense).

Evalúa el rendimiento (error, gráficos).

Si no capta bien los patrones, prueba con 2 o 3 capas LSTM.

Usa técnicas como:

**Validación cruzada temporal**

**Early stopping**

**Dropout** para regularización

# Prompt 20

Contexto: Asistencia en la generación de código de ejemplos de aplicación de algoritmos

Fecha: agosto 2025

Prompt: ¿Me das un ejemplo sencillo de LSTM de una capa para series temporales diarias con validación cruzada, early stopping y dropout?

Respuesta:

**Datos sintéticos diarios** (por ejemplo, una serie senoide con ruido).

Una **red LSTM de una sola capa**.

**Dropout** para regularización.

**Early stopping** para evitar sobreentrenamiento.

Y un **simulacro de validación cruzada tradicional para series temporales**: usando un **expanding window** (forma adecuada de hacer CV en series temporales).

✅ Características del ejemplo

**Serie temporal univariada** (puedes extenderlo fácilmente a multivariada).

**Horizonte de predicción = 1 día**

**seq\_length = 14 días** (ventana de entrada)

**n\_splits = 5** validaciones cruzadas

Mide **loss de validación** en cada split

🧠 Código completo (solo Python, sin notebook)

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Dropout

from tensorflow.keras.callbacks import EarlyStopping

# 1. Generar datos diarios sintéticos (ej. 1 año)

np.random.seed(42)

days = 365

t = np.arange(0, days)

data = np.sin(2 \* np.pi \* t / 30) + 0.2 \* np.random.randn(days) # patrón mensual + ruido

# 2. Escalar datos

scaler = MinMaxScaler()

data\_scaled = scaler.fit\_transform(data.reshape(-1, 1))

# 3. Crear secuencias

def create\_sequences(data, seq\_length):

X, y = [], []

for i in range(len(data) - seq\_length):

X.append(data[i:i+seq\_length])

y.append(data[i+seq\_length])

return np.array(X), np.array(y)

seq\_length = 14

X, y = create\_sequences(data\_scaled, seq\_length)

# 4. Simular validación cruzada con expanding window

n\_splits = 5

fold\_size = len(X) // (n\_splits + 1)

results = []

for i in range(n\_splits):

split\_idx = fold\_size \* (i + 1)

X\_train, y\_train = X[:split\_idx], y[:split\_idx]

X\_val, y\_val = X[split\_idx:split\_idx + fold\_size], y[split\_idx:split\_idx + fold\_size]

# 5. Construir modelo LSTM con Dropout y EarlyStopping

model = Sequential([

LSTM(50, input\_shape=(seq\_length, 1), return\_sequences=False),

Dropout(0.3),

Dense(1)

])

model.compile(optimizer='adam', loss='mse')

early\_stop = EarlyStopping(monitor='val\_loss', patience=5, restore\_best\_weights=True)

history = model.fit(

X\_train, y\_train,

validation\_data=(X\_val, y\_val),

epochs=100,

batch\_size=16,

callbacks=[early\_stop],

verbose=0 )

val\_loss = model.evaluate(X\_val, y\_val, verbose=0)

results.append(val\_loss)

print(f"Fold {i+1}, Validation Loss: {val\_loss:.5f}")

# 6. Mostrar resultado final

print("\nValidation Loss por fold:", [round(r, 5) for r in results])

print("Promedio Validation Loss:", round(np.mean(results), 5))

📈 ¿Qué hace este código?

1. Crea una **serie temporal con estacionalidad mensual** y ruido.
2. Escala y estructura los datos en secuencias de 14 días para predecir el siguiente.
3. Usa una validación cruzada con **expanding window**: cada fold entrena con más datos.
4. Usa **LSTM de una capa**, **Dropout** del 30%, y **early stopping** con patience=5.
5. Imprime el error de validación (mse) en cada fold.

🧪 ¿Qué puedes modificar fácilmente?

| **Elemento** | **Qué cambiar** |
| --- | --- |
| seq\_length = 14 | Longitud del historial usado como entrada |
| n\_splits = 5 | Número de validaciones cruzadas |
| Dropout(0.3) | Regularización (más o menos riesgo de overfitting) |
| patience=5 | Cuánto esperar sin mejora antes de detener entrenamiento |
| LSTM(50) | Capacidad de memoria de la red |

# Prompt 21

Contexto: Generación de imágenes de apoyo

Fecha: septiembre 2025

Prompt: Genera una imagen con los logos de PyCharm, Python, OpenAI y Google Colaboratory en una fila