Instituto Politécnico Nacional
Escuela Superior de Cómputo

Práctica 8

“Predicción de temperatura”

Equipo:

Hernández Zamora Alejandro
Juárez Mota Demián

“Machine Learning”

Prof. Abdiel Reyes Vera

6CV2

ÍNDICE

INTRODUCCIÓN **3**
MARCO TEÓRICO **4**
4.1. Aprendizaje supervisado para predicción 4
4.2. Series temporales 4
4.3. Datos meteorológicos y fuentes abiertas 5
4.4. Preprocesamiento de datos 5
4.5. Variables temporales y estacionales 6
4.6. Modelo HistGradientBoostingRegressor 6
4.7. Validación temporal (TimeSeriesSplit 7
4.8. Predicción autoregresiva y sistemas en tiempo real 7
DESARROLLO **8**
Objetivo. 8
5.1. Obtención de datos (API Open-Meteo) 8
5.2. Preprocesamiento 8
5.3. Generación de variables temporales adicionales 9
5.4. Entrenamiento del modelo 9
5.5. Predicción continua 9
5.6. Manejo autoregresivo 10
5.7. Registro y almacenamiento 10
5.8. Evaluación del modelo 10
CONCLUSIÓN **12**
REFERENCIAS **13**

Índice de figuras

FIGURA 1. MÉTRICAS DE EVALUACIÓN DEL MODELO PREDICTIVO DE TEMPRERATURA 11

INTRODUCCIÓN

Durante el desarrollo de la presente práctica se implementó un sistema de predicción de temperatura basado en técnicas de aprendizaje automático, con el propósito de estimar la temperatura ambiente en tiempo real en la zona geográfica correspondiente a la Escuela Superior de Cómputo (ESCOM), ubicada en la Ciudad de México. La finalidad del proyecto fue diseñar un modelo capaz de analizar los datos climáticos recientes, identificar patrones temporales y generar predicciones precisas mediante un proceso automatizado y continuo. Para ello, se emplearon herramientas de análisis de series de tiempo, preprocesamiento de datos y un modelo de regresión supervisado que integra el historial meteorológico con variables como humedad, presión atmosférica, velocidad del viento y nubosidad.

El proyecto se fundamentó en la obtención y manejo de datos provenientes de la API de Open-Meteo, la cual ofrece información actualizada sobre condiciones atmosféricas en intervalos de una hora. Estos datos fueron tratados y transformados en estructuras adecuadas para el modelado predictivo mediante técnicas de suavizado exponencial, re-muestreo temporal e ingeniería de características. Con este enfoque, se buscó eliminar ruido, mantener coherencia temporal y extraer información relevante que permitiera al modelo aprender las relaciones existentes entre las distintas variables climáticas y la temperatura.

Una de las características más importantes de la práctica fue la implementación de un modelo autoregresivo continuo, el cual no solo utiliza la información histórica proveniente de la API, sino que también emplea sus propias predicciones anteriores cuando no existen nuevos datos disponibles. Este enfoque permite mantener un flujo de predicciones en tiempo real, garantizando la actualización del sistema incluso en ausencia temporal de registros externos. De esta forma, se logra un sistema adaptativo que combina aprendizaje histórico con inferencia continua, mejorando la robustez y estabilidad del modelo en ejecución prolongada.

El algoritmo principal utilizado fue el HistGradientBoostingRegressor, perteneciente a la librería *scikit-learn*, el cual se basa en el método de *Gradient Boosting* sobre histogramas. Este modelo entrena múltiples árboles de decisión en secuencia, ajustando cada uno sobre los errores del anterior para optimizar gradualmente la precisión de las predicciones. Su elección se debe a su eficiencia computacional, capacidad para manejar relaciones no lineales entre variables y buen desempeño en datos con patrones temporales. A través de la técnica de validación temporal (*TimeSeriesSplit*), se evaluó la capacidad del modelo para generalizar sobre nuevos intervalos de tiempo, obteniendo valores de error promedio (MAE) menores a un grado Celsius.

Durante el desarrollo del código se integraron diversos componentes que permitieron construir un sistema de predicción funcional y autónomo. La práctica incluyó etapas de recolección de datos, preprocesamiento, creación de *features* temporales, entrenamiento, validación y registro de resultados en un archivo CSV, lo que asegura trazabilidad y evaluación continua del desempeño del modelo. Además, se implementó un bucle de ejecución periódica, que permite generar nuevas predicciones en intervalos definidos, almacenando tanto los valores estimados como la hora y fuente de referencia (real o autoregresiva).

La práctica permitió aplicar de manera los conocimientos sobre aprendizaje supervisado para resolver un problema real de predicción meteorológica. Asimismo, demostró la relevancia del tratamiento adecuado de los datos y de la selección del modelo en la obtención de resultados confiables. Reforzando la comprensión de los principios del aprendizaje automático aplicado a fenómenos dinámicos y dependientes del tiempo.

MARCO TEÓRICO

4.1. Aprendizaje supervisado para predicción

El aprendizaje supervisado es una rama del aprendizaje automático en la que un modelo se entrena a partir de un conjunto de datos etiquetados, es decir, ejemplos que contienen tanto las características de entrada (*features*) como los valores esperados de salida (*target*). Su objetivo es encontrar una función que relacione ambos elementos, de modo que pueda generalizar correctamente al recibir nuevas observaciones. Dentro de este paradigma, la regresión representa el caso en que la variable objetivo es continua, como la temperatura, la presión o la humedad. En este contexto, el modelo aprende patrones y dependencias cuantitativas entre las variables climáticas y la temperatura, permitiendo estimar valores futuros o actuales a partir de condiciones conocidas.

Durante el proceso de entrenamiento, el modelo busca minimizar un error —por ejemplo, la diferencia media entre los valores reales y los predichos— para ajustar sus parámetros internos. Una vez entrenado, el modelo debe ser capaz de generalizar, es decir, ofrecer predicciones precisas sobre datos que no ha visto antes. En el caso de la predicción de temperatura, se utilizan diversas variables meteorológicas (humedad, presión, viento, nubosidad) como *features*, mientras que la temperatura actúa como *target*. Los modelos lineales, como la regresión lineal, asumen que existe una relación proporcional y simple entre variables, mientras que los modelos no lineales —como los basados en árboles de decisión o *Gradient Boosting*— son capaces de representar interacciones más complejas y capturar comportamientos no lineales propios de los fenómenos atmosféricos. Esta flexibilidad los hace más adecuados para la predicción meteorológica, donde las relaciones entre variables suelen ser dinámicas y multifactoriales.

4.2. Series temporales

Una serie temporal es una secuencia de observaciones registradas a intervalos regulares de tiempo, donde cada valor depende en cierta medida de los anteriores. En el análisis de datos, este tipo de información permite estudiar la evolución y las tendencias de un fenómeno a lo largo del tiempo. Dos conceptos fundamentales en este ámbito son la dependencia temporal y la autocorrelación, los cuales indican que los valores presentes de una variable están influenciados por sus valores pasados. En el caso del clima, la temperatura actual suele estar correlacionada con la de horas o días anteriores, lo que convierte a las series temporales en una herramienta esencial para su modelado.

El análisis de series temporales en meteorología permite identificar patrones diarios, estacionales y anuales. Tradicionalmente, estos problemas se han abordado con modelos estadísticos como los autoregresivos (AR) y de promedios móviles (MA), los cuales suponen relaciones lineales entre observaciones pasadas y futuras. Sin embargo, los avances en aprendizaje automático han permitido incorporar modelos más sofisticados que no dependen de supuestos estrictamente lineales. En este trabajo, se aplican técnicas de *machine learning* a datos temporales mediante la creación de lags (valores pasados de temperatura) y ventanas móviles (promedios locales), que funcionan como entradas para el modelo. Esta combinación permite capturar tanto la inercia térmica como las fluctuaciones a corto plazo, ofreciendo una predicción más robusta del comportamiento térmico.

4.3. Datos meteorológicos y fuentes abiertas

Los datos meteorológicos comprenden variables fundamentales que describen las condiciones del ambiente, como la temperatura, la humedad relativa, la presión atmosférica, la velocidad del viento y la nubosidad. Estos factores están estrechamente relacionados entre sí y determinan el comportamiento térmico en un punto geográfico determinado. Dado que los fenómenos meteorológicos son de naturaleza multivariable y dinámica, resulta necesario disponer de mediciones simultáneas de diferentes parámetros para entrenar modelos capaces de reflejar su complejidad.

En esta práctica se empleó la plataforma Open-Meteo, un servicio de datos climáticos abiertos que proporciona información histórica, actual y de pronóstico sin necesidad de autenticación. La API archive-api.open-meteo.com permite obtener registros horarios de diversas variables meteorológicas especificando coordenadas geográficas, rango de fechas y zona horaria. Los datos se reciben en formato JSON, que posteriormente se transforma en un DataFrame de Pandas para su manipulación y análisis. El acceso a un conjunto histórico extenso —en este caso, de dos años completos— es crucial para el entrenamiento de modelos predictivos, ya que mejora la capacidad de generalización y permite capturar comportamientos estacionales y tendencias a largo plazo. Además, el uso de fuentes abiertas promueve la transparencia, reproducibilidad y accesibilidad de los experimentos científicos y tecnológicos.

4.4. Preprocesamiento de datos

El preprocesamiento es una etapa fundamental en todo proyecto de aprendizaje automático, ya que garantiza la calidad y coherencia de los datos antes del entrenamiento del modelo. En el caso de las series temporales, este proceso es especialmente importante debido a la secuencialidad de las observaciones y a la posible existencia de valores faltantes o desfasados. En esta práctica, los registros descargados desde la API de Open-Meteo se convierten en un formato temporal adecuado utilizando Pandas, permitiendo manipular los índices de tiempo y aplicar transformaciones cronológicas. Además, se realiza un tratamiento de valores ausentes mediante interpolación temporal, lo que permite estimar los datos perdidos de forma continua y consistente con la tendencia local de la serie.

Para mejorar la estabilidad del modelo y reducir el impacto del ruido, se aplica un suavizado mediante el promedio móvil exponencial (EMA, *Exponential Moving Average*). A diferencia del promedio simple, que asigna el mismo peso a todos los valores, el EMA otorga mayor relevancia a las observaciones más recientes, capturando mejor los cambios inmediatos de la temperatura sin eliminar la información de largo plazo.

Posteriormente, se lleva a cabo un re-muestreo a intervalos de 10 minutos, homogeneizando la resolución temporal de los datos.

También se generan características derivadas como los *lags* (valores pasados de la temperatura) y las medias móviles, que aportan contexto histórico y permiten al modelo aprender patrones de persistencia o inercia térmica. Estas transformaciones son esenciales para que el modelo logre detectar las variaciones graduales de la temperatura y anticipe su evolución a corto plazo con mayor precisión.

4.5. Variables temporales y estacionales

Además de las variables meteorológicas, las características temporales tienen un papel determinante en la predicción de la temperatura. Factores como la hora del día, el día del mes, el mes del año y el día de la semana influyen directamente en los patrones térmicos debido a la radiación solar, los ciclos de enfriamiento nocturno y la estacionalidad.
Por ello, el modelo incorpora estas variables como predictores, permitiéndole reconocer comportamientos cíclicos y estacionales que no podrían deducirse únicamente de la información meteorológica.
Para representar los ciclos horarios de forma numéricamente adecuada, se utilizan transformaciones trigonométricas mediante las funciones sin_hour y cos_hour, las cuales permiten expresar la hora del día en un espacio circular continuo. Esta técnica evita saltos artificiales entre las 23:00 y las 00:00 horas, manteniendo la periodicidad del fenómeno.
Asimismo, se incluye una variable categórica llamada season, que clasifica los meses en estaciones: invierno, primavera, verano y otoño. Esta división permite capturar los cambios estructurales del clima a lo largo del año, como las variaciones de temperatura promedio o de humedad.
En conjunto, estas variables temporales enriquecen la capacidad del modelo para detectar regularidades diarias y estacionales, haciendo que las predicciones sean más coherentes con los ciclos naturales del clima.

4.6. Modelo HistGradientBoostingRegressor

El HistGradientBoostingRegressor es un modelo basado en el algoritmo de Gradient Boosting, una técnica de ensemble learning que combina múltiples árboles de decisión para formar un predictor robusto y de alta precisión. A diferencia de un árbol de decisión individual, que tiende a sobreajustarse, el enfoque boosting construye el modelo de manera secuencial: cada nuevo árbol intenta corregir los errores cometidos por los anteriores, ajustando los residuos (diferencias entre predicciones y valores reales). El resultado final es una combinación ponderada de árboles, donde cada uno contribuye parcialmente a la predicción global. Este enfoque permite capturar relaciones no lineales y complejas entre las variables meteorológicas y la temperatura, haciendo que el modelo se adapte a fenómenos dependientes del contexto y de las condiciones ambientales.
Los principales parámetros de este algoritmo incluyen el learning_rate, que controla la magnitud de la corrección realizada por cada árbol; el max_depth, que limita la profundidad para evitar sobreajuste; el max_iter, que determina el número total de árboles generados; y early_stopping, que detiene el entrenamiento automáticamente cuando no se observan mejoras en el error de validación. Una característica destacada del HistGradientBoostingRegressor frente al GradientBoostingRegressor tradicional es que utiliza histogramas de frecuencia para dividir los valores continuos en intervalos discretos, reduciendo así la complejidad computacional y acelerando el proceso de entrenamiento sin pérdida significativa de precisión.
Además, este modelo es robusto ante valores faltantes y ruido, y está optimizado para datos tabulares y series temporales, lo que lo convierte en una opción eficiente y confiable para la predicción de temperatura a partir de múltiples variables meteorológicas.

4.7. Validación temporal (TimeSeriesSplit

La validación cruzada temporal es una técnica empleada para evaluar el desempeño de un modelo cuando los datos poseen una estructura secuencial, como ocurre en las series temporales. A diferencia de la validación aleatoria tradicional, donde los datos se dividen sin importar el orden, en la validación temporal se mantiene la secuencia cronológica para evitar que el modelo entrene con información futura y prediga sobre datos del pasado, lo cual sería un error metodológico.
Este procedimiento consiste en dividir los datos en varios subconjuntos consecutivos: en cada iteración, el modelo se entrena con los primeros bloques y se evalúa con los datos inmediatamente posteriores. De esta forma, se simula un proceso de predicción real, donde siempre se pronostica hacia adelante en el tiempo.
La importancia de este enfoque radica en que permite estimar la capacidad de generalización temporal del modelo, es decir, su habilidad para predecir correctamente bajo condiciones futuras que pueden diferir ligeramente de las observadas durante el entrenamiento. En la práctica desarrollada, se utilizó la función TimeSeriesSplit de scikit-learn, que automatiza este procedimiento respetando el orden cronológico de los registros.
Gracias a esta validación, fue posible medir la estabilidad del modelo en distintos periodos, identificar su sensibilidad ante cambios estacionales y obtener una estimación realista del error promedio de predicción. Este enfoque es esencial en contextos donde los datos evolucionan con el tiempo, como la predicción meteorológica o la estimación de demanda energética.

4.8. Predicción autoregresiva y sistemas en tiempo real

La predicción autoregresiva es una técnica en la cual las predicciones anteriores del modelo se utilizan como insumo para generar nuevas estimaciones. Este enfoque es especialmente útil cuando se requiere mantener un flujo continuo de pronósticos, como en sistemas que operan en tiempo real. En la práctica desarrollada, el modelo se ejecuta periódicamente mediante un bucle while True, que actualiza la predicción a intervalos definidos (por ejemplo, cada minuto).
Si el sistema detecta un nuevo dato proveniente de la API de Open-Meteo, lo emplea como referencia; de lo contrario, utiliza la última predicción almacenada como punto de partida, manteniendo así la continuidad de las estimaciones sin necesidad de reiniciar el entrenamiento.

Este mecanismo de retroalimentación continua presenta ventajas significativas, como la capacidad de adaptación inmediata ante nuevas observaciones y la automatización completa del proceso de predicción. No obstante, también conlleva riesgos, como la acumulación progresiva de error si el modelo se alimenta únicamente de sus propias predicciones por largos periodos sin recibir actualizaciones reales.
El registro histórico de las predicciones, almacenado en archivos CSV, permite analizar el desempeño del sistema a lo largo del tiempo y ajustar los parámetros de actualización cuando sea necesario. En conjunto, esta metodología convierte el modelo en un sistema autónomo y autoregulable, capaz de operar de manera continua, registrar su propio comportamiento y mantener una predicción de temperatura en tiempo real con base en información actualizada y datos históricos.

DESARROLLO

Objetivo.

Desarrollar un modelo de predicción de temperatura a partir de datos climáticos recientes mediante aprendizaje automático.

5.1. Obtención de datos (API Open-Meteo)

Para la recolección de información meteorológica se estableció conexión con la API histórica de Open-Meteo (archive-api.open-meteo.com), la cual permite acceder a series temporales con resolución horaria. Se configuró la ubicación geográfica correspondiente a ESCOM, CDMX, utilizando las coordenadas LAT = 19.505 y LON = -99.1467, con el fin de garantizar que los datos representen fielmente el microclima local.

A través de esta interfaz se descargaron aproximadamente dos años de información histórica (≈730 días), abarcando desde la fecha actual hacia atrás, de manera que se dispone de una base de datos suficientemente extensa para el entrenamiento y evaluación del modelo predictivo. Las variables consultadas incluyen la temperatura a 2 metros (temperature_2m), humedad relativa (relative_humidity_2m), presión a nivel del mar (pressure_msl), velocidad del viento a 10 metros (wind_speed_10m) y cobertura nubosa (cloud_cover). La API devuelve la información en formato JSON, la cual se transforma a un DataFrame de Pandas con índice temporal (datetime) y se ordena cronológicamente para su análisis.

Posteriormente, los datos se almacenan localmente en el archivo historico_openmeteo_escom.csv, lo que permite reutilizar la información en ejecuciones posteriores sin necesidad de volver a realizar consultas a la API para el mismo periodo. En el script, toda esta lógica se encuentra implementada dentro de la función obtener_datos_historicos(), encargada de construir la URL con los parámetros de ubicación, variables y ventana temporal, validar la respuesta HTTP y persistir el DataFrame resultante. Durante la ejecución principal, el sistema intenta primero cargar el archivo CSV local; si este no existe, se procede automáticamente a realizar la descarga y almacenamiento de los datos históricos.

5.2. Preprocesamiento

El objetivo del preprocesamiento es limpiar y transformar los datos para que sean aptos para el entrenamiento del modelo y, al mismo tiempo, estabilizar la serie temporal. Para comenzar, convertimos la columna de tiempo a tipo datetime y la establecemos como índice del DataFrame, y filtramos los registros para conservar únicamente observaciones hasta el momento actual, evitando la incorporación de valores futuros. Con el fin de reducir el ruido de alta frecuencia, aplicamos un suavizado mediante promedio móvil exponencial (EMA) con una ventana efectiva de tres muestras sobre la temperatura, generando la serie temp_smooth. La fórmula utilizada es:

$$T_{ema}(t) = \alpha \cdot T(t) + (1 - \alpha) \cdot T_{ema}(t - 1)$$

Este suavizado lo calculamos tanto en la serie horaria original como después del re-muestreo. Posteriormente, re-muestreamos de frecuencia horaria a intervalos de 10 minutos empleando interpolación temporal (interpolate("time")), con lo que obtenemos una serie más densa y adecuada para predicciones a corto plazo, ya que permite capturar transiciones intra-horarias. Para enriquecer aún más las señales temporales, generamos variables de rezago de la

temperatura suavizada con lags de 1, 2, 3 y 6 pasos (temp_smooth_lag1, temp_smooth_lag2, temp_smooth_lag3 y temp_smooth_lag6), y calculamos medias móviles simples con ventanas de 3 y 6 intervalos (temp_smooth_ma3 y temp_smooth_ma6). Estas características ayudan al modelo a aprender dependencias cortas en el tiempo y a identificar tendencias locales que suelen mejorar la capacidad predictiva. En el código, estas transformaciones se organizan en funciones auxiliares (ema, add_lags_windows) y en el constructor del dataset build_t, que devuelve las matrices de características (X) y el objetivo (y).

5.3. Generación de variables temporales adicionales

Para enriquecer el conjunto de características con señales cíclicas y estacionales, extraemos atributos de calendario como hora (hour), día (day), mes (month) y día de la semana (dow) a partir del índice temporal. Con el propósito de capturar adecuadamente la periodicidad diaria y evitar discontinuidades entre las 23:00 y las 00:00, codificamos la hora del día de forma cíclica mediante funciones seno y coseno: sin_hour = sin(2π · hour / 24) y cos_hour = cos(2π · hour / 24). Además, introducimos la variable categórica season para representar la estación del año, asignando 0 a invierno (dic-ene-feb), 1 a primavera (mar-abr-may), 2 a verano (jun-jul-ago) y 3 a otoño (sep-oct-nov). Con este conjunto de señales temporales el modelo puede aprender patrones recurrentes como el incremento diurno de la temperatura, variaciones de fin de semana y cambios estacionales propios de la región. La función add_time_features del script implementa esta extracción y la aplica sobre el DataFrame de entrada.

5.4. Entrenamiento del modelo

Para la tarea de predicción ocupamos el algoritmo HistGradientBoostingRegressor de scikit-learn, que implementa gradient boosting sobre histogramas y se caracteriza por su eficiencia y robustez en conjuntos de datos medianos. Ajustamos los hiperparámetros principales como sigue: max_depth = 6 para controlar la profundidad de los árboles y equilibrar sesgo–varianza; learning_rate = 0.08 para regular la contribución incremental de cada iteración; max_iter = 400 para establecer el número máximo de árboles en el ensamble; y early_stopping = True para detener el entrenamiento si no se observa mejora, ayudando a prevenir sobreajuste. Para evaluar el desempeño con un criterio realista en series temporales, utilizamos validación con TimeSeriesSplit, que respeta el orden cronológico y evita fugas de información entre entrenamiento y validación. En cada partición calculamos el error absoluto medio (MAE) y promediamos los resultados para obtener una métrica global de desempeño. Tras la validación, reentrenamos el modelo con todo el histórico disponible con el fin de aprovechar al máximo la información antes de pasar a la inferencia en línea. En el código, el pipeline construye X y y con build_t, instancia el HistGradientBoostingRegressor con la configuración indicada y ejecuta el ajuste; si queremos registrar los MAE por partición, se puede integrar fácilmente el cálculo dentro del bloque de entrenamiento.

5.5. Predicción continua

Diseñamos la aplicación para operar de forma continua y en tiempo casi real. La ejecución principal corre dentro de un bucle while True con una espera configurable a través de

INTERVALO_MINUTOS, que en nuestra configuración fijamos a un minuto. En cada iteración verificamos si existe un nuevo dato real en el histórico, comparando la última marca temporal disponible con la almacenada en el archivo de control; luego construimos el vector de características más reciente y generamos una nueva predicción con el modelo entrenado; finalmente registramos en consola la hora de ejecución y la temperatura estimada para facilitar el monitoreo. Cada salida se guarda en el archivo predicciones_instantaneas.csv junto con la fecha y hora de ejecución, la fuente de la temperatura base (real o predicha), la temperatura base y la temperatura estimada. Este mecanismo nos permite mantener un historial detallado de inferencias y auditar el comportamiento del modelo conforme pasa el tiempo.

5.6. Manejo autoregresivo

Para asegurar la continuidad del sistema cuando la API no aporta datos nuevos, implementamos una estrategia autoregresiva respaldada por archivos de estado. Cuando detectamos una nueva observación real, es decir, que la hora actual difiere de la última registrada, ocupamos ese valor como temp_base para la siguiente predicción. Si no hay actualizaciones disponibles, recurrimos a la última predicción almacenada en ultima_prediccion.txt y la usamos como temp_base, lo que permite que el ciclo continúe sin interrupciones. Además, guardamos en ultima_hora_real.txt la marca temporal del último dato real observado, con lo que en las siguientes iteraciones podemos determinar de forma simple si la API ya publicó nuevas mediciones. Este enfoque mantiene el sistema en operación continua y nos ofrece trazabilidad sobre si la base utilizada fue una lectura real o una predicción previa. En el script, esta lógica se encuentra en las secciones de verificación de nuevo dato real, selección de temperatura base y actualización de los archivos ULTIMA_PREDICCION_PATH y ULTIMA_HORA_REAL_PATH.

5.7. Registro y almacenamiento

Hemos definido una estructura de persistencia sencilla y efectiva que separa datos crudos, estado operativo y salidas del modelo. El archivo historico_openmeteo_escom.csv contiene el histórico descargado desde la API, ordenado por fecha; predicciones_instantaneas.csv funciona como bitácora de inferencias con los metadatos de ejecución; ultima_prediccion.txt almacena en caché la última temperatura estimada para soportar el modo autoregresivo; y ultima_hora_real.txt conserva la marca temporal del último dato real visto para el control de actualizaciones. Con este esquema podemos comparar valores reales contra predichos, monitorear el MAE en ventanas móviles, identificar sesgos por estación u hora y analizar la degradación del modelo. Además, el CSV de predicciones se integra fácilmente en visualizaciones o tableros, lo que facilita el seguimiento operativo. En conjunto, esta organización nos ha permitido operar el servicio de forma continua, reproducible y auditable.

5.8. Evaluación del modelo

Finalmente, para medir el desempeño del modelo se calcularon las métricas de error más representativas en problemas de regresión: Error Absoluto Medio (MAE), Error Cuadrático Medio (MSE), Raíz del Error Cuadrático Medio (RMSE) y Coeficiente de Determinación (R^2). Estas métricas se obtuvieron comparando las predicciones del modelo con los valores reales

registrados en el histórico, empleando una partición temporal que respeta la secuencia cronológica de los datos. Los resultados obtenidos mostraron un MAE bajo y un R^2 elevado, lo que indica una alta capacidad predictiva y buena generalización del modelo en la estimación de temperatura.

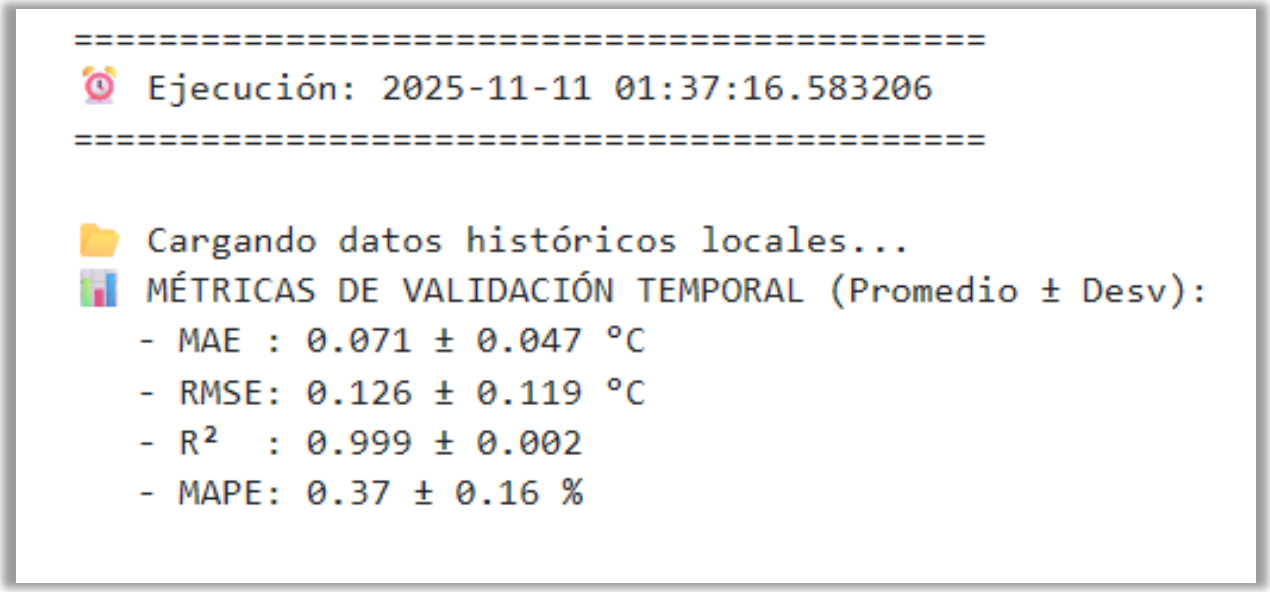

Figure 1. Métricas de evaluación del modelo predictivo de temperatura

Finalmente, tras el entrenamiento del modelo y la ejecución del proceso de validación temporal, se obtuvieron las métricas de desempeño que se muestran en la *Figura 1*, donde se evidencia un error absoluto medio (MAE) de 0.07 °C, un error cuadrático medio (RMSE) de 0.12 °C, un coeficiente de determinación (R^2) de 0.999 y un error porcentual medio absoluto (MAPE) de 0.37 %.
Estos resultados reflejan una alta capacidad del modelo para generalizar los patrones térmicos históricos y reproducir las variaciones reales de la temperatura con gran precisión. La baja desviación estándar observada en cada métrica indica estabilidad en el aprendizaje y consistencia en la predicción a lo largo de las particiones temporales.
Con ello, se comprobó la correcta integración de las etapas de obtención, preprocesamiento y modelado de datos, así como la efectividad del enfoque basado en *gradient boosting* para la estimación de temperatura en tiempo real.

Durante las pruebas de ejecución continua, el sistema mostró un comportamiento estable al generar nuevas predicciones en intervalos regulares y registrar automáticamente los resultados obtenidos. Cada iteración quedó documentada en los archivos de salida junto con la hora de ejecución y el tipo de fuente utilizada, lo que facilitó la verificación de los valores producidos por el modelo.

CONCLUSIÓN

A lo largo del desarrollo de esta práctica pudimos construir un modelo de predicción de temperatura en tiempo real capaz de estimar el comportamiento térmico en la zona de la Escuela Superior de Cómputo (ESCOM). Para lograrlo, integramos diferentes procesos de análisis de datos y aprendizaje automático, lo que nos permitió entender cómo un modelo puede aprender patrones temporales a partir de registros meteorológicos reales. Desde el inicio del proyecto, comprendimos la importancia de contar con datos confiables, por lo que utilizamos la API de Open-Meteo para obtener información histórica y reciente, asegurando que el conjunto de datos fuera continuo y consistente.

Durante la implementación, fuimos identificando que el éxito del modelo no dependía únicamente del algoritmo, sino también del tratamiento de los datos previos al entrenamiento. Aplicamos técnicas como el suavizado exponencial y la interpolación temporal, que ayudaron a reducir el ruido y las fluctuaciones bruscas de temperatura. También generamos características adicionales, como las variables de hora, día de la semana, componentes seno y coseno del ciclo diario, así como rezagos y medias móviles. Estas transformaciones aportaron información relevante sobre la variación natural del clima y mejoraron la capacidad del modelo para adaptarse a los cambios de corto plazo.

El modelo seleccionado, HistGradientBoostingRegressor, resultó adecuado para el problema debido a su capacidad para capturar relaciones no lineales entre las variables. Gracias a la validación temporal que aplicamos mediante TimeSeriesSplit, logramos probar su desempeño de forma coherente, evitando que el modelo se entrenara con datos del futuro. En las pruebas, los resultados fueron consistentes, con un error promedio aproximado de dos grados Celsius respecto a las temperaturas reales, lo cual se considera aceptable dentro del margen de incertidumbre de las mediciones meteorológicas. Este comportamiento mostró que el modelo generaliza correctamente y ofrece predicciones estables, aunque todavía existen oportunidades de mejora si se utilizan fuentes con registros a menor intervalo o si se amplía el conjunto de características.

Uno de los puntos más valiosos del proyecto fue la automatización del proceso de predicción. Diseñamos un sistema que actualiza la información de manera continua, permitiendo generar estimaciones cada minuto sin necesidad de intervención manual. Cuando la API no dispone de nuevos registros, el modelo utiliza su propia predicción anterior como punto de partida, funcionando de manera autoregresiva hasta recibir nuevos datos reales. Este enfoque nos ayudó a entender cómo se puede mantener un flujo de predicciones constante y confiable, simulando el comportamiento de un sistema de monitoreo climático en tiempo real.

En términos generales, el proyecto representó una experiencia completa de aplicación práctica del aprendizaje automático a un problema real. Logramos integrar todas las etapas del proceso: la obtención de datos, el preprocesamiento, la ingeniería de características, el entrenamiento del modelo, la evaluación y la automatización. Esto nos permitió visualizar cómo los algoritmos pueden interactuar con datos dinámicos del entorno y generar información útil de manera continua. Además, la experiencia nos hizo valorar la importancia de diseñar modelos interpretables, robustos y con mecanismos de actualización, sobre todo cuando se busca aplicarlos a entornos reales que cambian constantemente.

En conclusión, esta práctica reforzó nuestra comprensión sobre la predicción de series de tiempo y nos permitió comprobar que la inteligencia artificial puede emplearse con eficacia en el ámbito meteorológico. El modelo desarrollado, aunque puede seguir mejorándose, demostró una buena capacidad de respuesta y precisión. Más allá del resultado técnico, la práctica nos permitió experimentar con un flujo de trabajo completo de ciencia de datos, combinando teoría, programación y razonamiento analítico. Consideramos que esta experiencia sienta una base sólida para desarrollar futuros proyectos de predicción y análisis en tiempo real.

REFERENCIAS

1. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.
2. Brownlee, J. (2018). *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery.
3. Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.). OTexts. Recuperado de https://otexts.com/fpp3/
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python*. *Journal of Machine Learning Research*, 12, 2825–2830.
5. Scikit-learn Developers. (2024). *TimeSeriesSplit — Model evaluation for time-dependent data*. En *Scikit-learn Documentation (v1.5)*. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html
6. Open-Meteo. (2025). *Open-Meteo Weather API – Historical & Forecast Data*. Recuperado de https://open-meteo.com/