INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

UNIDAD DE APRENDIZAJE

Procesamiento de Lenguaje Natural | Machine Learning

PROFESOR

Abdiel Reyes Vera

Práctica 8: Regresor para predecir el clima

INTEGRANTES

González Martínez Silvia

López Reyes José Roberto

GRUPO

6CM2

FECHA DE ENTREGA

12 de noviembre de 2025

Índice

Índice de figuras

Índice de cuadros

1 Introducción

En el contexto del Machine Learning hay tres tipos de aprendizaje: supervisado, no supervisado y reforzado. El aprendizaje **supervisado** los algoritmo son entrenados usando datos etiquetados (la etiqueta describe lo que estamos viendo). El **no supervisado**, utiliza conjuntos de datos que no se encuentran etiquetados, solo poseen características y, solo escogerán las más significativas. Y finalmente el **reforzado**, que este toma el agente, ambiente y acciones.

Para el desarrollo de esta práctica tomaremos el aprendizaje supervisado como punto de partida. Este tipo tiene dos tipos de problemas: problemas de **regresión** y problemas de **clasificación**. ¿Cuál nos permitirá realizar predicciones en el futuro? descartemos de inicio la clasificación, ésta tiene como resultado una categoría o una clase, es decir, es capaz de separar elementos. Lo más habitual es que se clasifique solamente entre dos posibles clases, comúnmente: Verdadero o falso. Por el otro lado, la regresión se utiliza para *predecir* valores numéricos continuos a partir de variables de entrada.

La regresión lineal es el algoritmo más fundamental y ampliamente utilizado En los problemas de regresión los valores que se predicen son valores continuos. Se debe identitificar cuales son las variables independientes, para establecer una fórmula matemática, que se encargará de asociar el valor de la variable dependiente con las variables independientes. Esta variable dependientes es el valor a predecir. Un modelo de regresión intenta explicar el comportamiento de una variable dependiente en relación con otras variables independientes.

Para minimizar el error entre valores predichos y reales, la regresión lineal utiliza el **error cuadrático medio (MSE)** como función de costo de coste. Este método penaliza más los errores grandes debido al cuadrado de las diferencias, incentivando al modelo a encontrar el mejor ajuste posible. Por otro lado, también encontramos al **erro absoluto medio (MAE)** el cual representa el promedio de las diferencias absolutas entre predicciones y valores reales. A diferencia del MSE, el MAE no eleva al cuadrado los errores, por lo que es menos sensible a valores atípicos y proporciona una medida mpas intuitiva del error promedio. Finalmente, el **coeficiente de determinación (R^2)** indica qué proporción de la variabilidad en los datos es explicada por el modelo. Es decir, si un R^2 es de 0,85 significa que el modelo explica el 85 por ciento de la varianza en los datos, dejando solo un 15 por ciento sin explicar. Los valores de R^2 oscilan entre 0 (el modelo no explica nada) y 1 (explica perfectamente la vairanza). Es importante no interpretar R^2 de manera asilada.

El clima siempre está cambiando, hoy podemos tener temperarturas dentro del rango permitido pero, el día de mañana puede ser diferente. El clima se refiere a las condiciones atomosféricas promedio de una región durante un largo período, incluyendo variables como la temperaturam la humadad, la precipitación y el viento. La meteorología estudia estos fenómenos para **predecir** eventos climáticos futuros, lo que es crucial para diversas actividades.

Y es ahí en donde se vuelven uno mismo: la predicción del clima y la regresión. Realizar un análisis de regresión nos permitirá detectar patrones en los datos y poryectarlos hacia el futuro ¿cómo se aplica en el clima? para realizar una buena predicción, se debe incluir la observación y obtención de datos históricos, además, teniendo en cuenta que el clima puede ser influencia por factores como la latitud, altitud, corriente oceánicas, relieve y albedo. Estos factores no actúan de manera aislada; interactúan entre sí para crear el clima característico de cada región.

La capidad de elegir el modelo adecuado de regresión puede tener un impacto significativo en la calidad de las decisiones. La elección debe basarse no solo en la precisión del modelo, sino también en su alineación con lo objetivos que se tienen (predecir el clima). A lo largo de está práctica, se mostrará la serie de pasos que se siguieron para conseguir el modelo ideal para poder predecir el clima.

2 Objetivo

Desarrollar un modelo de regresión capaz de predecir el clima en la Alcaldía Gustavo A. Madero, utilizando datos históricos, con el fin de encontrar el mejor ajuste del modelo a los datos de la región.

3 Marco teórico

3.1 Regresión en Machine Learning

En el aprendizaje automático, la **regresión** es una familia de algoritmos de **aprendizaje supervisado** cuyo objetivo principal es predecir un valor numérico *continuo*.

A diferencia de los problemas de clasificación, que buscan asignar una etiqueta o una categoría (por ejemplo, lluvioso o soleado), los problemas de regresión buscan responder a la pregunta '¿cuánto?' o '¿cuál será el valor?'.

Para lograr esto, el modelo de regresión aprende una función de mapeo (f) a partir de un conjunto de datos de entrenamiento. Esta función busca modelar la relación matemática entre un conjunto de variables de entrada, conocidas como características o *features* (X), y la variable de salida continua, conocida como *target* (y).

$$y \approx f(X)$$

3.2 Series Temporales

Una **serie temporal** (time series) es una secuencia de puntos de datos recopilados en intervalos de tiempo sucesivos y ordenados. A diferencia de los problemas de regresión estándar, donde se asume que cada observación es independiente de las demás, en una serie temporal el *orden* cronológico de los datos es la característica más fundamental.

En este proyecto, el conjunto de datos de temperatura horaria es un ejemplo clásico de serie temporal. El valor de la temperatura en un instante t no es aleatorio, sino que depende fuertemente de los valores observados en instantes anteriores (ejemplo: la temperatura a las 10:00 AM está fuertemente influenciada por la de las 9:00 AM).

Tradicionalmente, una serie temporal se puede descomponer en cuatro componentes principales:

- **Autocorrelación (o Dependencia Temporal):** Es la correlación de la serie consigo misma en diferentes puntos en el tiempo. Esta propiedad es la base de la predicción, ya que implica que los valores pasados contienen información sobre los valores futuros. Es la justificación para crear *features* de *lags* (como `temperatura_hace_1hora`).
- **Estacionalidad (Seasonality):** Se refiere a los patrones cíclicos, predecibles y repetitivos en los datos que ocurren en intervalos de tiempo fijos. Nuestro conjunto de datos exhibe dos formas claras de estacionalidad:
    - **Ciclo Diurno (Diario):** La temperatura sube sistemáticamente durante el día y baja durante la noche.
    - **Ciclo Anual (Estacional):** La temperatura promedio es más alta en los meses de primavera/verano y más baja en otoño/invierno.

- **Tendencia (Trend):** Es el movimiento o dirección a largo plazo de la serie (ej. un aumento o disminución gradual de la temperatura promedio a lo largo de varios años, lo que podría estar asociado al calentamiento global).
- **Ruido (Noise o Residual):** Es la componente aleatoria e impredecible de la serie que no puede ser explicada por las otras tres componentes. Eventos atípicos, como un **frente frío** repentino, se manifiestan como ruido. El objetivo de un modelo de regresión robusto es capturar con éxito la autocorrelación y la estacionalidad, dejando solo el ruido como el error irreducible (medido por nuestro RMSE).

3.3 Ingeniería de Características para Series Temporales

Los modelos de regresión, como `HistGradientBoostingRegressor` o `SVR`, no pueden interpretar una marca de tiempo (timestamp) en su formato nativo (p. ej., `"2025-11-10 21:00:00"`). Para que el modelo pueda aprender de los patrones temporales, primero debemos cuantificar el tiempo.

La *ingeniería de características* (feature engineering) es el proceso de transformar estos datos crudos en un conjunto de *features* numéricas que representen explícitamente la información contenida en el tiempo. Para este proyecto, se implementaron tres técnicas principales para capturar la estacionalidad y la autocorrelación de la serie.

3.3.1. Codificación Cíclica (Seno/Coseno)

Las características temporales como la hora del día, el mes del año o el día del año son **cíclicas**. Por ejemplo, la hora 23:00 (11 PM) está a solo una hora de la 00:00 (medianoche). Sin embargo, si se codifican numéricamente como 23 y 0, el modelo los percibe como valores extremadamente distantes, lo cual es incorrecto.

Para resolver esto, se aplica una **codificación cíclica** que mapea estos números a un espacio bidimensional (un círculo), utilizando las funciones seno y coseno. Esto asegura que los extremos del ciclo (como 23 y 0, o diciembre y enero) sean adyacentes en este nuevo espacio.

$$X_{\text{sin}} = \text{sin}\left(\frac{2\pi \cdot X}{X_{\text{max}}}\right)$$

$$X_{\text{cos}} = \text{cos}\left(\frac{2\pi \cdot X}{X_{\text{max}}}\right)$$

En este proyecto, se generaron 6 *features* cíclicas: `hora_sin`/`hora_cos` (con $X_{\text{max}} = 24$), `mes_sin`/`mes_cos` (con $X_{\text{max}} = 12$), y `dia_anio_sin`/`dia_anio_cos` (con $X_{\text{max}} = 365$).

3.3.2. Lags (Retrasos)

Las *features* de **lag** (retraso) son la forma más directa de modelar la autocorrelación de la serie temporal. Se basan en la premisa de que el valor futuro de la temperatura depende fuertemente de sus valores pasados.

Un lag se crea ”desplazando”la serie de tiempo hacia adelante por k períodos. Esto crea una nueva característica donde cada fila en el tiempo t contiene el valor que la serie tenía en el instante $t-k$. Por ejemplo, la *feature* `temperatura_hace_1hora` le permite al modelo establecer una relación directa entre la temperatura de la hora anterior y la temperatura de la hora actual.

Se pueden crear múltiples lags para capturar la memoria a corto plazo (ej. 1 hora) y a largo plazo o estacional (ej. 24 horas, 1 semana, o incluso 1 año).

3.3.3. Ventanas Móviles (Rolling Windows)

Mientras que los lags proporcionan el valor de un único punto en el pasado, las **ventanas móviles** (o *rolling windows*) resumen un rango de puntos pasados. Esto permite al modelo capturar la tendencia reciente y lo hace más robusto al ruido.º a fluctuaciones momentáneas.

Se define una ventana de tamaño w y se calcula una estadística (como la media, mediana, o desviación estándar) sobre los w puntos anteriores. Por ejemplo, la *feature* `temp_media_ultimas_3h` calcula el promedio de la temperatura de las 3 horas previas. Esta *feature* le informa al modelo si la temperatura está ”subiendo.º ”bajandorecientemente.

3.4 Modelos de Regresión Evaluados

Para determinar el mejor enfoque para predecir la temperatura, se evaluó un amplio espectro de algoritmos de regresión. Cada modelo tiene una arquitectura y un enfoque matemático fundamentalmente diferente para aprender la relación entre las *features* temporales y la temperatura objetivo.

3.4.1. RandomForestRegressor

Es un modelo de ensamble que opera construyendo una multitud de árboles de decisión durante el entrenamiento (de ahí el ”bosque”). Cada árbol se entrena con una submuestra aleatoria de los datos (*bootstrapping*). La predicción final es el promedio de las predicciones de todos los árboles individuales, lo que lo hace muy robusto contra el sobreajuste (overfitting).

3.4.2. GradientBoostingRegressor

A diferencia de RandomForest, este es un modelo de *boosting*. Construye los árboles de decisión de forma secuencial: cada nuevo árbol se entrena para corregir los errores residuales del árbol anterior. Es un algoritmo extremadamente potente, pero su entrenamiento es computacionalmente muy lento y puede ser sensible al sobreajuste.

3.4.3. HistGradientBoostingRegressor

Es la implementación moderna y de alto rendimiento de Scikit-Learn, inspirada en LightGBM. Su principal ventaja es la velocidad: en lugar de buscar el çorte”óptimo en cada *feature*, primero discretiza las *features* continuas en *bins* (histogramas). Crucialmente para este proyecto, tiene la capacidad de manejar valores faltantes (NaN) de forma nativa, lo que lo hace más rápido y robusto.

3.4.4. XGBRegressor

Es una implementación optimizada y altamente paralelizable de Gradient Boosting. Famoso por su dominio en competencias de ciencia de datos, XGBoost incorpora regularización L1

(Lasso) y L2 (Ridge) directamente en su función de costo, lo que ayuda a controlar el sobreajuste y, a menudo, conduce a una precisión superior.

3.4.5. Ridge

Es un modelo de regresión lineal que introduce un término de regularización L2. Esta penalización (la suma de los cuadrados de los coeficientes) evita que los coeficientes del modelo crezcan de manera descontrolada. Es muy eficaz para prevenir el sobreajuste en modelos lineales y manejar la multicolinealidad (cuando las *features* están correlacionadas entre sí).

3.4.6. Lasso

Similar a Ridge, Lasso es una regresión lineal con regularización, pero utiliza una penalización L1 (la suma de los valores absolutos de los coeficientes). Su propiedad más distintiva es que puede reducir los coeficientes de las *features* menos importantes a exactamente cero, realizando así una selección automática de características.

3.4.7. Support Vector Regressor (SVR)

Es la contraparte de regresión de las Máquinas de Vectores de Soporte (SVM). A diferencia de los modelos lineales que intentan minimizar el error de *todos* los puntos, SVR intenta ajustar la mayor cantidad de datos posible dentro de un "tubo.º margen de error (definido por el hiperparámetro ϵ). Solo los puntos fuera de este tubo (los "vectores de soporte") influyen en el modelo, haciéndolo robusto a valores atípicos. Es altamente dependiente del escalado de datos.

3.4.8. KNeighborsRegressor

Es un algoritmo no paramétrico "perezoso"(*lazy learner*). No .ªprendeüna función f durante el entrenamiento; simplemente memoriza todo el conjunto de datos. Para hacer una predicción, busca los k puntos de datos más cercanos (vecinos) en el historial (según la distancia de sus *features*) y predice el promedio de sus temperaturas. Es muy sensible al escalado de las características.

3.4.9. Multilayer Perceptron (MLP)

Es un modelo de red neuronal artificial. Consiste en una capa de entrada (nuestras *features*), una o más capas ocultas con funciones de activación no lineales (como ReLU), y una capa de salida (la temperatura). Al optimizar los "pesos"de las conexiones entre neuronas, el MLP es capaz de aprender relaciones extremadamente complejas y no lineales. Es sensible al escalado y computacionalmente costoso de entrenar.

3.5 Metodología de Evaluación y Optimización

La elección de un algoritmo de regresión es solo la primera parte del proceso. Para asegurar que el modelo tenga el mejor rendimiento posible y que su precisión sea medida de forma fiable, es fundamental implementar una metodología robusta de optimización y evaluación.

3.5.1. Afinación de Hiperparámetros (GridSearchCV)

A diferencia de los parámetros internos de un modelo (que se aprenden durante el entrenamiento), un hiperparámetro es una configuración externa que se debe definir *antes* del entrenamiento (por ejemplo, el número de árboles en un RandomForest o la profundidad máxima de un árbol).

La **Afinación de Hiperparámetros** (Hyperparameter Tuning) es el proceso de encontrar la combinación óptima de estos valores. Para este proyecto, se utilizó **GridSearchCV** (Búsqueda en Cuadrícula con Validación Cruzada). Este método realiza una búsqueda exhaustiva: se le proporciona una çuadrícula"(diccionario) de posibles valores para cada hiperparámetro, y `GridSearchCV` evalúa metódicamente cada combinación posible, seleccionando aquella que produce el mejor rendimiento según la métrica de evaluación (en nuestro caso, `neg_mean_squared_error`).

3.5.2. Validación Cruzada para Series Temporales (TimeSeriesSplit)

En los problemas de regresión estándar, la Validación Cruzada (como K-Fold) baraja los datos aleatoriamente para crear los pliegues (folds) de entrenamiento y prueba. En una serie temporal, este enfoque es fatalmente incorrecto.

Si los datos se barajan, el modelo podría ser entrenado con datos del futuro (ej. Lunes 10 de Nov.) para predecir el pasado (ej. Viernes 7 de Nov.), un fenómeno conocido como fuga de datos (data leakage) que produce métricas de rendimiento falsamente optimistas.

Para evitar esto, se utiliza **TimeSeriesSplit**. Este método crea pliegues que respetan el orden cronológico, donde los datos de entrenamiento siempre son anteriores a los datos de validación. Por ejemplo:

- **Pliegue 1:** Entrena con [Semana 1], Valida con [Semana 2].
- **Pliegue 2:** Entrena con [Semanas 1, 2], Valida con [Semana 3].
- **Pliegue 3:** Entrena con [Semanas 1, 2, 3], Valida con [Semana 4].

Esto asegura que el modelo siempre se entrene con datos del "pasado"para validar su rendimiento en el "futuroïnmediato, simulando un escenario de predicción realista.

3.5.3. Métricas de la Regresión

Para evaluar cuantitativamente el rendimiento de los modelos de regresión, se utilizaron tres métricas estándar:

1. Error Cuadrático Medio (Mean Squared Error - MSE): Es el promedio de los errores al cuadrado. Su principal característica es que penaliza fuertemente los errores grandes.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2. Raíz del Error Cuadrático Medio (Root Mean Squared Error - RMSE): Es la raíz cuadrada del MSE. Esta fue la métrica principal para la optimización de hiperparámetros. Su gran ventaja es que el resultado se expresa en las mismas unidades que la variable objetivo. Un RMSE de

1.63, en este proyecto, significa que el modelo se equivoca, en promedio, por 1.63°C.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

3. Error Absoluto Medio (Mean Absolute Error - MAE): Es el promedio de las diferencias absolutas entre la predicción y el valor real. Es menos sensible a los valores atípicos (outliers) que el RMSE y es muy fácil de interpretar.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

4. Coeficiente de Determinación (R^2): Mide la proporción de la varianza en la variable objetivo (temperatura) que es predecible a partir de las características de entrada. Un valor de 1.0 indica un ajuste perfecto, mientras que un valor de 0.0 indica que el modelo no es mejor que simplemente predecir la media histórica.

4 Desarrollo

4.1 Obtención de los datos

Para la recolección de datos se buscaron en diferentes páginas web pero, la que más se adaptó y facilitó dicha recolección fue:

visualcrossing - Historical weather data for Cuidad de México, CDMX, México

Una vez que entrabas al sitio, podías visualizar la siguiente pantalla:

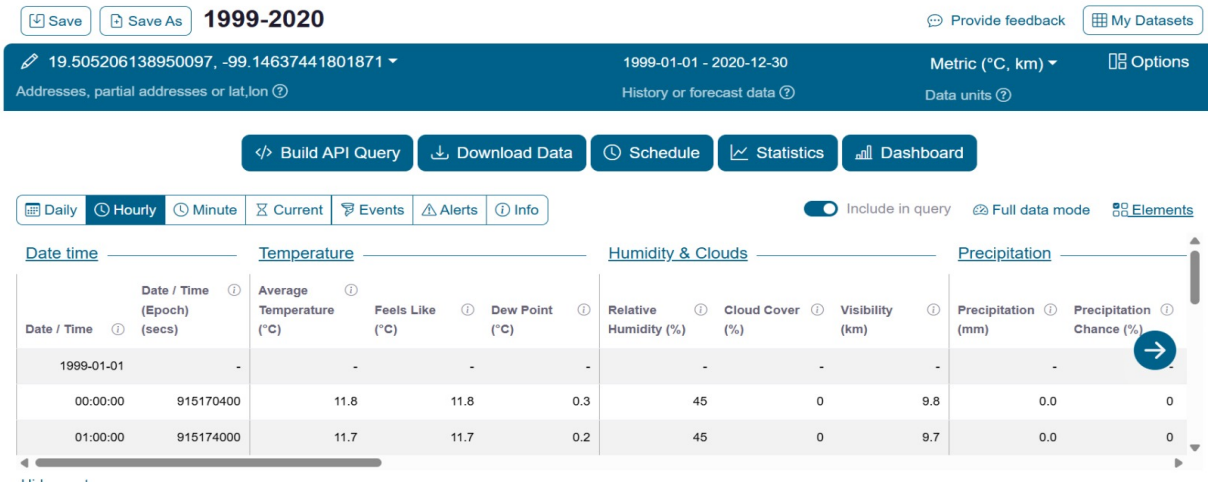

Figura 1: Pantalla que muestra la distribución de la información en la página web

El formato de los datos fue:

- Temperatura: °C
- Hora: 00:00:00

4.2 Limpieza de los datos

La limpieza de los datos se llevaron de la siguiente forma:

- Se unificaron en un sola columna fecha con tipo **datetime**, combinando fecha y hora

```
        df['fecha'] = pd.to_datetime(df['fecha'] + ' ' + df['hora'])

```

- Se ordenaron los registros cronológicamente en una nueva columna llamada **fecha**

```
        df = df.sort_values('fecha').reset_index(drop=True)

```

- Se agregaron nuevas columnas que se derivaron de la fecha: dia/anio, mes/actual, anio/actual, hora/actual

4.3 Feature Engineering

Después de la limpieza de los datos, el conjunto tenía únicamente presentes sus respectivas *timestamps* y la temperatura, también se le añadieron las categorías propias de esta Serie Temporal, tomando las temperaturas de diversos estados previos al actual.

Sin embargo, esto no es suficiente para el modelo, ya que surgen algunos problemas adicionales a la hora de utilizar algunas características de este conjunto de datos, para ello, se utilizaron diversas técnicas propias de la **Ingeniería de Características** (*Feature Engineering*) en varias de estas *features*.

Codificación de Características Cíclicas

El modelo no entiende explícitamente el ciclo que siguen las horas en nuestro sistema horario (ejemplo: no sabe que 23:00 es cercano a la hora 00:00 del siguiente día).

Esto mismo ocurre con los meses (no sabe que el mes 12 es cercano al mes 1 del siguiente año) y con los días (tampoco sabrá que el día 365 de un año es cercano al 1 del siguiente).

Para ello, se transforman estas categorías cíclicas usando una codificación de Seno y Coseno, así el modelo entenderá estas series que siguen los ciclos a través de los años.

```
# Hora (0-23, 24 valores)
df['hora_sin'] = np.sin(2 * np.pi * df['hora_actual'] / 24.0)
df['hora_cos'] = np.cos(2 * np.pi * df['hora_actual'] / 24.0)

# Mes (1-12, 12 valores)
df['mes_sin'] = np.sin(2 * np.pi * df['mes_actual'] / 12.0)
df['mes_cos'] = np.cos(2 * np.pi * df['mes_actual'] / 12.0)

# Día del año (1-365)
df['dia_anio_sin'] = np.sin(2 * np.pi * df['dia_anio'] / 365.0)
df['dia_anio_cos'] = np.cos(2 * np.pi * df['dia_anio'] / 365.0)
```

Listing 1: Creación de features cíclicas (Seno/Coseno).

Creación de Lags (Retrasos)

El modelo no va a entender con simplemente tener varios datos de fechas con su temperatura, ya que aprenderá un patrón que no nos servirá al tener una gran fluctuación en general.

Para darle una especie de memoria al modelo y que aprenda un patrón real de como varía la temperatura a través del tiempo, se utilizan los retrasos (*lags*) para darle este aprendizaje del pasado, creando estos a corto (horas), mediano (días) y largo plazo (años).

```
# Lags de Horas
df['temperatura_hace1hora'] = df[COLUMNA_TARGET].shift(1)
df['temperatura_hace3horas'] = df[COLUMNA_TARGET].shift(3)
df['temperatura_hace6horas'] = df[COLUMNA_TARGET].shift(6)
df['temperatura_hace12horas'] = df[COLUMNA_TARGET].shift(12)

# Lags de Días
df['temperatura_hace1dia'] = df[COLUMNA_TARGET].shift(24)
df['temperatura_hace3dias'] = df[COLUMNA_TARGET].shift(24 * 3)
df['temperatura_hace1semana'] = df[COLUMNA_TARGET].shift(24 * 7)

# Lags de Año
df['temperatura_hace1anio'] = df[COLUMNA_TARGET].shift(365 * 24)
```

Listing 2: Creación de features de lag (retraso).

Creación de Ventanas Móviles (Rolling Windows)

Lo último realizado en este proceso fue una manera de hacer más robusto y que capture la tendencia reciente, se crearon estas últimas características llamadas ventanas móviles (*Rolling Windows*).

Estas características calculan la temperatura promedio sobre los períodos recientes, los cuales fueron la temperatura hace 3 horas y hace 24 horas (1 día).

```
# Media de las últimas 3 horas
df['temp_media_ultimas_3h'] = df[COLUMNA_TARGET].shift(1).rolling(window=3)
    .mean()

# Media de las últimas 24 horas
df['temp_media_ultimas_24h'] = df[COLUMNA_TARGET].shift(1).rolling(window
    =24).mean()
```

Listing 3: Creación de features de ventana móvil.

4.4 Selección y entrenamiento del modelo

Para poder llevar a cabo la selección del modelo, fue necesario someterlos a ciertas métricas escoger al mejor algoritmo, el que se adaptara al problema planteado. La siguiente tabla muestra el vaciado de los reusltados obtenidos:

4.5 Código Principal de Predicciones

Finalmente, al haber limpiado por completo el cojunto de datos, haber aplicado la Ingeniería de Características y haber seleccionado el mejor modelo en la comparativa (`HistGradientBoosting`), el último paso a realizar era la programación de las predicciones a la hora actual de la prueba.

Algoritmo/Métrica	MAE	MSE	RMSE	R^2
RandomForestRegressor	0.875985823	1.518676222	1.232345821	0.886279766
GradientBoostingRegressor	0.880981395	1.477838270	1.215663716	0.889337759
HistGradientBoostingRegressor	0.859986202	1.466785584	1.211109237	0.890165397
XGBRegressor	0.905657461	1.568791410	1.252514036	0.882527083
Ridge	0.938584828	1.676965670	1.294977092	0.874426869
Lasso	0.950130565	1.687791905	1.299150455	0.873616188
SVR	0.872155670	1.503904493	1.226337838	0.887385890
KNeighborsRegressor	1.109291167	2.207474373	1.485757172	0.834701763
MLP*	1.463591579	3.638082849	1.907375906	0.727576144

Cuadro 1: Comparación de algoritmos de regresión según métricas de desempeño

División de Datos

El conjunto de datos se dividió en un 80 % para entrenamiento y un 20 % para pruebas.

A pesar de que comúnmente la división de estos datos es aleatoria, en este caso dicho enfoque podría producir anomalías al no respetarse el tiempo cronológico de los datos. En general, las Series Temporales usan un distinto enfoque de partición, respetando esta cronología a la hora de hacer la división.

```python
# División de datos
X_train, X_test = X.iloc[:int(len(X) * 0.8)], X.iloc[int(len(X) * 0.8):]
y_train, y_test = y.iloc[:int(len(X) * 0.8)], y.iloc[int(len(X) * 0.8):]

# Escalado de características
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Listing 4: División de datos cronológica (sin barajar).

Optimización de Hiperparámetros

Para encontrar la mejor configuración del modelo que devolviera los mejores resultados en la evaluación de métricas, se implementó una búsqueda en cuadrícula (`GridSearchCV`) sobre el conjunto de datos de entrenamiento.

Se utilizó el `TimeSeriesSplit` como la estrategia de validación cruzada, lo que asegura que el modelo siempre se entrene con datos del pasado para validar su rendimiento en el futuro, previniendo la fuga de datos.

La métrica de optimización fue el error cuadrático medio negativo (`neg_mean_squared_error`), con el objetivo de minimizar el RMSE.

```python
# Parámetros a probar
parameters = {
    'max_iter': [200, 400, 600],
    'learning_rate': [0.05, 0.1],
    'max_depth': [7, 10, 15],
    'l2_regularization': [0.5, 1.0],
    'min_samples_leaf': [20, 40]
}

# Usando TSS
tscv = TimeSeriesSplit(n_splits=5)
```

```

# Configurar GridSearchCV
gb = HistGradientBoostingRegressor(random_state=42)
grid_search = GridSearchCV(gb, parameters, cv=tscv,
                           scoring='neg_mean_squared_error',
                           n_jobs=-1, verbose=1)
grid_search.fit(X_train_scaled, y_train)

# Obtener mejores parámetros
best_params = grid_search.best_params_
```

Listing 5: Configuración y ejecución de GridSearchCV.

Entrenamiento del Modelo Final

Una vez identificados los mejores hiperparámetros, se entrenó el modelo final una última vez, utilizando todo el conjunto de datos para asegurar que aprenda de toda la información histórica disponible antes de hacer predicciones futuras.

```
# Re-entrenar el escalador con todos los datos
final_scaler = StandardScaler()
X_full_scaled = final_scaler.fit_transform(X)

# Re-entrenar el modelo con los mejores parámetros y todos los datos
final_model = HistGradientBoostingRegressor(**best_params, random_state=42)
final_model.fit(X_full_scaled, y)
```

Listing 6: Re-entrenamiento del modelo final con todos los datos.

5 Pruebas

5.1 Resultados de Optimización

Después de la optimización y entrenamiento del modelo, se obtuvieron los siguientes resultados, se eligió una conofiguración un un error (RMSE) decente.

```
best_params = {
    'l2_regularization': 1.0,
    'learning_rate': 0.05,
    'max_depth': 10,
    'max_iter': 200,
    'min_samples_leaf': 20
}
```

Listing 7: Hiperparámetros óptimos encontrados por GridSearchCV.

Luego, se realizó la evaluación de las métricas obtenidas por el modelo final con los mejores parámetros.

Métrica	Valor
RMSE (Error Promedio)	**1.68 °C**
MAE (Error Absoluto)	1.13 °C
Coeficiente R^2	0.88
MSE	2.81

Cuadro 2: Métricas de rendimiento del modelo final sobre el conjunto de prueba (Test Set).

El resultado más importante es el **RMSE de 1.68 °C**. Esta métrica establece nuestra línea base de rendimiento: en un escenario controlado de predicción a 1 hora (1-step-ahead), se espera que nuestro modelo tenga un error promedio de 1.67 grados centígrados. Un coeficiente R^2 de 0.88 indica que el modelo logra explicar el 88 % de la variabilidad de la temperatura, lo cual representa un ajuste estadístico muy robusto.

Visualmente, el rendimiento del modelo sobre el conjunto de prueba se puede observar en la gráfica siguiente. La línea de predicción (naranja) sigue muy de cerca a la línea de temperatura real (azul), demostrando la capacidad del modelo para capturar tanto la estacionalidad diaria como los picos y valles de temperatura.

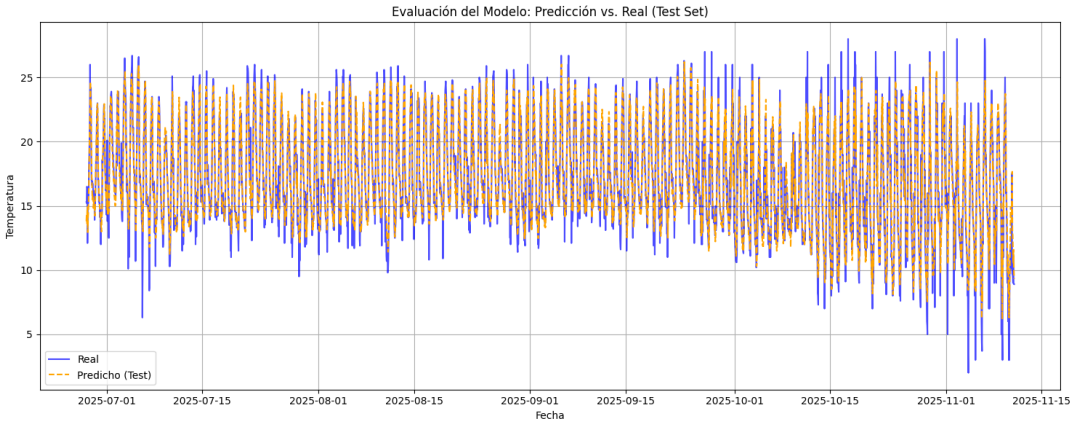

Figura 2: Comparativa de la temperatura real (azul) vs. la predicción del modelo (naranja) sobre el conjunto de prueba.

5.2 Predicción Recursiva (Análisis de Deriva)

Si bien la métrica RMSE (1.67 °C) nos da el error teórico para una predicción de un solo paso (1-step-ahead), en un escenario de despliegue real, el objetivo es predecir múltiples horas en el futuro (ej. T+1, T+2, ..., T+24).

El modelo, sin embargo, fue entrenado con *features* de lag (como `temperatura_hace_1hora`). Para predecir la hora T+2, el modelo necesita la temperatura de T+1, la cual aún no existe. Esto introduce la necesidad de una **predicción recursiva**.

La estrategia para predecir el futuro se implementó en la función `predecir_futuro_recursivo`, descrita en la sección de Desarrollo. El proceso lógico es el siguiente:

- **Paso 1 (T+1):** El modelo predice la hora T+1, utilizando únicamente datos históricos reales (ej. la temperatura de la hora T, T-1, T-2...).
- **Paso 2 (T+2):** El modelo predice la hora T+2. Para la *feature* `temperatura_hace_1hora`, utiliza la **predicción** generada en el Paso 1.
- **Paso 3 (T+n):** El proceso se repite, donde cada nueva predicción depende de las predicciones generadas en los pasos anteriores.

El modelo comienza a .ªlimentarse de sus propias predicciones", creando un bucle de retroalimentación.

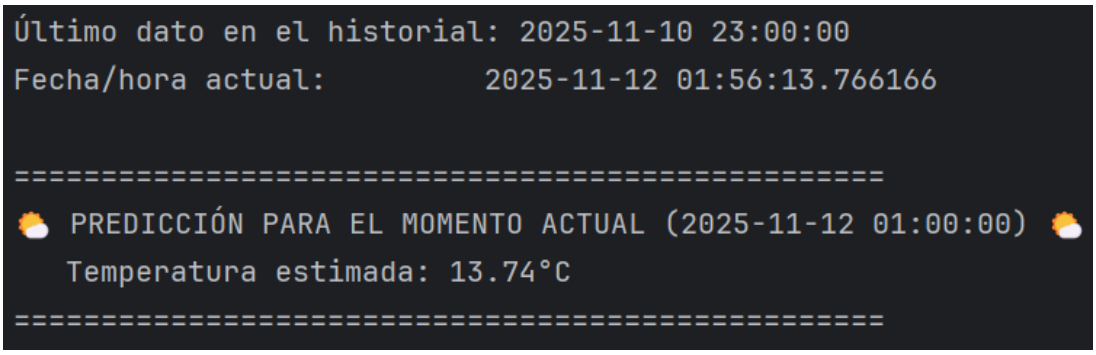

Figura 3: Prueba del sistema de predicción del clima en un momento dado.

5.2.1. Acumulación de Error

El fallo crítico de esta estrategia es la **Acumulación de Error** (Error Compounding). El modelo tiene un error inevitable en el Paso 1 (nuestro RMSE de 1.67°C). Al usar esta predicción (ya errónea) como entrada para el Paso 2, el error del Paso 2 será aún mayor.

Esto crea un fenómeno conocido como **paseo aleatorio** (*random walk*) de la predicción. El error "derivaçon cada paso:

- En un escenario de **"mala suerte"**, los pequeños errores se suman ('+1.5, +0.5, +2.0...'), y la predicción se dispara a valores absurdos.
- En un escenario de **"buena suerte"**, los errores se cancelan entre sí ('+1.5, -0.8, -0.7...'), y la predicción final puede parecer milagrosamente precisa.

La predicción a largo plazo se vuelve, por tanto, una cuestión de **suerte estadística** más que de precisión del modelo.

6 Conclusión

El desarrollo de esta práctica consisitió en realizar un modelo de regresión capaz de predecir el clima dentro de la Alcaldía Gustavo A. Madero. ¿Por qué se hizo uso de un regresor? recordemos que este nos permite predecir valores numéricos continuos a partir de variables de entrada, de ahí el uso de regresores.

El siguiente paso y el más importante fue, la obtención de los datos. Esta fase fue la que más tiempo demandó debido a que se trataban de datos históricos de años anteriores. La búsqueda fue cautelosa duarante la búsqueda de datos; no se perdió el objetivo el cual era: buscar las temperaturas de años anteriores a partir de la hora de su registro, es decir, la temperatura de hace 1,2,3 horas,etc. Una vez obtenida la información, se procedió a concentrarla para crear el dataset ideal que cumpliera con las especificaciones requeridas, una de ellas fue registrarlas por horas con el fin de tener un margen de error más grande que pequeño y así, evitar que los datos recolectados quedaran en uno solo.

Para escoger el regresor que se adapatara a las necesidades del problema fue sencilla. Primeramente se hizo un recuento de cuántos modelos de regresión nos podían servir para el plateamiento del problema pero, para obtener una mejor respuesta se emplearon métricas necesarias para evaluar modelos de regresión. De los cuales fueron: MSE, RMSE, MAE Y R^2. El primero de ellos permitió conocer el promedio de los errores (su principal característica penalizar fuertemente los errores grandes), el RMSE busca la optimización de los hiperparámetros, MAE es el promedio de las diferencias absolutas entre la predicción y el valor final, para culminar, R^2 mide la proporción de la varianza en la variable objetivo (temperatura) que es predecible a partir de las características de entrada. Un valor de 1.0 indica un ajusto perfecto, mientrs que un valor de 0.0 indica que el modelo no es mejor que simplemente predecir media histórica. Después de haber evaluado dichas métricas, la que se adoptó a la perfección fue **HistGradienteBoostingRegressor**, que a diferencia de RandomForeste, este es un modelo de boostinf. Construye los árboles de decisión de forma secuencial: cada nuevo árbol se entrena para corregir los errores residuales del árbol anteior. Es un algoritmo extremedamente potente, pero su tiempo computacionalmente es muy lento y puede ser sensible al sobreajusto.

Dicha práctica permitií saber que el clima es un fenómenos complejo y no completamente predecible. Aunque los metorólogos pueden predecir el clima a corto plazo con cierta precisión, la certeza se reduce significativamente a medida que se aleja del horizonte temporal de unos días. Estos se debe a las caacidad caótica de la atmósfera, donde incluso un pequeño error en la información incial puede llevar pronósticos erróneos. Por lo tanto, aunque se pueden hacer proyecciones a largo plazo basadas en tendencias generales el clima sigue siendo un concepto impredecible en el largo plazo.

Durante el entrenamiento, el modelo se compotó de diversas formas. Una de ellas manifestó tener buena predicción del clima, variando así alrededor de 0.2° sin embarg, en otros días contenía un margen de error bastante alto. De esta manera se planteron dos cosas: la primera de ellas era que, el dataset aún contenía información que no era útil para la ocasión por lo que, se requierió una liempeza para que los datos fueran compatibles. Finalmente, se decidió aplicar un valición cruzado (cross validation). La validación cruzada es una técnica esencial en el aprendizaje automático para eveluar la capacidad predictiva de un modelo. SU ibjetivo principal es estimar con mayor precisión el rendimiento del modelo en datos no vistos, reduciendo el riesgo de (overfitting). La validación cruzada simula la llegada de nuevas observaciones, separando los datos en varios grupos y proponiendo un subconjunto para entrenar el modelo y otro para probarlo. Este proceso se repite varias veces, cmabiando qué datos se utiliza para entrenar y cuáles para probar, lo que permite obtener una estimación más precisa del rendimiento del modelo.

Se cumplieron con los objetivos de la práctica. Se solucionaron los pruebas para concretar los resultados y así, predecir la temperatura de cualquier localidad.

7 Bibliografía

- **BertIA**. (2024, 17 de septiembre). *Algoritmos de Regresión en Machine Learning: Definición, tipos y evaluación*. BertIA. https://bertia.es/algoritmos-regresion-ml/
- **Buitrago, B.** (2020, 14 de septiembre). *Machine Learning — Modelos de Regresión I*. iWannaBeDataDriven. https://medium.com/iwannabedatadriven/
- **Chen, T., & Guestrin, C.** (2016). XGBoost: A Scalable Tree Boosting System. En *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785
- **McKinney, W.** (2010). Data structures for statistical computing in Python. En *Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 56-61). https://doi.org/10.25080/Majora-92bf1922-00a
- **OpenWeatherMap.** (s.f.). *Current Weather Data API*. Recuperado el 12 de noviembre de 2025, de https://openweathermap.org/api
- **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Dubourg, V.** (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.
- **Visual Crossing.** (s.f.). *Historical Weather Data for Mexico City, Mexico*. Visual Crossing Weather. Recuperado el 12 de noviembre de 2025, de https://www.visualcrossing.com/weather-history/Mexico+City%2CMexico/