

## Análisis del Mercado Energético Colombiano y Predicción de Precios

El propósito fundamental de este código de Python es analizar y modelar el comportamiento del precio spot de la energía eléctrica en Colombia dentro de un contexto energético global. El script se encarga de manera fluida de la adquisición, limpieza y consolidación de datos históricos de precios spot de Colombia (COP/kWh), Noruega (Nordpool) y Alemania/Francia (EUR/kWh), incluyendo también el nivel agregado de embalses en Colombia. Una vez procesados, el código visualiza las tendencias de precios a través de seis gráficas clave, destacando una fuerte correlación negativa ( $\approx -0.70$ ) entre el nivel de los embalses y el precio en Colombia. Finalmente, incorpora una funcionalidad de Machine Learning al entrenar y evaluar un modelo básico de Regresión Lineal para predecir el precio spot diario, demostrando la aplicación de técnicas predictivas mediante variables de rezago y contextuales, con un rendimiento medido por un  $R^2$  de  $\approx 0.49$ .

### 1. Importación de Librerías y Configuración Inicial

Este bloque de código se encarga de dos tareas principales:

- Importar Librerías:** Se traen todas las bibliotecas de Python necesarias para el análisis de datos, visualización y machine learning (como `pandas`, `requests`, `matplotlib`, `numpy`, `sklearn`, etc.).
- Configuración de Constantes y APIs:** Se definen variables globales como el rango de fechas para el análisis (`ANIO_INICIO`, `ANIO_FIN`), la tasa de conversión de moneda (`TASA_CAMBIO_COP_POR_EUR`), y las URLs de las APIs de donde se obtendrán los datos. También se inicializan series de Pandas vacías que se utilizarán para almacenar los datos mensuales procesados y se definen nombres de columnas estándar para mantener la consistencia. La función `clean_col_name_strict` ayuda a estandarizar los nombres de las columnas.

Ejecuta este bloque para configurar el entorno y las variables iniciales.

```
import pandas as pd
import requests
import matplotlib.pyplot as plt
import numpy as np
import unicodedata
import re

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


ANIO_INICIO = '2020-01-01'
ANIO_FIN = '2021-12-31'
TASA_CAMBIO_COP_POR_EUR = 4000


NORDPOOL_API = "https://sheetdb.io/api/v1/thma2z0rowstm"
COLOMBIA_API = "https://api.sheety.co/88787cbd1a8fa2c4c990e0a0e44a6475/preciodebolsaponderadoColombia/preciodebolsaponderadoacda7E04137D49049C5CF56Ae063390FCsv"
EMBALSES_API = "https://sheetdb.io/api/v1/cqwmh3juhhsdr"
ALEMANIA_FRANCIA_API = "https://api.sheety.co/88787cbd1a8fa2c4c990e0a0e44a6475/alemania0Francia/alemania0Francia"


df_eu_monthly_eur = pd.Series(dtype='float64')
df_co_monthly_cop = pd.Series(dtype='float64')
df_embalses_monthly_pct = pd.Series(dtype='float64')
df_de_fr_monthly_eur = pd.Series(dtype='float64')


DATE_COL_CO = 'fecha'
VALUE_COL_CO = 'valor'
DATE_EMBALSE_COL = 'Fecha'
REGION_EMBALSE_COL = 'Embalse'
VOLUME_COL_EXACT = 'Volumen_Util_Energia_GWh'
CAPACITY_COL_EXACT = 'Capacidad_Util_Energia_GWh'
VALUE_EMBALSE_COL = 'Nivel_Embalse_Porcentaje'


def clean_col_name_strict(name):
    name = unicodedata.normalize('NFKD', name).encode('ascii', 'ignore').decode('utf-8')
    name = name.replace(' ', '_').replace('-', '_')
    name = re.sub(r'^a-zA-Z0-9_', '', name)
    return name.lower()


print(f"1. Período de análisis solicitado: {ANIO_INICIO} a {ANIO_FIN}")
print(f"    Tasa de conversión: 1 EUR = {TASA_CAMBIO_COP_POR_EUR} COP")
```

1. Período de análisis solicitado: 2020-01-01 a 2021-12-31  
Tasa de conversión: 1 EUR = 4000 COP

### 2. Procesamiento de Datos de Colombia (COP/kWh)

Este bloque se enfoca en obtener y preparar los datos de precios spot de Colombia:

- Descarga y Carga:** Realiza una petición `GET` a la `COLOMBIA_API` y carga la respuesta JSON en un DataFrame de pandas.
- Limpieza de Datos:** Renombra columnas para estandarización, convierte las columnas de fecha (`fecha`) y valor (`valor`) a los tipos correctos (`datetime` y `float`).
- Manejo de Nulos:** Elimina filas con valores faltantes esenciales.
- Agregación y Filtrado:** Convierte los datos a un promedio diario, filtrando solo las fechas dentro del `ANIO_INICIO` y `ANIO_FIN`.
- Cálculo Mensual:** Finalmente, calcula el promedio mensual de los precios y lo almacena en `df_co_monthly_cop`.

Ejecuta este bloque para procesar los datos de Colombia.

```
print("\n2. Procesando datos de Colombia")
try:
    response_co = requests.get(COLOMBIA_API, timeout=30)
    response_co.raise_for_status()

    data_json = response_co.json()
    if isinstance(data_json, dict) and data_json:
        data_list = data_json.get(list(data_json.keys())[0], [])
        df_co_raw = pd.DataFrame(data_list)
    else:
        df_co_raw = pd.DataFrame(data_json)

    df_co_raw.columns = df_co_raw.columns.str.lower()
    df_co_raw.rename(columns={'fecha': DATE_COL_CO, 'valor': VALUE_COL_CO}, inplace=True)
```

```
df_co_raw[DATE_COL_CO] = pd.to_datetime(df_co_raw[DATE_COL_CO], errors='coerce')
df_co_raw[VALUE_COL_CO] = pd.to_numeric(
    df_co_raw[VALUE_COL_CO].astype(str).str.replace(',', '.', regex=False), errors='coerce'
).astype(float)

df_co_raw.dropna(subset=[DATE_COL_CO, VALUE_COL_CO], inplace=True)

df_co_diario = df_co_raw.set_index(DATE_COL_CO).sort_index()
df_co_diario = df_co_diario[[VALUE_COL_CO]].resample('D').mean()
df_co_diario = df_co_diario.loc[ANIO_INICIO:ANIO_FIN].dropna().copy()

df_co_monthly_cop = df_co_diario[VALUE_COL_CO].resample('ME').mean()
print(f" Colombia OK. Meses procesados: {len(df_co_monthly_cop)}.")

except Exception as e:
    print(f" Error al procesar datos de Colombia: {e}")
df_co_monthly_cop = pd.Series(dtype='float64')
df_co_diario = pd.DataFrame()
```

2. Procesando datos de Colombia  
Colombia OK. Meses procesados: 22.

3. Procesamiento de Datos de Noruega (EUR/kWh)

Esta sección se dedica a obtener y procesar los datos de precios spot de Noruega:

- **Descarga y Carga:** Realiza una petición `GET` a la `NORDPOOL_API` y carga los datos JSON en un `DataFrame`.
- **Limpieza y Conversión:** Convierte la columna `SpotPriceEUR` a tipo numérico y `HourUTC` a formato de fecha y hora, estableciéndola como índice del `DataFrame`.
- **Filtrado y Relleno de Nulos:** Filtra el `DataFrame` para incluir solo el período de análisis ( `ANIO_INICIO` a `ANIO_FIN` ) y rellena los valores faltantes en `SpotPriceEUR` con la media de la columna.
- **Cálculo Mensual:** Calcula el promedio mensual de `SpotPriceEUR` y lo almacena en `df_eu_monthly_eur`.

Ejecuta este bloque para procesar los datos de Noruega.

```
print("\n3. Procesando datos de Noruega")
try:
    response_eu = requests.get(NORDPOOL_API, timeout=30)
    response_eu.raise_for_status()

    df_eu = pd.DataFrame(response_eu.json())

    df_eu['SpotPriceEUR'] = pd.to_numeric(df_eu['SpotPriceEUR'].astype(str).str.replace(',', '.', regex=False), errors='coerce')
    df_eu['HourUTC'] = pd.to_datetime(df_eu['HourUTC'])
    df_eu = df_eu.set_index('HourUTC').sort_index()

    df_eu_filtrado = df_eu.loc[ANIO_INICIO:ANIO_FIN, 'SpotPriceEUR'].copy()
    df_eu_filtrado = df_eu_filtrado.fillna(df_eu_filtrado.mean())

    df_eu_monthly_eur = df_eu_filtrado.resample('ME').mean()

    print(f" Noruega OK. Meses procesados: {len(df_eu_monthly_eur)}.")

except Exception as e:
    print(f" Error al procesar datos de Noruega: {e}")
df_eu_monthly_eur = pd.Series(dtype='float64')
```

3. Procesando datos de Noruega  
Noruega OK. Meses procesados: 16.

4. Procesamiento de Datos de Alemania/Francia (EUR/kWh)

Este bloque de código se encarga de procesar los datos de precios de Alemania y Francia:

- **Descarga y Carga:** Obtiene los datos de la `ALEMANIA_FRANCIA_API` y los carga en un `DataFrame`.
- **Normalización de Columnas:** Ajusta automáticamente los nombres de las columnas para asegurar que se identifiquen correctamente las columnas de fecha y valor, incluso si tienen nombres ligeramente diferentes en la API.
- **Conversión de Tipos:** Convierte las columnas de fecha y valor a los tipos `datetime` y `float`.
- **Agregación Diaria e Interpolación:** Calcula el promedio diario de los precios, filtra por el período de análisis y rellena los valores faltantes mediante **interpolación lineal**, que es útil para estimar valores intermedios en una serie temporal.
- **Cálculo Mensual:** Obtiene el promedio mensual de los precios y lo almacena en `df_de_fr_monthly_eur`.

Ejecuta este bloque para procesar los datos de Alemania/Francia.

```
print("\n4. Procesando datos de Alemania/Francia")
try:
    response_de_fr = requests.get(ALEMANIA_FRANCIA_API, timeout=30)
    response_de_fr.raise_for_status()

    data_json_de_fr = response_de_fr.json()

    if isinstance(data_json_de_fr, dict) and data_json_de_fr:
        data_list_de_fr = data_json_de_fr.get(list(data_json_de_fr.keys())[0], [])
        df_de_fr_raw = pd.DataFrame(data_list_de_fr)
    else:
        df_de_fr_raw = pd.DataFrame(data_json_de_fr)

    if df_de_fr_raw.shape[1] == 2:
        df_de_fr_raw.columns = [DATE_COL_CO, VALUE_COL_CO]
    else:
        df_de_fr_raw.columns = df_de_fr_raw.columns.str.lower()
        if DATE_COL_CO not in df_de_fr_raw.columns:
            date_col_match = [col for col in df_de_fr_raw.columns if 'fecha' in col or 'date' in col]
            if date_col_match:
                df_de_fr_raw.rename(columns={date_col_match[0]: DATE_COL_CO}, inplace=True)
        if VALUE_COL_CO not in df_de_fr_raw.columns:
            value_col_match = [col for col in df_de_fr_raw.columns if 'valor' in col or 'precio' in col or 'price' in col]
            if value_col_match:
                df_de_fr_raw.rename(columns={value_col_match[0]: VALUE_COL_CO}, inplace=True)

    df_de_fr_raw[DATE_COL_CO] = pd.to_datetime(df_de_fr_raw[DATE_COL_CO], errors='coerce')
```

```
df_de_fr_raw[VALUE_COL_CO] = pd.to_numeric(
    df_de_fr_raw[VALUE_COL_CO].astype(str).str.replace(',', '.', regex=False), errors='coerce'
).astype(float)

df_de_fr_diario = df_de_fr_raw.dropna(subset=[DATE_COL_CO, VALUE_COL_CO]).set_index(DATE_COL_CO).sort_index()
df_de_fr_diario = df_de_fr_diario[[VALUE_COL_CO]].resample('D').mean()
df_de_fr_diario = df_de_fr_diario.loc[ANIO_INICIO:ANIO_FIN].copy()

df_de_fr_diario[VALUE_COL_CO] = df_de_fr_diario[VALUE_COL_CO].interpolate(method='linear')

df_de_fr_monthly_eur = df_de_fr_diario[VALUE_COL_CO].resample('ME').mean().dropna()

print(f"  Alemania/Francia OK. Meses procesados: {len(df_de_fr_monthly_eur)}.")

except Exception as e:
    print(f"  Error al procesar datos de Alemania/Francia: {e}")
    df_de_fr_monthly_eur = pd.Series(dtype='float64')
```

4. Procesando datos de Alemania/Francia  
Alemania/Francia OK. Meses procesados: 14.

5. Procesamiento de Datos del Nivel Agregado de Embalses (%)

Este bloque procesa los datos del nivel de los embalses en Colombia:

- **Descarga y Carga:** Realiza una petición `GET` a la `EMBALSES_API` y crea un `DataFrame` con la respuesta JSON.
- **Limpieza de Columnas:** Aplica la función `clean_col_name_strict` a todos los nombres de las columnas para estandarizarlos.
- **Conversión de Tipos:** Convierte las columnas `fecha`, `volumen_util_energia_gwh` y `capacidad_util_energia_gwh` a sus tipos de datos correctos (`datetime` y `numeric`).
- **Filtrado y Cálculo del Nivel:** Filtra los datos por el período de análisis, luego agrupa por día para sumar el volumen útil y la capacidad total de los embalses, calculando el `Nivel_Embalse_Porcentaje` diario.
- **Cálculo Mensual:** Finalmente, calcula el promedio mensual del nivel de embalses y lo almacena en `df_embalses_monthly_pct`.

Ejecuta este bloque para procesar los datos de los embalses.

```
print("\n5. Procesando datos del Nivel Agregado de Embalses")
try:
    response_emb = requests.get(EMBALSES_API, timeout=30)
    response_emb.raise_for_status()

    df_emb = pd.DataFrame(response_emb.json())
    df_emb.columns = [clean_col_name_strict(col) for col in df_emb.columns]

    date_col = clean_col_name_strict(DATE_EMBALSE_COL)
    volumen_col = clean_col_name_strict(VOLUME_COL_EXACT)
    capacidad_col = clean_col_name_strict(CAPACITY_COL_EXACT)

    df_emb[date_col] = pd.to_datetime(df_emb[date_col], errors='coerce')
    df_emb[volumen_col] = pd.to_numeric(df_emb[volumen_col], errors='coerce')
    df_emb[capacidad_col] = pd.to_numeric(df_emb[capacidad_col], errors='coerce')
    df_emb.dropna(subset=[date_col, volumen_col, capacidad_col], inplace=True)

    start_date = pd.to_datetime(ANIO_INICIO)
    end_date = pd.to_datetime(ANIO_FIN)

    df_filtrado_fechas = df_emb.loc[
        (df_emb[date_col].dt.normalize() >= start_date) &
        (df_emb[date_col].dt.normalize() <= end_date)
    ].copy()

    df_emb_diario = df_filtrado_fechas.groupby(df_filtrado_fechas[date_col].dt.normalize()).agg(
        Total_Volumen=(volumen_col, 'sum'),
        Total_Capacidad=(capacidad_col, 'sum')
    )

    df_emb_diario[VALUE_EMBALSE_COL] = (
        df_emb_diario['Total_Volumen'] / df_emb_diario['Total_Capacidad']
    ) * 100
    df_emb_diario.index.name = DATE_COL_CO

    df_embalses_monthly_pct = df_emb_diario[VALUE_EMBALSE_COL].resample('ME').mean()

    if len(df_embalses_monthly_pct) == 0:
        raise ValueError(f"No hay datos de embalses en el rango de fechas.")

    print(f"  Embalses OK. Meses procesados: {len(df_embalses_monthly_pct)}.")

except Exception as e:
    print(f"  Error al procesar datos de Embalses: {e}")
    df_embalses_monthly_pct = pd.Series(dtype='float64')
    df_emb_diario = pd.DataFrame()
```

5. Procesando datos del Nivel Agregado de Embalses  
Embalses OK. Meses procesados: 24.

6. Preparación y Entrenamiento del Modelo de Predicción (Machine Learning)

Este bloque de código prepara los datos y entrena un modelo de regresión lineal para predecir el precio spot de Colombia:

- **Fusión de Datos:** Combina el `DataFrame` de precios diarios de Colombia (`df_co_diario`) con el de niveles diarios de embalses (`df_emb_diario`).
- **Creación de Características (Feature Engineering):** Genera nuevas variables predictoras como el mes, el día de la semana, y los valores de precio y embalse del día anterior (`Precio_Ayer`, `Embalse_Ayer`). Estas "características de rezago" son cruciales para un modelo de series de tiempo.
- **División de Datos:** Separa el conjunto de datos en dos partes: un 80% para **entrenamiento** del modelo (`X_train`, `Y_train`) y un 20% para **prueba** (`X_test`, `Y_test`), para evaluar su rendimiento con datos no vistos.
- **Entrenamiento del Modelo:** Instancia y entrena un modelo de `LinearRegression` con los datos de entrenamiento.
- **Evaluación del Modelo:** Realiza predicciones sobre el conjunto de prueba (`Y_pred`) y calcula métricas clave como el `mean_squared_error` (MSE), `rmse` (Raíz del Error Cuadrático Medio) y `r2_score` (Coeficiente de Determinación R²), que indican la precisión y capacidad explicativa del modelo.

Ejecuta este bloque para preparar los datos y entrenar el modelo de Machine Learning.

```
print("\n6. Preparando Modelo de Predicción de Precios (ML)")

try:
    if df_co_diario.empty or df_emb_diario.empty:
        raise ValueError("Datos de Colombia o Embalses vacíos. No se puede entrenar el modelo.")

    df_ml = df_co_diario.rename(columns={VALUE_COL_CO: 'Precio_COP'}).copy()
    df_ml = df_ml.merge(
        df_emb_diario[[VALUE_EMBALSE_COL]],
        left_index=True,
        right_index=True,
        how='inner'
    )

    df_ml['Mes'] = df_ml.index.month
    df_ml['Dia_Semana'] = df_ml.index.dayofweek
    df_ml['Precio_Ayer'] = df_ml['Precio_COP'].shift(1)
    df_ml['Embalse_Ayer'] = df_ml[VALUE_EMBALSE_COL].shift(1)
    df_ml.dropna(inplace=True)

    if df_ml.empty:
        raise ValueError("El dataset ML está vacío después de crear features.")

    X = df_ml[['Precio_Ayer', 'Embalse_Ayer', 'Mes', 'Dia_Semana']]
    Y = df_ml['Precio_COP']

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, shuffle=False)

    model = LinearRegression()
    model.fit(X_train, Y_train)

    Y_pred = model.predict(X_test)
    mse = mean_squared_error(Y_test, Y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(Y_test, Y_pred)

    print(f"\n Resultados de la Regresión Lineal:")
    print(f"  RMSE: {rmse:.2f} COP/kWh | R^2: {r2:.4f}")

except Exception as e:
    print(f"  Error al ejecutar el Modelo de Predicción: {e}")
    Y_test = pd.Series(dtype='float64')
    Y_pred = np.array([])
    rmse = np.nan
    r2 = np.nan
```

6. Preparando Modelo de Predicción de Precios (ML)

Resultados de la Regresión Lineal:  
RMSE: 9.41 COP/kWh | R^2: 0.4924

7. Ajuste de Datos para Gráficas y Consolidación de Series

Este bloque de código se encarga de preparar y ajustar los datos finales antes de la visualización:

- **Conversión de Moneda:** Convierte el `df_co_monthly_cop` a euros ( `df_co_monthly_eur` ) para permitir una comparación directa con los datos europeos.
- **Identificación de Fechas Comunes:** Busca las fechas que están presentes en **todas** las series mensuales de datos (Colombia, Noruega, Alemania/Francia y Embalses). Esto es fundamental para asegurar que todos los gráficos se muestren en un período consistente.
- **Ajuste de Series:** Todas las series mensuales se ajustan para incluir solo las `fechas_comunes` identificadas.
- **Creación de DataFrames para Gráficas:** Se construyen dos DataFrames específicos:
  - `df_comparision` : Contiene los precios mensuales de Noruega, Alemania/Francia y Colombia (convertido a EUR) para la gráfica comparativa.
  - `df_relacion` : Incluye el precio spot de Colombia en COP y el nivel de embalses en porcentaje, para analizar la relación entre ellos.
- **Cálculo de Correlación:** Calcula el coeficiente de correlación de Pearson entre el `Precio_Spot_COP` y el `Nivel_Embalse_Pct` para cuantificar la relación entre estas dos variables.

Ejecuta este bloque para preparar los datos para las gráficas.

```
df_co_monthly_eur = df_co_monthly_cop / TASA_CAMBIO_COP_POR_EUR

if df_co_monthly_cop.empty or df_eu_monthly_eur.empty or df_embalses_monthly_pct.empty or df_de_fr_monthly_eur.empty:
    print("\nError: No se puede graficar. Una o más series están vacías.")
    fechas_comunes = pd.Index([])
else:
    print("\n7. Consolidando series en fechas comunes")

    fechas_comunes = df_co_monthly_eur.index.intersection(df_eu_monthly_eur.index).intersection(df_de_fr_monthly_eur.index).intersection(df_embalses_monthly_pct.index)

    if len(fechas_comunes) == 0:
        print(f"No se encontraron fechas comunes entre las series.")
    else:
        df_co_monthly_eur_ajustado = df_co_monthly_eur.loc[fechas_comunes]
        df_eu_monthly_eur_ajustado = df_eu_monthly_eur.loc[fechas_comunes]
        df_de_fr_monthly_eur_ajustado = df_de_fr_monthly_eur.loc[fechas_comunes]
        df_co_monthly_cop_ajustado = df_co_monthly_cop.loc[fechas_comunes]
        df_embalses_monthly_pct_ajustado = df_embalses_monthly_pct.loc[fechas_comunes]

if len(fechas_comunes) > 0:

    df_comparision = pd.DataFrame({
        'Noruega (Nordpool)': df_eu_monthly_eur_ajustado,
        'Alemania/Francia': df_de_fr_monthly_eur_ajustado,
        'Colombia (Convertido)': df_co_monthly_eur_ajustado
    }).dropna()

    df_relacion = pd.DataFrame({
```

```
'Precio_Spot_COP': df_co_monthly_cop_ajustado,
'Nivel_Embalse_Pct': df_embalses_monthly_pct_ajustado
}).dropna()

correlacion = df_relacion['Precio_Spot_COP'].corr(df_relacion['Nivel_Embalse_Pct'])
print(f"\nCorrelación Precio Spot vs. Nivel Embalse: {correlacion:.4f}")
```

7. Consolidando series en fechas comunes

Correlación Precio Spot vs. Nivel Embalse: -0.7037

8. Generación de Gráficas

Este último bloque de código genera y muestra las seis gráficas que resumen el análisis:

- **Gráfica 1: Precio Spot Promedio Mensual de Colombia (COP/kWh):** Muestra la evolución del precio de la energía en Colombia.
- **Gráfica 2: Precio Spot Promedio Mensual de Noruega (Nordpool) (EUR/kWh):** Ilustra la tendencia del precio en el mercado Nordpool.
- **Gráfica 3: Precio Spot Promedio Mensual de Alemania/Francia (EUR/kWh):** Muestra el comportamiento del precio en la región franco-alemana.
- **Gráfica 4: Comparación del Precio Spot Promedio Mensual Multi-País (EUR/kWh):** Compara visualmente las tendencias de precios de los tres mercados (Colombia, Noruega y Alemania/Francia) en una misma unidad.
- **Gráfica 5: Predicción del Precio Spot de Colombia (Regresión Lineal):** Compara los valores reales del precio spot de Colombia con las predicciones del modelo de Machine Learning, incluyendo las métricas RMSE y R².
- **Gráfica 6: Relación entre Precio Spot y Nivel de Embalses en Colombia:** Utiliza un doble eje Y para superponer la evolución del precio spot en Colombia y el nivel agregado de los embalses, destacando su correlación.

Ejecuta este bloque para generar todas las visualizaciones.

```
print("\n8. Generando 6 Gráficas en el orden solicitado")

if len(fechas_comunes) > 0:

    # Gráfica 1: Precio Spot Mensual de Colombia (COP/kWh)
    plt.figure(figsize=(12, 5))
    plt.plot(df_co_monthly_cop_ajustado.index, df_co_monthly_cop_ajustado, label='Colombia (COP/kWh)', marker='^', linestyle='--', color='red')
    plt.title('Gráfica 1: Precio Spot Promedio Mensual de Colombia')
    plt.suptitle(f'Unidad: COP/kWh', fontsize=10, y=0.92)
    plt.xlabel('Fecha')
    plt.ylabel('Precio Promedio Mensual (COP/kWh)')
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.xticks(rotation=45)
    plt.legend()
    plt.tight_layout(rect=[0, 0, 1, 0.95])
    plt.show()

    # Gráfica 2: Precio Spot Mensual de Noruega (EUR/kWh)
    plt.figure(figsize=(12, 5))
    plt.plot(df_eu_monthly_eur_ajustado.index, df_eu_monthly_eur_ajustado.values,
             label='Noruega (EUR/kWh)', marker='o', linestyle='-', color='blue')
    plt.title('Gráfica 2: Precio Spot Promedio Mensual de Noruega (Nordpool)')
    plt.suptitle(f'Unidad: EUR/kWh', fontsize=10, y=0.92)
    plt.xlabel('Fecha')
    plt.ylabel('Precio Promedio Mensual (EUR/kWh)')
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.xticks(rotation=45)
    plt.legend()
    plt.tight_layout(rect=[0, 0, 1, 0.95])
    plt.show()

    # Gráfica 3: Precio Spot Mensual de Alemania/Francia (EUR/kWh)
    plt.figure(figsize=(12, 5))
    plt.plot(df_de_fr_monthly_eur_ajustado.index, df_de_fr_monthly_eur_ajustado.values,
             label='Alemania/Francia (EUR/kWh)', marker='s', linestyle='-', color='purple')
    plt.title('Gráfica 3: Precio Spot Promedio Mensual de Alemania/Francia')
    plt.suptitle(f'Unidad: EUR/kWh', fontsize=10, y=0.92)
    plt.xlabel('Fecha')
    plt.ylabel('Precio Promedio Mensual (EUR/kWh)')
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.xticks(rotation=45)
    plt.legend()
    plt.tight_layout(rect=[0, 0, 1, 0.95])
    plt.show()

    # Gráfica 4: Comparación Multi-País (EUR/kWh)
    plt.figure(figsize=(15, 6))

    plt.plot(df_comparacion.index, df_comparacion['Noruega (Nordpool)'], label='Noruega (EUR/kWh)', marker='o', linestyle='-', color='blue')
    plt.plot(df_comparacion.index, df_comparacion['Alemania/Francia'], label='Alemania/Francia (EUR/kWh)', marker='s', linestyle='-', color='purple')
    plt.plot(df_comparacion.index, df_comparacion['Colombia (Convertido)'], label='Colombia (EUR/kWh)', marker='^', linestyle='--', color='red')

    plt.title(f'Gráfica 4: Comparación del Precio Spot Promedio Mensual Multi-País')
    plt.suptitle(f'Unidad: EUR/kWh (Tasa de conversión: 1 EUR = {TASA_CAMBIO_COP_POR_EUR} COP)', fontsize=10, y=0.92)
    plt.xlabel('Fecha')
    plt.ylabel('Precio Promedio Mensual (EUR/kWh)')
    plt.legend(title='País')
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.xticks(rotation=45)
    plt.tight_layout(rect=[0, 0, 1, 0.95])
    plt.show()

    if not Y_test.empty and not df_relacion.empty:

        # Gráfica 5: Predicción ML (COP/kWh)
        plt.figure(figsize=(15, 6))
        plt.plot(Y_test.index, Y_test, label='Real (Test)', color='green', linewidth=2)
        plt.plot(Y_test.index, Y_pred, label='Predicción (Modelo)', color='orange', linestyle='--')
        plt.title('Gráfica 5: Predicción del Precio Spot de Colombia (Regresión Lineal)')
        plt.suptitle(f'RMSE: {rmse:.2f} COP/kWh | R^2: {r2:.4f}', fontsize=10, y=0.92)
        plt.xlabel('Fecha')
        plt.ylabel('Precio (COP/kWh)')
        plt.legend()
        plt.grid(True, linestyle=':', alpha=0.6)
        plt.xticks(rotation=45)
        plt.tight_layout(rect=[0, 0, 1, 0.95])
```



```
plt.show()

# Gráfica 6: Relación Precio vs. Embalses (Doble Eje Y)
fig, ax1 = plt.subplots(figsize=(15, 6))

color = 'red'
ax1.set_xlabel('Fecha')
ax1.set_ylabel('Precio Spot Promedio (COP/kWh)', color=color)
ax1.plot(df_relacion.index, df_relacion['Precio_Spot_COP'], color=color, label='Precio Spot (COP/kWh)', marker='^', linestyle='--')
ax1.tick_params(axis='y', labelcolor=color)
ax1.grid(True, linestyle='--', alpha=0.5)

ax2 = ax1.twinx()
color = 'blue'
ax2.set_ylabel('Nivel Agregado de Embalses (%)', color=color)
ax2.plot(df_relacion.index, df_relacion['Nivel_Embalse_Pct'], color=color, label='Nivel Embalses (%)', marker='o', linestyle='-')
ax2.tick_params(axis='y', labelcolor=color)

plt.title('Gráfica 6: Relación entre Precio Spot y Nivel de Embalses en Colombia')
plt.suptitle(f'Correlación de Pearson: {correlacion:.4f}', fontsize=10, y=0.92)
fig.legend(loc="upper left", bbox_to_anchor=(0.1, 0.9))
plt.xticks(rotation=45)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()

else:
    print("\nADVERTENCIA: Las gráficas de Predicción ML y Relación Embalses no se pudieron generar por falta de datos o errores en el modelo.")
```

8. Generando 6 Gráficas en el orden solicitado

