# N04. Proyección de precios de energía en bolsa mensuales

Una de las variables que explican el comportamiento de los precios de energía en bolsa en Colombia, son los 'Aportes de energía hidráulica convertidos a energía'.

Por ello, se requiere proyectar esta variable para los próximos 78 meses.

## 4.1. Proyección de los 'Aportes de energía hidráulica en energía'

Para proyectar los 'Aportes de energía hidráulica convertidos a energía' se seguirá la siguiente metodología:

* Se encontrará la relación entre 'Aportes de energía hidráulica (energía)' y 'Aportes de energía hidráulica (caudal)'.
* Se contrará la relación y la 'Diferencia de aportes' entre los 'Aportes de energía hidráulica históricos' y 'Aportes de energía hidráulica real'.
* Se encontrará la relación entre el MEI (Índice multivariado El Niño) y la 'Diferencia de aportes'.

Se crearán inicialmente dos DataFrame: 'aportes' y 'capacidad'.

### 4.1.1. Carga paquetes para la proyección

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.1.1. Carga paquetes requeridos para leer en base de datos y proyección.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
import numpy as np  
import pandas as pd  
import datetime as dt  
import warnings  
from statsmodels.tools.sm\_exceptions import ConvergenceWarning  
warnings.filterwarnings("ignore", category=FutureWarning)  
warnings.filterwarnings("ignore", category=ConvergenceWarning)  
warnings.filterwarnings("ignore", category=UserWarning)  
from sqlalchemy import create\_engine  
from sqlalchemy.orm import sessionmaker  
  
import matplotlib.pyplot as plt  
%matplotlib widget  
  
import statsmodels.api as sm  
import statsmodels.formula.api as smf

## 4.2. Variables de Aportes de caudales y de energía de caudales

XM publica las siguientes variables relacionadas con los 'aportes', registradas de manera diaria, las cuales se deben transformar en variables mensuales:

* AporEner
* AporEnerMediHist
* AporCaudalRio
* AporCaudalMediHistRio
* AporCaudalRio (m3/s). Valores de la hidrologia de los caudales de los rios del SIN, en metros cubicos por segundo.
* AporCaudalMediHistRio (m3/s). Caudal medio mensual histórico para los rios del SIN, obtenido como el promedio de los valores de cada mes para todos años con información disponibles.
* AporEner (kWh). Caudales en energia de los rios que aportan agua a algun embalse del SIN (Sistema).
* AporEnerRio (kWh). Caudales en energia de los rios que aportan agua a algun embalse del SIN (Río).
* AporEnerMediHist Aportes Media Histórica Energía Sistema
* AporEnerMediHist (kWh). Caudal medio mensual histórico en energia para los rios del SIN, obtenido como el promedio de los valores de cada mes para todos años con información disponibles (Sistema).
* AporCaudalMediHistRio (kWh). Caudal medio mensual histórico en energia para los rios del SIN, obtenido como el promedio de los valores de cada mes para todos años con información disponibles (Río).

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.2.1. Crea la máquina para conectarse a la base de datos dbXm.db.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
engine = create\_engine('sqlite:///dbXm.db', echo=False)  
Session = sessionmaker(bind=engine)  
session = Session()

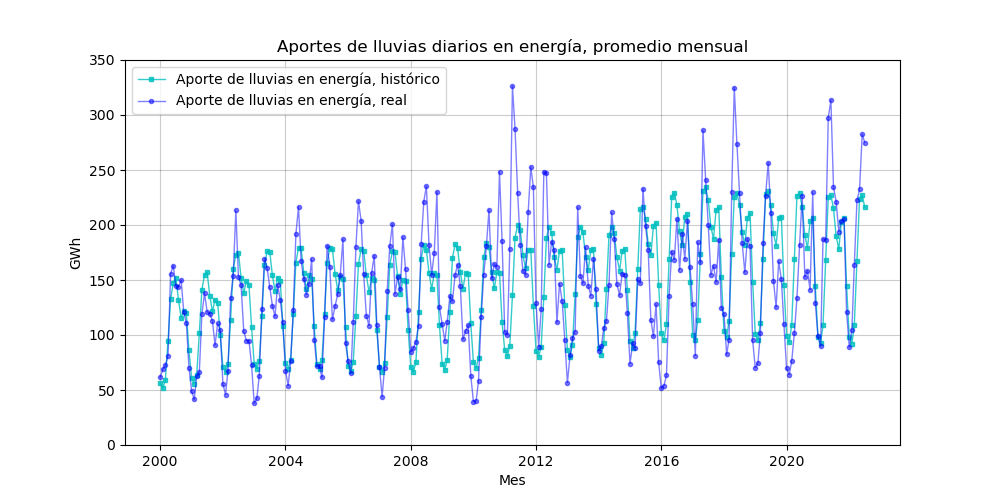
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.2.2. Lee las variables de 'aportes' y ajusta.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
tablas = ['AporEner', 'AporEnerMediHist', \  
 'AporEnerRio', 'AporEnerMediHistRio', \  
 'AporCaudalRio', 'AporCaudalMediHistRio']  
nombresDf = ['aporEner', 'aporEnerMediHist', \  
 'aporEnerRio', 'aporEnerMediHistRio', \  
 'aporCaudalRio', 'aporCaudalMediHistRio']  
df = {}  
  
for i, tabla in enumerate(tablas):  
 consulta = "SELECT \* FROM " + tabla  
 df[nombresDf[i]] = pd.read\_sql(consulta, engine, parse\_dates=["Date"])  
#Ajusta las columnas, los índices y los nombres de los DataFrames.  
for i, nombre in enumerate(nombresDf):  
 df[nombre] = df[nombre].iloc[:, 1:] if i <= 1 else df[nombre].iloc[:, [1, 3]]  
 df[nombre] = df[nombre].set\_index('Date')  
 df[nombre].columns = [nombre]  
 df[nombre + '\_m'] = df[nombre] #Crea los contenedores DataFrame mensuales.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.2.3. Agrupa los valores para obtener los valores de mensuales.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
for i, nombre in enumerate(nombresDf):  
 if i <= 1: #Diarios, por Sistema.  
 df[nombre + '\_m'] = df[nombre + '\_m']\  
 .groupby(pd.to\_datetime(df[nombre + '\_m'].index)\  
 .to\_period('M')).mean()  
 else: #Diarios, por ríos.  
 df[nombre + '\_m'] = df[nombre + '\_m']\  
 .groupby(pd.to\_datetime(df[nombre + '\_m'].index).\  
 to\_period('D')).sum()  
 df[nombre + '\_m'] = df[nombre + '\_m']\  
 .groupby((df[nombre + '\_m'].index)\  
 .to\_timestamp().to\_period('M')).mean()  
 #  
 df[nombre + '\_m'].index = df[nombre + '\_m'].index.to\_timestamp()

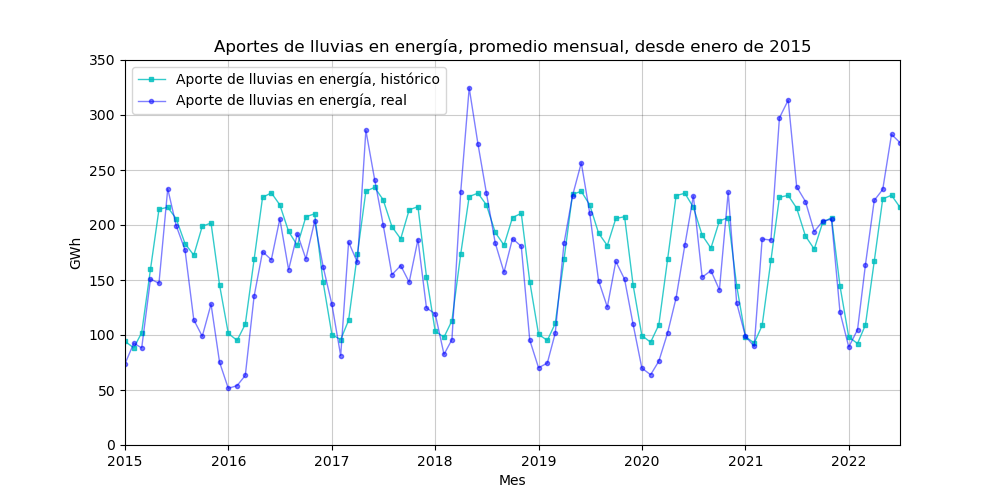
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.2.4. Integra las tablas de base de datos en un DataFrame aportes\_m.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
aportes\_m = df['aporEner\_m']/1e6  
nombresDf\_m = ['aporEnerMediHist\_m', \  
 'aporEnerRio\_m', 'aporEnerMediHistRio\_m', \  
 'aporCaudalRio\_m', 'aporCaudalMediHistRio\_m']  
for i, nombre in enumerate(nombresDf\_m):  
 if i == 0:  
 aportes\_m = pd.merge(aportes\_m, df[nombre]/1e6, left\_on="Date", right\_on="Date")  
 else:  
 aportes\_m = pd.merge(aportes\_m, df[nombre], left\_on="Date", right\_on="Date")  
  
aportes\_m = aportes\_m.drop(columns=['aporEnerRio', 'aporEnerMediHistRio'])  
aportes\_m.insert(loc=len(aportes\_m.columns), column="diferenciaAportes", \  
 value=(aportes\_m['aporEner'] - aportes\_m['aporEnerMediHist']).values, \  
 allow\_duplicates=True)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.2.5. Salva y lee aportes\_m.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva aportes\_m  
# aportes\_m.to\_parquet('parquet/aportes\_m.parquet.gzip', compression='gzip', \  
# engine='fastparquet')  
# #  
# aportes\_m.to\_excel('xlsx/aportes\_m.xlsx', sheet\_name='hoja\_1')  
#Lee aportes\_m  
aportes\_m = pd.read\_parquet('parquet/aportes\_m.parquet.gzip', \  
 engine='fastparquet')

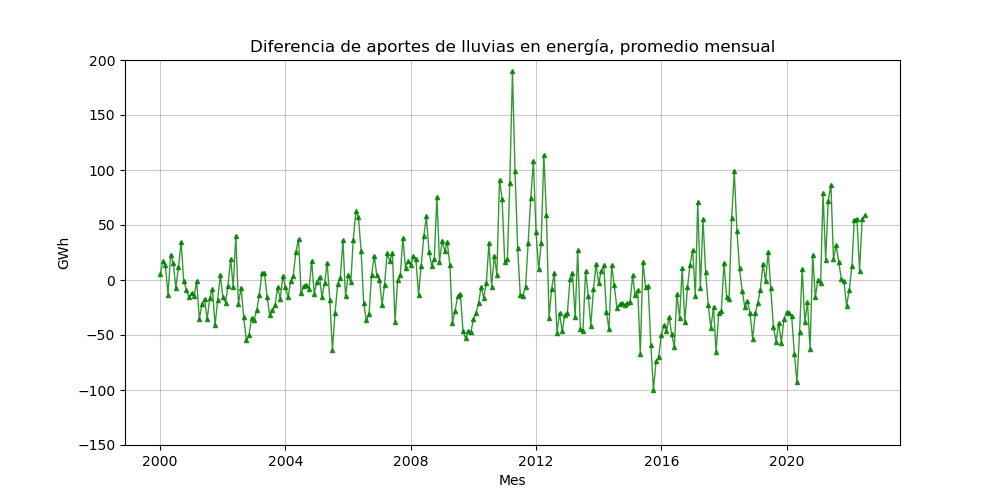
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.2.6. Grafica aportes\_m.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
ax1.set\_title ('Aportes de lluvias diarios en energía, promedio mensual')  
ax1.set\_ylabel ('GWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(aportes\_m.index, aportes\_m.aporEnerMediHist, 'c-s', \  
 label='Aporte de lluvias en energía, histórico', linewidth=1, markersize=3, alpha=0.8)  
ax1.plot(aportes\_m.index, aportes\_m.aporEner, 'b-o', \  
 label='Aporte de lluvias en energía, real', linewidth=1, markersize=3, alpha=0.5)  
# ax1.plot(meiT.index, meiT.meiP1988\_08, 'r-', label='Escenario alto', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP2010\_07, 'y-', label='Escenario Medio', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP1998\_08, 'g-', label='Escenario Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-06-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
plt.legend(loc='best')  
#plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 350)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



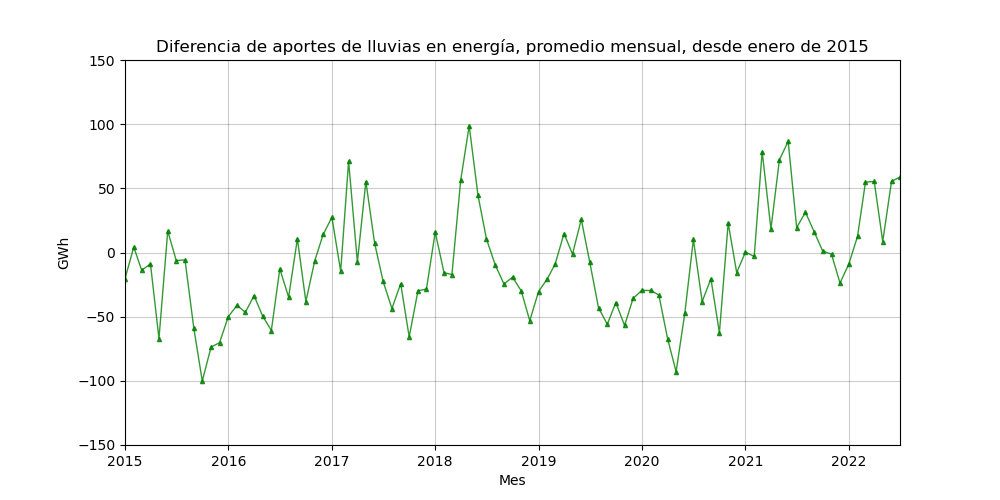
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.2.7. Grafica aportes\_m desde 2015-01.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
ax1.set\_title ('Aportes de lluvias en energía, promedio mensual, desde enero de 2015')  
ax1.set\_ylabel ('GWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(aportes\_m.index, aportes\_m.aporEnerMediHist, 'c-s', \  
 label='Aporte de lluvias en energía, histórico', linewidth=1, markersize=3, alpha=0.8)  
ax1.plot(aportes\_m.index, aportes\_m.aporEner, 'b-o', \  
 label='Aporte de lluvias en energía, real', linewidth=1, markersize=3, alpha=0.5)  
# ax1.plot(meiT.index, meiT.meiP1988\_08, 'r-', label='Escenario alto', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP2010\_07, 'y-', label='Escenario Medio', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP1998\_08, 'g-', label='Escenario Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-06-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
plt.legend(loc='best')  
plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 350)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



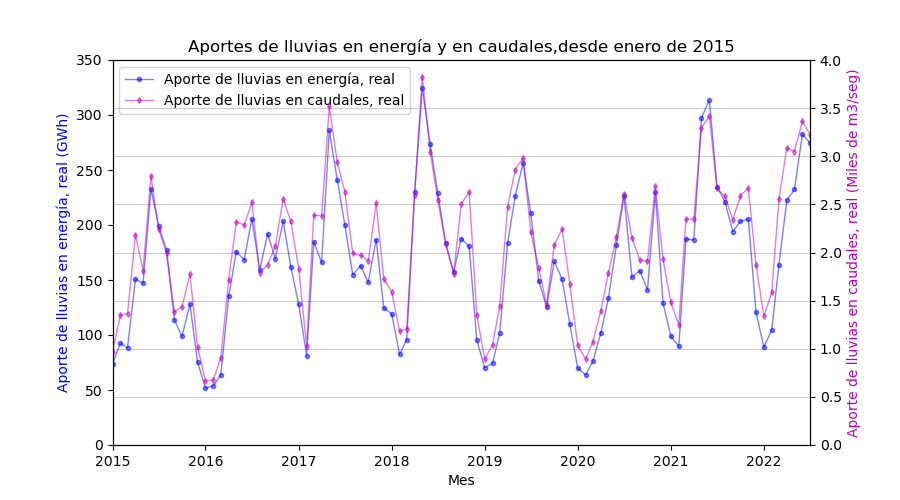
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.2.8. Grafica diferencia de aportes de lluvia, promedio mensual.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
ax1.set\_title ('Diferencia de aportes de lluvias en energía, promedio mensual')  
ax1.set\_ylabel ('GWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(aportes\_m.index, aportes\_m.diferenciaAportes, 'g-^', \  
 label='Diferencia de aporte de lluvias en energía', linewidth=1, markersize=3, alpha=0.8)  
# ax1.plot(meiT.index, meiT.meiP1988\_08, 'r-', label='Escenario alto', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP2010\_07, 'y-', label='Escenario Medio', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP1998\_08, 'g-', label='Escenario Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-06-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
# plt.legend(loc='best')  
#plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(-150, 200)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.2.9. Grafica diferencia de aportes de lluvia, desde enero de 2015.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
ax1.set\_title ('Diferencia de aportes de lluvias en energía, promedio mensual, desde enero de 2015')  
ax1.set\_ylabel ('GWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(aportes\_m.index, aportes\_m.diferenciaAportes, 'g-^', \  
 label='Diferencia de aporte de lluvias en energía', linewidth=1, markersize=3, alpha=0.8)  
# ax1.plot(meiT.index, meiT.meiP1988\_08, 'r-', label='Escenario alto', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP2010\_07, 'y-', label='Escenario Medio', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP1998\_08, 'g-', label='Escenario Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-06-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
# plt.legend(loc='best')  
plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(-150, 150)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.2.10. Grafica aporEner Vs aporCaudalRio desde 2015-01.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(9,5))  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
ax2 = ax1.twinx()  
  
ax1.set\_title ('Aportes de lluvias en energía y en caudales,desde enero de 2015')  
  
ax1.set\_xlabel("Mes")  
ax1.set\_ylabel("Aporte de lluvias en energía, real (GWh)", color='b')  
ax2.set\_ylabel("Aporte de lluvias en caudales, real (Miles de m3/seg)", color='m')  
  
linea1 = ax1.plot(aportes\_m.index, aportes\_m.aporEner, 'b-o', \  
 label="Aporte de lluvias en energía, real", \  
 linewidth=1, markersize=3, alpha=0.5)  
  
linea2 = ax2.plot(aportes\_m.index, aportes\_m.aporCaudalRio/1e3, 'm-d', \  
 label="Aporte de lluvias en caudales, real", linewidth=1, \  
 markersize=3, alpha=0.5)  
  
  
# added these three lines  
lineas = linea1 + linea2  
etiquetas = [l.get\_label() for l in lineas]  
ax1.legend(lineas, etiquetas, loc='best')  
  
plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2022-07-01'))  
#plt.ylim(0, 400)  
#plt.par.ylim(0, 400)  
ax1.set\_ylim(0, 350)  
ax2.set\_ylim(0, 4)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()

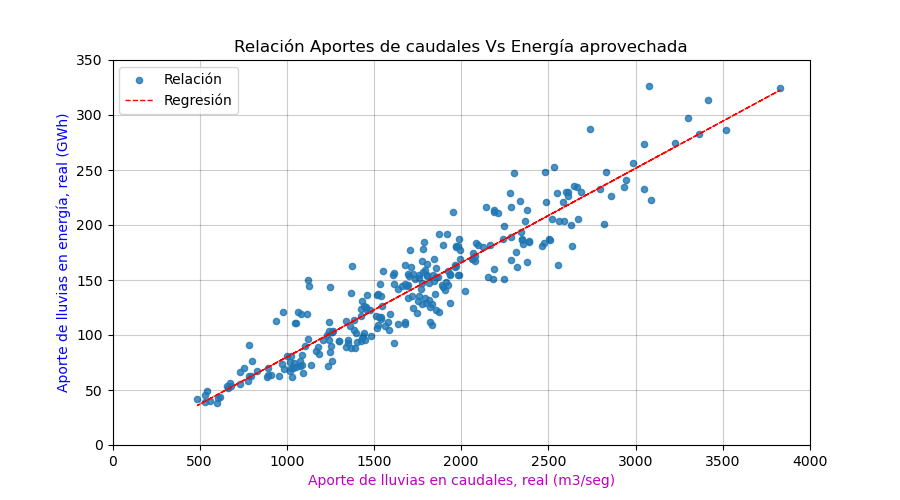


#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.2.11. Calcula el coeficiente de correlación entre aporCaudalRio Vs aporEner.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
import scipy.stats  
# aportes\_m.aporCaudalRio.corr(aportes\_m.aporEner) #0.9375  
r, p = scipy.stats.pearsonr(aportes\_m.aporCaudalRio, aportes\_m.aporEner)  
r, p # r (correlación de Pearson): 0.9375; valor p: 0.  
#(0.9375748320891986, 2.1212045210040593e-125)

(0.9375748320891986, 2.1212045210040593e-125)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.2.12. Calcula la regresión entre la dispersión de aporCaudalRio Vs aporEner.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
m, b = np.polyfit(aportes\_m.aporCaudalRio, aportes\_m.aporEner, 1)  
aportes\_m.insert(5, 'aporEner\_hat', aportes\_m.aporCaudalRio\*m + b, allow\_duplicates=True)   
# aportes\_m  
# m, b  
# m = 0.08577574033953389  
# b = -5.7997463853853946  
# m \* X + b

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.2.13. Grafica dispersión de aporCaudalRio Vs aporEner con la regresión.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(9,5))  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, \  
 hspace=None)  
  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
  
ax1.set\_title ('Relación Aportes de caudales Vs Energía aprovechada')  
  
# ax1.set\_xlabel("Mes")  
# ax1.set\_ylabel("Aporte de lluvias en energía, real (GWh)", color='b')  
# ax2.set\_ylabel("Aporte de lluvias en caudales, real (m3/seg)", color='m')  
  
plt.scatter(aportes\_m.aporCaudalRio, aportes\_m.aporEner, label='Relación', \  
 alpha=0.8, s=20, marker="o")  
ax1.plot(aportes\_m.aporCaudalRio, aportes\_m.aporEner\_hat, 'r--', \  
 label='Regresión', linewidth=1)  
plt.xlabel("Aporte de lluvias en caudales, real (m3/seg)", color='m')  
plt.ylabel("Aporte de lluvias en energía, real (GWh)", color='b')  
# plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 350)  
plt.xlim(0, 4000)  
# plt.par.ylim(0, 400)  
#plt.set\_ylim(0, 350)  
# ax2.set\_ylim(0, 4500)  
plt.legend(loc='best')  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



La relación es: aporEner = (aporCaudalRio \* 0.0857) - 5.799

## 4.3. Relación entre el MEI y los 'Aportes de caudales de lluvia reales'

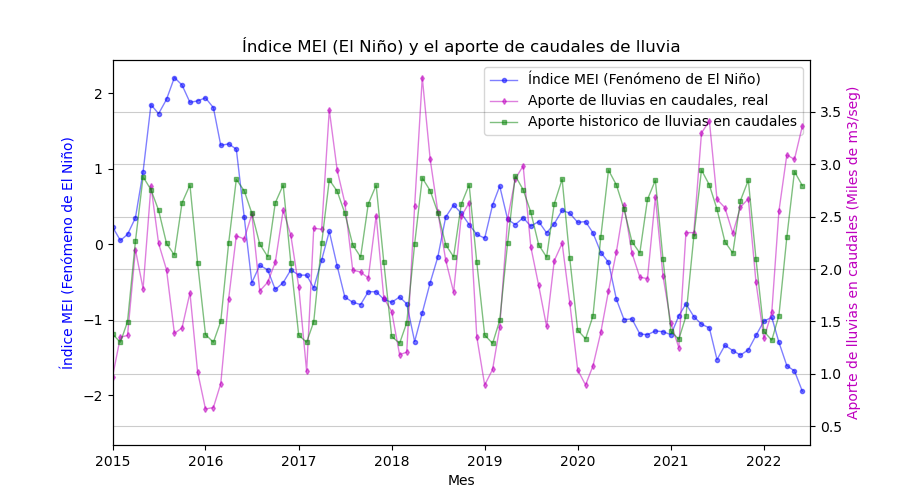
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.3.1. Lee el meiT y aportes\_m.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
meiT = pd.read\_parquet('parquet/meiT.parquet.gzip', engine='fastparquet')  
aportes\_m = pd.read\_parquet('parquet/aportes\_m.parquet.gzip', \  
 engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.3.2. Acondiciona meiT para poder comparar relacionar con aportes\_m.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
mei\_real = meiT.loc[meiT['mei'].notna(), 'mei']  
mei\_real.index.name = 'Date'  
caudal\_mei = pd.merge(aportes\_m.aporCaudalRio/1e3, mei\_real, left\_on="Date", right\_on="Date")  
caudal\_mei = pd.merge(caudal\_mei, aportes\_m.aporCaudalMediHistRio/1e3, left\_on="Date", right\_on="Date")  
# caudal\_mei

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.3.3. Corrige valores atípicos de aporCaudalMediHistRio.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# caudal\_mei.loc[pd.to\_datetime('2013-05-01'), 'aporCaudalMediHistRio'] #3.41  
# caudal\_mei.loc[pd.to\_datetime('2015-10-01'), 'aporCaudalMediHistRio'] #5.20  
# caudal\_mei.loc[pd.to\_datetime('2015-11-01'), 'aporCaudalMediHistRio'] #5.29  
# caudal\_mei.loc[pd.to\_datetime('2015-12-01'), 'aporCaudalMediHistRio'] #4.07  
#  
# caudal\_mei.loc[pd.to\_datetime('2014-05-01'), 'aporCaudalMediHistRio'] #2.13  
# caudal\_mei.loc[pd.to\_datetime('2016-10-01'), 'aporCaudalMediHistRio'] #2.63  
# caudal\_mei.loc[pd.to\_datetime('2016-11-01'), 'aporCaudalMediHistRio'] #2.80  
# caudal\_mei.loc[pd.to\_datetime('2016-12-01'), 'aporCaudalMediHistRio'] #2.05  
#  
caudal\_mei\_c = caudal\_mei.copy()  
caudal\_mei\_c.loc['2013-05-01', 'aporCaudalMediHistRio'] = \  
 caudal\_mei\_c.loc['2014-05-01', 'aporCaudalMediHistRio']  
caudal\_mei\_c.loc['2015-10-01', 'aporCaudalMediHistRio'] = \  
 caudal\_mei\_c.loc['2016-10-01', 'aporCaudalMediHistRio']  
caudal\_mei\_c.loc['2015-11-01', 'aporCaudalMediHistRio'] = \  
 caudal\_mei\_c.loc['2016-11-01', 'aporCaudalMediHistRio']  
caudal\_mei\_c.loc['2015-12-01', 'aporCaudalMediHistRio'] = \  
 caudal\_mei\_c.loc['2016-12-01', 'aporCaudalMediHistRio']  
caudal\_mei = caudal\_mei\_c

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.3.4. Calcula mei1 (un retardo en el mei).  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
caudal\_mei.insert(loc=len(caudal\_mei.columns), column='mei1', value=caudal\_mei.mei.shift(), allow\_duplicates=True)  
caudal\_mei = caudal\_mei.dropna()

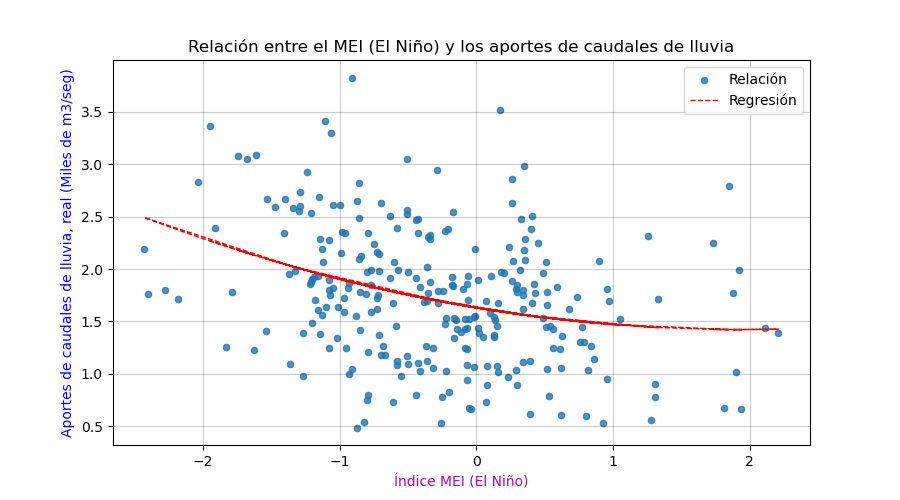
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.3.5. Grafica mei, aportCaudalRio y aporCaudalMediHistRio.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(9,5))  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
ax2 = ax1.twinx()  
  
ax1.set\_title ("Índice MEI (El Niño) y el aporte de caudales de lluvia")  
  
ax1.set\_xlabel("Mes")  
ax1.set\_ylabel("Índice MEI (Fenómeno de El Niño)", color='b')  
ax2.set\_ylabel("Aporte de lluvias en caudales (Miles de m3/seg)", color='m')  
  
linea1 = ax1.plot(caudal\_mei.index, caudal\_mei.mei, 'b-o', \  
 label="Índice MEI (Fenómeno de El Niño)", \  
 linewidth=1, markersize=3, alpha=0.5)  
  
linea2 = ax2.plot(caudal\_mei.index, caudal\_mei.aporCaudalRio, 'm-d', \  
 label="Aporte de lluvias en caudales, real", linewidth=1, \  
 markersize=3, alpha=0.5)  
  
linea3 = ax2.plot(caudal\_mei.index, caudal\_mei.aporCaudalMediHistRio, 'g-s', \  
 label="Aporte historico de lluvias en caudales", linewidth=1, \  
 markersize=3, alpha=0.5)  
  
# added these three lines  
lineas = linea1 + linea2 + linea3  
etiquetas = [l.get\_label() for l in lineas]  
ax1.legend(lineas, etiquetas, loc='best')  
  
# plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2022-06-01'))  
plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2022-07-01'))  
#plt.ylim(0, 400)  
#plt.par.ylim(0, 400)  
# ax1.set\_ylim(0, 350)  
# ax2.set\_ylim(0, 4500)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.3.6. Calcula coeficiente de correlación entre mei y aporCaudalRio.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
import scipy.stats  
r, p = scipy.stats.pearsonr(caudal\_mei.mei, caudal\_mei.aporCaudalRio)  
#r, p # r (correlación de Pearson): -0.3067; valor p: 0.  
#mei: (-0.31433351138495325, 1.3979868319505354e-07)  
#mei1: (-0.32745039954951416, 4.075025773408061e-08)  
#mei2: (-0.30851561209096595, 2.559915197528141e-07)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.3.7. Calcula la regresión entre caudal\_mei.mei Vs caudal\_mei.aporCaudalRio.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
a, b, c = np.polyfit(caudal\_mei.mei, caudal\_mei.aporCaudalRio, 2) #grado 2.  
caudal\_mei.insert(4, 'aporCaudalRio\_hat', \  
 a \* np.square(caudal\_mei.mei) + b \* caudal\_mei.mei + c, allow\_duplicates=True)   
# caudal\_mei  
# a, b, c  
# (0.056633704635569736, -0.21706142481276552, 1.6296385955059782)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.3.8. Gráfica de dispersión de mei Vs aporCaudalRio, con la regresión.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(9,5))  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
  
ax1.set\_title ('Relación entre el MEI (El Niño) y los aportes de caudales de lluvia')  
  
# ax1.set\_xlabel("Mes")  
# ax1.set\_ylabel("Aporte de lluvias en energía, real (GWh)", color='b')  
# ax2.set\_ylabel("Aporte de lluvias en caudales, real (m3/seg)", color='m')  
  
plt.scatter(caudal\_mei.mei, caudal\_mei.aporCaudalRio, label='Relación', \  
 alpha=0.8, s=20, marker="o")  
ax1.plot(caudal\_mei.mei, caudal\_mei.aporCaudalRio\_hat, 'r--', \  
 label='Regresión', linewidth=1)  
plt.xlabel("Índice MEI (El Niño)", color='m')  
plt.ylabel("Aportes de caudales de lluvia, real (Miles de m3/seg)", color='b')  
# plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2022-07-01'))  
# plt.ylim(0, 350)  
# plt.xlim(0, 4000)  
# plt.par.ylim(0, 400)  
#plt.set\_ylim(0, 350)  
# ax2.set\_ylim(0, 4500)  
plt.legend(loc='best')  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



## 4.4. Modelo entre el MEI + 'Aportes de caudales históricos' con los 'Aportes de caudales de lluvia reales'

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.1. Modelo de regresión lineal aporCaudalRio ~ mei + aporCaudalMediHistRio.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
regresion\_lineal = smf.ols(formula = 'aporCaudalRio ~ mei + aporCaudalMediHistRio', data=caudal\_mei)  
modelo\_ajustado\_caudal = regresion\_lineal.fit()  
modelo\_ajustado\_caudal.summary()

<class 'statsmodels.iolib.summary.Summary'>  
"""  
 OLS Regression Results   
==============================================================================  
Dep. Variable: aporCaudalRio R-squared: 0.720  
Model: OLS Adj. R-squared: 0.717  
Method: Least Squares F-statistic: 341.3  
Date: Tue, 16 Aug 2022 Prob (F-statistic): 3.59e-74  
Time: 09:57:08 Log-Likelihood: -86.508  
No. Observations: 269 AIC: 179.0  
Df Residuals: 266 BIC: 189.8  
Df Model: 2   
Covariance Type: nonrobust   
=========================================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------------------  
Intercept 0.0220 0.071 0.308 0.758 -0.118 0.162  
mei -0.2603 0.024 -10.739 0.000 -0.308 -0.213  
aporCaudalMediHistRio 0.9139 0.038 24.267 0.000 0.840 0.988  
==============================================================================  
Omnibus: 14.032 Durbin-Watson: 1.140  
Prob(Omnibus): 0.001 Jarque-Bera (JB): 15.770  
Skew: 0.474 Prob(JB): 0.000376  
Kurtosis: 3.713 Cond. No. 8.33  
==============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
"""

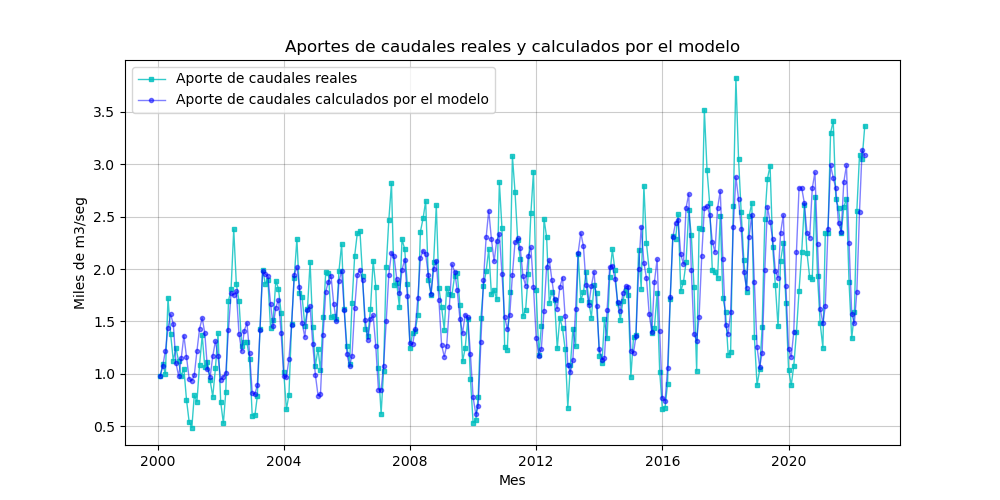
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.2. Salva y lee el modelo\_ajustado\_caudal.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Salva modelo\_ajustado\_caudal  
# modelo\_ajustado\_caudal.save('modelos/modelo\_ajustado\_caudal.pickle')  
#Lee modelo\_ajustado\_caudal  
modelo\_ajustado\_caudal = sm.load('modelos/modelo\_ajustado\_caudal.pickle')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.3. Matriz de correlación.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
corr = caudal\_mei.iloc[:, 0:3].corr()  
corr.style.background\_gradient(cmap='coolwarm').set\_precision(2)

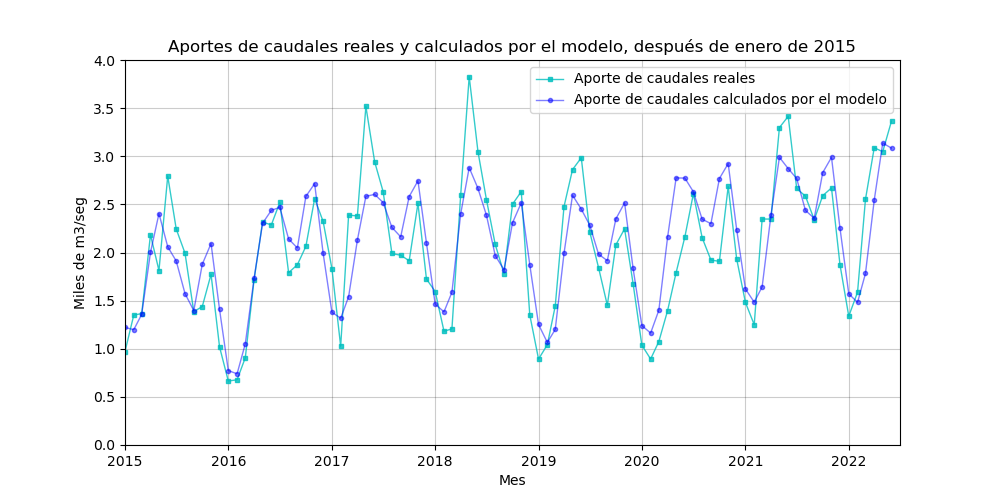
<pandas.io.formats.style.Styler at 0x16b36e71730>

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.4. Incluye los valores ajustados de aportes de caudal en caudal\_mei.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
valores\_ajustados = modelo\_ajustado\_caudal.predict(caudal\_mei)  
caudal\_mei.insert(loc=4, column='aporCaudal\_hat', value=valores\_ajustados, allow\_duplicates=True)  
# caudal\_mei

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.5. Grafica aporte de caudales reales y ajustados.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
ax1.set\_title ('Aportes de caudales reales y calculados por el modelo')  
ax1.set\_ylabel ('Miles de m3/seg')  
ax1.set\_xlabel ('Mes')  
ax1.plot(caudal\_mei.index, caudal\_mei.aporCaudalRio, 'c-s', \  
 label='Aporte de caudales reales', linewidth=1, markersize=3, alpha=0.8)  
ax1.plot(caudal\_mei.index, caudal\_mei.aporCaudal\_hat, 'b-o', \  
 label='Aporte de caudales calculados por el modelo', linewidth=1, markersize=3, alpha=0.5)  
# ax1.plot(meiT.index, meiT.meiP1988\_08, 'r-', label='Escenario alto', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP2010\_07, 'y-', label='Escenario Medio', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP1998\_08, 'g-', label='Escenario Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-06-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
plt.legend(loc='best')  
#plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2029-01-01'))  
# plt.ylim(0, 350)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.6. Grafica aporte de caudales reales y ajustados, después de 2015.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
ax1.set\_title ('Aportes de caudales reales y calculados por el modelo, después de enero de 2015')  
ax1.set\_ylabel ('Miles de m3/seg')  
ax1.set\_xlabel ('Mes')  
ax1.plot(caudal\_mei.index, caudal\_mei.aporCaudalRio, 'c-s', \  
 label='Aporte de caudales reales', linewidth=1, markersize=3, alpha=0.8)  
ax1.plot(caudal\_mei.index, caudal\_mei.aporCaudal\_hat, 'b-o', \  
 label='Aporte de caudales calculados por el modelo', linewidth=1, markersize=3, alpha=0.5)  
# ax1.plot(meiT.index, meiT.meiP1988\_08, 'r-', label='Escenario alto', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP2010\_07, 'y-', label='Escenario Medio', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP1998\_08, 'g-', label='Escenario Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-06-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
plt.legend(loc='best')  
plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 4)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.7. Define el mape.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#mean\_absolute\_percentage\_error, La media del porcentaje de error absoluto.   
def mape(y\_true, y\_pred):  
 y\_true, y\_pred = np.array(y\_true), np.array(y\_pred)  
 return np.mean(np.abs((y\_true - y\_pred) / y\_true)) \* 100

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.8. Aplica el mape a todo el conjunto de datos.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
mape(caudal\_mei.aporCaudalRio, caudal\_mei.aporCaudal\_hat) #16.38%.

16.38407717086482

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.9. Crea caudal\_mei\_m (para modelo) subconjuntos de train y test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
caudal\_mei\_m = caudal\_mei.drop(columns=['mei1', 'aporCaudal\_hat'], axis=1)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.10. Salva y lee caudal\_mei\_m (para el modelo con train y test).  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Salva caudal\_mei\_m  
# caudal\_mei\_m.to\_parquet('parquet/caudal\_mei\_m.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee caudal\_mei\_m  
caudal\_mei\_m = pd.read\_parquet('parquet/caudal\_mei\_m.parquet.gzip', \  
 engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.11. Crea subconjuntos de train y test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
caudal\_mei\_m = caudal\_mei.drop(columns=['aporCaudal\_hat'], axis=1)  
train = caudal\_mei\_m.iloc[:-78]  
test = caudal\_mei\_m.iloc[-78:]

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.12. Modelo de regresión lineal aplicado a train.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
regresion\_lineal = smf.ols(formula = 'aporCaudalRio ~ mei + aporCaudalMediHistRio', data=train)  
modelo\_ajustado\_caudal\_t = regresion\_lineal.fit()  
modelo\_ajustado\_caudal\_t.summary()

<class 'statsmodels.iolib.summary.Summary'>  
"""  
 OLS Regression Results   
==============================================================================  
Dep. Variable: aporCaudalRio R-squared: 0.666  
Model: OLS Adj. R-squared: 0.662  
Method: Least Squares F-statistic: 187.4  
Date: Tue, 16 Aug 2022 Prob (F-statistic): 1.70e-45  
Time: 09:59:38 Log-Likelihood: -45.545  
No. Observations: 191 AIC: 97.09  
Df Residuals: 188 BIC: 106.8  
Df Model: 2   
Covariance Type: nonrobust   
=========================================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------------------  
Intercept -0.0310 0.089 -0.349 0.728 -0.206 0.144  
mei -0.2641 0.027 -9.802 0.000 -0.317 -0.211  
aporCaudalMediHistRio 0.9556 0.052 18.477 0.000 0.854 1.058  
==============================================================================  
Omnibus: 13.811 Durbin-Watson: 1.207  
Prob(Omnibus): 0.001 Jarque-Bera (JB): 15.377  
Skew: 0.572 Prob(JB): 0.000458  
Kurtosis: 3.789 Cond. No. 8.97  
==============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
"""

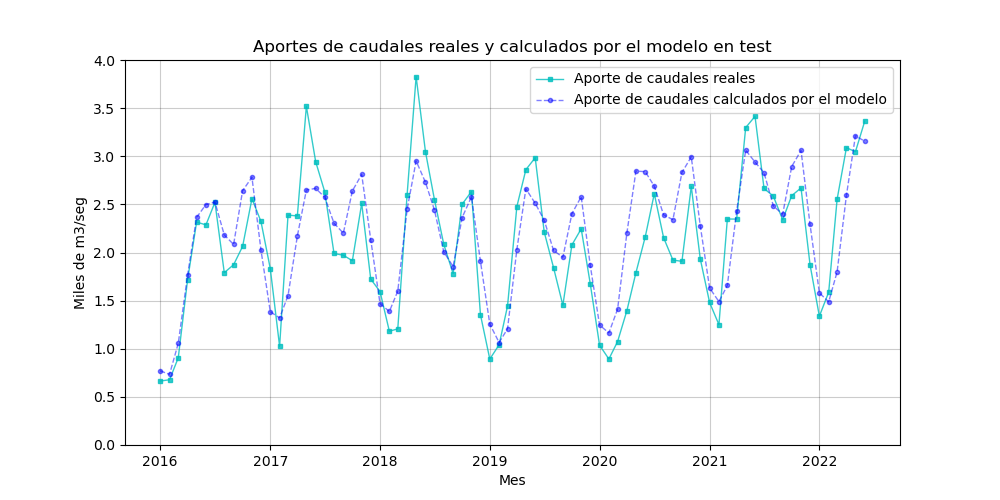
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.13. Salva y lee el modelo\_ajustado\_caudal\_t (aplicado a train)  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Salva modelo\_ajustado\_caudal\_t  
# modelo\_ajustado\_caudal\_t.save('modelos/modelo\_ajustado\_caudal\_t.pickle')  
#Lee modelo\_ajustado\_caudal  
modelo\_ajustado\_caudal\_t = sm.load('modelos/modelo\_ajustado\_caudal\_t.pickle')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.14. Incluye los valores ajustados de test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
valores\_ajustados = modelo\_ajustado\_caudal\_t.predict(test)  
test.insert(loc=3, column='aporCaudal\_hat', value=valores\_ajustados, allow\_duplicates=True)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.15. Calcula el modelo en test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
mape(test.aporCaudalRio, test.aporCaudal\_hat) #16.5%

16.548927941040834

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.4.16. Grafica aporte de caudales reales y calculados por el modelo en test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
ax1.set\_title ('Aportes de caudales reales y calculados por el modelo en test')  
ax1.set\_ylabel ('Miles de m3/seg')  
ax1.set\_xlabel ('Mes')  
ax1.plot(test.index, test.aporCaudalRio, 'c-s', \  
 label='Aporte de caudales reales', linewidth=1, markersize=3, alpha=0.8)  
ax1.plot(test.index, test.aporCaudal\_hat, 'b--o', \  
 label='Aporte de caudales calculados por el modelo', linewidth=1, markersize=3, alpha=0.5)  
# ax1.plot(meiT.index, meiT.meiP1988\_08, 'r-', label='Escenario alto', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP2010\_07, 'y-', label='Escenario Medio', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP1998\_08, 'g-', label='Escenario Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-06-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2015-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 4)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



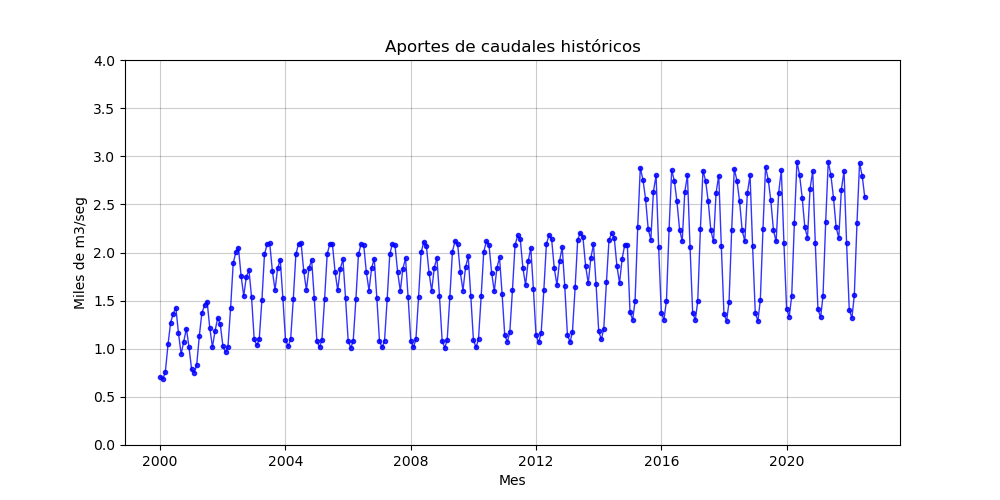
## 4.5. Proyección de los 'Aportes de caudales de lluvia históricos'

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.5.1. Lee aportes\_m.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee aportes\_m  
aportes\_m = pd.read\_parquet('parquet/aportes\_m.parquet.gzip', \  
 engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.5.2. Crea aportes\_h (históricos).  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
aportes\_h = aportes\_m  
aportes\_h = aportes\_h.drop(columns=['aporEner', 'aporEnerMediHist', 'aporCaudalRio', \  
 'diferenciaAportes'], axis=1)  
aportes\_h['aporCaudalMediHistRio'] = aportes\_h['aporCaudalMediHistRio']/1e3

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.5.3. Corrige valores atípicos de aporCaudalMediHistRio.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# aportes\_h.loc[pd.to\_datetime('2013-05-01'), 'aporCaudalMediHistRio'] #3.41  
# aportes\_h.loc[pd.to\_datetime('2015-10-01'), 'aporCaudalMediHistRio'] #5.20  
# aportes\_h.loc[pd.to\_datetime('2015-11-01'), 'aporCaudalMediHistRio'] #5.29  
# aportes\_h.loc[pd.to\_datetime('2015-12-01'), 'aporCaudalMediHistRio'] #4.07  
#  
# aportes\_h.loc[pd.to\_datetime('2014-05-01'), 'aporCaudalMediHistRio'] #2.13  
# aportes\_h.loc[pd.to\_datetime('2016-10-01'), 'aporCaudalMediHistRio'] #2.63  
# aportes\_h.loc[pd.to\_datetime('2016-11-01'), 'aporCaudalMediHistRio'] #2.80  
# aportes\_h.loc[pd.to\_datetime('2016-12-01'), 'aporCaudalMediHistRio'] #2.05  
#  
aportes\_h\_c = aportes\_h.copy()  
aportes\_h\_c.loc['2013-05-01', 'aporCaudalMediHistRio'] = \  
 aportes\_h\_c.loc['2014-05-01', 'aporCaudalMediHistRio']  
aportes\_h\_c.loc['2015-10-01', 'aporCaudalMediHistRio'] = \  
 aportes\_h\_c.loc['2016-10-01', 'aporCaudalMediHistRio']  
aportes\_h\_c.loc['2015-11-01', 'aporCaudalMediHistRio'] = \  
 aportes\_h\_c.loc['2016-11-01', 'aporCaudalMediHistRio']  
aportes\_h\_c.loc['2015-12-01', 'aporCaudalMediHistRio'] = \  
 aportes\_h\_c.loc['2016-12-01', 'aporCaudalMediHistRio']  
aportes\_h = aportes\_h\_c

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.5.4. Grafica de aporte de caudales históricos.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
ax1.set\_title ('Aportes de caudales históricos')  
ax1.set\_ylabel ('Miles de m3/seg')  
ax1.set\_xlabel ('Mes')  
ax1.plot(aportes\_h.index, aportes\_h.aporCaudalMediHistRio, 'b-o', \  
 label='Aporte de caudales históricos', linewidth=1, markersize=3, alpha=0.8)  
# ax1.plot(caudal\_mei.index, caudal\_mei.aporCaudal\_hat, 'b-o', \  
# label='Aporte de caudales calculados por el modelo', linewidth=1, markersize=3, alpha=0.5)  
# ax1.plot(meiT.index, meiT.meiP1988\_08, 'r-', label='Escenario alto', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP2010\_07, 'y-', label='Escenario Medio', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP1998\_08, 'g-', label='Escenario Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-06-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
# plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2020-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 4)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



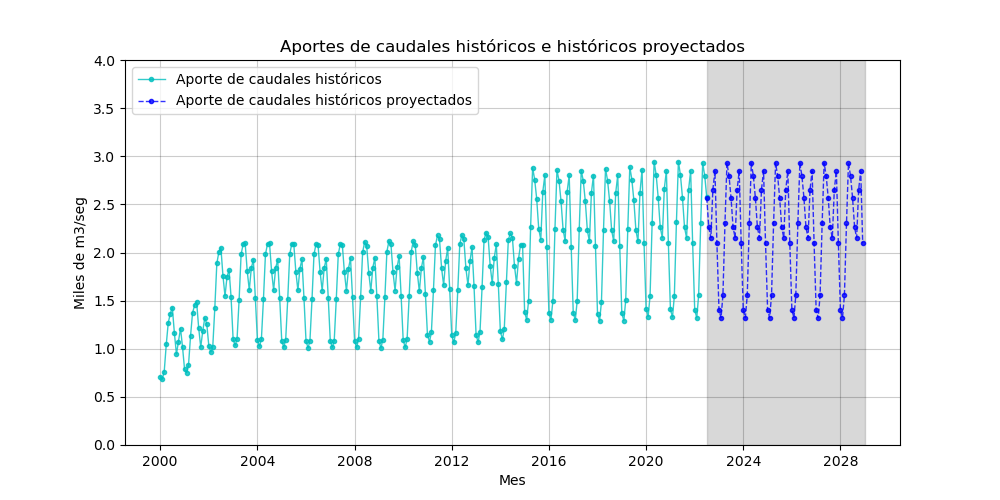
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.5.5. Crea los aportes futuros de caudales históricos.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
aportes\_repetidos = pd.DataFrame(aportes\_h.loc['2021-07-01': '2022-06-01', \  
 'aporCaudalMediHistRio'].values, index=range(0,12))  
aportes\_futuros = pd.DataFrame()  
for i in range(1, 7):  
 aportes\_futuros = aportes\_futuros.append(aportes\_repetidos)  
aportes\_futuros = aportes\_futuros.append(aportes\_repetidos[0:6])  
aportes\_futuros = aportes\_futuros.reset\_index()  
aportes\_futuros = aportes\_futuros.drop(columns=['index'], axis=1)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.5.6. Crea DataFrame de proyección.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
aportes\_h\_py = pd.DataFrame({'aporCaudalMediHistRio': \  
 aportes\_futuros[0].values}, index=pd.date\_range(start='2022-07-01', \  
 periods = 78, freq='MS'))  
aportes\_h\_py.index.name = 'Date'

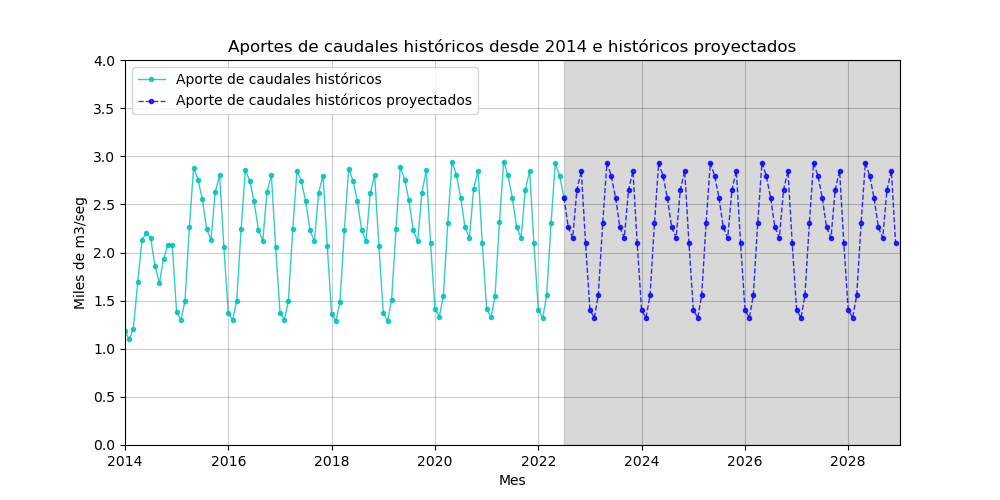
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.5.7. Integra DataFrame de proyección con aportes históricos.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
aportes\_h\_py = aportes\_h\_py.rename(columns={'aporCaudalMediHistRio': \  
 'aporCaudalHistProy'})  
aportes\_h = aportes\_h.join(aportes\_h\_py, how='outer')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.5.8. Salva y lee aportes\_h.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva aportes\_h  
# aportes\_h.to\_parquet('parquet/aportes\_h.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee caudal\_mei\_m  
aportes\_h = pd.read\_parquet('parquet/aportes\_h.parquet.gzip', \  
 engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.5.9. Grafica de aporte de caudales históricos e históricos proyectados.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)  
ax1.set\_title ('Aportes de caudales históricos e históricos proyectados')  
ax1.set\_ylabel ('Miles de m3/seg')  
ax1.set\_xlabel ('Mes')  
ax1.plot(aportes\_h.index, aportes\_h.aporCaudalMediHistRio, 'c-o', \  
 label='Aporte de caudales históricos', linewidth=1, markersize=3, alpha=0.8)  
ax1.plot(aportes\_h.index, aportes\_h.aporCaudalHistProy, 'b--o', \  
 label='Aporte de caudales históricos proyectados', linewidth=1, markersize=3, alpha=0.8)  
# ax1.plot(caudal\_mei.index, caudal\_mei.aporCaudal\_hat, 'b-o', \  
# label='Aporte de caudales calculados por el modelo', linewidth=1, markersize=3, alpha=0.5)  
# ax1.plot(meiT.index, meiT.meiP1988\_08, 'r-', label='Escenario alto', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP2010\_07, 'y-', label='Escenario Medio', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP1998\_08, 'g-', label='Escenario Bajo', markersize=2)  
ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
 color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2020-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 4)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.5.10. Grafica de caudales históricos y proyectados, desde 2014.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Aportes de caudales históricos desde 2014 e históricos proyectados')  
ax1.set\_ylabel ('Miles de m3/seg')  
ax1.set\_xlabel ('Mes')  
ax1.plot(aportes\_h.index, aportes\_h.aporCaudalMediHistRio, 'c-o', \  
 label='Aporte de caudales históricos', linewidth=1, markersize=3, alpha=0.8)  
ax1.plot(aportes\_h.index, aportes\_h.aporCaudalHistProy, 'b--o', \  
 label='Aporte de caudales históricos proyectados', linewidth=1, \  
 markersize=3, alpha=0.8)  
# ax1.plot(caudal\_mei.index, caudal\_mei.aporCaudal\_hat, 'b-o', \  
# label='Aporte de caudales calculados por el modelo', linewidth=1, markersize=3, alpha=0.5)  
# ax1.plot(meiT.index, meiT.meiP1988\_08, 'r-', label='Escenario alto', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP2010\_07, 'y-', label='Escenario Medio', markersize=2)  
# ax1.plot(meiT.index, meiT.meiP1998\_08, 'g-', label='Escenario Bajo', markersize=2)  
ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
 color='#808080', alpha=0.3)  
plt.legend(loc='best')  
plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 4)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



## 4.6. Proyección de los 'Aportes de caudales de lluvias'

La proyección de los aportes de caudales de lluvias se deben hacer para tres escenarios hasta diciembre de 2028.

La variable objetivo es 'aporCaudalRio' y las variables explicativas son: 'mei' y 'aporCaudalMediHistRio'.

La variable 'mei' es la que genera los tres escenarios.

Ya se cuenta con el modelo, elaborado en el numeral 4.4. de este documento. Ahora, se requiera elaborar el nuevo conjunto de datos sobre los que se aplicará la proyección.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.6.1. Se carga el modelo\_ajustado\_caudal basado en el conjunto 'train'.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
modelo\_ajustado\_caudal\_t = sm.load('modelos/modelo\_ajustado\_caudal\_t.pickle')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.6.2. Se leen las variables requeridas para proyectar los aportes de lluvias.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee el meiT que contiene las proyecciones de mei en tres escenarios.  
meiT = pd.read\_parquet('parquet/meiT.parquet.gzip', \  
 engine='fastparquet')  
#Lee el aportes\_h que contiene las proyecciones aportes históricos proyectados.  
aportes\_h = pd.read\_parquet('parquet/aportes\_h.parquet.gzip', \  
 engine='fastparquet')  
#Lee el caudal\_mei\_m que contiene el formato para aplicar al modelo.  
caudal\_mei\_m = pd.read\_parquet('parquet/caudal\_mei\_m.parquet.gzip', \  
 engine='fastparquet')

El fenómeno de El Niño, representado por el Índice Multivariado El Niño (MEI) se comporta de manera cíclica. Aunque no es posible predecir la frecuencia de estos ciclos, ni la amplitud de la variable, sí se conoce que su variación es continua y depende de sus valores anteriores. Es decir, normalmente cuando el valor de la variable empieza a descender sigue descendiendo y cuando asciende sigue ascendiendo, a menos que se presente un cambio en la dirección en algún momento.

Cuando el valor del MEI desciende por debajo de -0,5 se dice que se presenta un Fenómeno de La Niña. Cuando la variable supera el valor de 0,5 se considera que se presenta un Fenómeno de El Niño.

Se tienen registros mensuales del MEI desde hace 40 años y hay ciertos patrones de comportamiento que se repiten. Con el fin de realizar la evaluación de las ofertas en tres escenarios, se asumirán tres de estos patrones de comportamientos que sean consistentes con las actuales condiciones del MEI. Se escogen estos patrones de comportamiento del MEI de forma que ellos representen condiciones de precios altos, precios medios y precios bajos. Las tres series escogidas representan periodos con 78 meses continuos, empezando a partir de 1988-08, 1998-08 y 2010-07.

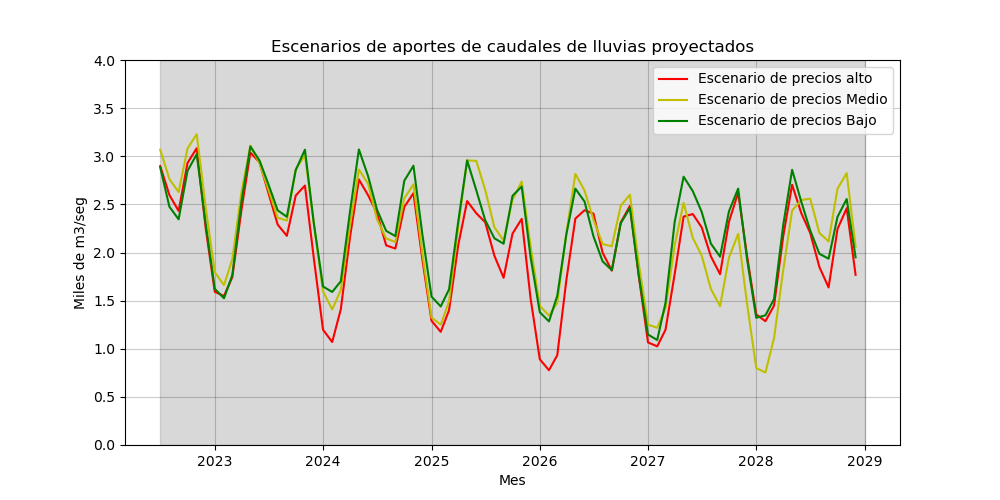
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.6.3. Crea tres escenarios de caudal\_mei para proyectar aportes de lluvias.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
aportes\_h = aportes\_h.drop(columns=['aporCaudalMediHistRio'], axis=1)  
caudal\_mei\_m0 = caudal\_mei\_m.join(aportes\_h, how='right')  
caudal\_mei\_m0['aporCaudalMediHistRio'] = caudal\_mei\_m0['aporCaudalHistProy']  
caudal\_mei\_m0 = caudal\_mei\_m0.loc[pd.to\_datetime('2022-07-01'):pd.to\_datetime('2028-12-01'), \  
 ['aporCaudalRio', 'mei', 'aporCaudalMediHistRio']]  
meiT = meiT.loc[pd.to\_datetime('2022-07-01'):pd.to\_datetime('2028-12-01'), \  
 ['meiP1988\_08', 'meiP1998\_08', 'meiP2010\_07']]  
meiT = meiT.rename(columns={'meiP1988\_08': 'EscenarioAlto', \  
 'meiP1998\_08': 'EscenarioBajo', 'meiP2010\_07': 'EscenarioMedio'})  
#  
caudal\_mei\_Alto = caudal\_mei\_m0.copy()  
caudal\_mei\_Medio = caudal\_mei\_m0.copy()  
caudal\_mei\_Bajo = caudal\_mei\_m0.copy()  
#  
caudal\_mei\_Alto['mei'] = meiT['EscenarioAlto'].values  
caudal\_mei\_Medio['mei'] = meiT['EscenarioMedio'].values  
caudal\_mei\_Bajo['mei'] = meiT['EscenarioBajo'].values

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.6.4. Calcula los valores proyectados en los tres escenarios.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
valores\_ajustados\_Alto = modelo\_ajustado\_caudal\_t.predict(caudal\_mei\_Alto)  
caudal\_mei\_Alto['aporCaudalRio'] = valores\_ajustados\_Alto  
caudal\_mei\_Alto = caudal\_mei\_Alto.rename(columns={'aporCaudalRio': \  
 'aporCaudalAlto'})  
  
valores\_ajustados\_Medio = modelo\_ajustado\_caudal\_t.predict(caudal\_mei\_Medio)  
caudal\_mei\_Medio['aporCaudalRio'] = valores\_ajustados\_Medio  
caudal\_mei\_Medio = caudal\_mei\_Medio.rename(columns={'aporCaudalRio': \  
 'aporCaudalMedio'})  
  
valores\_ajustados\_Bajo = modelo\_ajustado\_caudal\_t.predict(caudal\_mei\_Bajo)  
caudal\_mei\_Bajo['aporCaudalRio'] = valores\_ajustados\_Bajo  
caudal\_mei\_Bajo = caudal\_mei\_Bajo.rename(columns={'aporCaudalRio': \  
 'aporCaudalBajo'})  
  
caudal\_escenarios = caudal\_mei\_Alto.copy()  
caudal\_escenarios = caudal\_escenarios.drop(columns=['mei', 'aporCaudalMediHistRio'], axis=1)  
caudal\_escenarios.insert(loc=len(caudal\_escenarios.columns), column='aporCaudalMedio', \  
 value=caudal\_mei\_Medio.loc[:, 'aporCaudalMedio'].values)  
caudal\_escenarios.insert(loc=len(caudal\_escenarios.columns), column='aporCaudalBajo', \  
 value=caudal\_mei\_Bajo.loc[:, 'aporCaudalBajo'].values)  
  
caudal\_escenarios

aporCaudalAlto aporCaudalMedio aporCaudalBajo  
Date   
2022-07-01 2.897537 3.066567 2.884331  
2022-08-01 2.596691 2.763080 2.475200  
2022-09-01 2.432000 2.627441 2.344844  
2022-10-01 2.929834 3.080377 2.847960  
2022-11-01 3.083672 3.228933 3.020286  
... ... ... ...  
2028-08-01 1.849260 2.203167 1.986597  
2028-09-01 1.637029 2.115068 1.935473  
2028-10-01 2.243148 2.663083 2.369921  
2028-11-01 2.460373 2.824844 2.555452  
2028-12-01 1.768022 2.061184 1.950258  
  
[78 rows x 3 columns]

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.6.5. Salva y lee caudal\_escenarios.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva caudal\_escenarios  
# caudal\_escenarios.to\_parquet('parquet/caudal\_escenarios.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee caudal\_escenarios  
caudal\_escenarios = pd.read\_parquet('parquet/caudal\_escenarios.parquet.gzip', \  
 engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.6.6. Grafica de caudales de lluvias proyectados, en tres escenarios.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Escenarios de aportes de caudales de lluvias proyectados')  
ax1.set\_ylabel ('Miles de m3/seg')  
ax1.set\_xlabel ('Mes')  
ax1.plot(caudal\_escenarios.index, caudal\_escenarios.aporCaudalAlto, 'r-', \  
 label='Escenario de precios alto', markersize=2)  
ax1.plot(caudal\_escenarios.index, caudal\_escenarios.aporCaudalMedio, 'y-', \  
 label='Escenario de precios Medio', markersize=2)  
ax1.plot(caudal\_escenarios.index, caudal\_escenarios.aporCaudalBajo, 'g-', \  
 label='Escenario de precios Bajo', markersize=2)  
ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
 color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 4)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



## 4.7. Proyección de los 'Aportes de lluvias en energía'

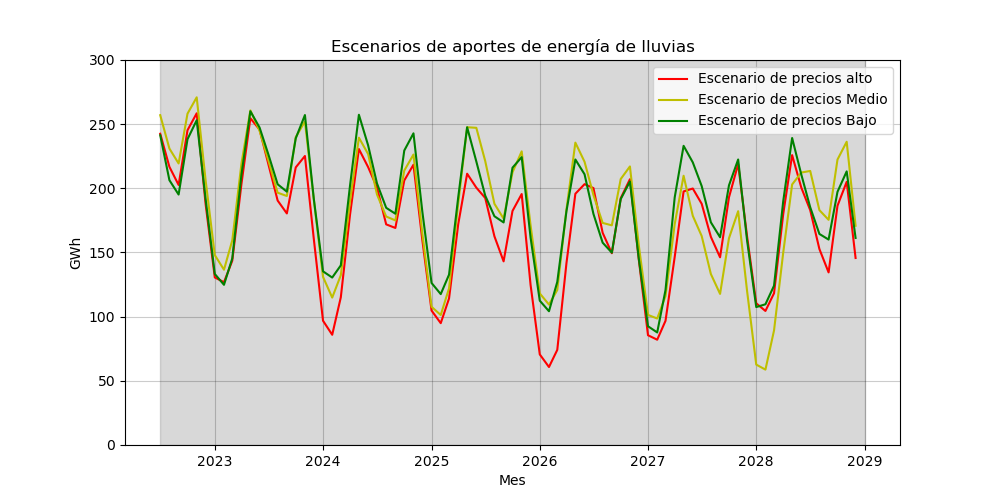
Ya se cuenta con el factor de conversión y es aporEner = (aporCaudalRio \* 0.0857) - 5.799

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.7.1. Crea aporte\_energia\_escenarios, en tres escenarios.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
aporte\_energia\_escenarios = caudal\_escenarios.copy()  
  
aporte\_energia\_escenarios.insert(loc=3, column='aporteEnergiaAlto', \  
 value=(aporte\_energia\_escenarios.aporCaudalAlto \* 1e3 \* 0.0857) - 5.799, \  
 allow\_duplicates=True)  
  
aporte\_energia\_escenarios.insert(loc=4, column='aporteEnergiaMedio', \  
 value=(aporte\_energia\_escenarios.aporCaudalMedio \* 1e3 \* 0.0857) - 5.799, \  
 allow\_duplicates=True)  
  
aporte\_energia\_escenarios.insert(loc=5, column='aporteEnergiaBajo', \  
 value=(aporte\_energia\_escenarios.aporCaudalBajo \* 1e3 \* 0.0857) - 5.799, \  
 allow\_duplicates=True)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.7.2. Solo deja escenarios de aporte\_energia, elimina caudales.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
aporte\_energia\_escenarios = aporte\_energia\_escenarios.drop(columns=['aporCaudalAlto', 'aporCaudalMedio', \  
 'aporCaudalBajo'], axis=1)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.7.3. Salva y lee aporte\_energia\_escenarios.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Salva aporte\_energia\_escenarios  
# aporte\_energia\_escenarios.to\_parquet\  
# ('parquet/aporte\_energia\_escenarios.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee aporte\_energia\_escenarios  
aporte\_energia\_escenarios = pd.read\_parquet\  
 ('parquet/aporte\_energia\_escenarios.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.7.4. Grafica los aportes de energía de lluvias en tres escenarios.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Escenarios de aportes de energía de lluvias')  
ax1.set\_ylabel ('GWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaAlto, 'r-', \  
 label='Escenario de precios alto', markersize=2)  
ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaMedio, 'y-', \  
 label='Escenario de precios Medio', markersize=2)  
ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaBajo, 'g-', \  
 label='Escenario de precios Bajo', markersize=2)  
ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
 color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 300)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



## 4.8. Proyección de los Precios de Bolsa Mensuales (Pbm)

Los Pbm a proyectar tendrán las siguientes condiciones:

1. Serán los precios sin Cee. Al final, a los precios proyectados se les debe agregar el Cee.
2. Serán los logaritmos de los precios (pbm\_log). Esto con el fin de garantizar que los precios sean siempre positivos y que el rango de los precios no sean tan amplios. Al final los precios se deben convertir con la fórmula EXP(pbm\_log).

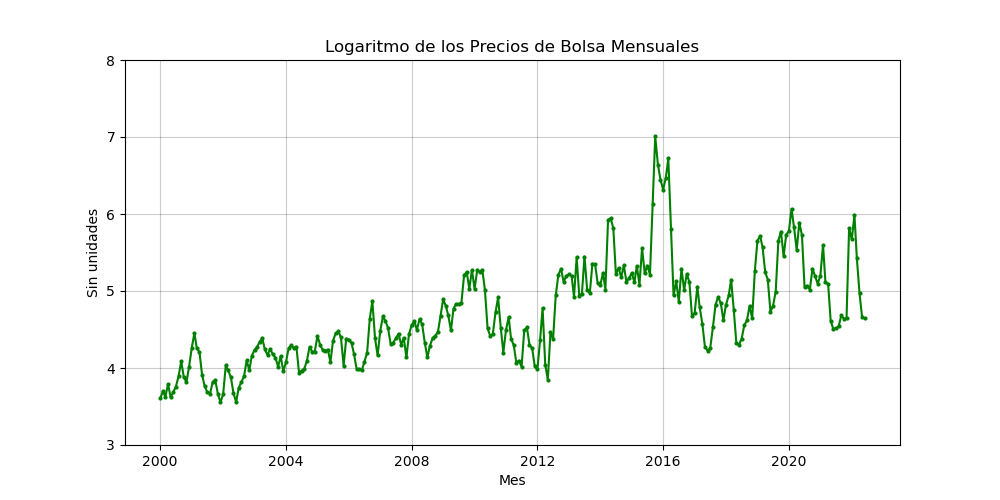
### 4.8.1. Creación del Pbm\_log

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.1. Se leen pbm y cee.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee pbm.  
pbm = pd.read\_parquet('parquet/pbm.parquet.gzip', engine='fastparquet') #271, se debe quitar 2022-07.  
#Lee cee\_cere.  
cee\_cere = pd.read\_parquet('parquet/cee\_cere.parquet.gzip', \  
 engine='fastparquet')  
#Lee cee\_cere2014\_07.  
cee\_cere2014\_07 = pd.read\_parquet('parquet/cee\_cere2014\_07.parquet.gzip', \  
 engine='fastparquet') #270 filas.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.2. Se integra el cee al pbm.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Se crea pbm\_cee y se ajusta el nombre de índice, se eliminan columnas y fila.  
pbm\_cee = pbm.copy()  
pbm\_cee.index.name = 'Date'  
pbm\_cee = pbm\_cee.drop(columns=['preEsca', 'preConEsca'], axis=1)  
pbm\_cee = pbm\_cee.drop(index=pd.to\_datetime('2022-07-01'), axis=0)  
  
#Se integran las columna de pbm y cee.  
pbm\_cee = pbm\_cee.join(cee\_cere, how='inner')  
  
#Se crea pbm\_log  
pbm\_cee.insert(loc=len(pbm\_cee.columns), column='pbm\_log', \  
 value=np.log(pbm\_cee.precio), allow\_duplicates=True)  
  
#Se resta cee a precio y se obtiene pbm\_sc (pbm sin cee).  
pbm\_cee.insert(loc=len(pbm\_cee.columns), column='pbm\_sc', value=pbm\_cee.precio - pbm\_cee.cee, \  
 allow\_duplicates=True)  
#Se aplica el logaritmo pbm\_sc y se obtiene pbm\_sc\_log.  
pbm\_cee.insert(loc=len(pbm\_cee.columns), column='pbm\_sc\_log', value=np.log(pbm\_cee.pbm\_sc), \  
 allow\_duplicates=True)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.3. Salva y lee pbm\_cee.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Salva pbm\_cee  
# pbm\_cee.to\_parquet\  
# ('parquet/pbm\_cee.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee pbm\_cee  
pbm\_cee = pd.read\_parquet\  
 ('parquet/pbm\_cee.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.4. Grafica pbm\_log.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Logaritmo de los Precios de Bolsa Mensuales')  
ax1.set\_ylabel ('Sin unidades')  
ax1.set\_xlabel ('Mes')  
ax1.plot(pbm\_cee.index, pbm\_cee.pbm\_log, 'g-o', \  
 label='Logaritmo de Pbm', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaAlto, 'r-', \  
# label='Escenario de precios alto', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaMedio, 'y-', \  
# label='Escenario de precios Medio', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaBajo, 'g-', \  
# label='Escenario de precios Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
# plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(3, 8)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



### 4.8.2. Creación del modelo de predicción de pbm\_log

Se explorarán diferentes variables que pueden influenciar o explicar los precios de energía en bolsa:

1. Aportes de lluvias en energía.
2. Diferencia entre los aportes de lluvia en energía y los aportes históricos de lluvias en energía (expectativas).
3. Capacidad de generación instalada.
4. Demanda de energía.

#### 4.8.2.1. Adquiere 'aportes de lluvia en energía' y 'diferencia de aportes de lluvia'

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.5. Lee aportes\_m.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee aportes\_m  
aportes\_m = pd.read\_parquet\  
 ('parquet/aportes\_m.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.6. Ajusta aportes\_m, elimina columnas y fila. Se integra con pbm\_cee.   
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
aportes\_m = aportes\_m.drop(columns=['aporCaudalRio', 'aporCaudalMediHistRio'], \  
 axis=1)  
aportes\_m = aportes\_m.drop(index=pd.to\_datetime('2022-07-01'), axis=0)  
#Se integran las columna de pbm y cee.  
pbm\_cee = pbm\_cee.join(aportes\_m, how='inner') #Con aportes de energía.  
pbm\_cee\_a = pbm\_cee.copy() #Con aportes de energía.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.7. Salva y lee pbm\_cee\_a.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva pbm\_cee\_a  
# pbm\_cee\_a.to\_parquet\  
# ('parquet/pbm\_cee\_a.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee pbm\_cee\_a  
pbm\_cee\_a = pd.read\_parquet\  
 ('parquet/pbm\_cee\_a.parquet.gzip', engine='fastparquet')

#### 4.8.2.2. Adquiere las variables 'capacidad de generación' y 'demanda'

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.8. Crea la máquina para conectarse a la base de datos dbXm.db.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
engine = create\_engine('sqlite:///dbXm.db', echo=False)  
Session = sessionmaker(bind=engine)  
session = Session()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.9. Lee las variables de 'capacidades' y 'demanda' diarias.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
tablas = ['CapaUtilDiarEner', 'DemaSin']  
nombresDf = ['capaUtilDiarEner', 'demaSin']  
df = {}  
  
for i, tabla in enumerate(tablas):  
 consulta = "SELECT \* FROM " + tabla  
 df[nombresDf[i]] = pd.read\_sql(consulta, engine, parse\_dates=["Date"])  
#Ajusta las columnas, los índices y los nombres de los DataFrames.  
for i, nombre in enumerate(nombresDf):  
 df[nombre] = df[nombre].iloc[:, 1:] if i <= 1 else df[nombre].iloc[:, [1, 3]]  
 df[nombre] = df[nombre].set\_index('Date')  
 df[nombre].columns = [nombre]

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.10. Convierte 'capacidades' y 'demandas' diarias a promedio mensuales.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
capacidad\_m = df['capaUtilDiarEner']  
capacidad\_m = capacidad\_m.groupby(capacidad\_m.index.to\_period('M')).mean()  
capacidad\_m.index = capacidad\_m.index.to\_timestamp()  
capacidad\_m = capacidad\_m.drop(index=pd.to\_datetime('2022-07-01'), axis=0)  
capacidad\_m = capacidad\_m.rename(columns={'capaUtilDiarEner':'capacidad'})  
capacidad\_m.capacidad = capacidad\_m.capacidad/1e9 #TWh  
  
demanda\_m = df['demaSin']  
demanda\_m = demanda\_m.groupby(demanda\_m.index.to\_period('M')).mean()  
demanda\_m.index = demanda\_m.index.to\_timestamp()  
demanda\_m = demanda\_m.drop(index=pd.to\_datetime('2022-07-01'), axis=0)  
demanda\_m = demanda\_m.rename(columns={'demaSin':'demanda'})  
demanda\_m.demanda = demanda\_m.demanda/1e9 #TWh

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.11. Salva 'capacidad\_m' y 'demanda\_m', son promedio mensuales en TWh.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva capacidad\_m  
# capacidad\_m.to\_parquet\  
# ('parquet/capacidad\_m.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
# #Salva demanda\_m  
# demanda\_m.to\_parquet\  
# ('parquet/demanda\_m.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee capacidad\_m  
capacidad\_m = pd.read\_parquet\  
 ('parquet/capacidad\_m.parquet.gzip', engine='fastparquet')  
#Lee demanda\_m  
demanda\_m = pd.read\_parquet\  
 ('parquet/demanda\_m.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.12. Integra 'capacidades' y 'demandas' mensuales a pbm\_cee\_a.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee pbm\_cee\_a  
pbm\_cee\_a = pd.read\_parquet\  
 ('parquet/pbm\_cee\_a.parquet.gzip', engine='fastparquet')  
#Convierte aportes a TWh.  
pbm\_cee\_a['aporEner'] = pbm\_cee\_a['aporEner']/1e3   
pbm\_cee\_a['aporEnerMediHist'] = pbm\_cee\_a['aporEnerMediHist']/1e3   
pbm\_cee\_a['diferenciaAportes'] = pbm\_cee\_a['diferenciaAportes']/1e3   
  
pbm\_cee\_a = pbm\_cee\_a.join(capacidad\_m, how='inner')  
pbm\_cee\_a = pbm\_cee\_a.join(demanda\_m, how='inner')  
pbm\_cee\_acd = pbm\_cee\_a.copy() #pbm con acd: aportesEnergia, capacidad, demanda.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.13. Salva y lee pbm\_cee\_acd.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva pbm\_cee\_acd  
# pbm\_cee\_acd.to\_parquet\  
# ('parquet/pbm\_cee\_acd.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee pbm\_cee\_acd  
pbm\_cee\_acd = pd.read\_parquet\  
 ('parquet/pbm\_cee\_acd.parquet.gzip', engine='fastparquet')

#### 4.8.2.3. Elabora el modelo de regresión del objetivo pbm\_log.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.14. Crea subconjuntos de train y test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
train = pbm\_cee\_acd.iloc[:-30]  
test = pbm\_cee\_acd.iloc[-30:]

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.15. Matriz de correlación.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
corr = test.iloc[:, [3, 6, 7, 8, 9, 10]].corr()  
corr.style.background\_gradient(cmap='coolwarm').set\_precision(2)

<pandas.io.formats.style.Styler at 0x1fda1fdad90>

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.16. Modelo de regresión lineal aplicado a train.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
regresion\_lineal = smf.ols(formula = 'pbm\_log ~ aporEner + \  
 aporEnerMediHist + capacidad + demanda', data=train)  
modelo\_ajustado\_pbm\_acd = regresion\_lineal.fit()  
modelo\_ajustado\_pbm\_acd.summary()

<class 'statsmodels.iolib.summary.Summary'>  
"""  
 OLS Regression Results   
==============================================================================  
Dep. Variable: pbm\_log R-squared: 0.714  
Model: OLS Adj. R-squared: 0.710  
Method: Least Squares F-statistic: 147.0  
Date: Tue, 16 Aug 2022 Prob (F-statistic): 9.22e-63  
Time: 10:37:22 Log-Likelihood: -79.724  
No. Observations: 240 AIC: 169.4  
Df Residuals: 235 BIC: 186.9  
Df Model: 4   
Covariance Type: nonrobust   
====================================================================================  
 coef std err t P>|t| [0.025 0.975]  
------------------------------------------------------------------------------------  
Intercept 3.0681 0.389 7.878 0.000 2.301 3.835  
aporEner -5.9763 0.616 -9.694 0.000 -7.191 -4.762  
aporEnerMediHist 2.8967 0.800 3.621 0.000 1.321 4.473  
capacidad -0.0948 0.030 -3.130 0.002 -0.154 -0.035  
demanda 22.5465 1.204 18.726 0.000 20.174 24.918  
==============================================================================  
Omnibus: 32.845 Durbin-Watson: 0.606  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 78.811  
Skew: 0.627 Prob(JB): 7.70e-18  
Kurtosis: 5.511 Cond. No. 890.  
==============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
"""

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.17. Salva y lee el modelo\_ajustado\_pbm\_acd (aplicado a train)  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva modelo\_ajustado\_pbm\_acd  
# modelo\_ajustado\_pbm\_acd.save('modelos/modelo\_ajustado\_pbm\_acd.pickle')  
#Lee modelo\_ajustado\_caudal  
modelo\_ajustado\_pbm\_acd = sm.load('modelos/modelo\_ajustado\_pbm\_acd.pickle')

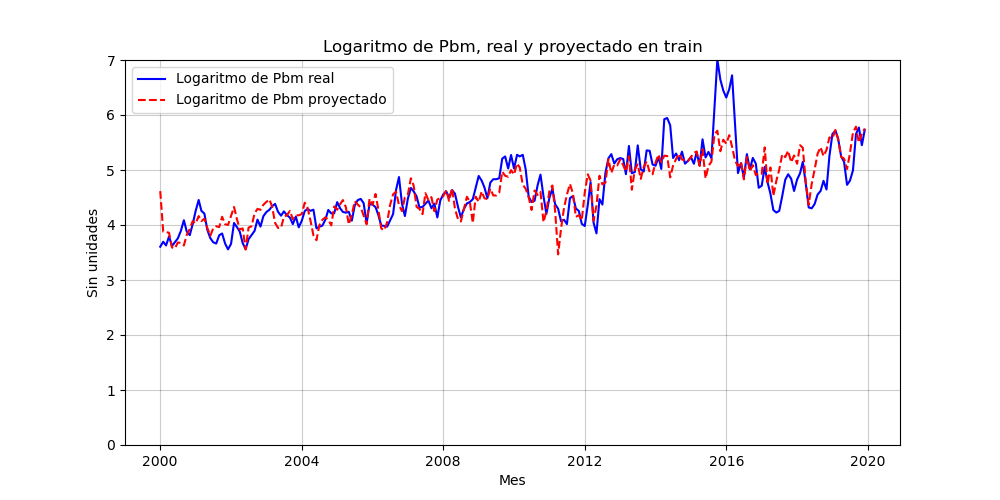
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.18. Incluye los valores ajustados de train.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
valores\_ajustados = modelo\_ajustado\_pbm\_acd.predict(train)  
train.insert(loc=len(train.columns), column='pbm\_log\_hat', \  
 value=valores\_ajustados, allow\_duplicates=True)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.19. Calcula el error del modelo en train.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
mape(train.pbm\_log, train.pbm\_log\_hat) #5%

5.08715198194615

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.20. Grafica pbm\_log real y proyectado en train.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Logaritmo de Pbm, real y proyectado en train')  
ax1.set\_ylabel ('Sin unidades')  
ax1.set\_xlabel ('Mes')  
ax1.plot(train.index, train.pbm\_log, 'b-', \  
 label='Logaritmo de Pbm real', markersize=2)  
ax1.plot(train.index, train.pbm\_log\_hat, 'r--', \  
 label='Logaritmo de Pbm proyectado', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaAlto, 'r-', \  
# label='Escenario de precios alto', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaMedio, 'y-', \  
# label='Escenario de precios Medio', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaBajo, 'g-', \  
# label='Escenario de precios Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 7)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()

C:\Users\ericd\AppData\Local\Temp\ipykernel\_18044\3229617231.py:4: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max\_open\_warning`).  
 fig = plt.figure(figsize=(10,5))

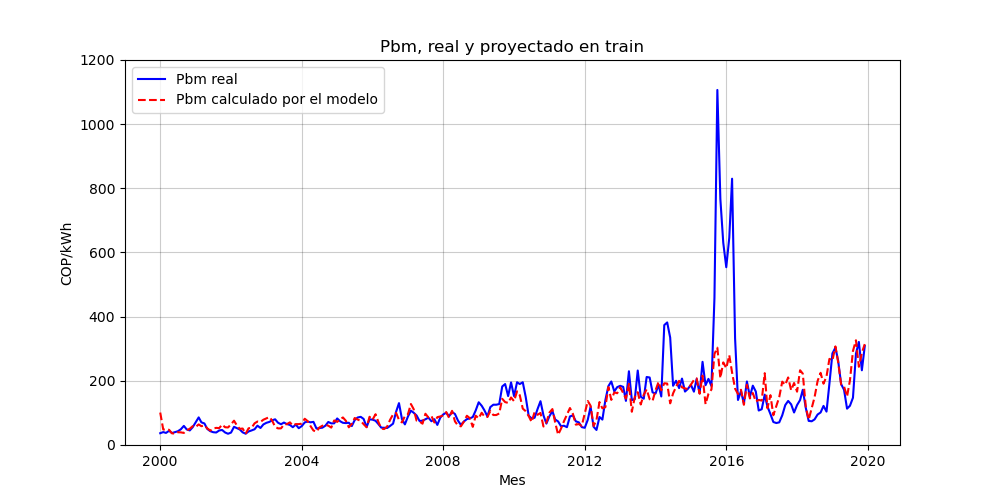


#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.21. Incluye los valores de pbm convertidos a train.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
pbm\_convertidos = np.exp(train.pbm\_log\_hat)  
train.insert(loc=len(train.columns), column='pbm\_hat', value=pbm\_convertidos, allow\_duplicates=True)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.22. Calcula el error del modelo en train sobre pbm.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
mape(train.precio, train.pbm\_hat) #24%

24.45581294936937

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.23. Grafica pbm real y proyectado en train.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Pbm, real y proyectado en train')  
ax1.set\_ylabel ('COP/kWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(train.index, train.precio, 'b-', \  
 label='Pbm real', markersize=2)  
ax1.plot(train.index, train.pbm\_hat, 'r--', \  
 label='Pbm calculado por el modelo', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaAlto, 'r-', \  
# label='Escenario de precios alto', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaMedio, 'y-', \  
# label='Escenario de precios Medio', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaBajo, 'g-', \  
# label='Escenario de precios Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 1200)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()

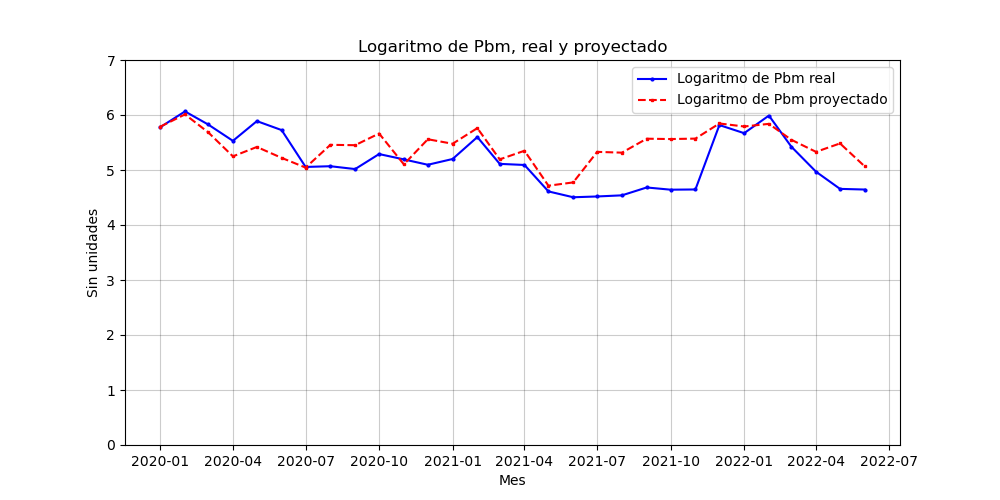


#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.24. Incluye los valores ajustados de test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
valores\_ajustados = modelo\_ajustado\_pbm\_acd.predict(test)  
test.insert(loc=len(test.columns), column='pbm\_log\_hat', value=valores\_ajustados, allow\_duplicates=True)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.25. Calcula el error del modelo en test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
mape(test.pbm\_log, test.pbm\_log\_hat) #7%

7.2749077007968355

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.26. Grafica pbm\_log real y proyectado en test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Logaritmo de Pbm, real y proyectado')  
ax1.set\_ylabel ('Sin unidades')  
ax1.set\_xlabel ('Mes')  
ax1.plot(test.index, test.pbm\_log, 'b-o', \  
 label='Logaritmo de Pbm real', markersize=2)  
ax1.plot(test.index, test.pbm\_log\_hat, 'r--s', \  
 label='Logaritmo de Pbm proyectado', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaAlto, 'r-', \  
# label='Escenario de precios alto', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaMedio, 'y-', \  
# label='Escenario de precios Medio', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaBajo, 'g-', \  
# label='Escenario de precios Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 7)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()

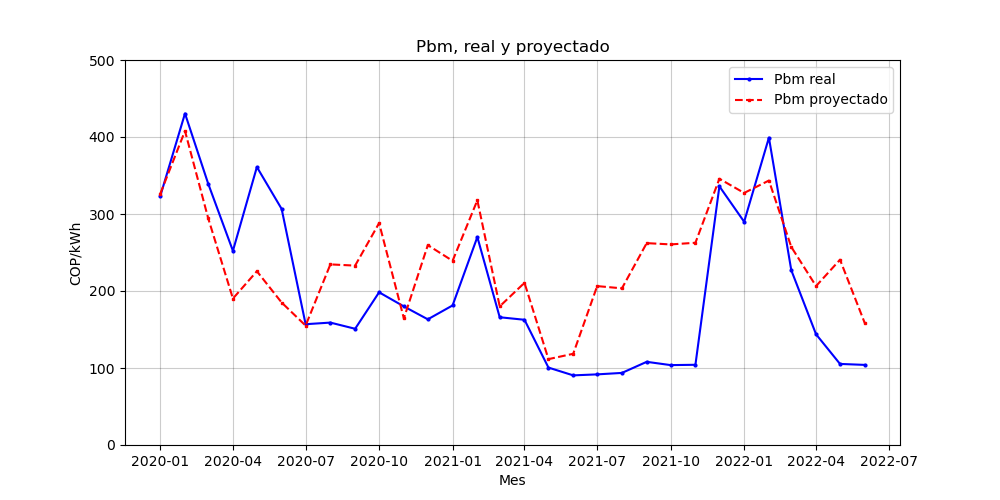


#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.27. Incluye los valores de pbm convertidos a test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
pbm\_convertidos = np.exp(test.pbm\_log\_hat)  
test.insert(loc=len(test.columns), column='pbm\_hat', value=pbm\_convertidos, allow\_duplicates=True)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.28. Calcula el error del modelo en test sobre pbm.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
mape(test.precio, test.pbm\_hat) #47%

47.411488340293204

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.29. Grafica pbm real y proyectado en test.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Pbm, real y proyectado')  
ax1.set\_ylabel ('COP/kWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(test.index, test.precio, 'b-o', \  
 label='Pbm real', markersize=2)  
ax1.plot(test.index, test.pbm\_hat, 'r--s', \  
 label='Pbm proyectado', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaAlto, 'r-', \  
# label='Escenario de precios alto', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaMedio, 'y-', \  
# label='Escenario de precios Medio', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaBajo, 'g-', \  
# label='Escenario de precios Bajo', markersize=2)  
# ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
# color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 500)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



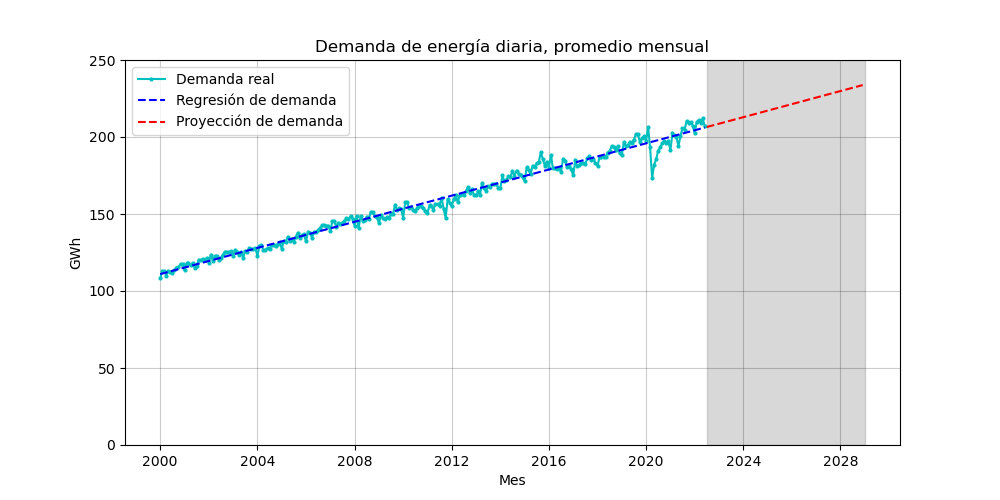
### 4.8.3. Proyecta demanda

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.30. Calcula la regresión lineal de la demanda.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee demanda\_m  
demanda\_m = pd.read\_parquet\  
 ('parquet/demanda\_m.parquet.gzip', engine='fastparquet')  
demanda\_m.demanda = demanda\_m.demanda \* 1e3  
demanda\_m.insert(loc=len(demanda\_m.columns), column='nMes', \  
 value=range(0, 270), allow\_duplicates=False)   
m, b = np.polyfit(demanda\_m.nMes, demanda\_m.demanda, 1)  
demanda\_m.insert(loc=len(demanda\_m.columns), column='regresion', \  
 value=m\*demanda\_m.nMes + b, allow\_duplicates=False)   
# aportes\_m  
# m, b  
# m = 0.3537137022942708  
# b = 110.97875816449749  
# m \* X + b

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.31. Crea DataFrame de proyección y lo integra a demanda\_m.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
demanda\_m.insert(loc=len(demanda\_m.columns), column='proyeccion', value=np.NaN, \  
 allow\_duplicates=True)  
  
demanda\_py = pd.DataFrame({'demanda': np.NaN, 'nMes': range(270, 348), \  
 'regresion': np.NaN, 'proyeccion': np.NaN}, \  
 index=pd.date\_range(start='2022-07-01', periods = 78, freq='MS'))  
demanda\_py.index.name = 'Date'  
demanda\_py['proyeccion'] = (m\*demanda\_py.nMes + b)  
# Integra.  
demanda\_py = demanda\_m.append(demanda\_py, ignore\_index = False)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.32. Salva y lee demanda\_py.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva demanda\_py  
# demanda\_py.to\_parquet\  
# ('parquet/demanda\_py.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee demanda\_py  
demanda\_py = pd.read\_parquet\  
 ('parquet/demanda\_py.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.33. Grafica de demanda de energía diaria promedio mensual.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Demanda de energía diaria, promedio mensual')  
ax1.set\_ylabel ('GWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(demanda\_py.index, demanda\_py.demanda, 'c-o', \  
 label='Demanda real', markersize=2)  
ax1.plot(demanda\_py.index, demanda\_py.regresion, 'b--', \  
 label='Regresión de demanda', markersize=2)  
ax1.plot(demanda\_py.index, demanda\_py.proyeccion, 'r--', \  
 label='Proyección de demanda', markersize=2)  
# ax1.plot(test.index, test.pbm\_hat, 'r--s', \  
# label='Pbm proyectado', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaAlto, 'r-', \  
# label='Escenario de precios alto', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaMedio, 'y-', \  
# label='Escenario de precios Medio', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaBajo, 'g-', \  
# label='Escenario de precios Bajo', markersize=2)  
ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
 color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 250)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()

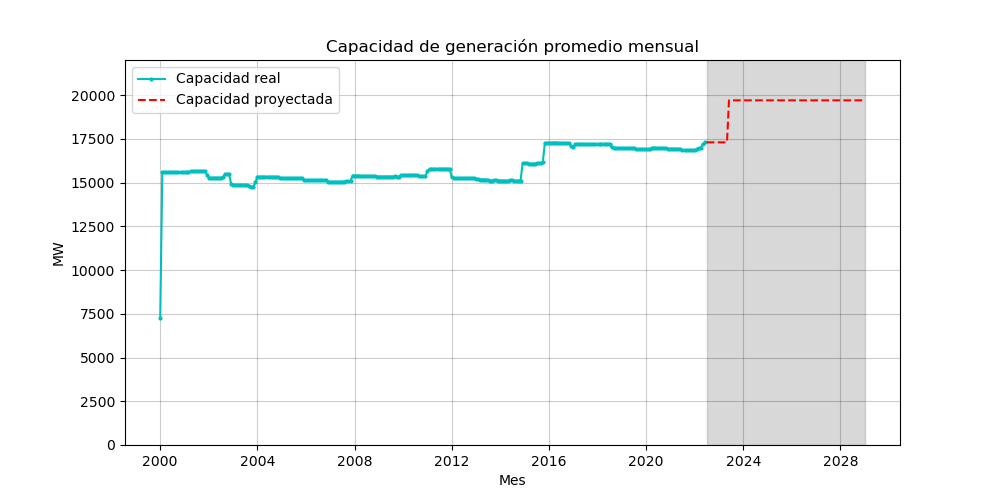


### 4.8.4. Proyecta capacidad

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.34. Crea DataFrame de proyección y lo integra a capacidad\_m.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee capacidad\_m  
capacidad\_m = pd.read\_parquet\  
 ('parquet/capacidad\_m.parquet.gzip', engine='fastparquet')  
capacidad\_m.insert(loc=len(capacidad\_m.columns), column='proyeccion', value=np.NaN, \  
 allow\_duplicates=True)  
#  
capacidad\_py = pd.DataFrame({'capacidad': np.NaN, 'proyeccion': np.NaN}, \  
 index=pd.date\_range(start='2022-07-01', periods = 78, freq='MS'))  
capacidad\_py.index.name = 'Date'  
capacidad\_py['proyeccion'] = capacidad\_m.iloc[269, 0]  
#Integra.  
capacidad\_py = capacidad\_m.append(capacidad\_py, ignore\_index = False)  
#Agrega la nueva capacidad de Hidroituango de 2400 MW a partir de 2023-06-01  
capacidad\_py.loc[pd.to\_datetime('2023-06-01') : pd.to\_datetime('2028-12-01'), \  
 ['proyeccion']] = (17.29361 + 2.4)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.35. Salva y lee capacidad\_py.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva capacidad\_py  
# capacidad\_py.to\_parquet\  
# ('parquet/capacidad\_py.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee capacidad\_py  
capacidad\_py = pd.read\_parquet\  
 ('parquet/capacidad\_py.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.36. Grafica de capacidad de generación de energía diaria, promedio mensual.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Capacidad de generación promedio mensual')  
ax1.set\_ylabel ('MW')  
ax1.set\_xlabel ('Mes')  
ax1.plot(capacidad\_py.index, capacidad\_py.capacidad \* 1e3, 'c-o', \  
 label='Capacidad real', markersize=2)  
ax1.plot(capacidad\_py.index, capacidad\_py.proyeccion \* 1e3, 'r--', \  
 label='Capacidad proyectada', markersize=2)  
# ax1.plot(demanda\_m.index, demanda\_m.regresion, 'b--', \  
# label='Regresión de demanda', markersize=2)  
# ax1.plot(demanda\_m.index, demanda\_m.proyeccion, 'r--', \  
# label='Proyección de demanda', markersize=2)  
# ax1.plot(test.index, test.pbm\_hat, 'r--s', \  
# label='Pbm proyectado', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaAlto, 'r-', \  
# label='Escenario de precios alto', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaMedio, 'y-', \  
# label='Escenario de precios Medio', markersize=2)  
# ax1.plot(aporte\_energia\_escenarios.index, aporte\_energia\_escenarios.aporteEnergiaBajo, 'g-', \  
# label='Escenario de precios Bajo', markersize=2)  
ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
 color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 22000)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



### 4.8.5. Prepara tres conjuntos de datos para proyección de pbm

Estos tres conjunto de datos (Alto, Medio y Bajo) contienen las variables explicativas proyectadas, que son las siguientes:

1. aporte\_energia\_escenarios, en tres Escenarios.
2. aportes\_h, que nos permitirán calcular las diferencias de aportes en energía.
3. capacidad\_py
4. demanda\_py

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.37. Lee las variables explicativas proyectadas.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
aporte\_energia\_escenarios = pd.read\_parquet\  
 ('parquet/aporte\_energia\_escenarios.parquet.gzip', engine='fastparquet')  
aportes\_h = pd.read\_parquet\  
 ('parquet/aportes\_h.parquet.gzip', engine='fastparquet')  
#La relación es: aporEner = (aporCaudalRio \* 0.0857) - 5.799 #m3/seg.  
aportes\_h.insert(loc=len(aportes\_h.columns), column='aporEnerMediHist', \  
 value=aportes\_h.aporCaudalHistProy \* 1e3 \* 0.0857 - 5.799, allow\_duplicates=True)  
aportes\_h = aportes\_h.drop(columns=['aporCaudalMediHistRio', \  
 'aporCaudalHistProy'], axis=1)  
#Capacidad.  
capacidad\_py = pd.read\_parquet\  
 ('parquet/capacidad\_py.parquet.gzip', engine='fastparquet')  
capacidad\_py = capacidad\_py.drop(columns=['capacidad'], axis=1)  
capacidad\_py = capacidad\_py[pd.to\_datetime('2022-07-01'):pd.to\_datetime('2029-01-01')]  
capacidad\_py = capacidad\_py.rename(columns={'proyeccion': 'capacidad'})  
#Demanda.  
demanda\_py = pd.read\_parquet\  
 ('parquet/demanda\_py.parquet.gzip', engine='fastparquet')  
demanda\_py = demanda\_py.drop(columns=['demanda', 'nMes', 'regresion'], axis=1)  
demanda\_py = demanda\_py[pd.to\_datetime('2022-07-01'):pd.to\_datetime('2029-01-01')]  
demanda\_py = demanda\_py.rename(columns={'proyeccion': 'demanda'})

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.38. IntegraLee las variables explicativas proyectadas`en pbm\_0`.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#1. Tres escenarios de aportes de energía.  
pbm\_0 = aporte\_energia\_escenarios.copy()  
#2. Proyección de aportes de lluvias en caudales.  
pbm\_0 = pbm\_0.join(aportes\_h, how='left')  
#3. Proyección de capacidad\_py.  
pbm\_0 = pbm\_0.join(capacidad\_py, how='left')  
#4. Proyección de demanda\_py.  
pbm\_0 = pbm\_0.join(demanda\_py, how='left')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.39. Salva y lee pbm\_0.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva pbm\_0  
# pbm\_0.to\_parquet\  
# ('parquet/pbm\_0.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee pbm\_0  
pbm\_0 = pd.read\_parquet\  
 ('parquet/pbm\_0.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.40. Crea los tres subconjuntos de proyección a partir de pbm\_0.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
pbm\_alto = pbm\_0.copy()  
pbm\_alto = pbm\_alto[['aporteEnergiaAlto', 'aporEnerMediHist', 'capacidad', 'demanda']]  
pbm\_alto = pbm\_alto.rename(columns={'aporteEnergiaAlto': 'aporEner'})  
pbm\_alto.aporEner = pbm\_alto.aporEner/1e3  
pbm\_alto.aporEnerMediHist = pbm\_alto.aporEnerMediHist/1e3  
pbm\_alto.demanda = pbm\_alto.demanda/1e3  
#  
pbm\_medio = pbm\_0.copy()  
pbm\_medio = pbm\_medio[['aporteEnergiaMedio', 'aporEnerMediHist', 'capacidad', 'demanda']]  
pbm\_medio = pbm\_medio.rename(columns={'aporteEnergiaMedio': 'aporEner'})  
pbm\_medio.aporEner = pbm\_medio.aporEner/1e3  
pbm\_medio.aporEnerMediHist = pbm\_medio.aporEnerMediHist/1e3  
pbm\_medio.demanda = pbm\_medio.demanda/1e3  
#  
pbm\_bajo = pbm\_0.copy()  
pbm\_bajo = pbm\_bajo[['aporteEnergiaBajo', 'aporEnerMediHist', 'capacidad', 'demanda']]  
pbm\_bajo = pbm\_bajo.rename(columns={'aporteEnergiaBajo': 'aporEner'})  
pbm\_bajo.aporEner = pbm\_bajo.aporEner/1e3  
pbm\_bajo.aporEnerMediHist = pbm\_bajo.aporEnerMediHist/1e3  
pbm\_bajo.demanda = pbm\_bajo.demanda/1e3

### 4.8.6. Proyecta los tres precios de bolsa Pbm

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.41. Lee el modelo\_ajustado\_pbm\_acd  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee el modelo\_ajustado\_pbm\_acd  
modelo\_ajustado\_pbm\_acd = sm.load('modelos/modelo\_ajustado\_pbm\_acd.pickle')

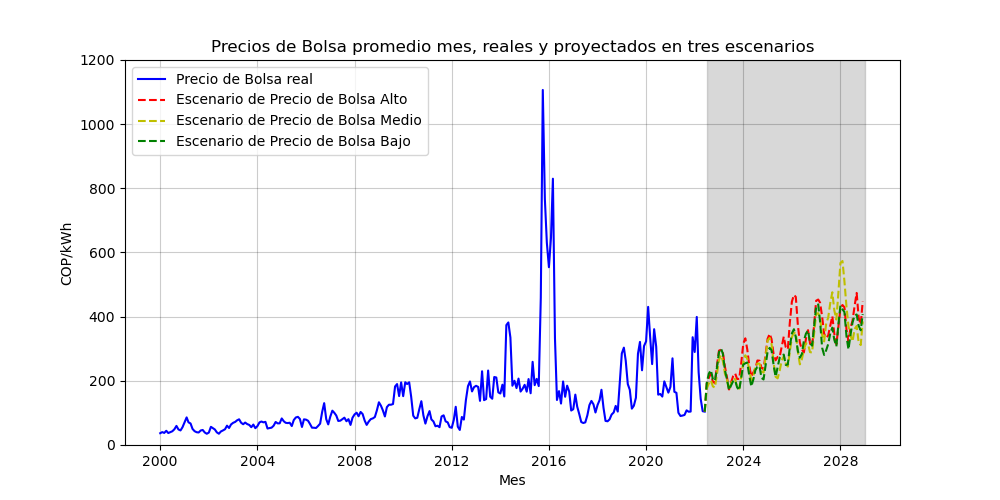
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.42. Calcula los valores proyectados y pasa de log a pbm.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
valores\_ajustados = modelo\_ajustado\_pbm\_acd.predict(pbm\_alto)  
pbm\_alto.insert(loc=len(pbm\_alto.columns), column='pbm\_alto', \  
 value=np.exp(valores\_ajustados), allow\_duplicates=True)  
  
valores\_ajustados = modelo\_ajustado\_pbm\_acd.predict(pbm\_medio)  
pbm\_medio.insert(loc=len(pbm\_medio.columns), column='pbm\_medio', \  
 value=np.exp(valores\_ajustados), allow\_duplicates=True)  
  
valores\_ajustados = modelo\_ajustado\_pbm\_acd.predict(pbm\_bajo)  
pbm\_bajo.insert(loc=len(pbm\_bajo.columns), column='pbm\_bajo', \  
 value=np.exp(valores\_ajustados), allow\_duplicates=True)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.43. Integra los pbm proyectados en pbm\_py.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
pbm\_py = pbm\_alto.copy()  
pbm\_py = pbm\_py[['pbm\_alto']]  
pbm\_py = pd.concat([pbm\_py, pbm\_medio[['pbm\_medio']]], axis=1)  
pbm\_py = pd.concat([pbm\_py, pbm\_bajo[['pbm\_bajo']]], axis=1)

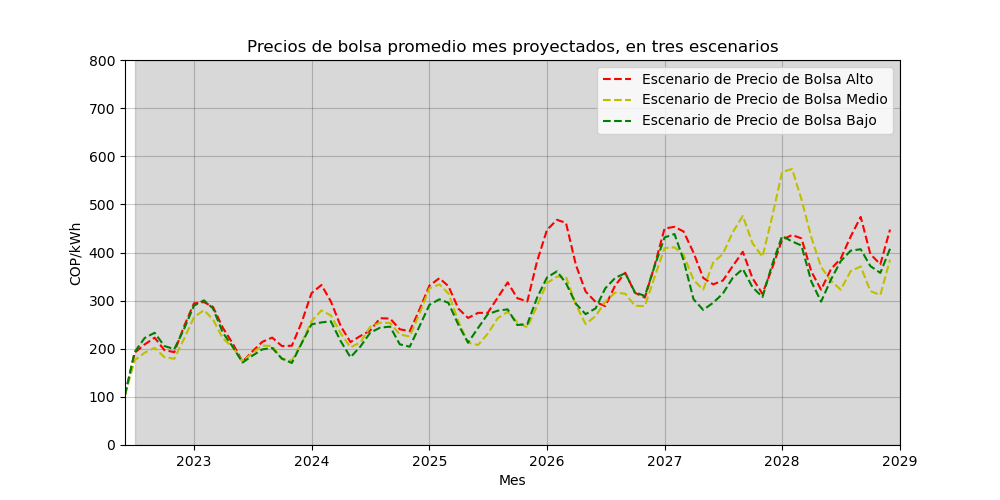
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.44. Integra con pbm reales.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee pbm\_cee  
pbm\_cee = pd.read\_parquet\  
 ('parquet/pbm\_cee.parquet.gzip', engine='fastparquet')  
pbm\_cee = pbm\_cee.drop(columns=['cee', 'cere', 'pbm\_log', 'pbm\_sc', 'pbm\_sc\_log'], axis=1)  
pbm\_cee = pbm\_cee.rename(columns={'precio': 'pbm'})  
  
pbm\_py = pbm\_py.join(pbm\_cee, how='outer')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.45. Carga datos en '2022-06-01.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
pbm\_py.loc[pd.to\_datetime('2022-06-01'), \  
 ['pbm\_alto', 'pbm\_medio', 'pbm\_bajo']] = \  
 pbm\_cee.loc[pd.to\_datetime('2022-06-01'), 'pbm']

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.46. Grafica los pbm reales y proyectados en tres escenarios.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Precios de Bolsa promedio mes, reales y proyectados en tres escenarios')  
ax1.set\_ylabel ('COP/kWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(pbm\_py.index, pbm\_py.pbm, 'b-', \  
 label='Precio de Bolsa real', markersize=2)  
ax1.plot(pbm\_py.index, pbm\_py.pbm\_alto, 'r--', \  
 label='Escenario de Precio de Bolsa Alto', markersize=2)  
ax1.plot(pbm\_py.index, pbm\_py.pbm\_medio, 'y--', \  
 label='Escenario de Precio de Bolsa Medio', markersize=2)  
ax1.plot(pbm\_py.index, pbm\_py.pbm\_bajo, 'g--', \  
 label='Escenario de Precio de Bolsa Bajo', markersize=2)  
ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
 color='#808080', alpha=0.3)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2014-01-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 1200)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.8.47. Grafica los pbm proyectados en tres escenarios.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Precios de bolsa promedio mes proyectados, en tres escenarios')  
ax1.set\_ylabel ('COP/kWh')  
ax1.set\_xlabel ('Mes')  
# ax1.plot(pbm\_py.index, pbm\_py.pbm, 'b-', \  
# label='Precio de Bolsa real', markersize=2)  
ax1.plot(pbm\_py.index, pbm\_py.pbm\_alto, 'r--', \  
 label='Escenario de Precio de Bolsa Alto', markersize=2)  
ax1.plot(pbm\_py.index, pbm\_py.pbm\_medio, 'y--', \  
 label='Escenario de Precio de Bolsa Medio', markersize=2)  
ax1.plot(pbm\_py.index, pbm\_py.pbm\_bajo, 'g--', \  
 label='Escenario de Precio de Bolsa Bajo', markersize=2)  
ax1.axvspan(pd.to\_datetime('2022-07-01'), pd.to\_datetime('2029-01-01'), \  
 color='#808080', alpha=0.3)  
plt.legend(loc='best')  
plt.xlim(pd.to\_datetime('2022-06-01'), pd.to\_datetime('2029-01-01'))  
plt.ylim(0, 800)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.48. Crea pbm\_py\_f.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
pbm\_py\_f = pbm\_py.copy() #Proyección pbm final.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.8.49. Salva y lee pbm\_py.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# # #Salva pbm\_py-f  
# pbm\_py\_f.to\_parquet\  
# ('parquet/pbm\_py\_f.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
# pbm\_py\_f.to\_excel('xlsx/pbm\_py\_f.xlsx', sheet\_name='hoja1')  
#Lee pbm\_py  
pbm\_py\_f = pd.read\_parquet\  
 ('parquet/pbm\_py\_f.parquet.gzip', engine='fastparquet')

## 4.9. Proyección del IPP

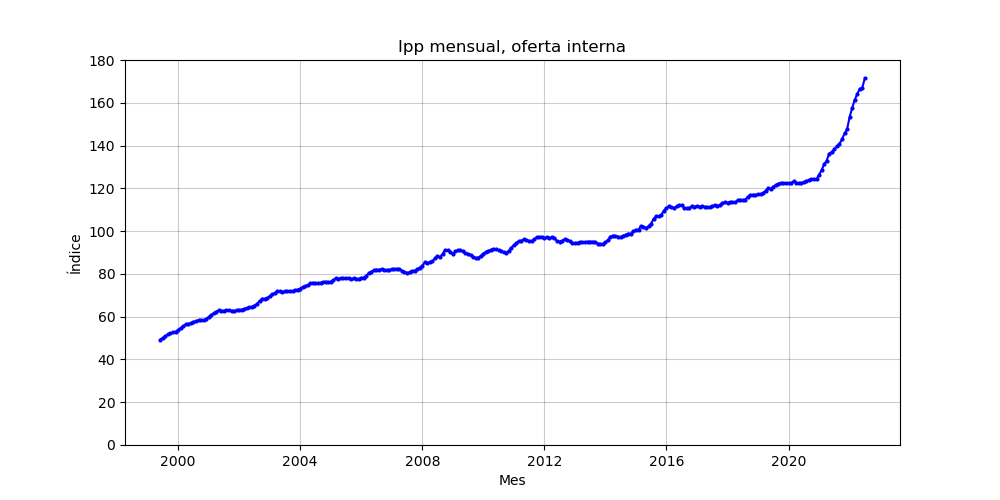
Para poder calcular los precios futuros, que permiten estimar los egresos de energía, se debe realizar una proyección del IPP, ya que los componentes de los precios de las ofertas (cargo C de comercialización y el precio fijo) están expresados en pesos de abril de 2022. Estos precios se deben ajustar mensualmente.

### 4.9.1. Obtiene el IPP mensual del DANE

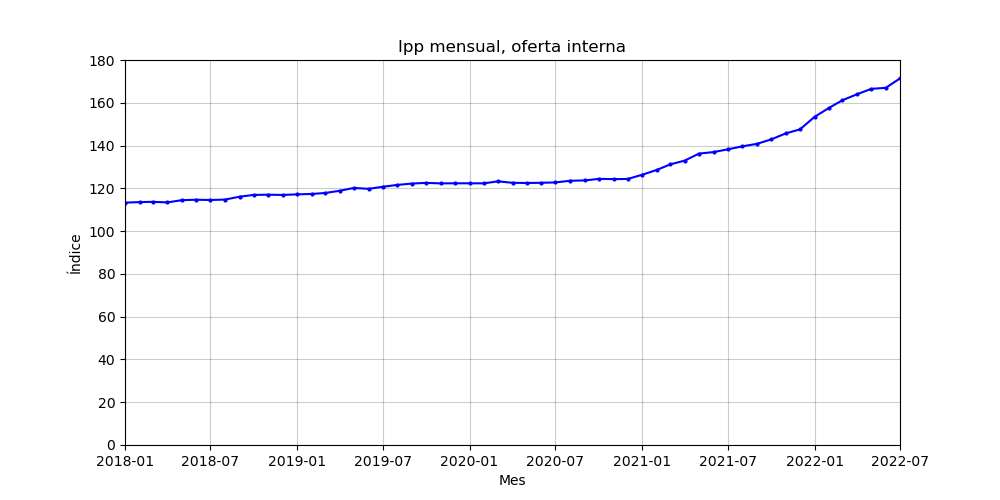
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.9.1. Obtiene el IPP histórico.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
ipp = pd.read\_excel("xlsx/anexo\_ipp1\_jul22.xlsx", sheet\_name='IPP Histórico', \  
 skiprows=4, skipfooter=9, usecols=[0, 1, 6])  
ipp.insert(loc=len(ipp.columns), column='Date', \  
 value=pd.date\_range(start='1999-06-01', end='2022-07-01', freq='MS'), \  
 allow\_duplicates=False)  
ipp = ipp.set\_index('Date')  
ipp.columns = ['columna1', 'columna2', 'ipp']  
ipp = ipp.drop(columns=['columna1', 'columna2'], axis=1)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.9.2. Salva y lee el IPP.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# # #Salva pbm\_py-f  
# ipp.to\_parquet\  
# ('parquet/ipp.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee ipp  
ipp = pd.read\_parquet\  
 ('parquet/ipp.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.9.3. Grafica el IPP.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Ipp mensual, oferta interna')  
ax1.set\_ylabel ('Índice')  
ax1.set\_xlabel ('Mes')  
ax1.plot(ipp.index, ipp.ipp, 'b-o', \  
 label='Ipp', markersize=2)  
# plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2018-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 180)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



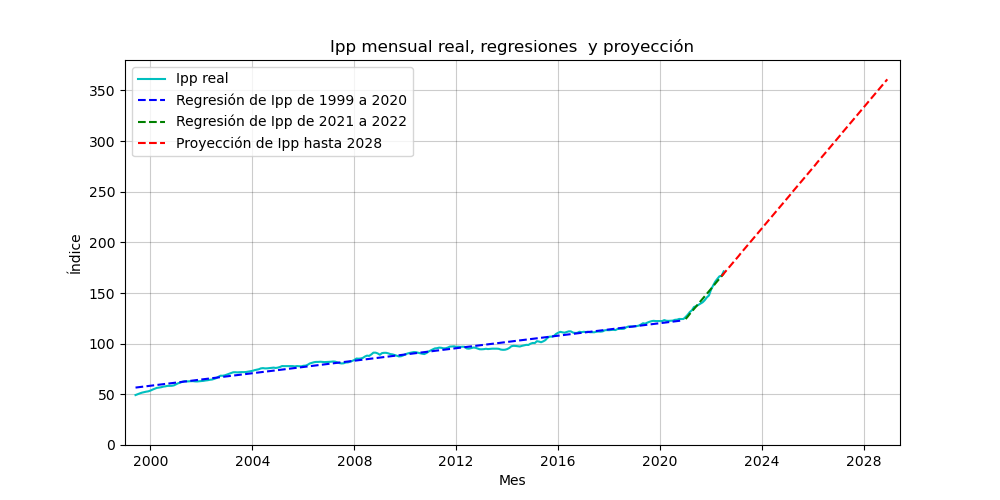
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.9.4. Grafica el IPP desde 2021.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Ipp mensual, oferta interna')  
ax1.set\_ylabel ('Índice')  
ax1.set\_xlabel ('Mes')  
ax1.plot(ipp.index, ipp.ipp, 'b-o', \  
 label='Ipp', markersize=2)  
# plt.legend(loc='best')  
plt.xlim(pd.to\_datetime('2018-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 180)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.9.5. Calcula dos regresiones de Ipp. Una hasta 2020-12 y otra en adelante.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Regresion99  
ipp.insert(loc=len(ipp.columns), column='n\_mes99', value=range(0, 278), \  
 allow\_duplicates=False)  
#Regresión de 1999-06 a 2020-12  
m99, b99 = np.polyfit(\  
 ipp.loc[pd.to\_datetime('1999-06-01'):pd.to\_datetime('2020-12-01'), :].\  
 n\_mes99, ipp.loc[pd.to\_datetime('1999-06-01'):\  
 pd.to\_datetime('2020-12-01'), :].ipp, 1)  
ipp.insert(loc=len(ipp.columns), column='regresion99', \  
 value=(ipp.n\_mes99 \* m99 + b99), allow\_duplicates=True)   
ipp.loc[pd.to\_datetime('2021-01-01'):pd.to\_datetime('2022-07-01'), 'regresion99'] = np.NaN  
#m21: 0.25%  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Regresion21  
ipp.insert(loc=len(ipp.columns), column='n\_mes21', value=np.NaN, \  
 allow\_duplicates=True)  
#Regresión de 2021-12 a 2022-07  
ipp.loc[pd.to\_datetime('2021-01-01'):pd.to\_datetime('2022-07-01'), 'n\_mes21'] = range(0, 19)  
m21, b21 = np.polyfit(\  
 ipp.loc[pd.to\_datetime('2021-01-01'):pd.to\_datetime('2022-07-01'), :].\  
 n\_mes21, ipp.loc[pd.to\_datetime('2021-01-01'):\  
 pd.to\_datetime('2022-07-01'), :].ipp, 1)  
ipp.insert(loc=len(ipp.columns), column='regresion21', \  
 value=(ipp.n\_mes21 \* m21 + b21), allow\_duplicates=True)  
#m21: 2,5%

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.9.6. Proyección hasta 2028-12.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#proyeccion28  
proyeccion28 = pd.DataFrame({'n\_mes28': range(17, 96)}, \  
 index=pd.date\_range(start=('2022-06-01'), end=('2028-12-01'), freq='MS'))  
proyeccion28.insert(loc=len(proyeccion28.columns), column='proyeccion28', \  
 value=proyeccion28.n\_mes28 \* m21 + b21, allow\_duplicates=True)  
ipp = ipp.join(proyeccion28, how='outer')  
ipp\_py = ipp.copy()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 4.9.7. Grafica el IPP.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Ipp mensual real, regresiones y proyección')  
ax1.set\_ylabel ('Índice')  
ax1.set\_xlabel ('Mes')  
ax1.plot(ipp\_py.index, ipp\_py.ipp, 'c-', \  
 label='Ipp real', markersize=2)  
ax1.plot(ipp\_py.index, ipp\_py.regresion99, 'b--', \  
 label='Regresión de Ipp de 1999 a 2020', markersize=2)  
ax1.plot(ipp\_py.index, ipp\_py.regresion21, 'g--', \  
 label='Regresión de Ipp de 2021 a 2022', markersize=2)  
ax1.plot(ipp\_py.index, ipp\_py.proyeccion28, 'r--', \  
 label='Proyección de Ipp hasta 2028', markersize=2)  
plt.legend(loc='best')  
plt.xlim(pd.to\_datetime('1999-01-01'), pd.to\_datetime('2029-06-01'))  
plt.ylim(0, 380)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 4.9.8. Salva y lee el ipp\_py (proyectado).  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# # #Salva ipp\_py  
# ipp\_py.to\_parquet\  
# ('parquet/ipp\_py.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee ipp  
ipp\_py = pd.read\_parquet\  
 ('parquet/ipp\_py.parquet.gzip', engine='fastparquet')