Hace forcast

Luego hace intervalos para forecast

Evalua el error

!pip install skforecast

from skforecast.ForecasterAutoregMultiOutput import ForecasterAutoregMultiOutput

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Lasso

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import make\_pipeline

from skforecast.model\_selection import grid\_search\_forecaster

import numpy as np

import pandas as pd

# Plots

import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight')

plt.rcParams['lines.linewidth'] = 1.5

%matplotlib inline

# Warnings configuration

import warnings

warnings.filterwarnings('ignore')

url = 'https://raw.githubusercontent.com/JoaquinAmatRodrigo/skforecast/master/data/h2o\_exog.csv'

data = pd.read\_csv(url, sep=',')

# Data preparation

data = data.rename(columns={'fecha': 'date'})

data['date'] = pd.to\_datetime(data['date'], format='%Y/%m/%d')

data = data.set\_index('date')

data = data.rename(columns={'x': 'y'})

data = data.asfreq('MS')

data = data.sort\_index()

# Verify that a temporary index is complete

(data.index == pd.date\_range(start=data.index.min(),

                             end=data.index.max(),

                             freq=data.index.freq)).all()

# Fill gaps in a temporary index

# data.asfreq(freq='30min', fill\_value=np.nan)

# Split data into train-test. Last 36 months are for test

steps = 36

data\_train = data[:-steps]  # BLUE

data\_test  = data[-steps:]  # RED

forecaster = ForecasterAutoregMultiOutput(

                    regressor = make\_pipeline(StandardScaler(),

Lasso(random\_state=123)),

                    steps     = 36,

                    lags      = 8

                )

param\_grid = {'lasso\_\_alpha': np.logspace(-5, 5, 10)}

lags\_grid = [5, 12, 20]

results\_grid = grid\_search\_forecaster(

                        forecaster  = forecaster,

                        y           = data\_train['y'],

                        param\_grid  = param\_grid,

                        lags\_grid = lags\_grid,

                        steps       = 36,

                        refit       = True,

                        metric      = 'mean\_squared\_error',

                        initial\_train\_size = int(len(data\_train)\*0.5),

                        return\_best = True,

                        verbose     = False

                    )

predictions = forecaster.predict()

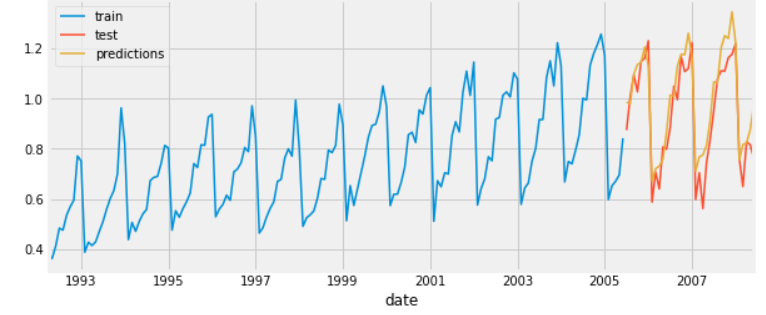
fig, ax = plt.subplots(figsize=(9, 4))

data\_train['y'].plot(ax=ax, label='train')

data\_test['y'].plot(ax=ax, label='test')

predictions.plot(ax=ax, label='predictions')

ax.legend();



error\_mse = mean\_squared\_error(y\_true = data\_test['y'], y\_pred = predictions)

print(f"Test error (mse) {error\_mse}")

Test error (mse) 0.009

# Create and train forecaster

from skforecast.ForecasterAutoreg import ForecasterAutoreg

forecaster = ForecasterAutoreg(

                    regressor = LinearRegression(),

                    lags = 15

                )

forecaster.fit(y=data\_train['y'])

# Prediction intervals

predictions = forecaster.predict\_interval(

                    steps    = steps,

                    interval = [1, 99],

                    n\_boot   = 500

              )

fig, ax=plt.subplots(figsize=(9, 4))

data\_test['y'].plot(ax=ax, label='test')

predictions['pred'].plot(ax=ax, label='prediction')

ax.fill\_between(

    predictions.index,

    predictions['lower\_bound'],

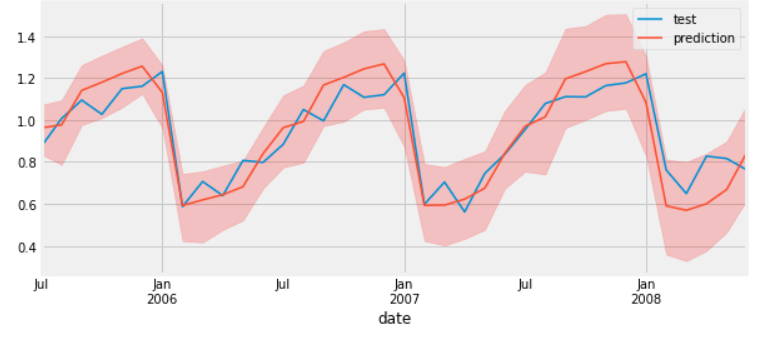
    predictions['upper\_bound'],

    color = 'red',

    alpha = 0.2

)

ax.legend();



El intervalo solo es para el área de la que se hizo el forecast.

# Backtest with prediction intervals

# ==============================================================================

n\_val = 36\*3

data\_train = data[:-n\_val]

data\_val  = data[-n\_val:]

steps = 36

forecaster = ForecasterAutoreg(

                regressor = make\_pipeline(StandardScaler(), Lasso(alpha=0.0215)),

                lags = 15

            )

from skforecast.model\_selection import backtesting\_forecaster

metric, predictions = backtesting\_forecaster(

                            forecaster         = forecaster,

                            y                  = data['y'],

                            initial\_train\_size = len(data\_train),

                            steps              = steps,

                            metric             = 'mean\_squared\_error',

                            refit              = True,

                            interval           = [1, 99],

                            n\_boot             = 100,

                            verbose            = True

                       )

print(f"Test error (mse): {error\_mse}")

# Plot

# ==============================================================================

fig, ax=plt.subplots(figsize=(9, 4))

data\_val['y'].plot(ax=ax, label='test')

predictions['pred'].plot(ax=ax, label='predicciones')

ax.fill\_between(

    predictions.index,

    predictions['lower\_bound'],

    predictions['upper\_bound'],

    color = 'red',

    alpha = 0.2

)

ax.legend();

