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')

!pip install skforecast

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.ForecasterAutoreg import ForecasterAutoreg

from skforecast.ForecasterAutoregCustom import ForecasterAutoregCustom

from skforecast.ForecasterAutoregMultiOutput import ForecasterAutoregMultiOutput

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

from skforecast.model\_selection import backtesting\_forecaster

from joblib import dump, load

# Data download

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

regressor = RandomForestRegressor(max\_depth=3, n\_estimators=100, random\_state=123)

forecaster = ForecasterAutoreg(

                regressor = regressor,

                lags      = 20

             )

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

predictions = forecaster.predict(steps=steps)

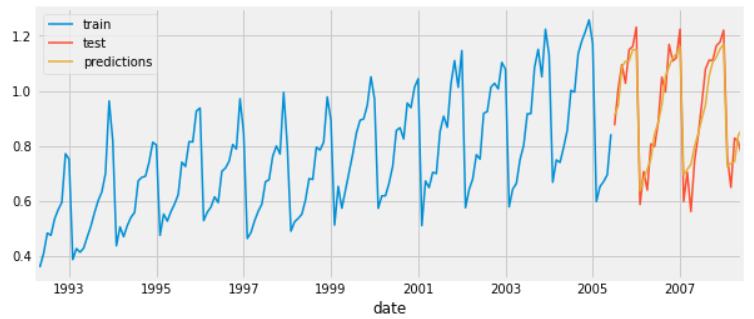
fig, ax = plt.subplots(figsize=(9, 4))

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): 0.00439269