Лабораторная работа 6

Анализ и прогнозирование временного ряда.

Цель лабораторной работы: изучение основных методов анализа и прогнозирование временных рядов.

Задание: Выберите набор данных (датасет) для решения задачи прогнозирования временного ряда. Визуализируйте временной ряд и его основные характеристики. Разделите временной ряд на обучающую и тестовую выборку. Произведите прогнозирование временного ряда с использованием как минимум двух методов. Визуализируйте тестовую выборку и каждый из прогнозов. Оцените качество прогноза в каждом случае с помощью метрик.

In [11]:
```
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

# Modeling and Forecasting
# ==============================================================================
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 sklearn.preprocessing import MinMaxScaler, StandardScaler

from skforecast.ForecasterAutoreg import ForecasterAutoreg
from skforecast.ForecasterAutoregCustom import ForecasterAutoregCustom
from skforecast.ForecasterAutoregMultiOutput import ForecasterAutoregMul
from skforecast.model_selection import grid_search_forecaster
from skforecast.model_selection import backtesting_forecaster

from joblib import dump, load

# Warnings configuration
# ==============================================================================
import warnings
warnings.filterwarnings('ignore')
```

In [21]:
```
data = pd.read_csv('1979-2021.csv', sep=',')
data
```

Out[21]:

	Date	United States(USD)	Europe(EUR)	Japan(JPY)	United Kingdom(GBP)	Canada(CAD)	Switzerlan
0	31-	233.7	144.8	45160.3	117.4	267.1	

	01-1979					
1	28-02-1979	251.3	154.6	50209.1	124.2	295.5
2	30-03-1979	240.1	148.0	50274.3	116.2	278.2
3	30-04-1979	245.3	152.8	54144.6	118.8	278.5
4	31-05-1979	274.6	172.0	61057.1	132.7	321.6
...	...	...	...	...	...	...
506	31-03-2021	1691.1	1438.8	186861.0	1225.7	2125.4
507	30-04-2021	1767.7	1468.4	193213.0	1276.7	2174.6
508	31-05-2021	1900.0	1554.0	207845.0	1336.6	2295.3
509	30-06-2021	1763.2	1486.8	195692.0	1276.3	2183.3
510	30-07-2021	1825.8	1539.7	200376.1	1313.2	2279.2

511 rows × 19 columns

Предварительная обработка

Удаляем все столбцы, кроме даты и USD:

In [22]:
```
data = data[['Date','United States(USD)']]
for i, row in data.iterrows():
    data.at[i, 'Date'] = '01'+row['Date'][2:]
data
```

Out[22]:

	Date	United States(USD)
0	01-01-1979	233.7
1	01-02-1979	251.3
2	01-03-1979	240.1
3	01-04-1979	245.3
4	01-05-1979	274.6
...	...	...
506	01-03-2021	1691.1
507	01-04-2021	1767.7
508	01-05-2021	1900.0

509	01-06-2021	1763.2
510	01-07-2021	1825.8

511 rows × 2 columns

In [23]:
```
data = data.rename(columns={'Date': 'date'})
data = data.rename(columns={'United States(USD)': 'y'})
data['date'] = pd.to_datetime(data['date'], format='%d-%m-%Y')
data = data.set_index('date')
data = data.asfreq(freq ='MS')
#data = data.sort_index()
data
```

Out[23]:

	y
date	
1979-01-01	233.7
1979-02-01	251.3
1979-03-01	240.1
1979-04-01	245.3
1979-05-01	274.6
...	...
2021-03-01	1691.1
2021-04-01	1767.7
2021-05-01	1900.0
2021-06-01	1763.2
2021-07-01	1825.8

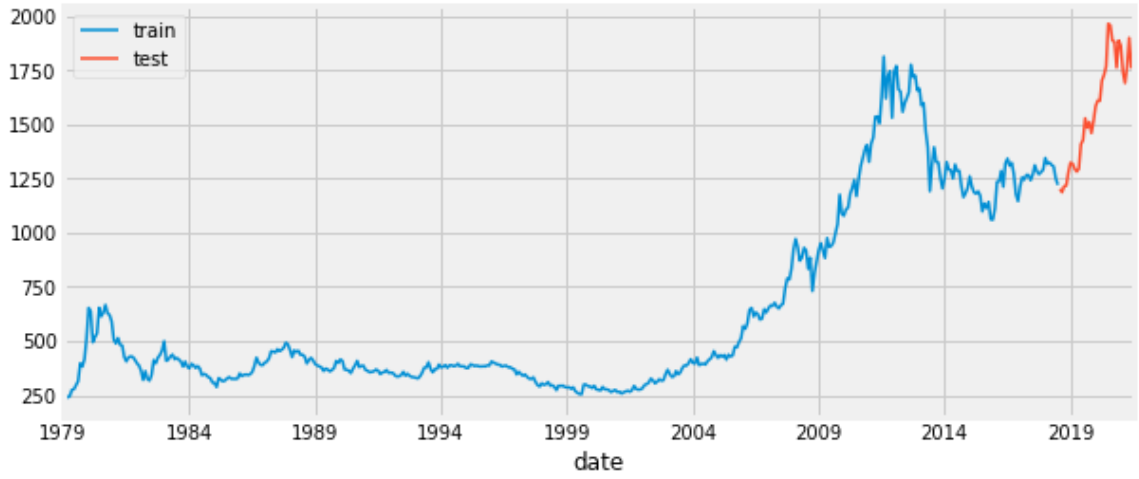
511 rows × 1 columns

Разделение выборки на обучающую и тестовую

In [24]:
```
import matplotlib.pyplot as plt
steps = 36
# scaler = MinMaxScaler().fit(data_train[['open']])

# data['open'] = scaler.transform(data[['open']])
data_train = data[:-steps]
data_test  = data[-steps:]
print(data)


print(f"Train dates : {data_train.index.min()} --- {data_train.index.max
print(f"Test dates  : {data_test.index.min()} --- {data_test.index.max()

fig, ax=plt.subplots(figsize=(9, 4))
data_train['y'].plot(ax=ax, label='train')
data_test['y'].plot(ax=ax, label='test')
ax.legend();
```

y

```
date
1979-01-01   233.7
1979-02-01   251.3
1979-03-01   240.1
1979-04-01   245.3
1979-05-01   274.6
...            ...
2021-03-01  1691.1
2021-04-01  1767.7
2021-05-01  1900.0
2021-06-01  1763.2
2021-07-01  1825.8

[511 rows x 1 columns]
Train dates : 1979-01-01 00:00:00 --- 2018-07-01 00:00:00  (n=475)
Test dates  : 2018-08-01 00:00:00 --- 2021-07-01 00:00:00  (n=36)
```



In [25]:
```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import median_absolute_error, r2_score

def test_model(predictions):
    print('mean_absolute_error: {}'.format(round(mean_absolute_error(data_test['y'], predictions), 2)))
    print('median_absolute_error: {}'.format(round(median_absolute_error(data_test['y'], predictions), 2)))
    print('r2_score: {}'.format(round(r2_score(data_test['y'], predictions), 2)))
```

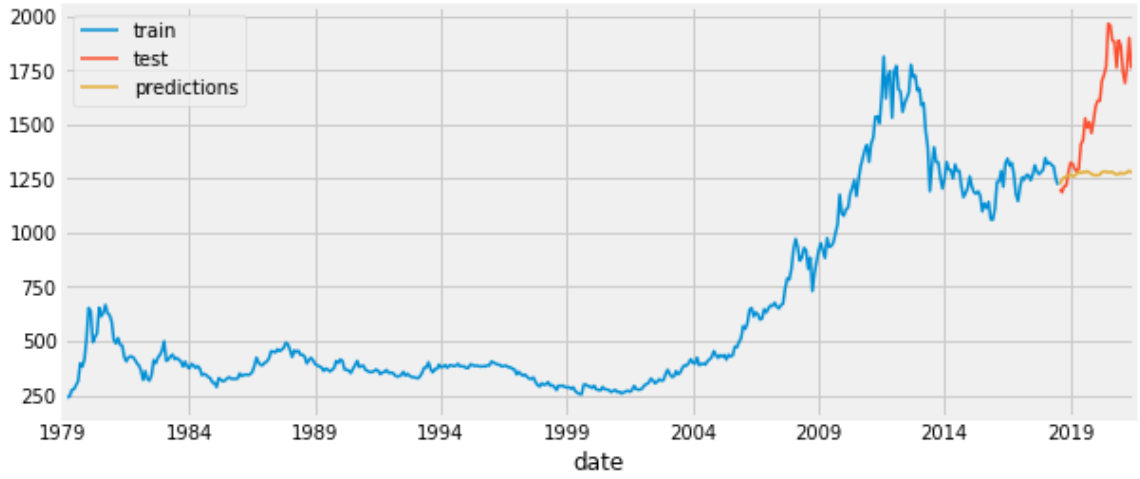
Обучение моделей

Skforecast-ForecasterAutoreg

In [26]:
```
forecaster = ForecasterAutoreg(
                regressor = RandomForestRegressor(random_state=123),
                lags = 6
                )

forecaster.fit(y=data_train['y'])
forecaster
steps = 36
predictions = forecaster.predict(steps=steps)
```

In [27]:
```
fig, ax2 = plt.subplots(figsize=(9, 4))
data_train['y'].plot(ax=ax2, label='train')
data_test['y'].plot(ax=ax2, label='test')
```

```
predictions.plot(ax=ax2, label='predictions')
ax2.legend();
```



In [28]:

```
lags_grid = [10, 16]

# Regressor's hyperparameters
param_grid = {'n_estimators': [100, 500],
              'max_depth': [3, 5, 10]}

results_grid = grid_search_forecaster(
                        forecaster         = forecaster,
                        y                  = data_train['y'],
                        param_grid         = param_grid,
                        lags_grid          = lags_grid,
                        steps              = steps,
                        refit              = True,
                        metric             = 'mean_squared_error',
                        initial_train_size = int(len(data_train)*0.5),
                        fixed_train_size   = False,
                        return_best        = True,
                        verbose            = False
               )
```

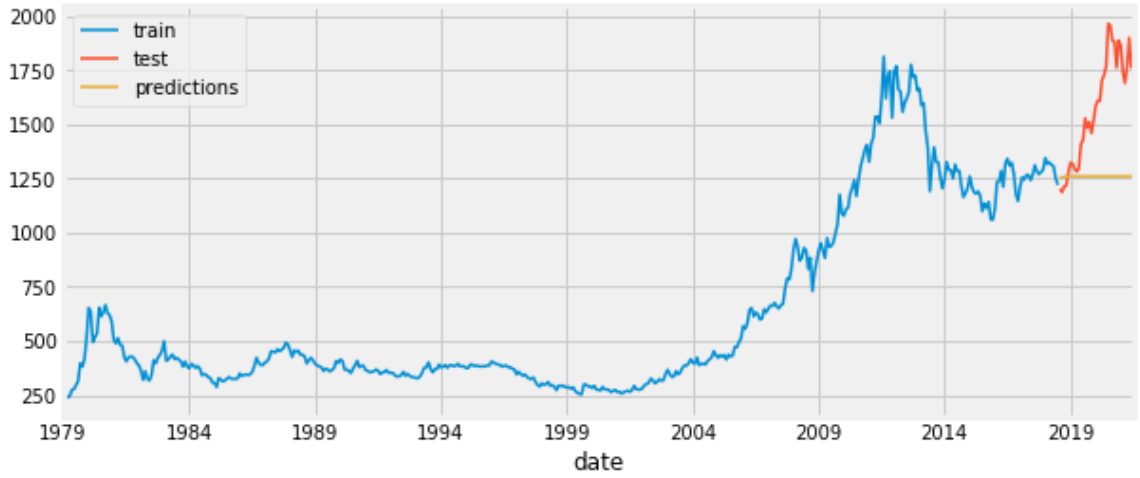
```
Number of models compared: 12
loop lags_grid:   0%|                                                    | 0/2
[00:00<?, ?it/s]
loop param_grid:   0%|                                                   | 0/6
[00:00<?, ?it/s]
loop param_grid:  17%|██████                                  | 1/6 [00:03
<00:18,  3.60s/it]
loop param_grid:  33%|████████████                            | 2/6 [00:18
<00:41, 10.42s/it]
loop param_grid:  50%|██████████████████                      | 3/6 [00:22<
00:21,  7.24s/it]
loop param_grid:  67%|████████████████████████                | 4/6 [00:39
<00:22, 11.11s/it]
loop param_grid:  83%|██████████████████████████████          | 5/6 [00:43
<00:08,  8.47s/it]
loop param_grid: 100%|████████████████████████████████████| 6/6 [01:02<
00:00, 12.14s/it]
loop lags_grid:  50%|██████████████████                      | 1/2 [01:02<
01:02, 62.37s/it]
loop param_grid:   0%|                                                   | 0/6
[00:00<?, ?it/s]
loop param_grid:  17%|██████                                  | 1/6 [00:03
```

```
<00:17,  3.47s/it]
loop param_grid:  33%|███████████                       | 2/6 [00:19
<00:43, 10.80s/it]
loop param_grid:  50%|████████████████                  | 3/6 [00:22<
00:22,  7.49s/it]
loop param_grid:  67%|█████████████████████             | 4/6 [00:40
<00:23, 11.56s/it]
loop param_grid:  83%|██████████████████████████        | 5/6 [00:44
<00:08,  8.86s/it]
loop param_grid: 100%|████████████████████████████████| 6/6 [01:06<
00:00, 13.09s/it]
loop lags_grid: 100%|████████████████████████████████| 2/2 [02:08<
00:00, 64.25s/it]
`Forecaster` refitted using the best-found lags and parameters, and the w
hole data set:
  Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
  Parameters: {'max_depth': 5, 'n_estimators': 500}
  Backtesting metric: 44788.07209342369
```

In [29]:

```
regressor = RandomForestRegressor(max_depth=5, n_estimators=500, random_
forecaster = ForecasterAutoreg(
                regressor = regressor,
                lags      = 16
             )

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();
test_model(predictions)
```

```
mean_absolute_error: 330.55
median_absolute_error: 336.26
r2_score: -1.7
```



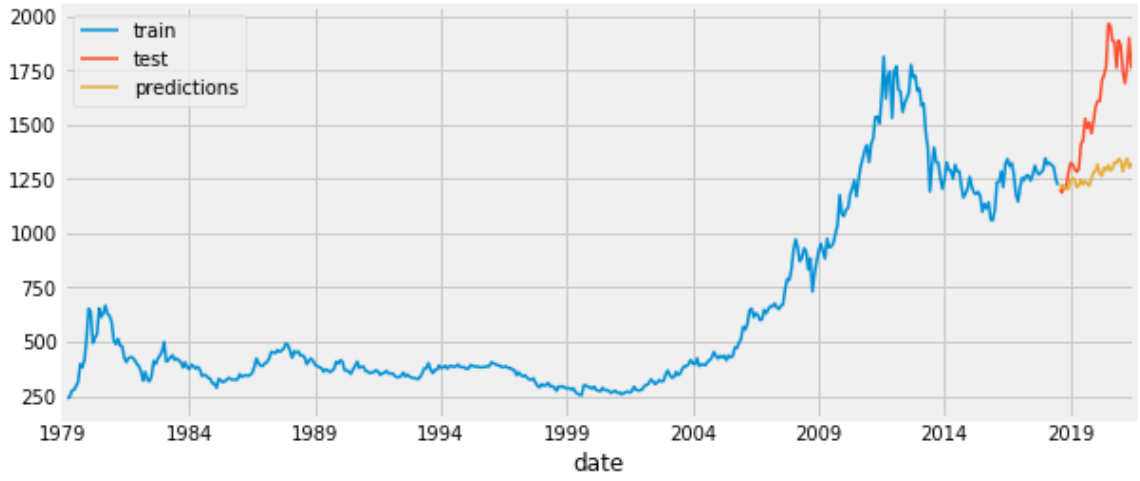
SARIMAX

In [37]:

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

SARIMAXmodel = SARIMAX(data_train['y'], order = (3, 1, 3), seasonal_orde
SARIMAXmodel = SARIMAXmodel.fit()
```

```
y_pred = SARIMAXmodel.get_forecast(len(data_test.index))
y_pred_df = y_pred.conf_int(alpha = 0.05)
y_pred_df["Predictions"] = SARIMAXmodel.predict(start = y_pred_df.index[
y_pred_df.index = data_test.index
y_pred_out = y_pred_df["Predictions"]
fig, ax2 = plt.subplots(figsize=(9, 4))
data_train['y'].plot(ax=ax2, label='train')
data_test['y'].plot(ax=ax2, label='test')
y_pred_out.plot(ax=ax2, label='predictions')
ax2.legend()
test_model(y_pred_out)
```

```
C:\Users\pstri\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9
_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\statsmode
ls\base\model.py:604: ConvergenceWarning: Maximum Likelihood optimization
failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
mean_absolute_error: 310.44
median_absolute_error: 295.07
r2_score: -1.31
```



In [ ]: