Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс   
«Технологии машинного обучения»  
 Отчет по лабораторной работе №6

«Разведочный анализ данных. Исследование и визуализация данных.»

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент группы ИУ5-63Б | преподаватель каф. ИУ5 |
| Тарновский Д.Р. | Гапанюк Ю.Е. |

Москва, 2022 г.

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

### 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]: | **date** | **y** |
|  | **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

In [24]:

# Разделение выборки на обучающую и тестовую

y

**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();

|  |  |
| --- | --- |
| 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 |

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(dat print('median\_absolute\_error: {}'**.**format(round(median\_absolute\_error print('r2\_score: {}'**.**format(round(r2\_score(data\_test['y'], predictio

[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 [26]:

# Обучение моделей

## Skforecast-ForecasterAutoreg

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%|██████████████████████████████████████|  00:00, 12.14s/it] | 6/6 | [01:02< |
| loop lags\_grid: 50%|███████████████████▌ | 01:02, 62.37s/it]  loop param\_grid: 0%| [00:00<?, ?it/s]  loop param\_grid: 17%|██████▎ | | 1/2  1/6 | [01:02<  | 0/6 [00:03 |

<00:17, 3.47s/it]

loop param\_grid: 33%|████████████▋

<00:43, 10.80s/it]

loop param\_grid: 50%|███████████████████ 00:22, 7.49s/it]

loop param\_grid: 67%|█████████████████████████▎

<00:23, 11.56s/it]

loop param\_grid: 83%|███████████████████████████████▋

<00:08, 8.86s/it]

| 2/6 [00:19

| 3/6 [00:22<

| 4/6 [00:40

| 5/6 [00:44

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'])

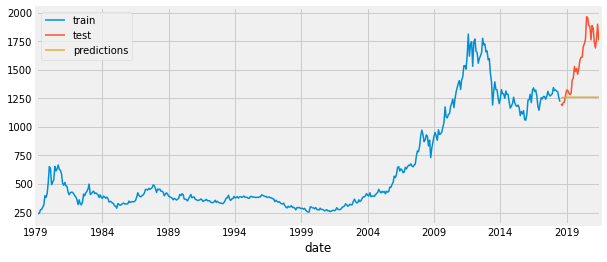
predictions **=** forecaster**.**predict(steps**=**steps) fig, ax **=** plt**.**subplots(figsize**=**(9, 4)) data\_train['y']**.**plot(ax**=**ax, label**=**'train') data\_test['y']**.**plot(ax**=**ax, label**=**'test') predictions**.**plot(ax**=**ax, label**=**'predictions') ax**.**legend();

test\_model(predictions)

mean\_absolute\_error: 330.55

median\_absolute\_error: 336.26

r2\_score: -1.7



In [37]:

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

SARIMAXmodel **=** SARIMAX(data\_train['y'], order **=** (3, 1, 3), seasonal\_orde SARIMAXmodel **=** SARIMAXmodel**.**fit()

## SARIMAX

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 "

In [ ]:

mean\_absolute\_error: 310.44

median\_absolute\_error: 295.07

r2\_score: -1.31

