Задание лабораторной работы

- Выбрать набор данных (датасет) для решения задачи прогнозирования временного ряда.
- Визуализировать временной ряд и его основные характеристики.
- Разделить временной ряд на обучающую и тестовую выборку.
- Произвести прогнозирование временного ряда с использованием как минимум двух методов.
- Визуализировать тестовую выборку и каждый из прогнозов.
- Оценить качество прогноза в каждом случае с помощью метрик.

Ячейки Jupyter-ноутбука

Выбор и загрузка данных

Текстовое описание

В качестве датасета для решения задачи прогнозирования временного ряда будем использовать набор данных, содержащий ежедневные климатические данные в городе Дели с 2013 по 2017 год. Данный набор доступен по адресу: https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data

Набор данных имеет следующие атрибуты:

- date - Дата - метка времени
- meantemp - Средняя температура - средняя температура, расчитанная по нескольким 3-часовым интервалам в день
- humidity - Влажность - показатель влажности в граммах воды на кубический метр воздуха
- wind_speed - Скорость ветра - скорость ветра в километрах в час
- meanpressure - Среднее давление - среднее давление в атмосферах

Импорт библиотек

Импортируем библиотеки с помощью команды import:

In [8]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot
import matplotlib.pyplot as plt
```

Уберем предупреждения:

In [9]:
```
import warnings
warnings.filterwarnings('ignore')
```

Загрузка данных

Выборка уже разделена. Для первичного анализа объединим тестовую и обучающую выборку:

In [12]:
```
data_test = pd.read_csv('DailyDelhiClimateTest.csv', header=0, parse_dates=['date'], index_col='date')
data_train = pd.read_csv('DailyDelhiClimateTrain.csv', header=0, parse_dates=['date'], index_col='date')
data = pd.concat([data_train, data_test], axis=0)
```

Первичная обработка данных и визуализация

Первичный анализ

Выведем первые 5 строк датасета:

In [13]:
```
data.head()
```

Out[13]:

	meantemp	humidity	wind_speed	meanpressure
date				
2013-01-01	10.000000	84.500000	0.000000	1015.666667
2013-01-02	7.400000	92.000000	2.980000	1017.800000
2013-01-03	7.166667	87.000000	4.633333	1018.666667
2013-01-04	8.666667	71.333333	1.233333	1017.166667
2013-01-05	6.000000	86.833333	3.700000	1016.500000

Определим размер датасета:

In [14]:
```
data.shape
```

Out[14]:(1576, 4)

Определим типы данных:

In [15]:
```
data.dtypes
```

Out[15]:
```
meantemp        float64
humidity        float64
wind_speed      float64
meanpressure    float64
dtype: object
```

Обработка данных

Оставим только столбец влажности для временного ряда:

In [16]:
```
data = data.drop(columns=['meantemp'], axis=1)
data = data.drop(columns=['wind_speed'], axis=1)
data = data.drop(columns=['meanpressure'], axis=1)
```

In [17]:
```
data.head()
```

Out[17]:

	humidity
date	
2013-01-01	84.500000
2013-01-02	92.000000
2013-01-03	87.000000
2013-01-04	71.333333
2013-01-05	86.833333

Основные статистические характеристки

Определим основные статистические характеристки временного ряда:

In [18]:
```
data.describe()
```

Out[18]:

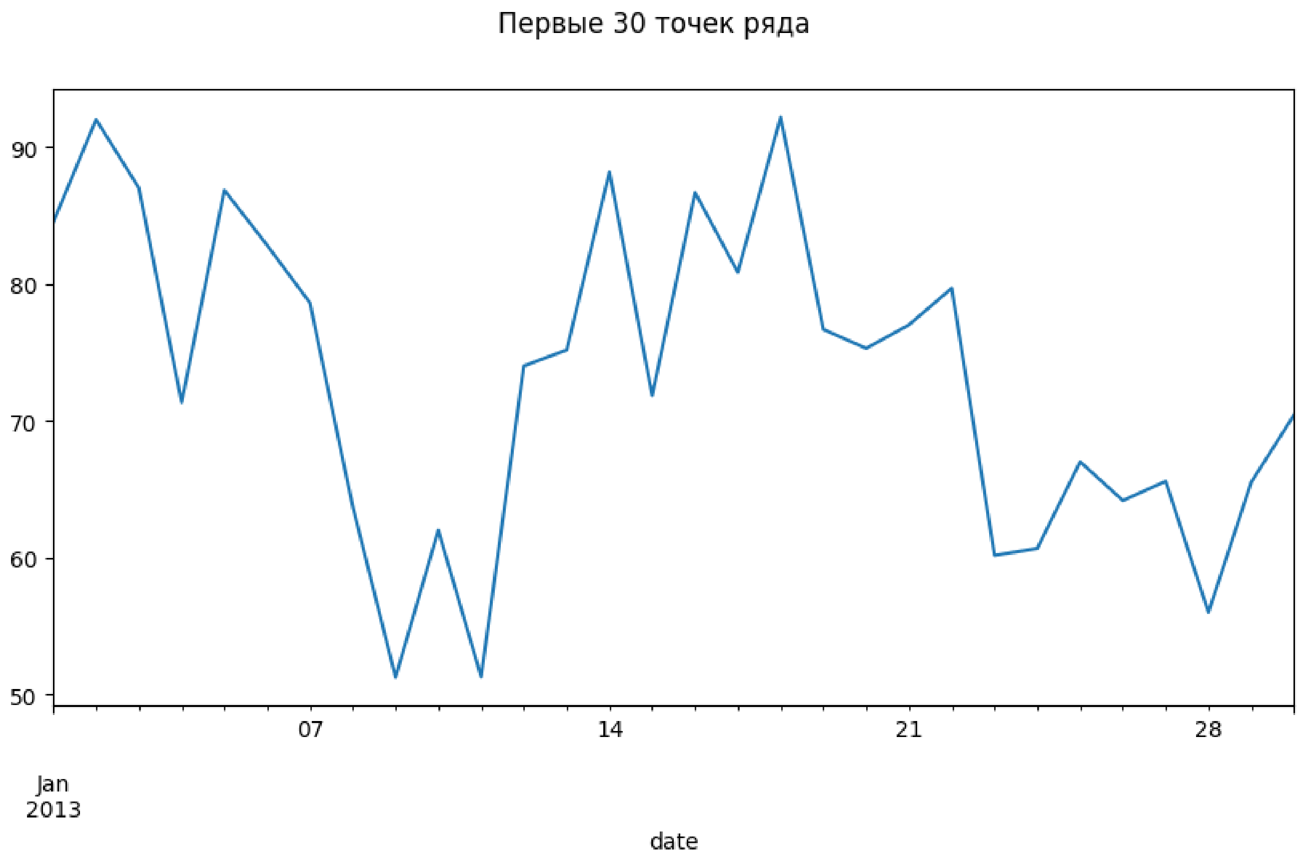
	humidity
count	1576.000000
mean	60.445229
std	16.979994
min	13.428571
25%	49.750000
50%	62.440476
75%	72.125000
max	100.000000

Визуализация исходного временного ряда

В виде графика:

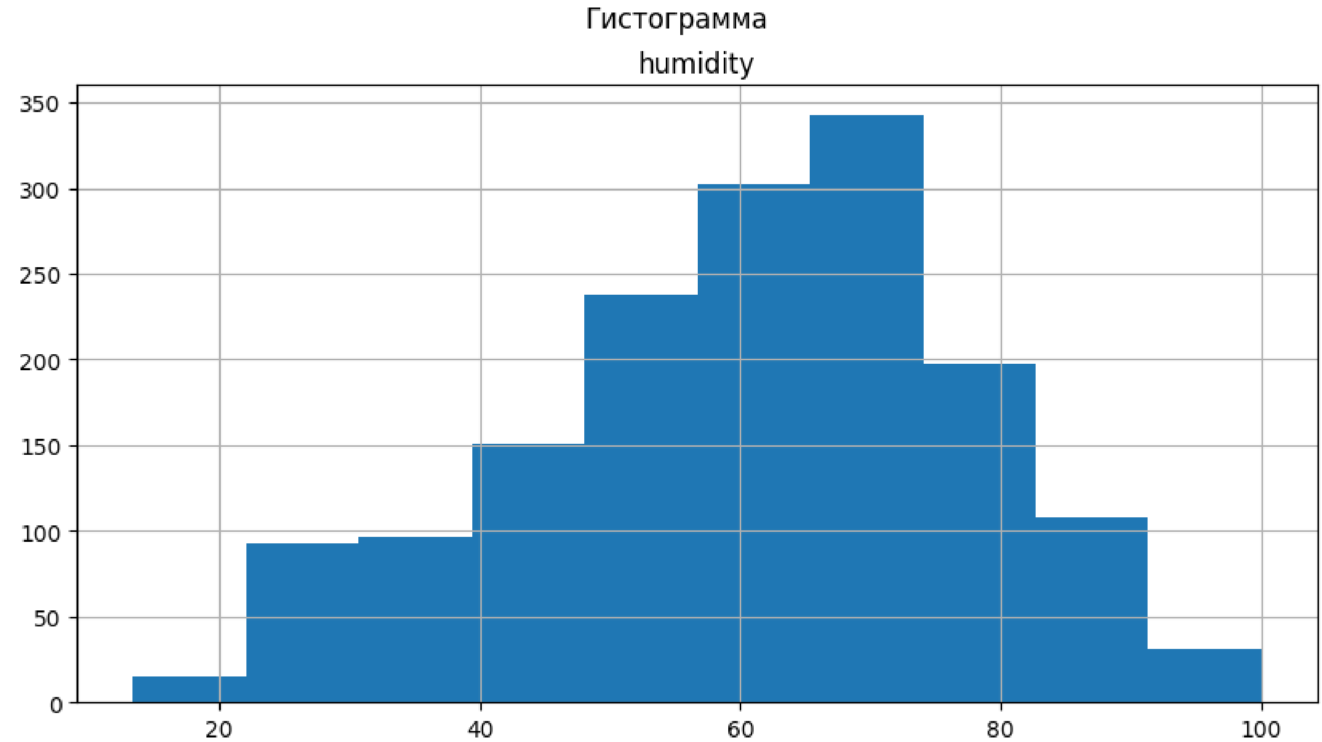
In [19]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Временной ряд в виде графика')
data.plot(ax=ax, legend=False)
pyplot.show()
```


```
In [20]:fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
         fig.suptitle('Первые 30 точек ряда')
         data[:30].plot(ax=ax, legend=False)
         pyplot.show()
```



В виде гистограммы:

```
In [21]:fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
         fig.suptitle('Гистограмма')
         data.hist(ax=ax, legend=False)
         pyplot.show()
```

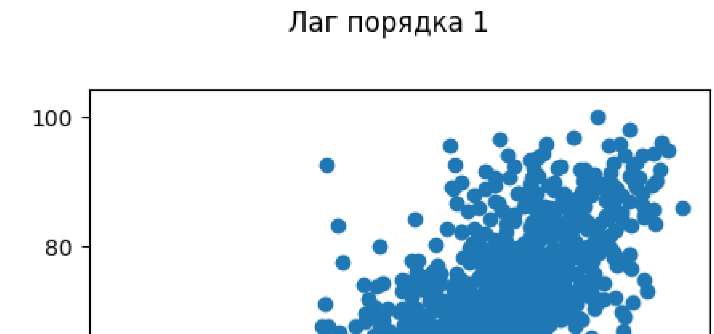


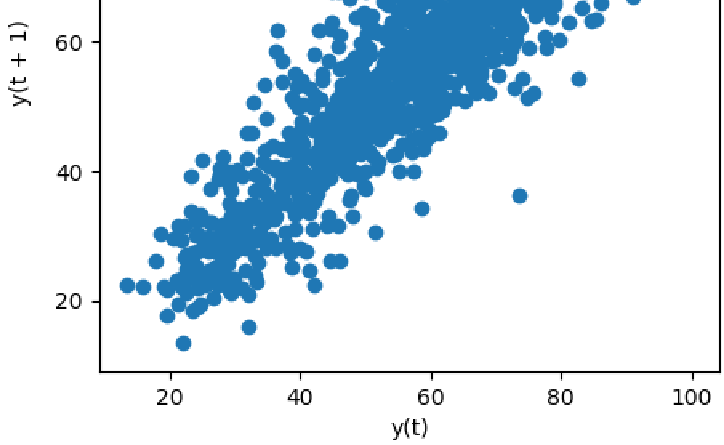
Вероятностная плотность распределения данных:

In [22]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Плотность вероятности распределения данных')
data.plot(ax=ax, kind='kde', legend=False)
pyplot.show()
```

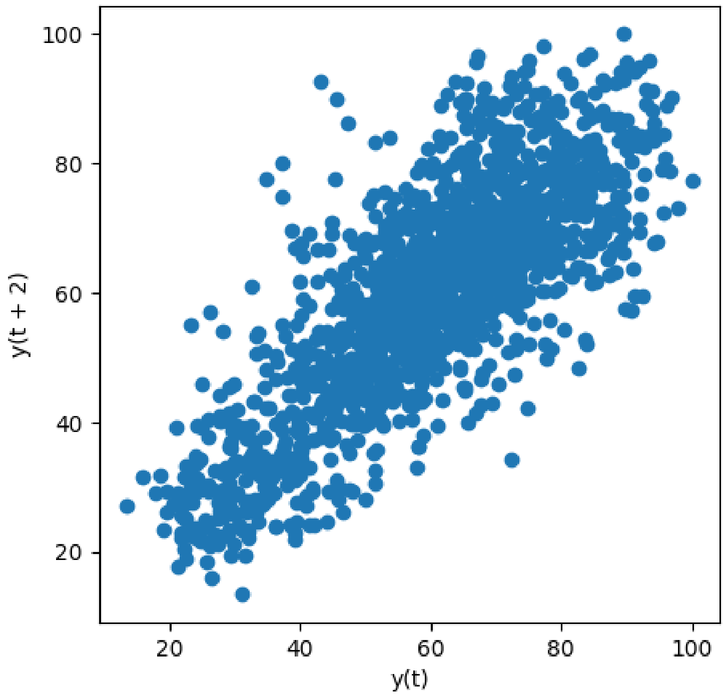
С помощью Lag Plot:

In [23]:
```
for i in range(1, 5):
    fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(5,5))
    fig.suptitle(f'Лаг порядка {i}')
    pd.plotting.lag_plot(data, lag=i, ax=ax)
    pyplot.show()
```

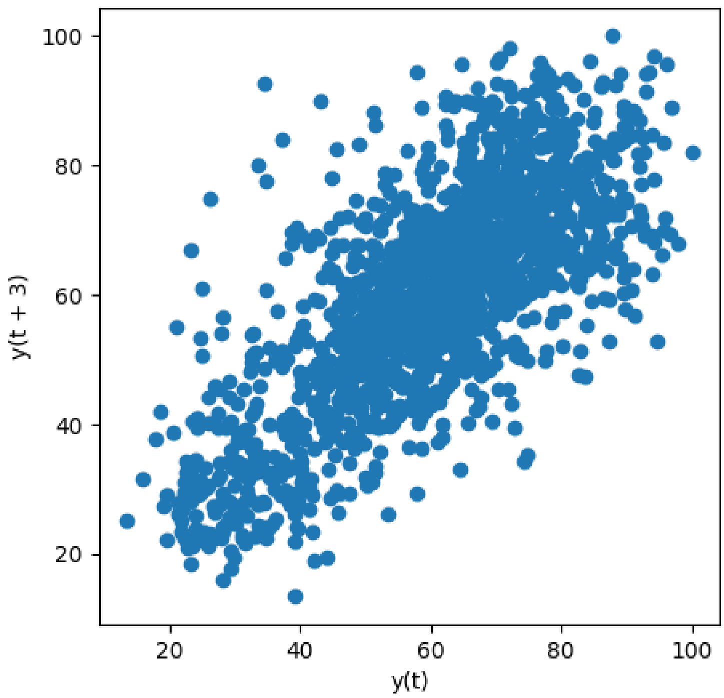




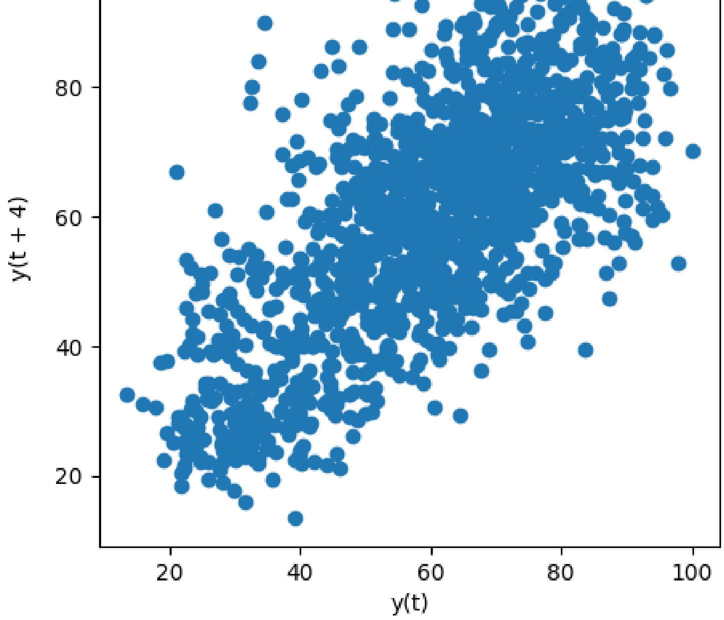
Лаг порядка 2



Лаг порядка 3



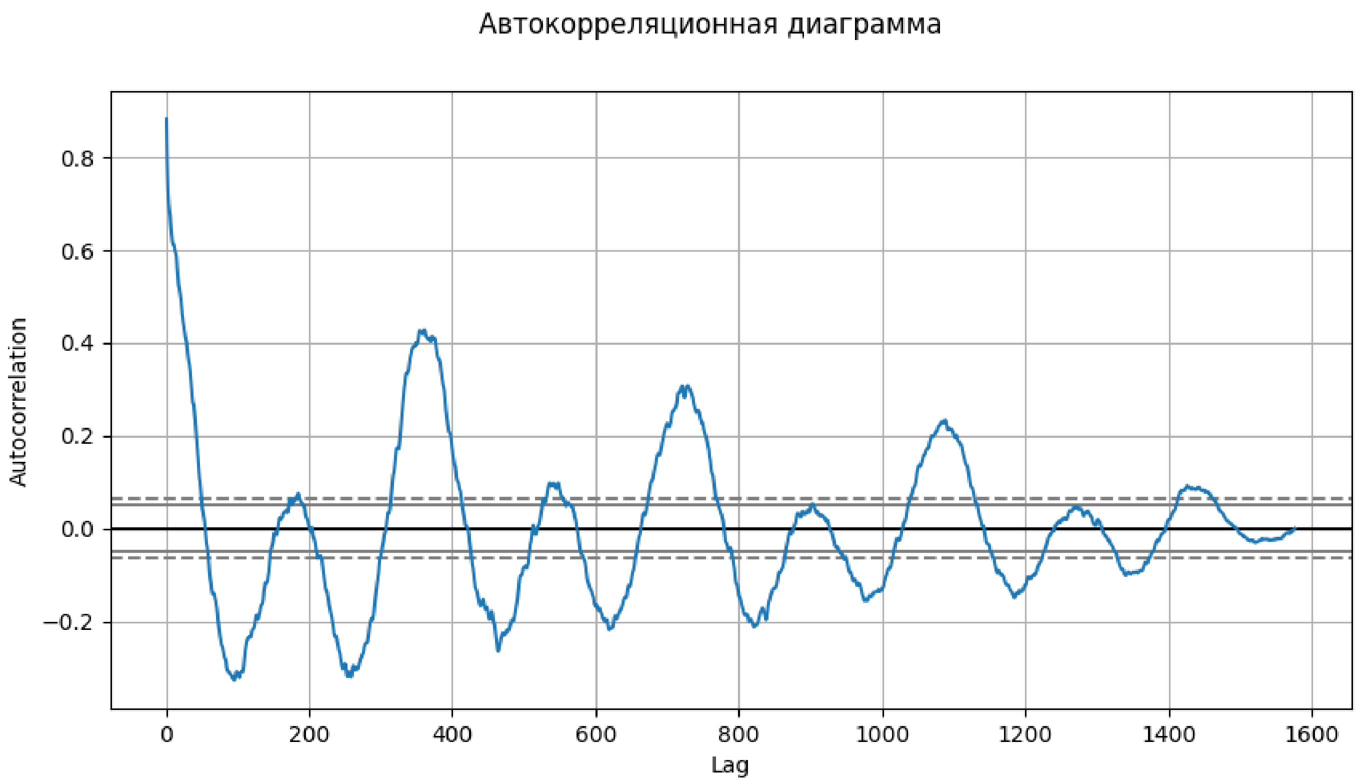
Лаг порядка 4



Наблюдается достаточно сильная положительная корреляция.

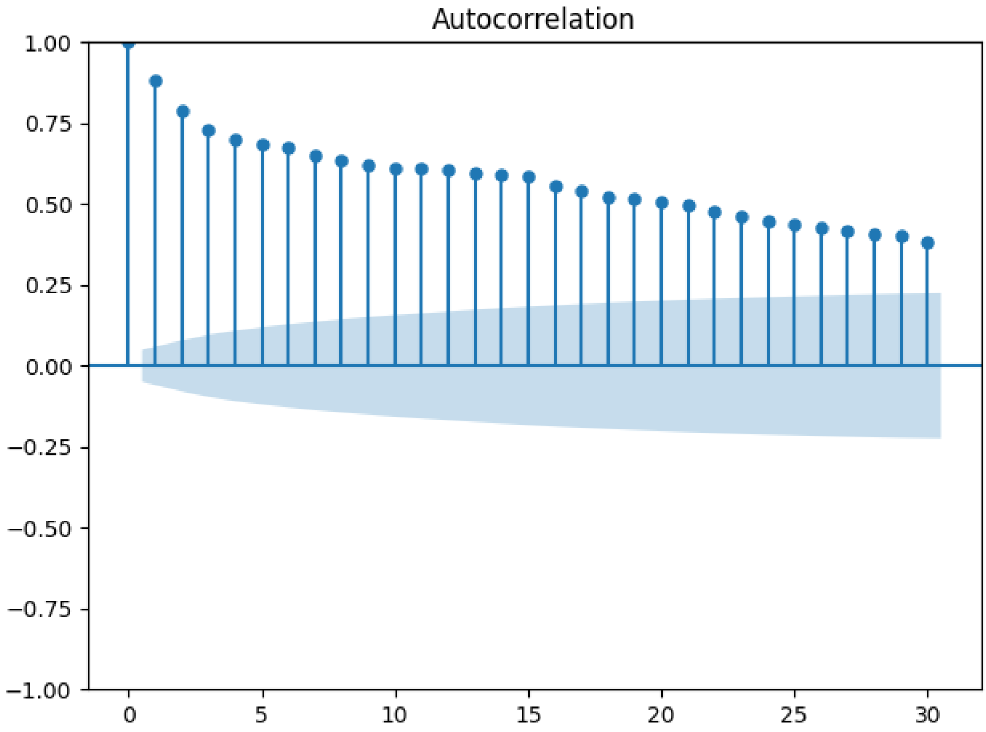
Автокорреляционная диаграмма:

In [24]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Автокорреляционная диаграмма')
pd.plotting.autocorrelation_plot(data, ax=ax)
pyplot.show()
```



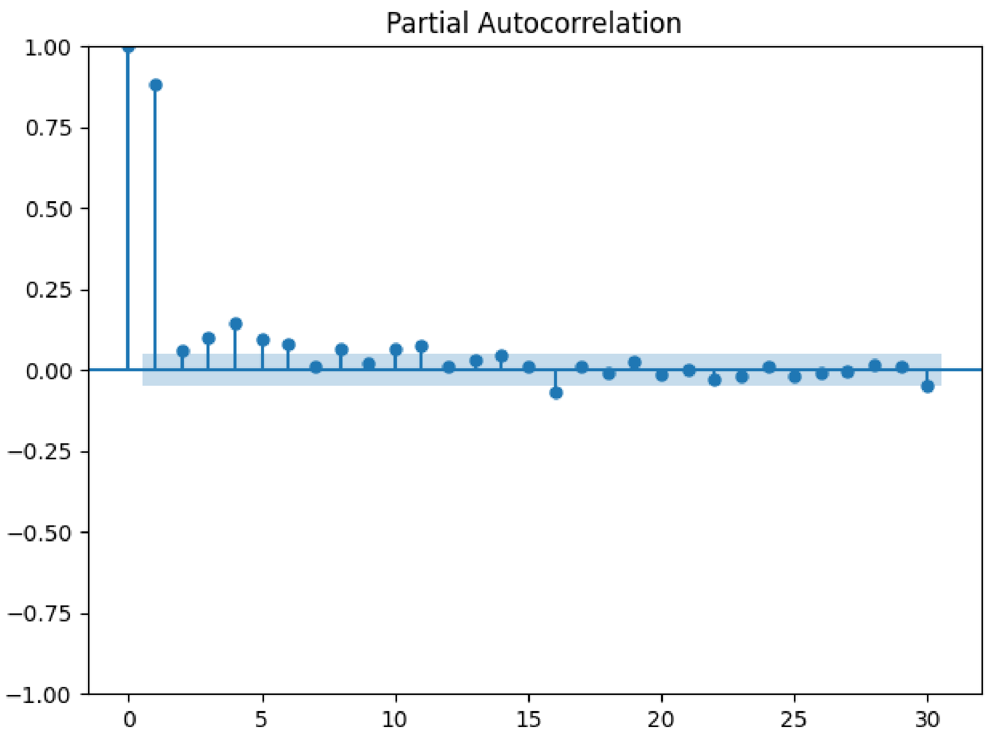
Автокорреляционная функция:

In [25]:
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(data, lags=30)
plt.tight_layout()
```



Частичная автокорреляционная функция:

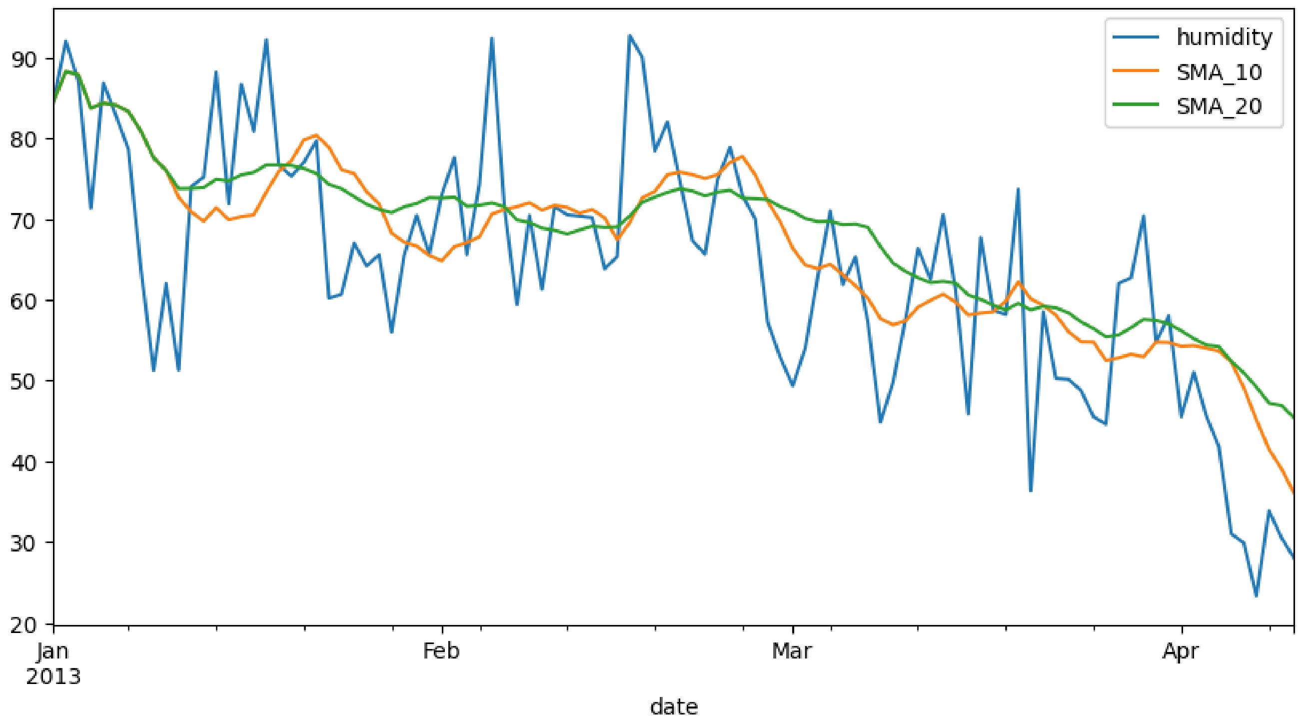
```
In [26]:from statsmodels.graphics.tsaplots import plot_pacf
        plot_pacf(data, lags=30)
        plt.tight_layout()
```



Временной ряд со скользящими средними:

```
In [27]:data2 = data.copy()
In [28]:data2['SMA_10'] = data2['humidity'].rolling(10, min_periods=1).mean()
        data2['SMA_20'] = data2['humidity'].rolling(20, min_periods=1).mean()
In [29]:fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
        fig.suptitle('Временной ряд со скользящими средними')
        data2[:100].plot(ax=ax, legend=True)
        pyplot.show()
```

Временной ряд со скользящими средними



Прогнозирование временного ряда с использованием авторегрессионного метода

Будем использовать авторегриссионный метод ARIMA:

In [30]:
```
from statsmodels.tsa.arima.model import ARIMA
```

Разделение выборки на обучающую и тестовую

In [31]:
```
xnum = list(range(data2.shape[0]))
Y = data2['humidity'].values
train_size = int(len(Y) * 0.7)
xnum_train, xnum_test = xnum[0:train_size], xnum[train_size:]
train, test = Y[0:train_size], Y[train_size:]
history_arima = [x for x in train]
```

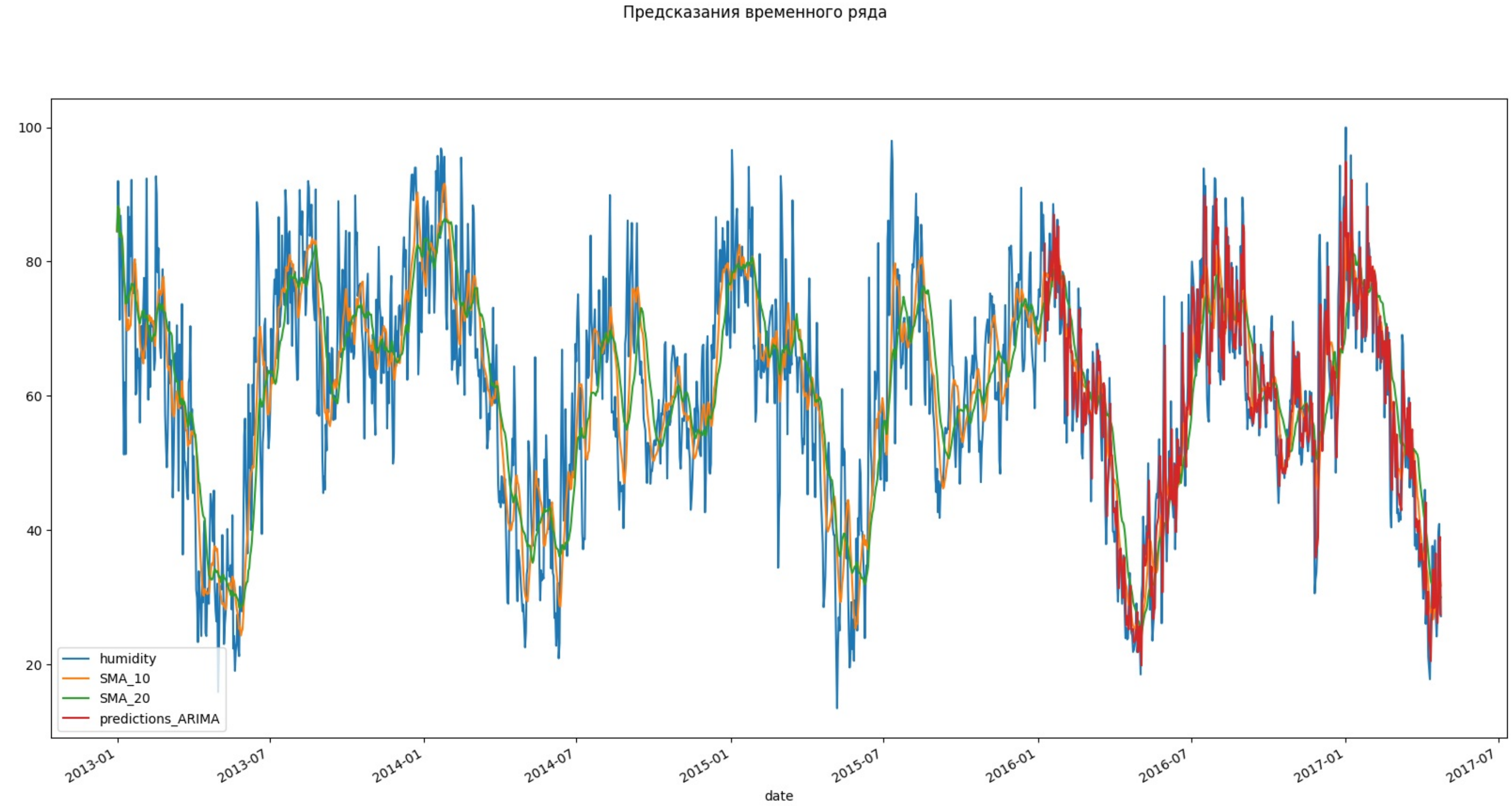
Прогноз ARIMA

In [35]:
```
arima_order = (6, 1, 0)
predictions_arima = list()
for t in range(len(test)):
    model_arima = ARIMA(history_arima, order=arima_order)
    model_arima_fit = model_arima.fit()
    yhat_arima = model_arima_fit.forecast()[0]
    predictions_arima.append(yhat_arima)
    history_arima.append(test[t])
```

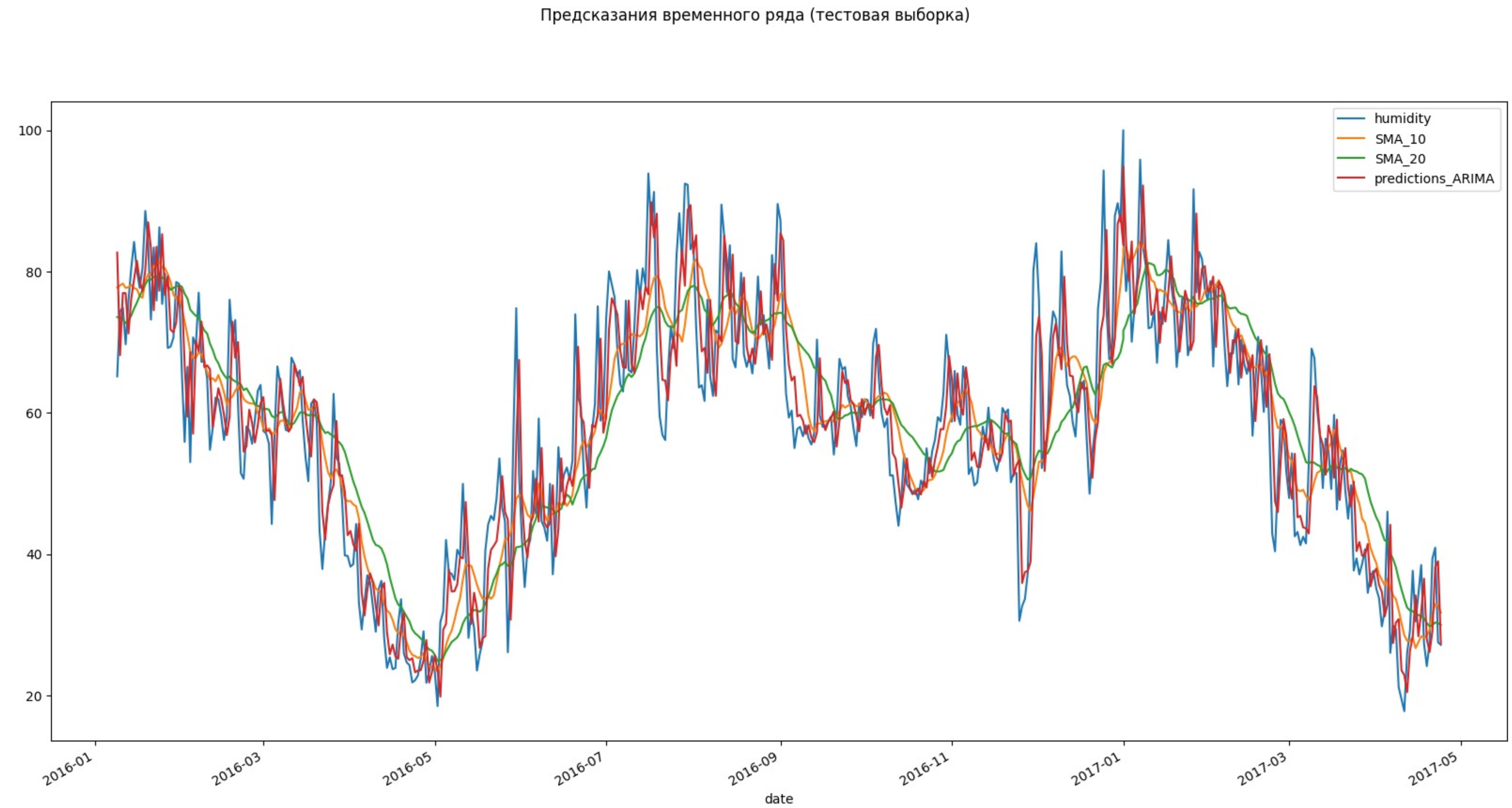
In [33]:
```
data2['predictions_ARIMA'] = (train_size * [np.NAN]) + list(predictions_arima)
```

Визуализация

In [34]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(20,10))
fig.suptitle('Предсказания временного ряда')
data2.plot(ax=ax, legend=True)
pyplot.show()
```




```
In [36]:fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(20,10))
        fig.suptitle('Предсказания временного ряда (тестовая выборка)')
        data2[train_size:].plot(ax=ax, legend=True)
        pyplot.show()
```



Предсказания ARIMA точны, близки к исходному, далеки от среднего скользящего.

Метрики

MAE и MSE:

```
In [37]:from sklearn.metrics import mean_absolute_error, mean_squared_error
In [38]:mean_squared_error(test, predictions_arima, squared=False)
Out[38]:7.491497821228328
In [39]:mean_absolute_error(test, predictions_arima)
Out[39]:5.595463345661848
```

Прогнозирование временного ряда с использованием метода символьной регрессии

Будем использовать библиотеку gplearn:

In [40]:**from** gplearn.genetic **import** SymbolicRegressor

Прогноз

```
In [41]:function_set = ['add', 'sub', 'mul', 'div', 'sin']
        est_gp = SymbolicRegressor(population_size=500, metric='mse',
                                   generations=200, stopping_criteria=0.01,
                                   init_depth=(4, 10), verbose=1, function_set=function_set,
                                   const_range=(-10, 10), random_state=0)
In [42]:est_gp.fit(np.array(xnum_train).reshape(-1, 1), train.reshape(-1, 1))
```

	Population Average		Best Individual			
Gen	Length	Fitness	Length	Fitness	OOB Fitness	Time Left
0	263.65	1.91324e+67	26	3366.8	N/A	5.85m
1	161.42	1.73488e+15	3	771.22	N/A	2.54m
2	62.67	3.99717e+14	3	771.22	N/A	1.53m
3	39.15	3.51722e+10	3	285.6	N/A	1.56m
4	24.00	3.38638e+11	3	285.6	N/A	56.82s
5	26.05	6.84991e+09	34	280.86	N/A	59.61s
6	11.13	1.4874e+10	35	280.438	N/A	47.45s
7	19.15	4.04141e+06	33	280.136	N/A	1.04m
8	33.94	2.44637e+10	62	279.776	N/A	1.22m
9	36.48	2.2103e+06	42	279.19	N/A	1.38m
10	45.82	1.61747e+09	39	279.026	N/A	1.31m
11	50.83	1.24868e+06	60	278.728	N/A	1.34m
12	51.02	1.20327e+06	72	278.686	N/A	1.26m
13	46.53	5.97296e+08	64	278.507	N/A	1.27m
14	59.07	988142	67	278.056	N/A	1.44m
15	80.40	1.4714e+06	70	277.651	N/A	1.91m
16	91.46	4.15928e+06	58	274.954	N/A	1.70m
17	94.69	1.16678e+06	58	274.954	N/A	1.75m
18	131.75	3.04158e+06	113	274.223	N/A	2.32m
19	154.79	599428	70	267.841	N/A	2.65m
20	129.60	5.39217e+06	128	267.662	N/A	2.63m
21	100.25	4.61995e+06	67	263.942	N/A	2.85m
22	92.04	274173	103	263.402	N/A	3.33m
23	107.35	193345	183	258.85	N/A	2.55m
24	108.87	140414	183	258.017	N/A	2.20m
25	123.21	185654	212	240.913	N/A	2.36m
26	180.34	297662	210	240.84	N/A	3.03m
27	208.77	143690	211	239.988	N/A	3.51m
28	213.35	338481	299	238.607	N/A	3.29m
29	222.05	231000	476	238.538	N/A	3.46m
30	267.90	200555	303	238.41	N/A	4.08m
31	298.85	110925	556	238.103	N/A	4.18m
32	309.06	185395	556	238.07	N/A	4.85m
33	340.90	132016	354	238.051	N/A	5.55m
34	326.51	129423	332	237.828	N/A	4.59m
35	314.32	939493	344	237.792	N/A	4.49m
36	327.52	129602	303	230.187	N/A	4.54m
37	318.18	7.70537e+07	340	220.34	N/A	4.36m
38	329.86	157729	366	220.279	N/A	4.51m
39	330.05	310550	329	219.403	N/A	5.05m
40	342.88	184113	348	218.34	N/A	5.12m
41	349.80	1.90276e+09	329	217.718	N/A	4.90m
42	360.93	303619	327	217.701	N/A	6.29m
43	344.29	226896	320	210.026	N/A	4.87m
44	337.52	231055	398	206.541	N/A	4.57m
45	340.60	294015	398	206.541	N/A	5.00m
46	359.81	256564	407	195.67	N/A	5.32m
47	407.65	152362	493	193.514	N/A	5.29m
48	424.48	5.85872e+06	450	190.798	N/A	5.26m
49	464.99	356433	450	190.793	N/A	5.66m
50	479.00	2.61636e+06	469	189.585	N/A	5.69m
51	463.20	97706.7	574	181.247	N/A	6.84m
52	486.36	314938	641	180.519	N/A	5.64m
53	533.12	319413	582	180.251	N/A	6.14m
54	599.20	154258	580	179.739	N/A	6.82m
55	605.87	115203	780	179.665	N/A	7.16m
56	607.26	1.10202e+06	580	161.751	N/A	7.07m
57	590.25	325810	607	157.107	N/A	6.39m
58	599.51	175627	498	154.816	N/A	6.50m
59	615.73	2.05937e+07	585	147.345	N/A	6.93m
60	572.38	381544	597	146.883	N/A	7.29m
61	576.44	289927	509	145.037	N/A	6.44m
62	557.31	243327	651	144.194	N/A	5.95m
63	574.89	2.80685e+06	579	142.065	N/A	6.23m

64	595.33	217064	582	140.262	N/A	7.12m
65	592.78	112236	578	139.268	N/A	6.31m
66	601.12	214792	687	139.167	N/A	6.23m
67	596.97	401058	580	138.77	N/A	6.13m
68	596.88	183980	731	138.407	N/A	6.74m
69	605.00	196923	645	138.124	N/A	6.02m
70	624.28	120101	702	134.96	N/A	6.22m
71	613.74	65220.9	700	134.95	N/A	6.00m
72	662.45	219994	706	134.663	N/A	7.47m
73	713.11	84137	720	134.383	N/A	6.76m
74	706.18	145495	708	134.371	N/A	6.63m
75	691.32	164370	734	133.882	N/A	6.58m
76	714.10	112927	859	133.105	N/A	7.25m
77	741.06	81064	920	132.395	N/A	6.75m
78	804.12	234355	1049	132.429	N/A	7.35m
79	822.98	90264.5	869	131.907	N/A	8.27m
80	832.85	205834	942	131.6	N/A	7.34m
81	860.39	295080	983	131.305	N/A	7.54m
82	891.89	244599	891	130.529	N/A	8.27m
83	941.01	236574	1051	130.064	N/A	7.94m
84	945.35	5.90819e+08	1051	129.819	N/A	7.93m
85	942.77	93379.8	1049	129.519	N/A	8.33m
86	983.41	235777	995	126.097	N/A	7.98m
87	1043.72	581588	999	125.898	N/A	8.37m
88	1142.51	286982	1005	124.618	N/A	9.85m
89	1031.67	108799	989	123.27	N/A	8.07m
90	1074.67	128401	981	123.027	N/A	9.00m
91	1026.87	5.90862e+08	987	121.965	N/A	7.98m
92	1003.39	2.34917e+09	1274	121.202	N/A	7.75m
93	1012.57	201797	982	120.63	N/A	8.48m
94	1065.47	128891	974	120.402	N/A	9.21m
95	1066.71	251783	1023	120.04	N/A	7.27m
96	1003.03	202755	1037	119.958	N/A	6.34m
97	981.58	159988	1001	119.906	N/A	6.40m
98	993.94	322564	989	119.464	N/A	6.41m
99	991.74	187031	946	119.374	N/A	7.64m
100	993.97	105857	1142	119.102	N/A	8.21m
101	976.85	79860.2	1144	119.079	N/A	7.44m
102	995.50	221920	951	118.929	N/A	6.94m
103	938.70	90457.6	950	118.854	N/A	6.49m
104	937.47	314656	939	118.68	N/A	6.86m
105	936.34	149304	919	118.526	N/A	6.38m
106	937.20	2.00517e+07	923	118.466	N/A	6.16m
107	941.85	8.91926e+09	1041	117.759	N/A	6.92m
108	943.66	159067	1041	117.646	N/A	6.11m
109	968.74	94109	1041	117.582	N/A	6.32m
110	1048.24	75924.5	1136	117.307	N/A	7.07m
111	1057.97	1.13477e+06	1180	117.163	N/A	6.63m
112	1076.84	236939	1182	116.834	N/A	7.32m
113	1128.41	73033.1	1188	116.809	N/A	6.86m
114	1120.40	256617	1178	116.745	N/A	6.75m
115	1142.22	139713	1205	116.588	N/A	7.31m
116	1161.78	119681	1389	116.536	N/A	6.78m
117	1177.39	163665	1523	116.336	N/A	7.15m
118	1174.59	1.49591e+06	1210	116.279	N/A	6.68m
119	1171.17	164129	1212	116.271	N/A	7.12m
120	1158.92	37142.5	1389	116.147	N/A	6.48m
121	1197.40	46742.8	1217	116.097	N/A	6.55m
122	1216.58	332484	1343	116.026	N/A	7.07m
123	1203.00	63012.6	1215	115.981	N/A	6.40m
124	1205.20	217140	1208	115.942	N/A	6.89m
125	1200.88	195967	1361	115.919	N/A	6.20m
126	1201.62	36773.3	1213	115.845	N/A	6.45m
127	1192.41	175546	1436	115.636	N/A	5.97m
128	1178.73	118886	1436	115.632	N/A	5.77m
129	1228.00	92349.2	1435	115.615	N/A	6.33m
130	1219.99	177369	1435	115.615	N/A	5.84m
131	1219.26	581658	1435	115.581	N/A	6.27m
132	1241.64	5.95807e+08	1338	115.248	N/A	5.73m
133	1238.89	278341	1361	115.15	N/A	5.97m
134	1248.58	1.60758e+11	1383	115.108	N/A	5.65m
135	1302.84	142129	1362	115.062	N/A	6.20m
136	1327.08	80862	1628	110.496	N/A	5.89m
137	1368.02	119268	1745	110.206	N/A	6.25m
138	1492.48	37613.6	1747	109.06	N/A	6.30m
139	1678.08	26897.3	1753	108.847	N/A	7.11m
140	1722.07	122838	1936	107.952	N/A	6.89m
141	1781.41	83720.5	2025	107.852	N/A	7.34m

```
142 1842.02      48335.6    1971        107.611          N/A     7.22m
143 1947.55      82681.7    1964        107.512          N/A     7.34m
144 1933.71     6.0061e+08    1970        107.395          N/A     7.41m
145 1972.54      74686.6    1970        106.999          N/A     7.25m
146 1954.03      64469.6    2011        106.981          N/A     7.01m
147 1951.31      8795.11    1942        106.773          N/A     7.13m
148 1955.85      975.374    1941        106.647          N/A     6.79m
149 1965.40   3.42713e+06    2020        106.646          N/A     6.81m
150 1947.16      78761.9    2019        106.512          N/A     6.78m
151 1933.35      58093.1    2018        106.506          N/A     6.43m
152 1964.62      57360.7    2004         106.35          N/A     7.60m
153 1970.03      69364.7    1881        106.234          N/A     6.80m
154 1950.37   5.95297e+08    1882        106.112          N/A     6.76m
155 1939.50       123477    1878        106.099          N/A     7.77m
156 1909.67       217390    1824        105.998          N/A     6.49m
157 1879.78      48951.3    1841        105.954          N/A     6.38m
158 1852.92   2.00151e+07    1828        105.831          N/A     6.02m
159 1834.71        41082    1828        105.831          N/A     5.37m
160 1817.06      74661.6    1832        105.797          N/A     5.38m
161 1814.77      3860.34    1832        105.783          N/A     5.64m
162 1808.57      62680.3    1842        105.664          N/A     5.77m
163 1758.15       203506    1712        105.417          N/A     4.38m
164 1690.86        92262    1712        105.394          N/A     3.88m
165 1692.49       116450    1741        105.261          N/A     3.78m
166 1727.47      66436.9    1739        105.171          N/A     4.42m
167 1716.24   3.89336e+11    1741        105.141          N/A     6.18m
168 1730.61   1.00493e+07    1750        105.092          N/A     3.93m
169 1741.79       571328    1742         104.97          N/A     3.29m
170 1733.52   1.78267e+07    1741        104.953          N/A     3.07m
171 1730.60       502739    1954        104.847          N/A     2.91m
172 1753.23       196115    1954        104.847          N/A     3.33m
173 1755.67   5.67425e+08    2047        104.254          N/A     3.11m
174 1757.01        82979    2047        104.254          N/A     3.60m
175 1806.70      93743.3    2049        103.817          N/A     3.02m
176 1954.89      35559.4    2022        103.736          N/A     2.94m
177 2026.45        73924    2036        103.596          N/A     2.90m
178 2044.62      87278.4    2048        103.544          N/A     2.79m
179 2045.47       124714    2047        103.372          N/A     2.64m
180 2031.76       130210    2134        103.226          N/A     2.50m
181 2055.03      35068.6    2631        102.926          N/A     2.51m
182 2066.12      72599.6    2633        102.919          N/A     2.41m
183 2030.26       161098    2032         103.01          N/A     2.22m
184 2020.91       136310    2076        102.829          N/A     2.08m
185 2018.65      30982.9    2009        102.519          N/A     2.12m
186 2003.83   6.01768e+08    2012        102.519          N/A     1.78m
187 2022.74      79395.3    2527        102.476          N/A     1.59m
188 2005.91      56386.1    2100        102.348          N/A     1.41m
189 2016.43       115070    2184        102.345          N/A     1.19m
190 2016.36      94111.4    2147        102.316          N/A     1.05m
191 2018.53       173633    2083        102.054          N/A    57.01s
192 2020.19       116259    2085        102.036          N/A    51.89s
193 2036.41       134852    2081        101.931          N/A    43.04s
194 2064.99      63033.3    2077        101.905          N/A    36.30s
195 2083.19      33114.6    2082        101.271          N/A    29.98s
196 2076.44       242556    2082        101.259          N/A    21.29s
197 2075.79       192377    2082        101.247          N/A    16.55s
198 2089.56   5.95726e+08    2101        101.067          N/A     7.75s
199 2086.89      58925.5    2051        101.046          N/A     0.00s
```

Out[42]:

SymbolicRegressor

```
sub(mul(7.676, 5.137), add(sub(add(sub(sub(sub(sub(sin(div(add(X0, X0), add(mul(7.676, 8.272), div(sub(sub(sub(sin(-3.873), mul(X0, -6.123
)), div(5.356, mul(7.676, 8.272))), add(-7.954, mul(7.676, sub(sub(sub(div(add(sub(sub(sub(sin(-3.873), mul(X0, -6.123)), sub(div(-2.805, 0.34
9), sub(sub(5.356, sub(3.716, -1.962)), sub(3.716, -1.962)))), add(-7.954, mul(7.676, sub(sub(sub(div(-2.805, 0.349), sub(mul(7.676, 5.137), a
dd(X0, sub(div(-2.805, 0.349), sub(3.716, -1.962))))), div(5.356, mul(7.676, 8.272))), add(-7.954, mul(7.676, 8.272)))))), sub(mul(X0, -6.123), a
dd(X0, 8.272))), sub(sub(sin(div(add(X0, add(-7.954, mul(7.676, sub(div(-2.805, 0.349), add(-7.954, sub(3.716, -1.962)))))), add(mul(7.676, 8.2
72), sub(3.716, -1.962)))), sub(3.716, -1.962)), div(3.103, X0))), sub(mul(7.676, 5.137), add(sub(sub(sin(-3.873), sub(div(sin(-3.873), X0), -1.9
62)), -6.123), sub(div(-2.805, 0.349), sub(div(3.103, X0), div(3.103, X0)))))), div(5.356, mul(7.676, 8.272))), add(-7.954, mul(7.676, 8.272)))))),
add(mul(7.676, div(add(X0, add(sin(div(add(X0, add(div(sub(div(sin(-3.873), X0), -1.962), X0), sub(div(-2.805, 0.349), sub(5.356, mul(7.676, 8.
272))))), add(mul(7.676, 8.272), div(add(X0, sub(mul(7.676, 7.676), sub(div(3.103, X0), sub(3.716, -1.962)))), X0)))), sub(X0, sub(sub(3.716, -
1.962), add(mul(7.676, 5.137), sub(div(-2.805, 0.349), sub(div(3.103, X0), sub(3.716, -1.962)))))))), add(mul(7.676, 8.272), div(5.356, X0)))), di
v(5.356, X0)))))), sub(3.716, -1.962)), div(7.676, sub(sin(div(add(X0, add(sub(div(-2.805, 0.349), div(-2.805, 0.349)), div(add(sub(sub(sub(sin(-
```

```
In [43]:y_gp = est_gp.predict(np.array(xnum_test).reshape(-1, 1))
        y_gp[:10]
```

```
Out[43]:array([73.80469798, 74.62276246, 74.81765215, 74.88961676, 74.91224874,
               74.90617581, 74.87934757, 74.83554142, 74.77687615, 74.70473071])
```

```
In [44]:data2['predictions_GPLEARN'] = (train_size * [np.NAN]) + list(y_gp)
```

Визуализация

Построим дерево по символьной регрессии:

```
In [47]:import graphviz
        import pydotplus
        from sklearn.tree import export_graphviz
In [50]:dot_data = est_gp._program.export_graphviz()
        pydot_graph = pydotplus.graph_from_dot_data(dot_data)
        pydot_graph.set_size(10)
        gvz_graph = graphviz.Source(pydot_graph.to_string())
        gvz_graph
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\backend\execute.py:79, in run_check(cmd, input_lines, encoding, quiet, **kwargs)
     78         kwargs['stdout'] = kwargs['stderr'] = subprocess.PIPE
---> 79     proc = _run_input_lines(cmd, input_lines, kwargs=kwargs)
     80 else:

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\backend\execute.py:99, in _run_input_lines(cmd, input_lines, kwargs)
     98 def _run_input_lines(cmd, input_lines, *, kwargs):
---> 99     popen = subprocess.Popen(cmd, stdin=subprocess.PIPE, **kwargs)
    101     stdin_write = popen.stdin.write

File ~\AppData\Local\Programs\Python\Python310\lib\subprocess.py:971, in Popen.__init__(self, args, bufsize, executable, stdin, stdout, stderr, preexec_fn, close_fds, shell, cwd, env, universal_newlines, startupinfo, creationflags, restore_signals, start_new_session, pass_fds, user, group, extra_groups, encoding, errors, text, umask, pipesize)
    968             self.stderr = io.TextIOWrapper(self.stderr,
    969                     encoding=encoding, errors=errors)
--> 971     self._execute_child(args, executable, preexec_fn, close_fds,
    972                         pass_fds, cwd, env,
    973                         startupinfo, creationflags, shell,
    974                         p2cread, p2cwrite,
    975                         c2pread, c2pwrite,
    976                         errread, errwrite,
    977                         restore_signals,
    978                         gid, gids, uid, umask,
    979                         start_new_session)
    980 except:
    981     # Cleanup if the child failed starting.

File ~\AppData\Local\Programs\Python\Python310\lib\subprocess.py:1440, in Popen._execute_child(self, args, executable, preexec_fn, close_fds, pass_fds, cwd, env, startupinfo, creationflags, shell, p2cread, p2cwrite, c2pread, c2pwrite, errread, errwrite, unused_restore_signals, unused_gid, unused_gids, unused_uid, unused_umask, unused_start_new_session)
   1439 try:
-> 1440     hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
   1441                              # no special security
   1442                              None, None,
   1443                              int(not close_fds),
   1444                              creationflags,
   1445                              env,
   1446                              cwd,
   1447                              startupinfo)
   1448 finally:
   1449     # Child is launched. Close the parent's copy of those pipe
   1450     # handles that only the child should have open.  You need
  (...)
   1453     # pipe will not close when the child process exits and the
   1454     # ReadFile will hang.

FileNotFoundError: [WinError 2] Не удается найти указанный файл

The above exception was the direct cause of the following exception:

ExecutableNotFound                        Traceback (most recent call last)
File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\IPython\core\formatters.py:974, in MimeBundleFormatter.__call__(self, obj, include, exclude)
    971 method = get_real_method(obj, self.print_method)
    973 if method is not None:
--> 974     return method(include=include, exclude=exclude)
    975 return None
    976 else:

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\jupyter_integration.py:98, in JupyterIntegration._repr_mimebundle_(self, include, exclude, **_)
     96 include = set(include) if include is not None else {self._jupyter_mimetype}
     97 include -= set(exclude or [])
```

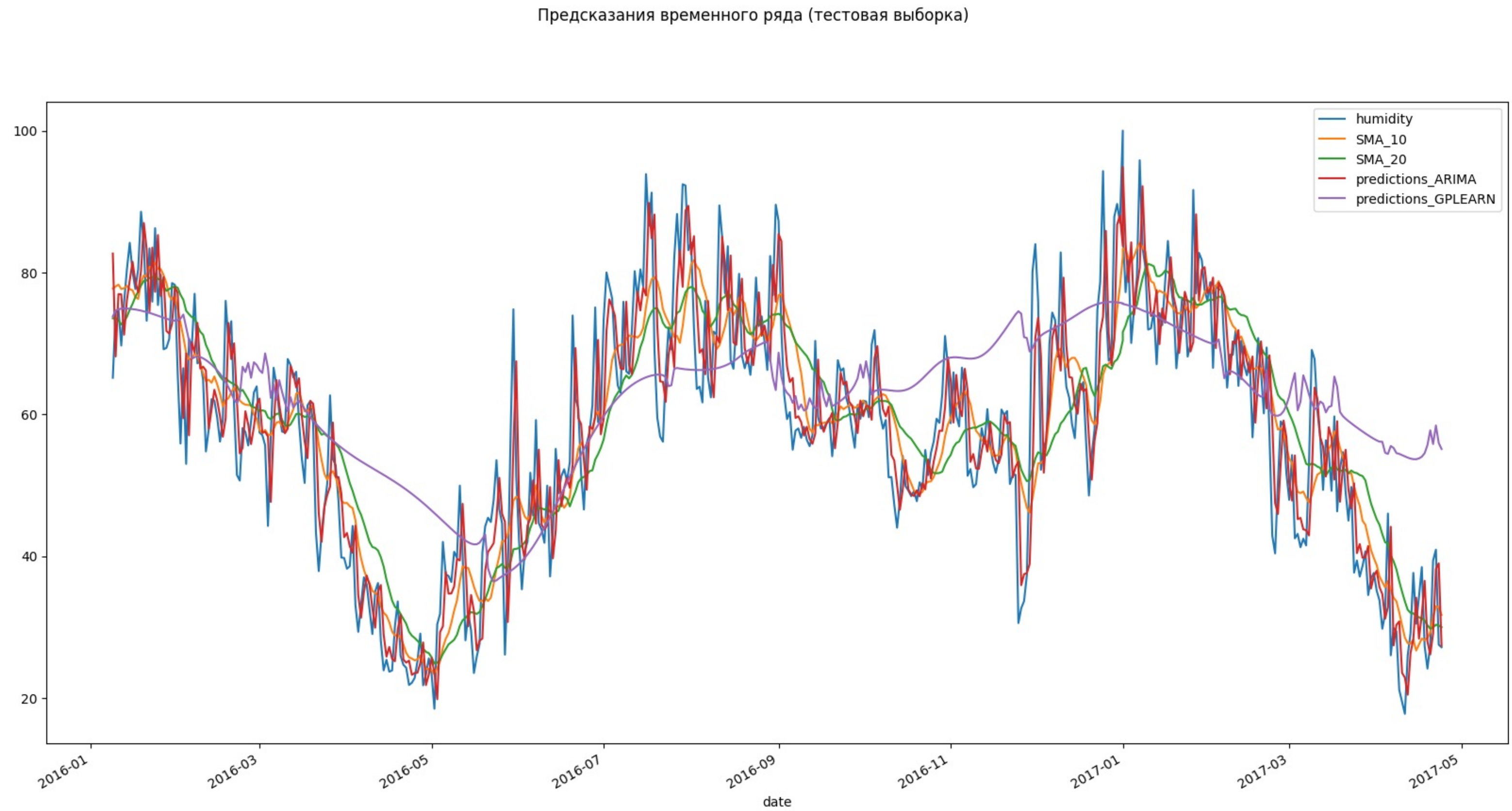
```
---> 98 return {mimetype: getattr(self, method_name)()
     99         for mimetype, method_name in MIME_TYPES.items()
    100          if mimetype in include}

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\jupyter_integration.py:98, in <dictcomp>(.0)
     96 include = set(include) if include is not None else {self._jupyter_mimetype}
     97 include -= set(exclude or [])
---> 98 return {mimetype: getattr(self, method_name)()
     99         for mimetype, method_name in MIME_TYPES.items()
    100          if mimetype in include}

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\jupyter_integration.py:112, in JupyterIntegration._repr_image_svg_xml(self)
    110 def _repr_image_svg_xml(self) -> str:
    111     """Return the rendered graph as SVG string."""
--> 112     return self.pipe(format='svg', encoding=SVG_ENCODING)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\piping.py:104, in Pipe.pipe(self, format, renderer, formatter, neato_no_op, quiet, engine, encoding)
     55 def pipe(self,
     56          format: typing.Optional[str] = None,
     57          renderer: typing.Optional[str] = None,
   (...)
     61          engine: typing.Optional[str] = None,
     62          encoding: typing.Optional[str] = None) -> typing.Union[bytes, str]:
     63     """Return the source piped through the Graphviz layout command.
     64 
     65     Args:
   (...)
    102          '<?xml version='
    103     """
--> 104     return self._pipe_legacy(format,
    105                              renderer=renderer,
    106                              formatter=formatter,
    107                              neato_no_op=neato_no_op,
    108                              quiet=quiet,
    109                              engine=engine,
    110                              encoding=encoding)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\_tools.py:171, in deprecate_positional_args.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    162     wanted = ', '.join(f'{name}={value!r}'
    163                        for name, value in deprecated.items())
    164     warnings.warn(f'The signature of {func.__name__} will be reduced'
    165                   f' to {supported_number} positional args'
    166                   f' {list(supported)}: pass {wanted}'
    167                   ' as keyword arg(s)',
    168                   stacklevel=stacklevel,
    169                   category=category)
--> 171 return func(*args, **kwargs)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\piping.py:121, in Pipe._pipe_legacy(self, format, renderer, formatter, neato_no_op, quiet, engine, encoding)
    112 @_tools.deprecate_positional_args(supported_number=2)
    113 def _pipe_legacy(self,
    114                  format: typing.Optional[str] = None,
   (...)
    119                  engine: typing.Optional[str] = None,
    120                  encoding: typing.Optional[str] = None) -> typing.Union[bytes, str]:
--> 121     return self._pipe_future(format,
    122                              renderer=renderer,
    123                              formatter=formatter,
    124                              neato_no_op=neato_no_op,
    125                              quiet=quiet,
    126                              engine=engine,
    127                              encoding=encoding)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\piping.py:149, in Pipe._pipe_future(self, format, renderer, formatter, neato_no_op, quiet, engine, encoding)
    146 if encoding is not None:
    147     if codecs.lookup(encoding) is codecs.lookup(self.encoding):
    148         # common case: both stdin and stdout need the same encoding
--> 149         return self._pipe_lines_string(*args, encoding=encoding, **kwargs)
    150     try:
    151         raw = self._pipe_lines(*args, input_encoding=self.encoding, **kwargs)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\backend\piping.py:212, in pipe_lines_string(engine, format, input_lines, encoding, renderer, formatter, neato_no_op, quiet)
```

```
206 cmd = dot_command.command(engine, format,
207                           renderer=renderer,
208                           formatter=formatter,
209                           neato_no_op=neato_no_op)
210 kwargs = {'input_lines': input_lines, 'encoding': encoding}
--> 212 proc = execute.run_check(cmd, capture_output=True, quiet=quiet, **kwargs)
213 return proc.stdout

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\graphviz\backend\execute.py:84, in run_check(cmd, input_lines, encoding, quiet, **kwargs)
 82 except OSError as e:
 83     if e.errno == errno.ENOENT:
---> 84         raise ExecutableNotFound(cmd) from e
 85     raise
 87 if not quiet and proc.stderr:

ExecutableNotFound: failed to execute WindowsPath('dot'), make sure the Graphviz executables are on your systems' PATH
```

Out[50]:<graphviz.sources.Source at 0x205a716a3b0>

Построим график по тестовой выборке:

In [51]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(20,10))
fig.suptitle('Предсказания временного ряда (тестовая выборка)')
data2[train_size:].plot(ax=ax, legend=True)
pyplot.show()
```



Визуально предсказания по методу сивольной регрессии менее точны, чем предсказания по ARIMA. Для повышения точности требуется настройка параметров метода, в частности увеличенное количество итераций цикла. Однако при этом сильно возрастут затраты времени.

Метрики

MAE и MSE:

In [52]:`mean_squared_error(test, y_gp, squared=False)`

Out[52]:13.52324614284193

In [53]:`mean_absolute_error(test, y_gp)`

Out[53]:10.607119049073066

Сранение качества моделей

Чем ближе значение MAE и MSE к нулю, тем лучше качество модели.

MAE для авторегрессионного метода ARIMA = 5.5, а для метода символьной регрессии = 10.6.

MSE для авторегрессионного метода ARIMA = 7.3, а для метода символьной регрессии = 13.5.

Качество модели для авторегрессионного метода ARIMA выше. Для выполенения ARIMA также требуется меньше времени.