



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ	ГОЛОВНОЙ УЧЕБНО-ИССЛЕДОВАТЕЛЬСКИЙ И МЕТОДИЧЕСКИЙ ЦЕНТР
	ПРОФЕССИОНАЛЬНОЙ РЕАБИЛИТАЦИИ ЛИЦ С ОГРАНИЧЕННЫМИ
	ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ
КАФЕДРА	СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

Отчёт по лабораторной работе №6 по курсу «Технологии машинного обучения».

«Анализ и прогнозирование временного ряда».

Выполнил:
Новиков С. А.
студент группы ИУ5-62Б

Проверил:
Гапанюк Ю.Е.

Подпись и дата:

Подпись и дата:

2025 г.

1. 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.

2. Ячейки Jupyter-ноутбука

2.1. Выбор и загрузка данных

2.1.1. 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, рассчитанная 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

2.1.2. Импорт библиотек

Импортируем библиотеки с помощью команды import:

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

Уберем предупреждения:

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

2.1.3. Загрузка данных

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

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

2.2. Первичная обработка данных и визуализация

2.2.1. Первичный анализ

Выведем первые 5 строк датасета:

```
[4]: data.head()
```

```
[4]:
```

	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:

```
[5]: data.shape
```

```
[5]: (1576, 4)
```

Определим типы данных:

```
[6]: data.dtypes
```

```
[6]: meantemp      float64
      humidity     float64
      wind_speed   float64
      meanpressure float64
      dtype: object
```

2.2.2. Обработка данных

Оставим только столбец влажности для временного ряда:

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

```
[8]: data.head()
```

```
[8]:
```

	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

2.2.3. Основные статистические характеристики

Определим основные статистические характеристики временного ряда:

```
[9]: data.describe()
```

[9]:

	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

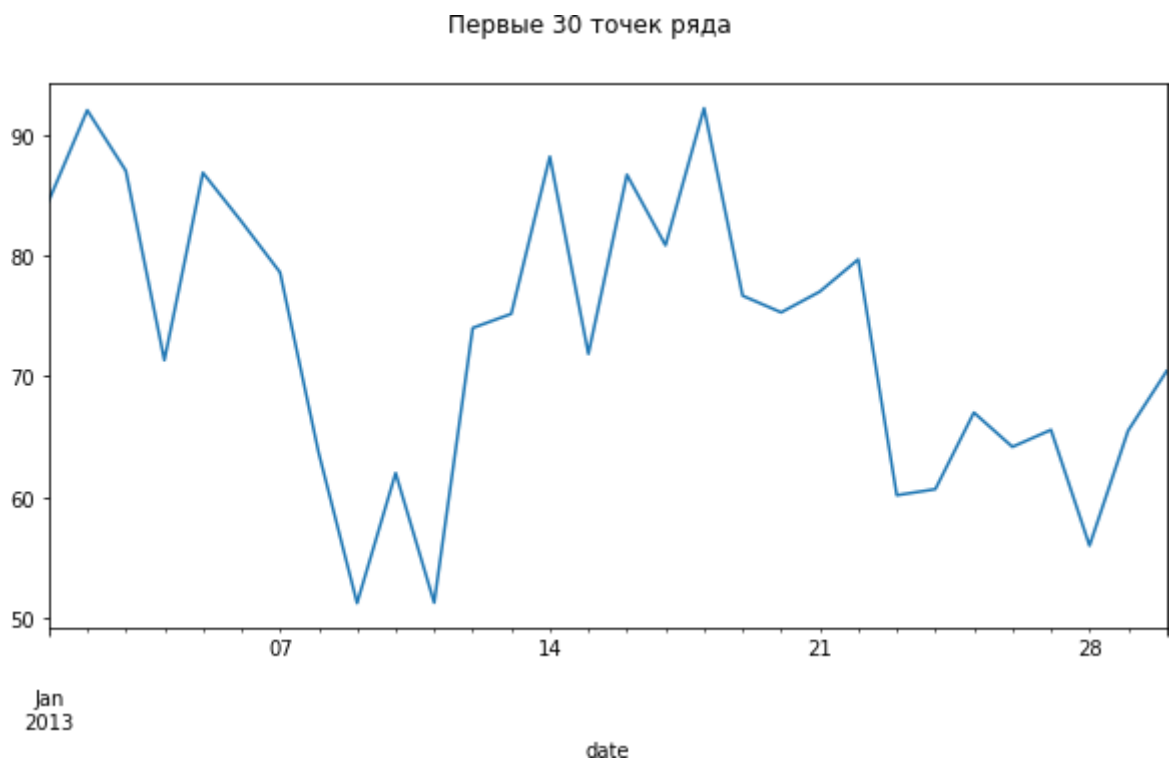
2.2.4. Визуализация исходного временного ряда

В виде графика:

```
[10]: 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()
```

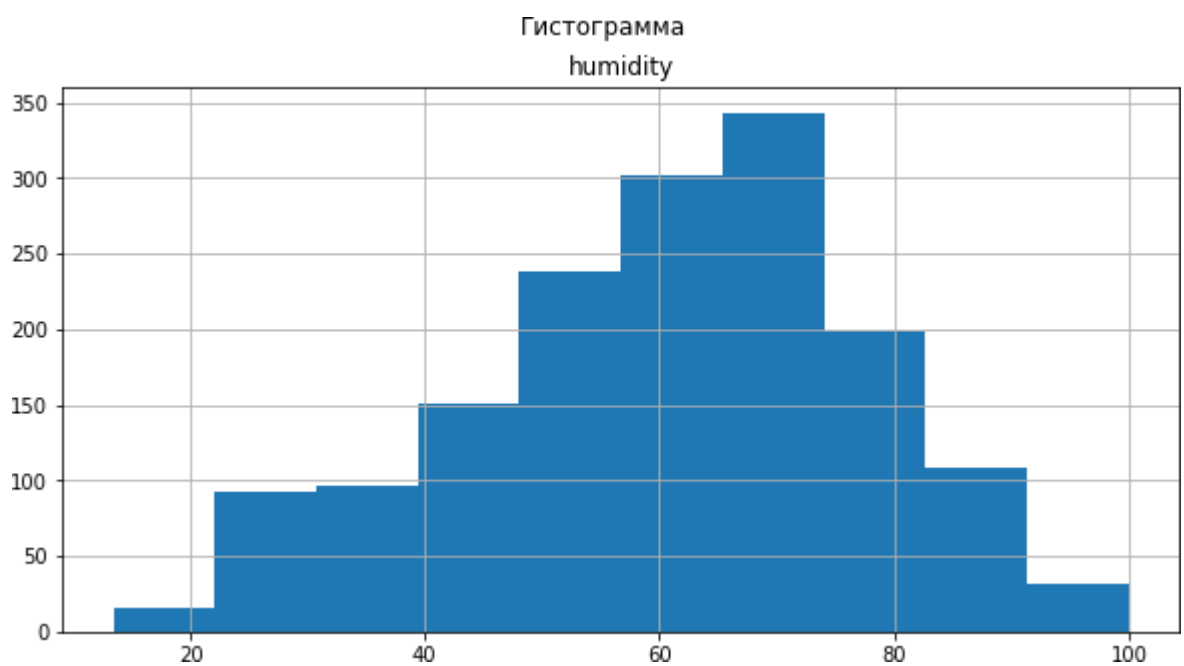


```
[11]: fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Первые 30 точек ряда')
data[:30].plot(ax=ax, legend=False)
pyplot.show()
```



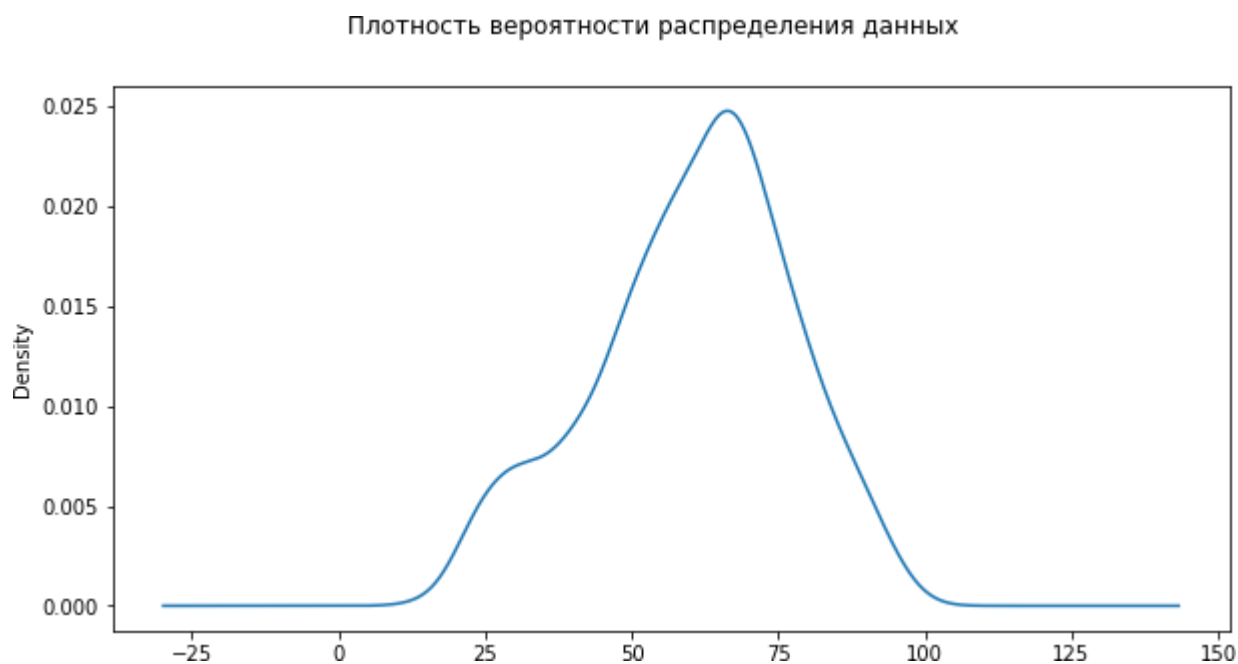
В виде гистограммы:

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



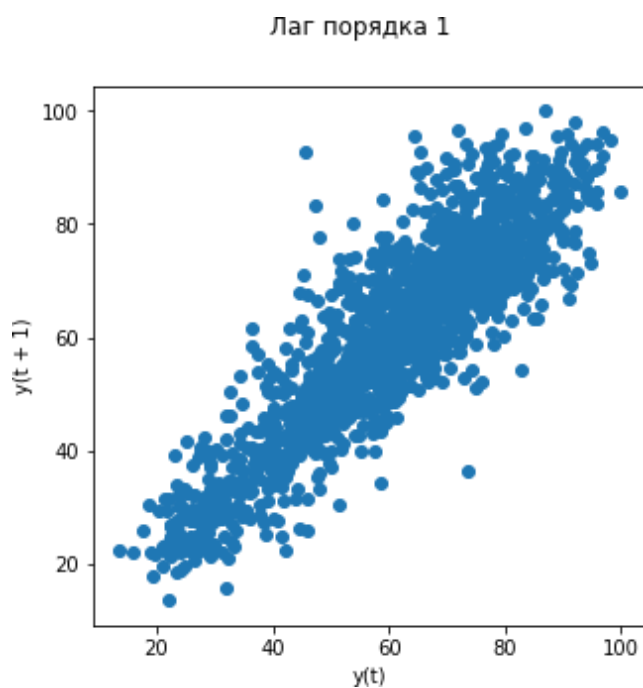
Вероятностная плотность распределения данных:

```
[13]: 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()
```

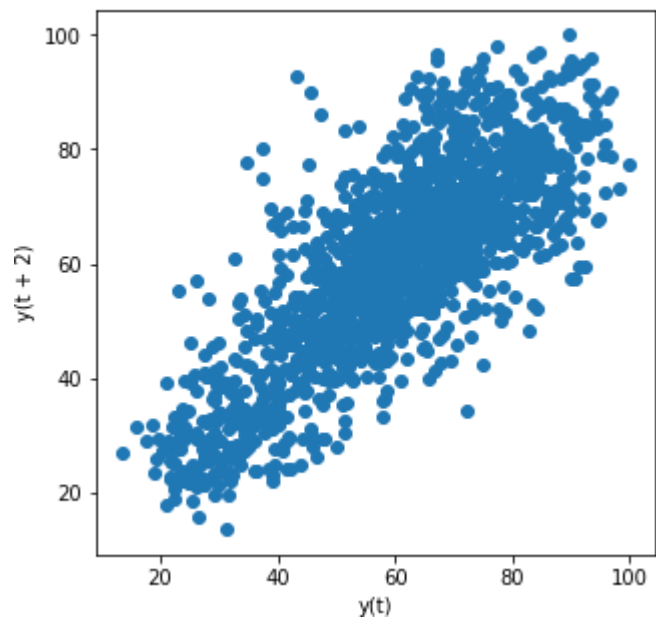


С помощью Lag Plot:

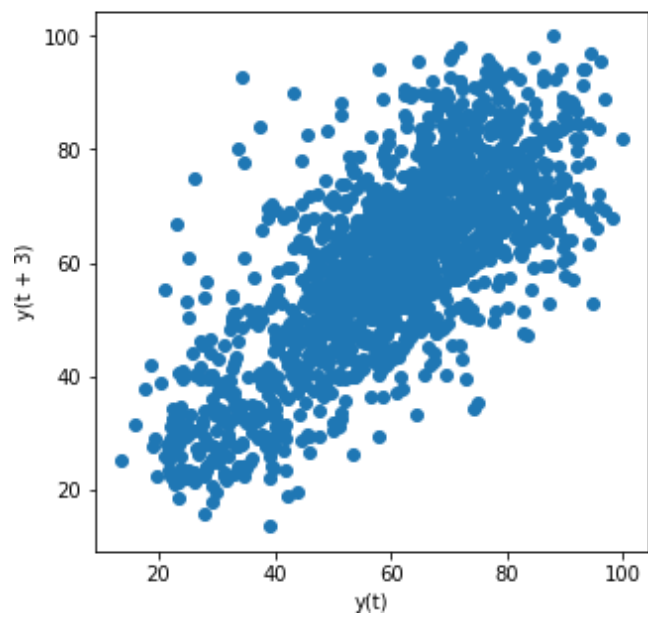
```
[14]: for i in range(1, 5):
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(5,5))
fig.suptitle(f'Лег порядка {i}')
pd.plotting.lag_plot(data, lag=i, ax=ax)
pyplot.show()
```

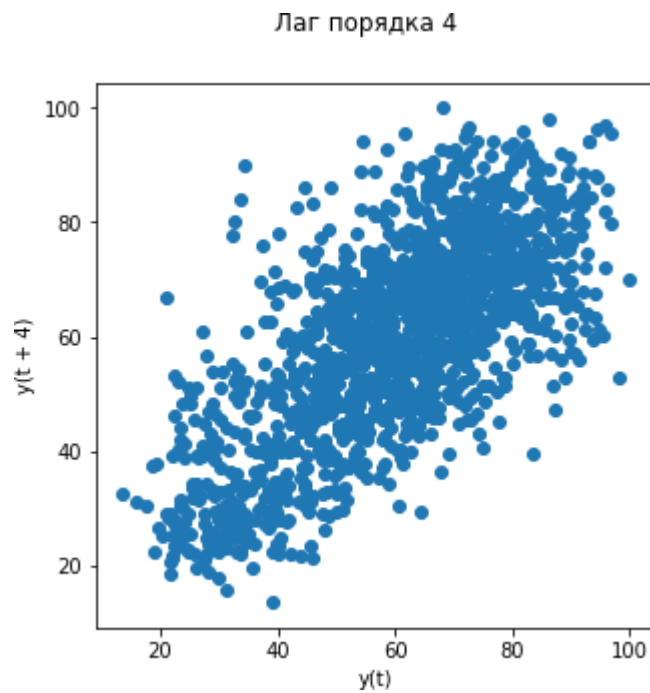


Лег порядка 2



Лег порядка 3

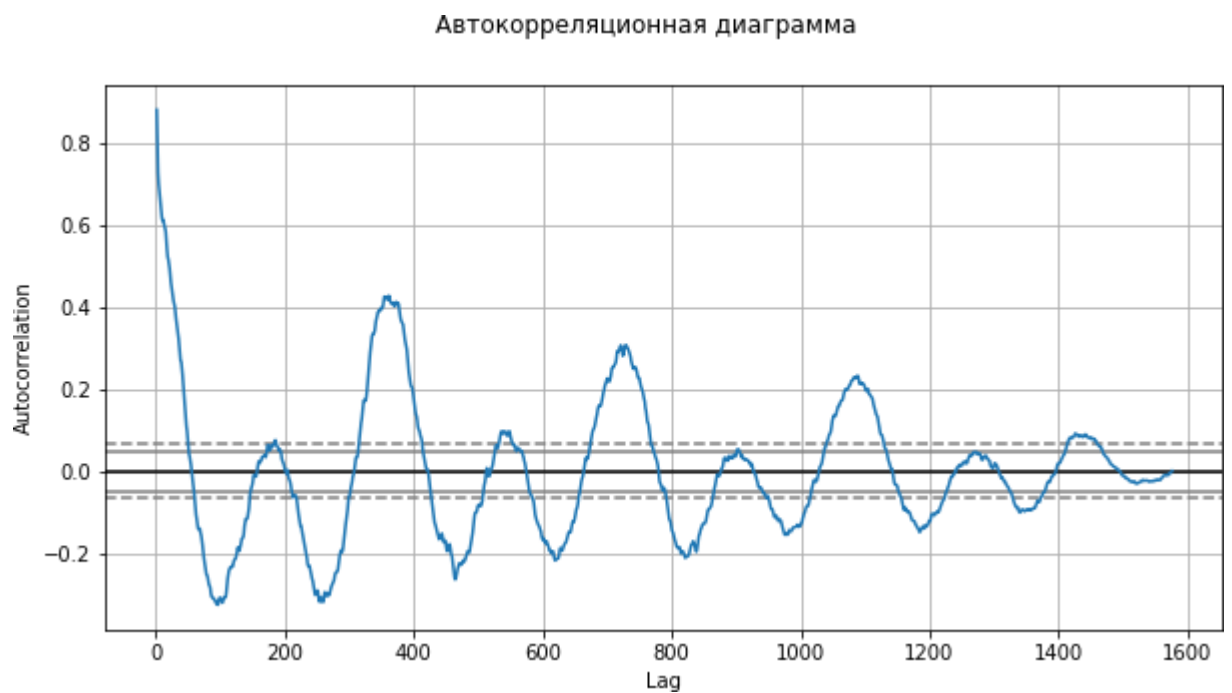




Наблюдается достаточно сильная положительная корреляция.

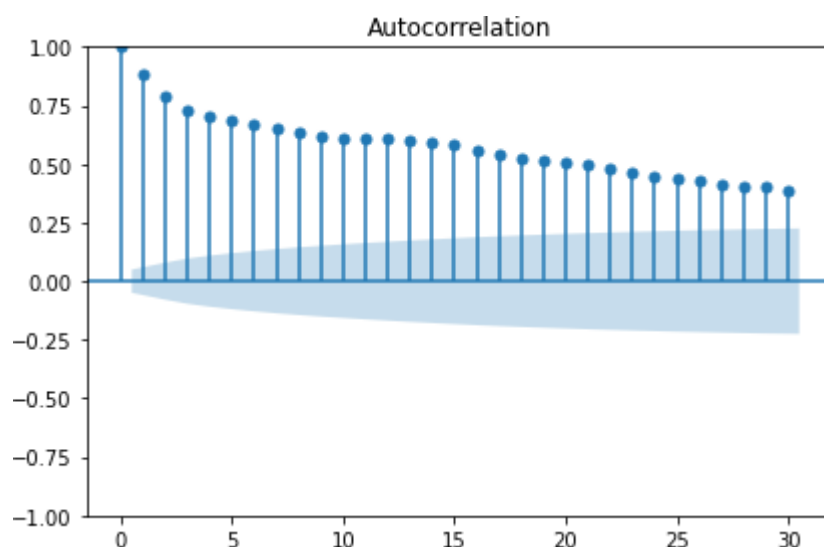
Автокорреляционная диаграмма:

```
[15]: 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()
```



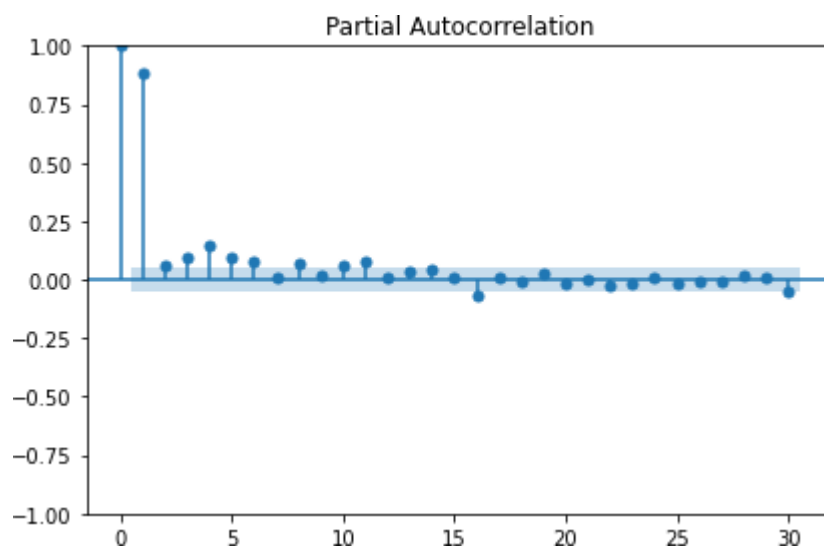
Автокорреляционная функция:


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



Частичная автокорреляционная функция:

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

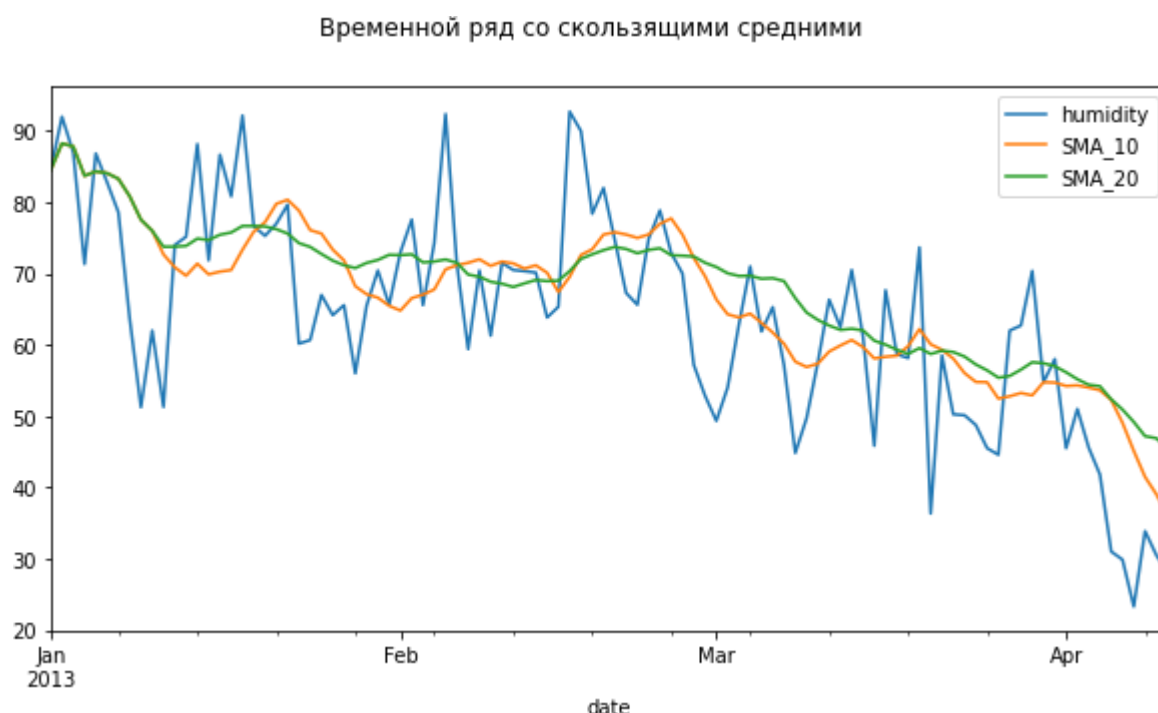


Временной ряд со скользящими средними:

```
[18]: data2 = data.copy()
```

```
[19]: data2['SMA_10'] = data2['humidity'].rolling(10, min_periods=1).mean()
      data2['SMA_20'] = data2['humidity'].rolling(20, min_periods=1).mean()
```

```
[20]: 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()
```



2.3. Прогнозирование временного ряда с использованием авторегрессионного метода

Будем использовать авторегрессионный метод ARIMA:

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

2.3.1. Разделение выборки на обучающую и тестовую

```
[22]: 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]
```

2.3.2. Прогноз ARIMA

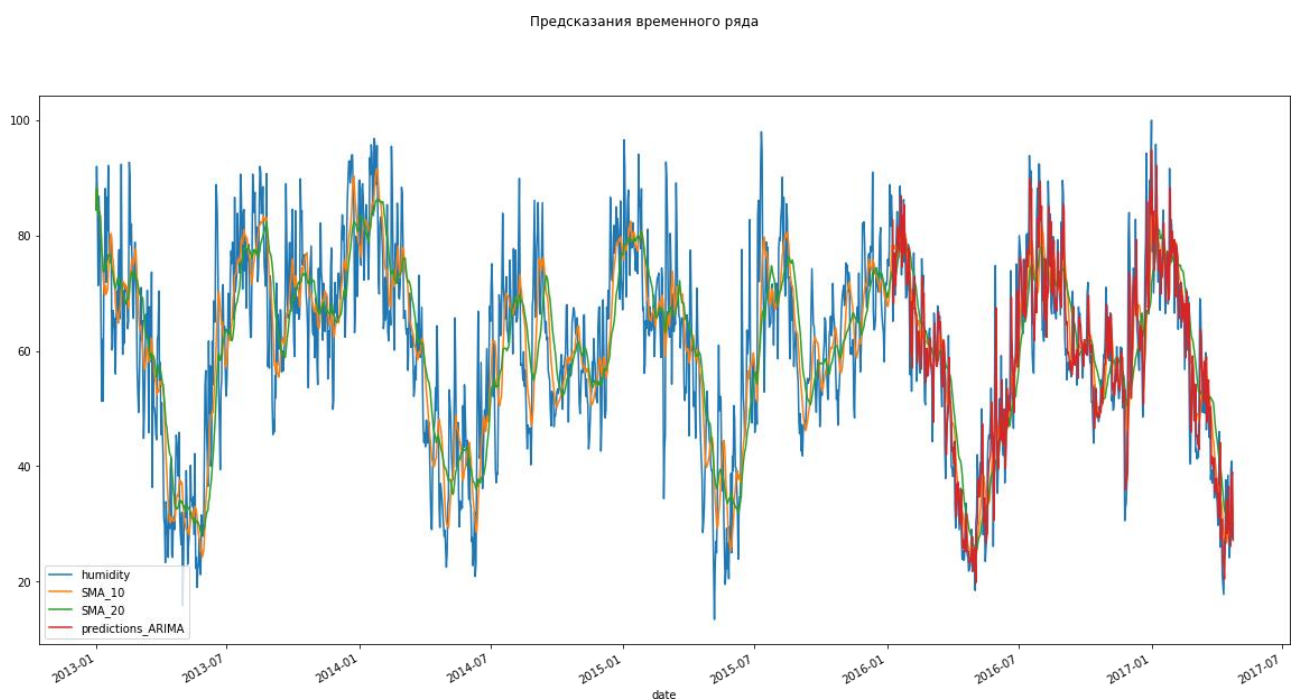
```
[23]: arima_order = (6, 1, 0)
predictions_arima = list()
for t in range(len(test)):
    model_arima = ARIMA(history_arima, order=arima_order)
    model_arima_fit = model_arima.fit()
```

```
yhat_arma = model_arma_fit.forecast()[0]
predictions_arma.append(yhat_arma)
history_arma.append(test[t])
```

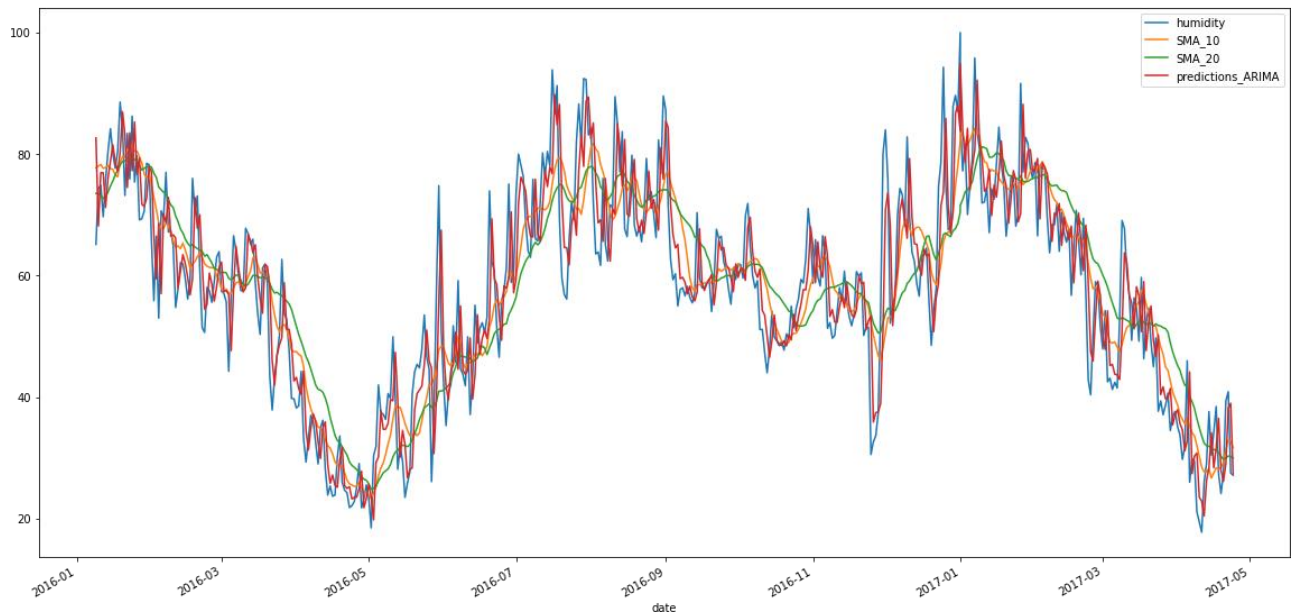
```
[24]: data2['predictions_ARIMA'] = (train_size * [np.NaN]) + list(predictions_arma)
```

2.3.3. Визуализация

```
[25]: 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()
```



```
[26]: 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.

2.3.4. Метрики

MAE и MSE:

```
[27]: from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
[28]: mean_squared_error(test, predictions_arima, squared=False)
```

```
[28]: 7.277480320087945
```

```
[29]: mean_absolute_error(test, predictions_arima)
```

```
[29]: 5.499332563097871
```

2.4. Прогнозирование временного ряда с использованием метода символьной регрессии

Будем использовать библиотеку gplearn:

```
[30]: from gplearn.genetic import SymbolicRegressor
```

2.4.1. Прогноз

```
[31]: 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)
```

[32]: `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	
6.89m						
1	161.42	1.73488e+15	3	771.22	N/A	
3.18m						
2	62.67	3.99717e+14	3	771.22	N/A	
1.78m						
3	39.15	3.51722e+10	3	285.6	N/A	
1.41m						
4	24.00	3.38638e+11	3	285.6	N/A	
1.24m						
5	26.05	6.84991e+09	34	280.86	N/A	
1.24m						
6	11.13	1.4874e+10	35	280.438	N/A	
1.02m						
7	19.15	4.04141e+06	33	280.136	N/A	
1.19m						
8	33.94	2.44637e+10	62	279.776	N/A	
1.37m						
9	36.48	2.2103e+06	42	279.19	N/A	
1.45m						
10	45.82	1.61747e+09	39	279.026	N/A	
1.60m						
11	50.83	1.24868e+06	60	278.728	N/A	
1.72m						
12	51.02	1.20327e+06	72	278.686	N/A	
1.65m						
13	46.53	5.97296e+08	64	278.507	N/A	
1.59m						
14	59.07	988142	67	278.056	N/A	
1.74m						
15	80.40	1.4714e+06	70	277.651	N/A	
1.95m						
16	91.46	4.15928e+06	58	274.954	N/A	
2.14m						
17	94.69	1.16678e+06	58	274.954	N/A	
2.24m						
18	131.75	3.04158e+06	113	274.223	N/A	
2.68m						
19	154.79	599428	70	267.841	N/A	
3.01m						
20	129.60	5.39217e+06	128	267.662	N/A	
2.65m						
21	100.25	4.61995e+06	67	263.942	N/A	
2.26m						
22	92.04	274173	103	263.402	N/A	
2.13m						

23	107.35	193345	183	258.85	N/A
2.34m					
24	108.87	140414	183	258.017	N/A
2.35m					
25	123.21	185654	212	240.913	N/A
2.52m					
26	180.34	297662	210	240.84	N/A
3.19m					
27	208.77	143690	211	239.988	N/A
3.55m					
28	213.35	338481	299	238.607	N/A
3.60m					
29	222.05	231000	476	238.538	N/A
3.85m					
30	267.90	200555	303	238.41	N/A
4.21m					
31	298.85	110925	556	238.103	N/A
4.54m					
32	309.06	185395	556	238.07	N/A
4.76m					
33	340.90	132016	354	238.051	N/A
5.06m					
34	326.51	129423	332	237.828	N/A
4.87m					
35	314.32	939493	344	237.792	N/A
4.77m					
36	327.52	129602	303	230.187	N/A
4.94m					
37	318.18	7.70537e+07	340	220.34	N/A
4.71m					
38	329.86	157729	366	220.279	N/A
4.92m					
39	330.05	310550	329	219.403	N/A
5.46m					
40	342.88	184113	348	218.34	N/A
5.47m					
41	349.80	1.90276e+09	329	217.718	N/A
5.27m					
42	360.93	303619	327	217.701	N/A
5.24m					
43	344.29	226896	320	210.026	N/A
4.91m					
44	337.52	231055	398	206.541	N/A
4.91m					
45	340.60	294015	398	206.541	N/A
4.94m					
46	359.81	256564	407	195.67	N/A
5.08m					
47	407.65	152362	493	193.514	N/A
5.65m					
48	424.48	5.85872e+06	450	190.798	N/A
5.74m					
49	464.99	356433	450	190.793	N/A

6.15m					
50	479.00	2.61636e+06	469	189.585	N/A
6.23m					
51	463.20	97706.7	574	181.247	N/A
5.96m					
52	486.36	314938	641	180.519	N/A
6.18m					
53	533.12	319413	582	180.251	N/A
6.65m					
54	599.20	154258	580	179.739	N/A
7.33m					
55	605.87	115203	780	179.665	N/A
7.32m					
56	607.26	1.10202e+06	580	161.751	N/A
7.30m					
57	590.25	325810	607	157.107	N/A
7.11m					
58	599.51	175627	498	154.816	N/A
7.55m					
59	615.73	2.05937e+07	585	147.345	N/A
7.40m					
60	572.38	381544	597	146.883	N/A
6.80m					
61	576.44	289927	509	145.037	N/A
6.85m					
62	557.31	243327	651	144.194	N/A
6.49m					
63	574.89	2.80685e+06	579	142.065	N/A
6.70m					
64	595.33	217064	582	140.262	N/A
6.93m					
65	592.78	112236	578	139.268	N/A
7.19m					
66	601.12	214792	687	139.167	N/A
6.90m					
67	596.97	401058	580	138.77	N/A
6.64m					
68	596.88	183980	731	138.407	N/A
6.66m					
69	605.00	196923	645	138.124	N/A
6.63m					
70	624.28	120101	702	134.96	N/A
6.68m					
71	613.74	65220.9	700	134.95	N/A
6.54m					
72	662.45	219994	706	134.663	N/A
6.93m					
73	713.11	84137	720	134.383	N/A
8.34m					
74	706.18	145495	708	134.371	N/A
8.62m					
75	691.32	164370	734	133.882	N/A
7.93m					

76	714.10	112927	859	133.105	N/A
7.28m					
77	741.06	81064	920	132.395	N/A
7.66m					
78	804.12	234355	1049	132.429	N/A
7.90m					
79	822.98	90264.5	869	131.907	N/A
8.02m					
80	832.85	205834	942	131.6	N/A
8.21m					
81	860.39	295080	983	131.305	N/A
9.52m					
82	891.89	244599	891	130.529	N/A
8.71m					
83	941.01	236574	1051	130.064	N/A
8.72m					
84	945.35	5.90819e+08	1051	129.819	N/A
8.33m					
85	942.77	93379.8	1049	129.519	N/A
8.01m					
86	983.41	235777	995	126.097	N/A
8.39m					
87	1043.72	581588	999	125.898	N/A
9.05m					
88	1142.51	286982	1005	124.618	N/A
9.94m					
89	1031.67	108799	989	123.27	N/A
8.37m					
90	1074.67	128401	981	123.027	N/A
8.70m					
91	1026.87	5.90862e+08	987	121.965	N/A
8.54m					
92	1003.39	2.34917e+09	1274	121.202	N/A
8.24m					
93	1012.57	201797	982	120.63	N/A
8.29m					
94	1065.47	128891	974	120.402	N/A
8.60m					
95	1066.71	251783	1023	120.04	N/A
8.50m					
96	1003.03	202755	1037	119.958	N/A
8.05m					
97	981.58	159988	1001	119.906	N/A
8.11m					
98	993.94	322564	989	119.464	N/A
7.99m					
99	991.74	187031	946	119.374	N/A
7.32m					
100	993.97	105857	1142	119.102	N/A
7.14m					
101	976.85	79860.2	1144	119.079	N/A
7.14m					
102	995.50	221920	951	118.929	N/A

7.40m					
103	938.70	90457.6	950	118.854	N/A
6.86m					
104	937.47	314656	939	118.68	N/A
6.62m					
105	936.34	149304	919	118.526	N/A
6.47m					
106	937.20	2.00517e+07	923	118.466	N/A
6.38m					
107	941.85	8.91926e+09	1041	117.759	N/A
6.61m					
108	943.66	159067	1041	117.646	N/A
6.67m					
109	968.74	94109	1041	117.582	N/A
6.73m					
110	1048.24	75924.5	1136	117.307	N/A
7.16m					
111	1057.97	1.13477e+06	1180	117.163	N/A
7.35m					
112	1076.84	236939	1182	116.834	N/A
7.29m					
113	1128.41	73033.1	1188	116.809	N/A
7.36m					
114	1120.40	256617	1178	116.745	N/A
7.28m					
115	1142.22	139713	1205	116.588	N/A
7.34m					
116	1161.78	119681	1389	116.536	N/A
7.36m					
117	1177.39	163665	1523	116.336	N/A
7.38m					
118	1174.59	1.49591e+06	1210	116.279	N/A
7.19m					
119	1171.17	164129	1212	116.271	N/A
7.12m					
120	1158.92	37142.5	1389	116.147	N/A
7.51m					
121	1197.40	46742.8	1217	116.097	N/A
7.93m					
122	1216.58	332484	1343	116.026	N/A
7.59m					
123	1203.00	63012.6	1215	115.981	N/A
7.06m					
124	1205.20	217140	1208	115.942	N/A
7.12m					
125	1200.88	195967	1361	115.919	N/A
6.74m					
126	1201.62	36773.3	1213	115.845	N/A
6.31m					
127	1192.41	175546	1436	115.636	N/A
6.46m					
128	1178.73	118886	1436	115.632	N/A
6.20m					

129	1228.00	92349.2	1435	115.615	N/A
6.29m					
130	1219.99	177369	1435	115.615	N/A
6.44m					
131	1219.26	581658	1435	115.581	N/A
6.30m					
132	1241.64	5.95807e+08	1338	115.248	N/A
6.27m					
133	1238.89	278341	1361	115.15	N/A
6.16m					
134	1248.58	1.60758e+11	1383	115.108	N/A
6.11m					
135	1302.84	142129	1362	115.062	N/A
6.43m					
136	1327.08	80862	1628	110.496	N/A
6.47m					
137	1368.02	119268	1745	110.206	N/A
6.30m					
138	1492.48	37613.6	1747	109.06	N/A
6.74m					
139	1678.08	26897.3	1753	108.847	N/A
7.39m					
140	1722.07	122838	1936	107.952	N/A
7.45m					
141	1781.41	83720.5	2025	107.852	N/A
7.60m					
142	1842.02	48335.6	1971	107.611	N/A
7.75m					
143	1947.55	82681.7	1964	107.512	N/A
7.94m					
144	1933.71	6.0061e+08	1970	107.395	N/A
7.95m					
145	1972.54	74686.6	1970	106.999	N/A
8.15m					
146	1954.03	64469.6	2011	106.981	N/A
6.84m					
147	1951.31	8795.11	1942	106.773	N/A
7.37m					
148	1955.85	975.374	1941	106.647	N/A
7.33m					
149	1965.40	3.42713e+06	2020	106.646	N/A
7.20m					
150	1947.16	78761.9	2019	106.512	N/A
7.08m					
151	1933.35	58093.1	2018	106.506	N/A
6.79m					
152	1964.62	57360.7	2004	106.35	N/A
7.04m					
153	1970.03	69364.7	1881	106.234	N/A
6.61m					
154	1950.37	5.95297e+08	1882	106.112	N/A
6.41m					
155	1939.50	123477	1878	106.099	N/A

6.23m					
156	1909.67	217390	1824	105.998	N/A
5.97m					
157	1879.78	48951.3	1841	105.954	N/A
6.11m					
158	1852.92	2.00151e+07	1828	105.831	N/A
5.57m					
159	1834.71	41082	1828	105.831	N/A
5.37m					
160	1817.06	74661.6	1832	105.797	N/A
5.10m					
161	1814.77	3860.34	1832	105.783	N/A
5.00m					
162	1808.57	62680.3	1842	105.664	N/A
4.96m					
163	1758.15	203506	1712	105.417	N/A
4.69m					
164	1690.86	92262	1712	105.394	N/A
4.39m					
165	1692.49	116450	1741	105.261	N/A
4.28m					
166	1727.47	66436.9	1739	105.171	N/A
4.22m					
167	1716.24	3.89336e+11	1741	105.141	N/A
4.23m					
168	1730.61	1.00493e+07	1750	105.092	N/A
4.06m					
169	1741.79	571328	1742	104.97	N/A
3.93m					
170	1733.52	1.78267e+07	1741	104.953	N/A
3.72m					
171	1730.60	502739	1954	104.847	N/A
3.58m					
172	1753.23	196115	1954	104.847	N/A
3.48m					
173	1755.67	5.67425e+08	2047	104.254	N/A
3.44m					
174	1757.01	82979	2047	104.254	N/A
3.40m					
175	1806.70	93743.3	2049	103.817	N/A
3.14m					
176	1954.89	35559.4	2022	103.736	N/A
3.22m					
177	2026.45	73924	2036	103.596	N/A
3.38m					
178	2044.62	87278.4	2048	103.544	N/A
3.16m					
179	2045.47	124714	2047	103.372	N/A
3.02m					
180	2031.76	130210	2134	103.226	N/A
2.83m					
181	2055.03	35068.6	2631	102.926	N/A
2.78m					

182	2066.12	72599.6	2633	102.919	N/A
2.62m					
183	2030.26	161098	2032	103.01	N/A
2.28m					
184	2020.91	136310	2076	102.829	N/A
2.29m					
185	2018.65	30982.9	2009	102.519	N/A
2.00m					
186	2003.83	6.01768e+08	2012	102.519	N/A
1.86m					
187	2022.74	79395.3	2527	102.476	N/A
1.78m					
188	2005.91	56386.1	2100	102.348	N/A
1.61m					
189	2016.43	115070	2184	102.345	N/A
1.44m					
190	2016.36	94111.4	2147	102.316	N/A
1.32m					
191	2018.53	173633	2083	102.054	N/A
1.21m					
192	2020.19	116259	2085	102.036	N/A
1.05m					
193	2036.41	134852	2081	101.931	N/A
53.90s					
194	2064.99	63033.3	2077	101.905	N/A
45.34s					
195	2083.19	33114.6	2082	101.271	N/A
36.69s					
196	2076.44	242556	2082	101.259	N/A
27.56s					
197	2075.79	192377	2082	101.247	N/A
18.45s					
198	2089.56	5.95726e+08	2101	101.067	N/A
9.32s					
199	2086.89	58925.5	2051	101.046	N/A
0.00s					

```
[32] : SymbolicRegressor(const_range=(-10, 10),
                        function_set=['add', 'sub', 'mul', 'div', 'sin'],
                        generations=200, init_depth=(4, 10), metric='mse',
                        population_size=500, random_state=0, stopping_criteria=0.01,
                        verbose=1)
```

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

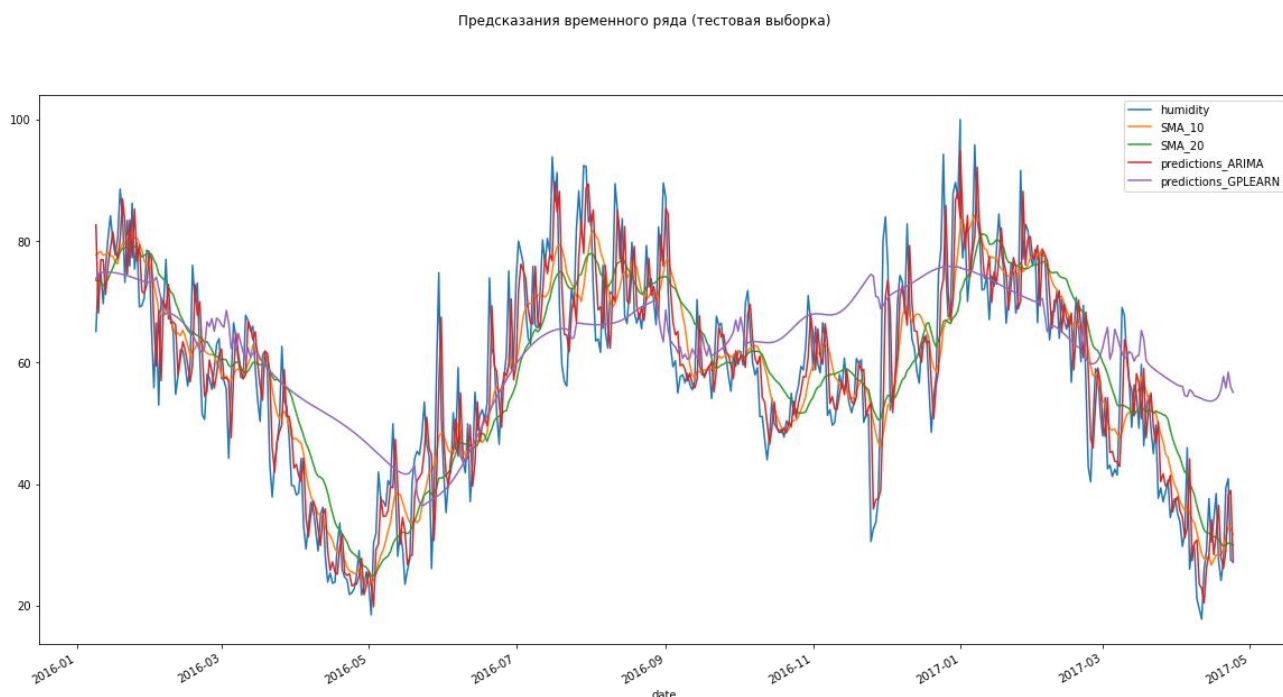
```
[33] : array([73.80469798, 74.62276246, 74.81765215, 74.88961676, 74.91224874,
              74.90617581, 74.87934757, 74.83554142, 74.77687615, 74.70473071])
```

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

2.4.2. Визуализация

Построим график по тестовой выборке:

```
[37]: 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 символьной 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.

2.4.3. Метрики

MAE и MSE:

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

```
[38]: 13.52324614284193
```

```
[39]: mean_absolute_error(test, y_gp)
```

```
[39]: 10.607119049073066
```

2.5. Сравнение 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 выполнения ARIMA также требуется меньше времени.