



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

**Факультет «Радиотехнический»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

**Лабораторная работа №6**

**по дисциплине «Технологии машинного обучения»**

**Выполнил:  
студент группы РТ5-61Б  
М.А. Ходосов**

**2022 г.**

## **Задание лабораторной работы:**

### **Задание:**

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

# Лабораторная работа 6

## Анализ и прогнозирование временного ряда.

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

Задание:

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

```
In [1]: import numpy as np
import pandas as pd
from matplotlib import pyplot
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from statsmodels.tsa.arima.model import ARIMA
from sklearn.model_selection import GridSearchCV
from gplearn.genetic import SymbolicRegressor
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from warnings import simplefilter
simplefilter('ignore')
```

Использован датасет, содержащий данные об изменении численности населения: <https://www.kaggle.com/datasets/census/population-time-series-data?datasetId=51748&sortBy=voteCount>

```
In [2]: data = pd.read_csv('../datasets/lab6_dataset.txt')
data.head()
```

Out[2]:

	date	value
0	1991-07-01	3.526591
1	1991-08-01	3.180891
2	1991-09-01	3.252221
3	1991-10-01	3.611003
4	1991-11-01	3.565869

```
In [3]: data.dtypes
```

Out[3]:

date	object
value	float64
dtype:	object

```
In [4]: # Преобразуем столбец "date" к datetime и установка его в качестве индекса
data['date'] = pd.to_datetime(data['date'])
data.set_index("date",inplace=True)
data.head()
```

Out[4]:

	value
date	
1991-07-01	3.526591
1991-08-01	3.180891
1991-09-01	3.252221
1991-10-01	3.611003
1991-11-01	3.565869

```
In [5]: data.head()
```

Out[5]:

	value
date	
1991-07-01	3.526591
1991-08-01	3.180891
1991-09-01	3.252221
1991-10-01	3.611003
1991-11-01	3.565869

```
In [6]: data.describe()
```

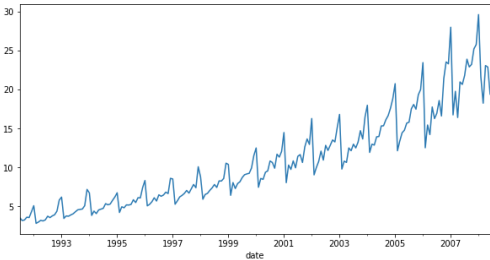
Out[6]:

	value
count	204.000000
mean	10.694430
std	5.956998
min	2.814520
25%	5.844095
50%	9.319345
75%	14.289964
max	29.665356

Визуализация временного ряда

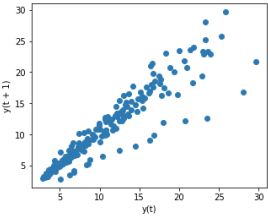
```
In [7]: fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Временной ряд в виде графика')
data.plot(ax=ax, legend=False)
pyplot.show()
```

Временной ряд в виде графика

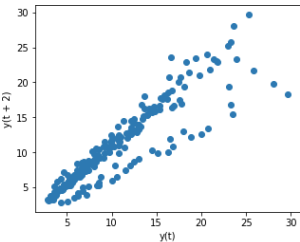


```
In [8]: for i in range(1, 5):
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(5,4))
fig.suptitle(f'Лар порядка {i}')
pd.plotting.lag_plot(data, lag=i, ax=ax)
pyplot.show()
```

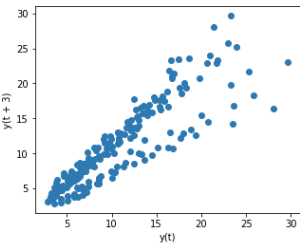
Лар порядка 1



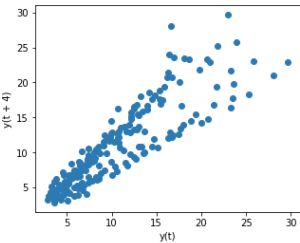
Лар порядка 2



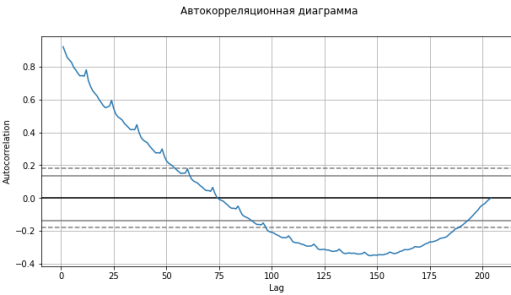
Лар порядка 3



Лар порядка 4

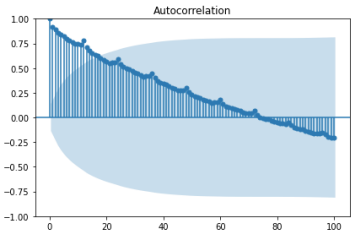


```
In [9]: fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Автокорреляционная диаграмма')
pd.plotting.autocorrelation_plot(data, ax=ax)
pyplot.show()
```



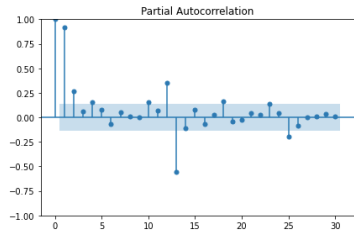
Автокорреляционная функция

```
In [10]: plot_acf(data, lags=100)
plt.tight_layout()
```



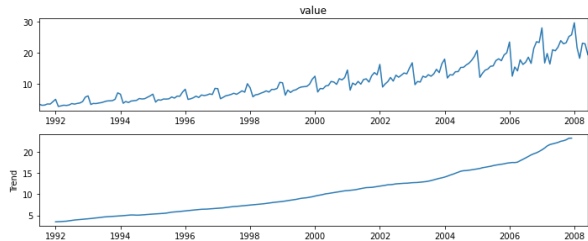
Частичная автокорреляционная функция

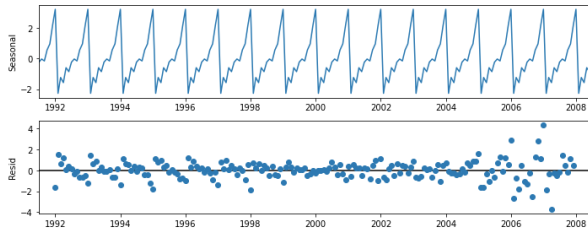
```
In [11]: plot_pacf(data, lags=30, method='ywm')
plt.tight_layout()
```



Декомпозиция временного ряда

```
In [12]: decomposed = seasonal_decompose(data['value'], model = 'add')
fig = decomposed.plot()
fig.set_size_inches((10, 8))
fig.tight_layout()
plt.show()
```





## Разделение временного ряда на обучающую и тестовую выборку

```
In [13]: data_2 = data.copy()
```

```
In [14]: # Целочисленная метка шкалы времени
xnum = list(range(data_2.shape[0]))
# Разделение выборки на обучающую и тестовую
Y = data_2['value'].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]
```

## Прогнозирование временного ряда авторегрессионным методом (ARIMA)

```
In [15]: # Параметры модели (p,d,q)
arima_order = (2,1,0)
# Формирование предсказаний
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])
# Вычисление метрики RMSE
error_arima = mean_squared_error(test, predictions_arima, squared=False)
```

```
In [16]: # Ошибка прогноза
np.mean(Y), error_arima
```

```
Out[16]: (10.694429582156861, 3.087041364389784)
```

```
In [17]: # Записываем предсказания в DataFrame
data_2['predictions_ARIMA'] = (train_size * [np.NaN]) + list(predictions_arima)
```

```
In [18]: fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Предсказания временного ряда (тестовая выборка)')
data_2.plot(ax=ax, legend=True)
pyplot.show()
```



```
In [19]: fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Предсказания временного ряда (тестовая выборка)')
data_2[train_size:].plot(ax=ax, legend=True)
pyplot.show()
```



## Прогнозирование временного ряда методом символьной регрессии

```
In [20]: function_set = ['add', 'sub', 'mul', 'div', 'sin']
SR = SymbolicRegressor(population_size=500, metric='mse',
                       generations=70, stopping_criteria=0.01,
                       init_depth=(4, 10), verbose=1, function_set=function_set,
                       const_range=(-100, 100), random_state=0)
```

```
In [21]: SR.fit(np.array(xnum_train).reshape(-1, 1), train.reshape(-1, 1))
```

Population Average			Best Individual		
Gen	Length	Fitness	Length	Fitness	OOB Fitness
0	263.65	2.60969e+51	137	52.4421	N/A
1	150.29	2.31086e+10	44	44.7186	N/A
2	100.75	3.3938e+08	1	13.3697	N/A
3	49.61	3.36747e+10	141	3.58406	N/A
4	40.66	4.94445e+10	106	2.88037	N/A
5	88.37	3.35228e+09	102	2.08309	N/A
6	107.64	5.72434e+08	41	1.96391	N/A
7	94.38	3.9808e+09	45	1.88267	N/A
8	74.23	2.83158e+10	47	1.85907	N/A
9	49.31	6.62331e+08	37	1.76829	N/A
10	42.09	1.09979e+09	41	1.7546	N/A
11	42.90	8.59967e+08	43	1.74225	N/A
12	37.92	2.01293e+09	41	1.73156	N/A
13	33.90	4.32481e+08	33	1.67804	N/A
14	34.83	4.30404e+08	43	1.678	N/A
15	38.83	5.605e+10	41	1.67615	N/A
16	38.28	356914	47	1.67239	N/A
17	36.49	199911	42	1.67236	N/A
18	34.92	370767	34	1.66796	N/A
19	32.78	195144	31	1.66366	N/A
20	32.41	439781	32	1.66181	N/A
21	31.77	361032	36	1.65974	N/A
22	32.02	895261	36	1.65974	N/A
23	31.27	307091	29	1.65937	N/A
24	30.23	330752	31	1.65753	N/A
25	30.31	387730	32	1.65762	N/A
26	30.38	222864	31	1.65753	N/A
27	29.91	344917	31	1.65753	N/A
28	29.93	306110	31	1.65753	N/A
29	29.25	167424	31	1.65753	N/A
30	29.89	335435	31	1.65753	N/A
31	30.03	627881	31	1.65753	N/A
32	29.41	211590	31	1.65753	N/A
33	29.95	210214	31	1.65753	N/A
34	29.91	181278	31	1.65753	N/A
35	29.72	363114	31	1.65753	N/A
36	30.56	290978	31	1.65753	N/A
37	29.56	204967	31	1.65753	N/A
38	29.19	181335	31	1.65753	N/A
39	29.11	154103	25	1.6573	N/A
40	27.55	102759	25	1.6573	N/A
41	24.98	61146.4	25	1.6573	N/A
42	23.32	34651.5	25	1.6573	N/A
43	24.03	1692.08	25	1.6573	N/A
44	23.19	167542	25	1.6573	N/A
45	23.82	125055	25	1.6573	N/A
46	24.12	4137.5	25	1.6573	N/A
47	23.86	27400.8	25	1.6573	N/A
48	24.34	1979.09	25	1.6573	N/A

```
Out[21]: SymbolicRegressor
add(div(X0, 14.579), sub(sub(div(X0, X0), sin(sub(sub(-10.392, 71.983), sin(sin(sin(sin(sin(div(X0, 14.579))))))))), sin(-83.389)))
```

```
In [22]: print(SR_program)
add(div(X0, 14.579), sub(sub(div(X0, X0), sin(sub(sub(-10.392, 71.983), sin(sin(sin(sin(sin(div(X0, 14.579))))))))), sin(-83.389)))
```

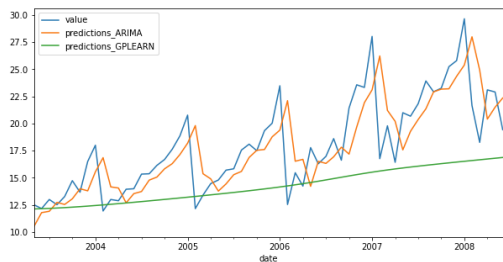
```
In [23]: # Предсказания
y_sr = SR.predict(np.array(xnum_test).reshape(-1, 1))
y_sr[:10]

Out[23]: array([12.1262825, 12.14957997, 12.17778464, 12.21103469, 12.24916232,
12.29180476, 12.33849853, 12.38874853, 12.44207316, 12.49803004])
```

```
In [24]: # Записываем предсказания в DataFrame
data_2['predictions_GPLEARN'] = (train_size * [np.NaN]) + list(y_sr)
```

```
In [25]: fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Предсказания временного ряда (тестовая выборка)')
data_2[train_size:].plot(ax=ax, legend=True)
pyplot.show()
```

Предсказания временного ряда (тестовая выборка)



```
In [26]: error_SR = mean_squared_error(test, y_sr, squared=False)
```

```
In [27]: # Ошибка прогноза  
np.mean(Y), error_SR
```

```
Out[27]: (10.694429582156861, 4.902532668025038)
```

### Качество прогноза моделей

```
In [28]: def print_metrics(y_test, y_pred):  
    print(f"R^2: {r2_score(y_test, y_pred)}")  
    print(f"MSE: {mean_squared_error(y_test, y_pred, squared=False)}")  
    print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
```

```
In [29]: print("ARIMA")  
print_metrics(test, predictions_arima)
```

```
print("\nGPLEARN")  
print_metrics(test, y_sr)
```

```
ARIMA  
R^2: 0.4702908041729138  
MSE: 3.087041364389784  
MAE: 2.2553908226247605
```

```
GPLEARN  
R^2: -0.3359604683725561  
MSE: 4.902532668025038  
MAE: 3.806162938096472
```

**Вывод:** Лучше оказалась ARIMA.