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 **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

О Т Ч Е Т
по лабораторной работе № 6

Дисциплина: Технологии машинного обучения
Тема: «Анализ и прогнозирование временного ряда»

Студент ИУ5Ц-82Б _____ А.Н. Свинцов
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Ю.Е. Гапанюк
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ А.Н. Нардид
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Лабораторная работа №6 : "Анализ и прогнозирование временного ряда"

Для работы используется датасет с информацией об изменеии числинности населения

```
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

data = pd.read_csv('POP.csv')
data.head()
```

```
  realtime_start     value        date realtime_end
0     2019-12-06  156309.0  1952-01-01   2019-12-06
1     2019-12-06  156527.0  1952-02-01   2019-12-06
2     2019-12-06  156731.0  1952-03-01   2019-12-06
3     2019-12-06  156943.0  1952-04-01   2019-12-06
4     2019-12-06  157140.0  1952-05-01   2019-12-06
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   realtime_start  816 non-null    object
 1   value           816 non-null    float64
 2   date            816 non-null    object
 3   realtime_end    816 non-null    object
dtypes: float64(1), object(3)
memory usage: 25.6+ KB
```

```
#Проверка на пропуски
data.isnull().sum()
```

```
realtime_start    0
value             0
date              0
```

```
realtime_end      0
dtype: int64
```

Проигнорируем данные о реальном времени, поскольку мы концентрируемся только на диапазоне дат, в котором меняется население.

```
data = data.drop(['realtime_start','realtime_end'],axis=1)

# Преобразование столбца даты в объект datetime и установка его в качестве индекса
data['date'] = pd.to_datetime(data['date'])
data.set_index('date',inplace=True)
data.head()
```

```
               value
date
1952-01-01  156309.0
1952-02-01  156527.0
1952-03-01  156731.0
1952-04-01  156943.0
1952-05-01  157140.0
```

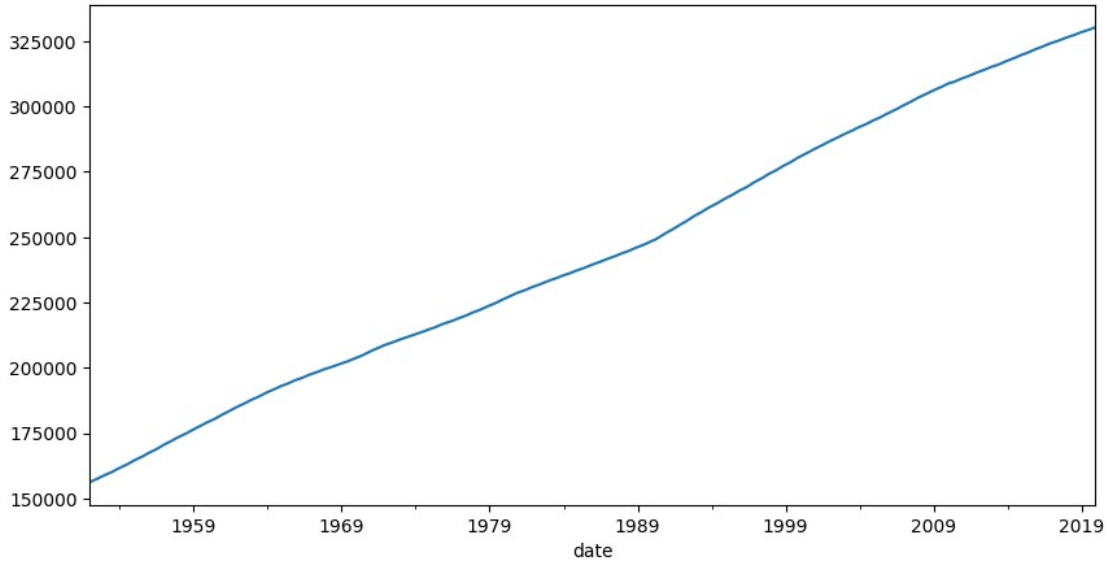
```
data.describe()
```

```
               value
count     816.000000
mean   243847.767826
std     50519.140567
min    156309.000000
25%    201725.250000
50%    239557.500000
75%    289364.250000
max    330309.946000
```

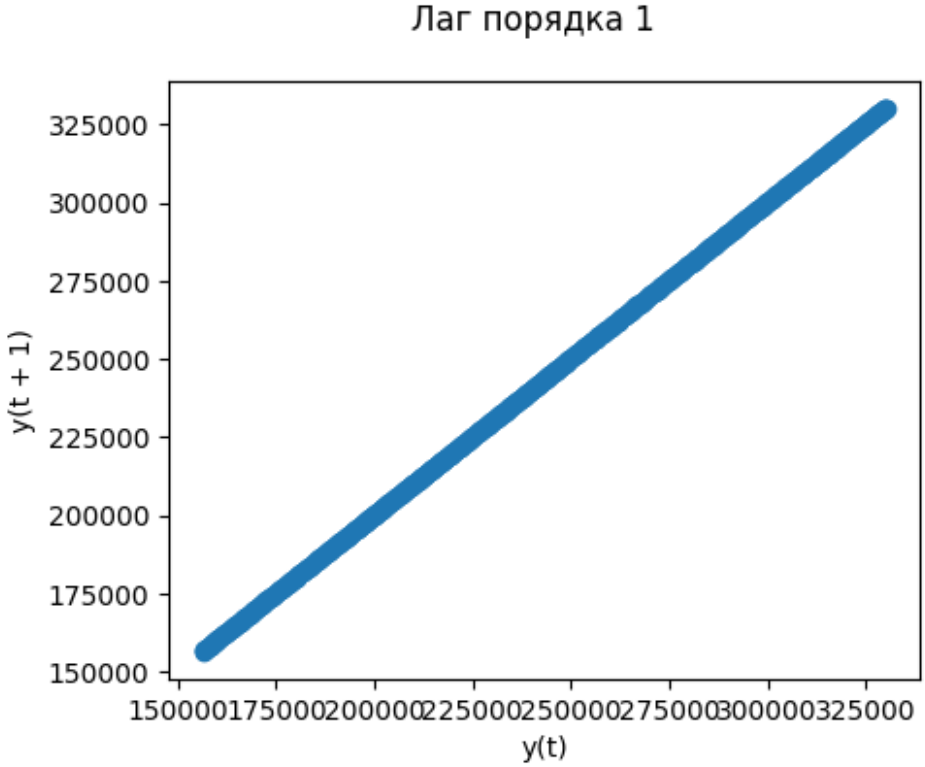
Визуализация временного ряда

```
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()
```

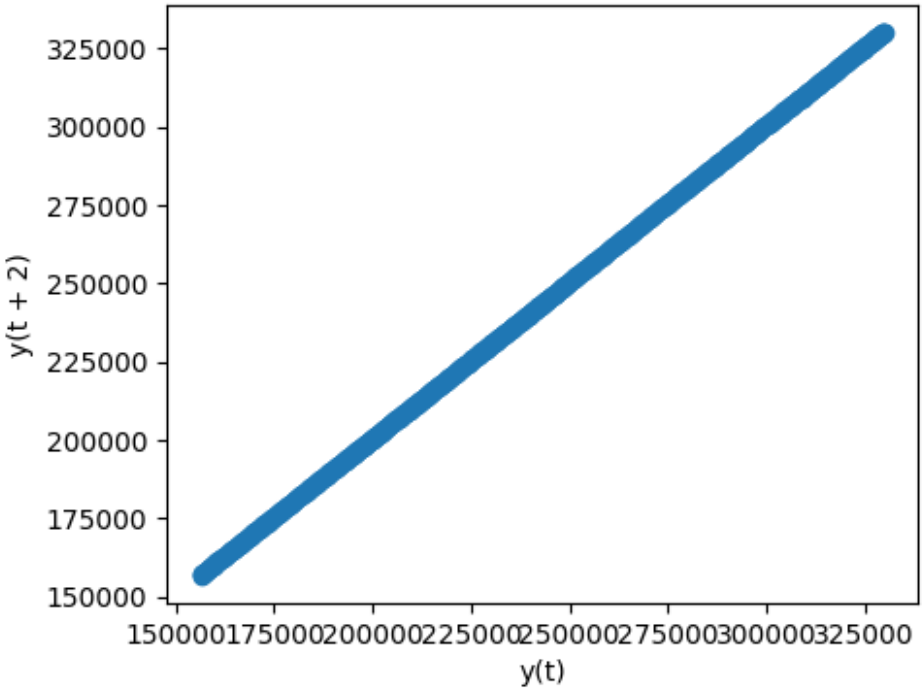
Временной ряд в виде графика



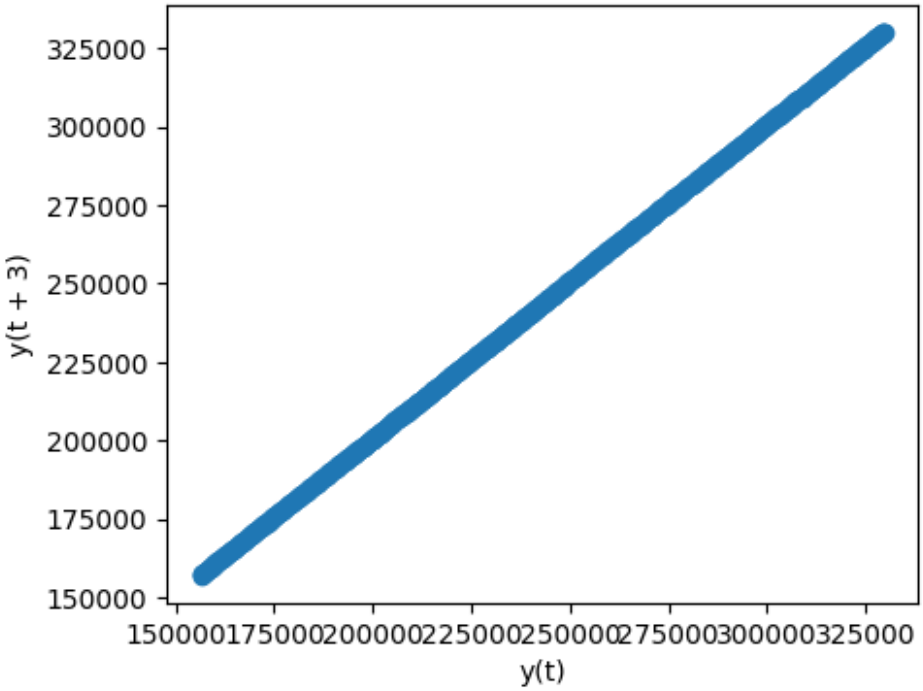

```
for i in range(1, 5):
    fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row',
figsize=(5,4))
    fig.suptitle(f'Лаг порядка {i}')
    pd.plotting.lag_plot(data, lag=i, ax=ax)
    pyplot.show()
```

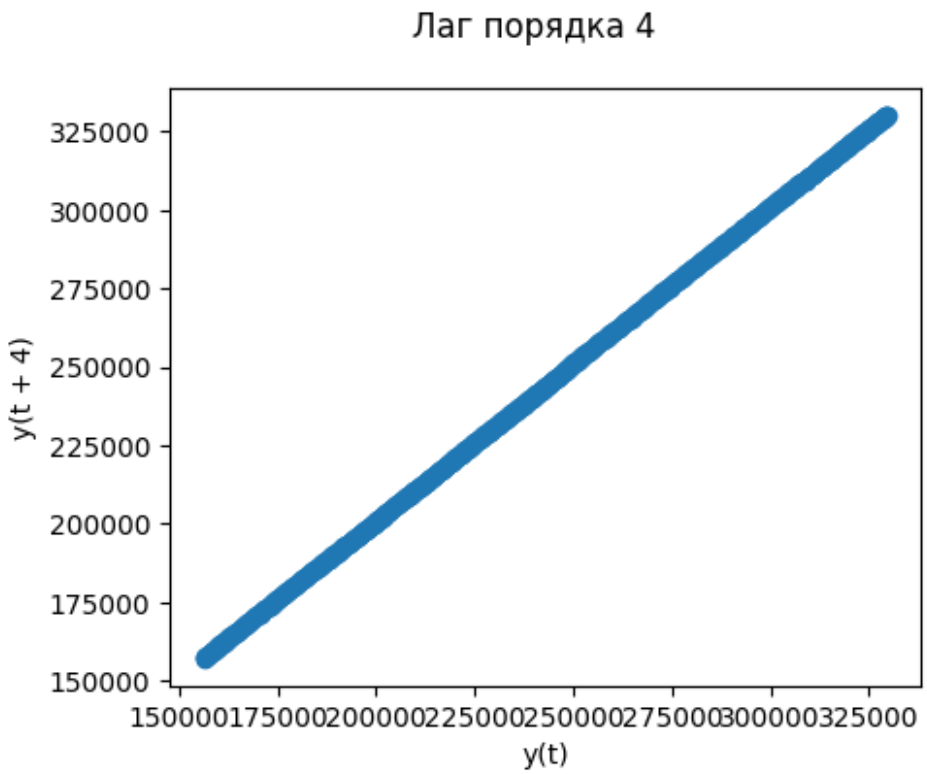


Лаг порядка 2

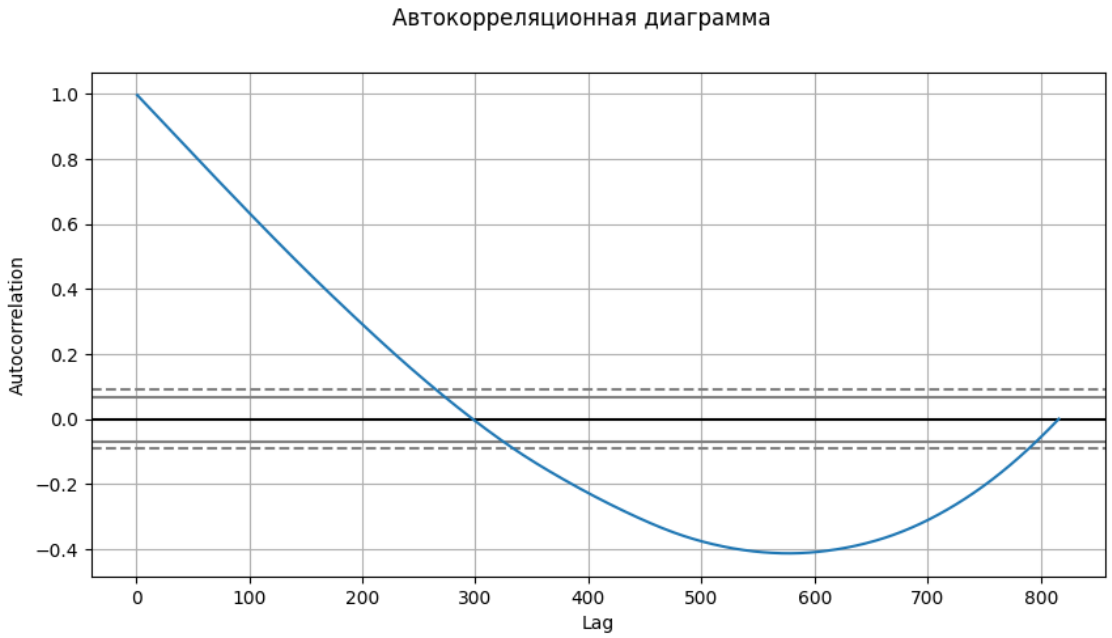


Лаг порядка 3



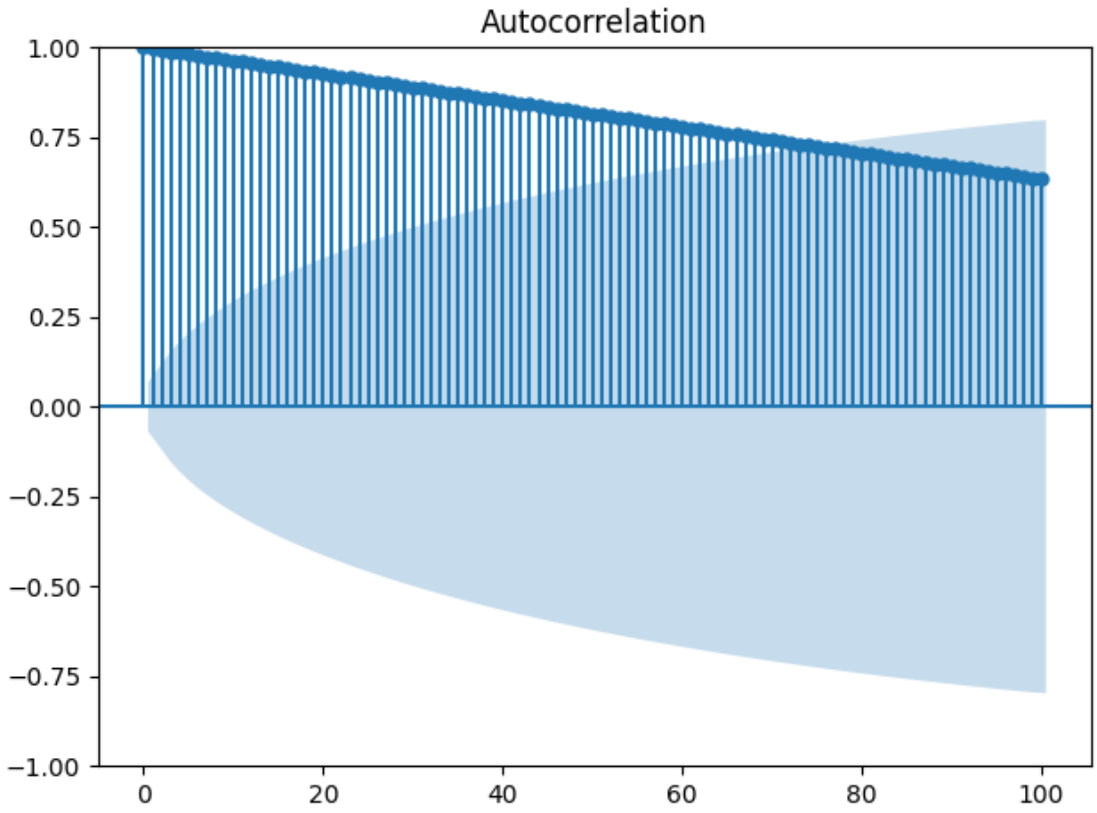



```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row',
figsize=(10,5))
fig.suptitle('Автокорреляционная диаграмма')
pd.plotting.autocorrelation_plot(data, ax=ax)
pyplot.show()
```



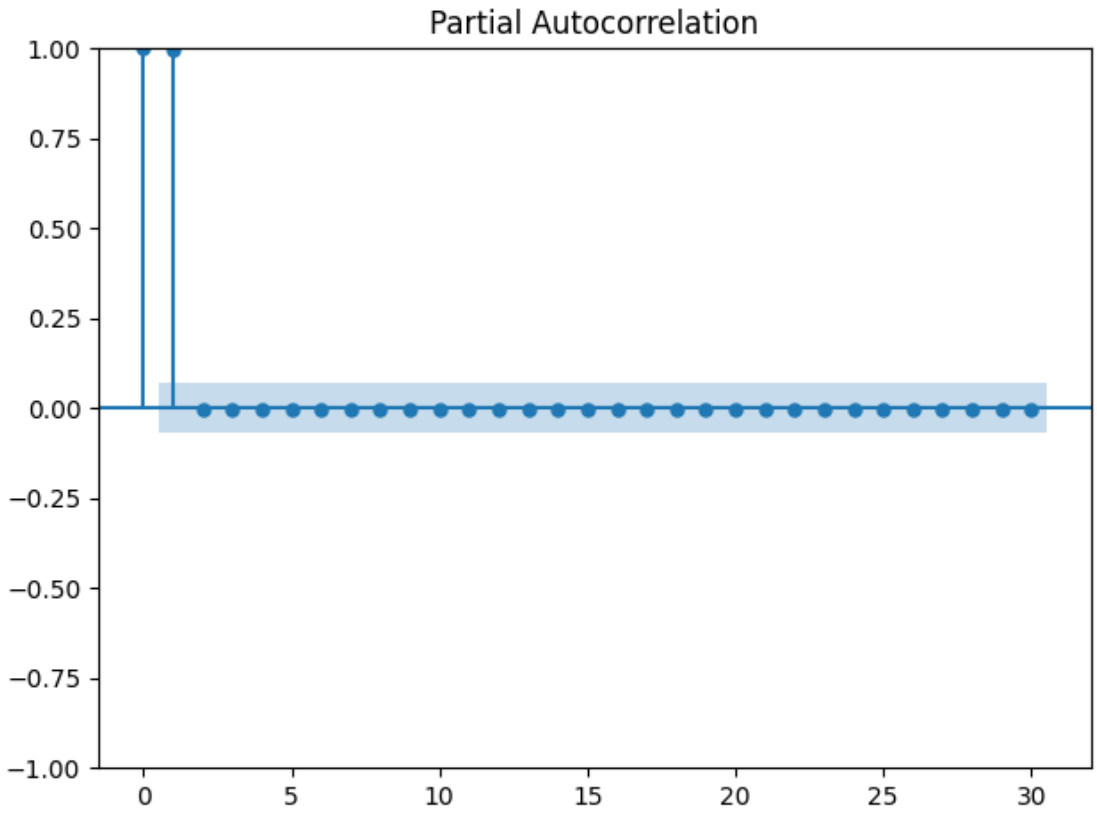
Автокорреляционная функция

```
plot_acf(data, lags=100)
plt.tight_layout()
```



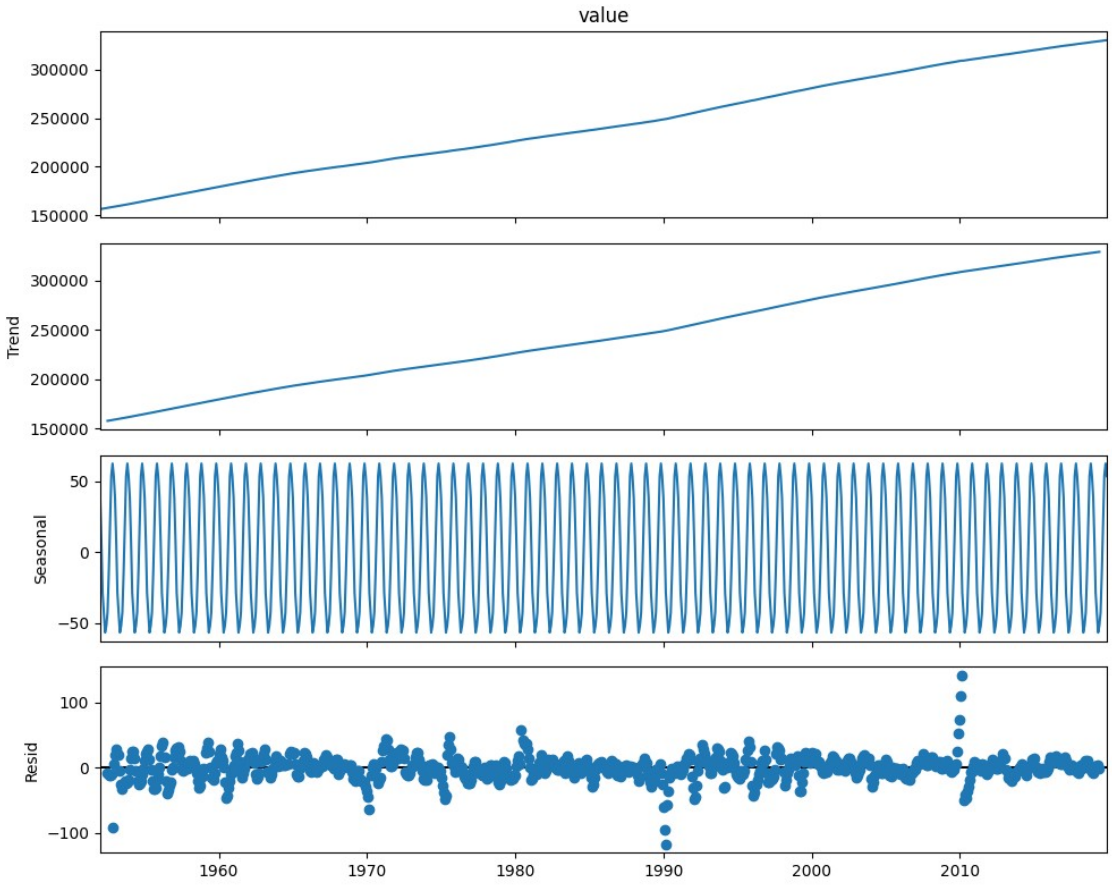
Частичная автокорреляционная функция

```
plot_pacf(data, lags=30)
plt.tight_layout()
```



Декомпозиция временного ряда

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



Наблюдается положительная динамика с 1952 по 2019 год.

Разделение временного ряда на обучающую и тестовую выборку

```
data_2 = data.copy()

# 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)

```
# 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]
```

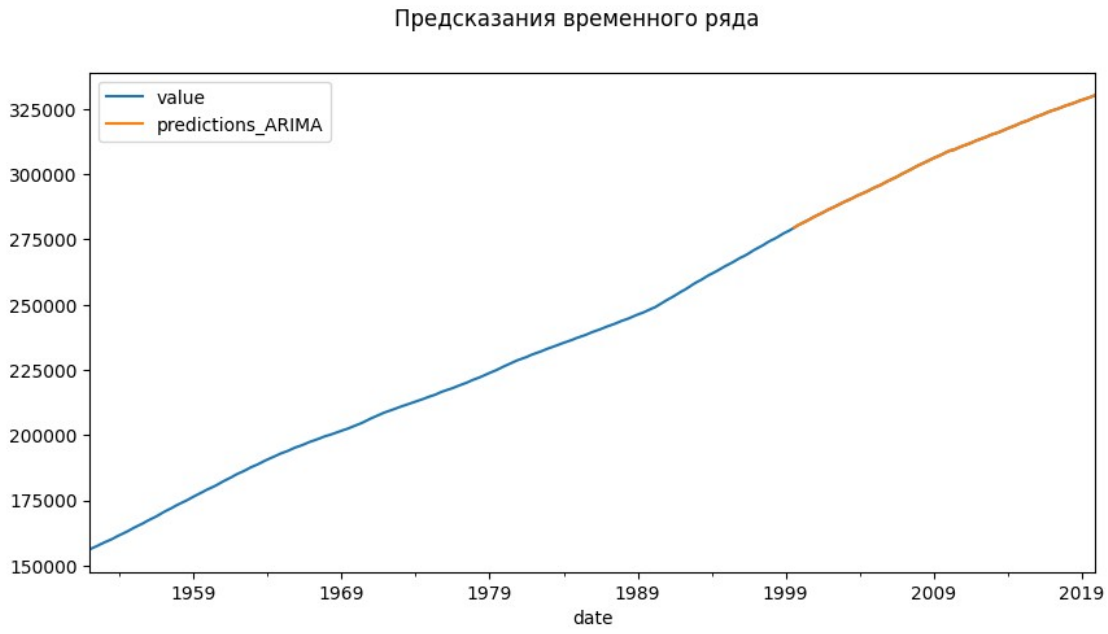
```
    predictions_arima.append(yhat_arima)
    history_arima.append(test[t])
# Вычисление метрики RMSE
error_arima = mean_squared_error(test, predictions_arima,
squared=False)

# Ошибка прогноза
np.mean(Y), error_arima
```

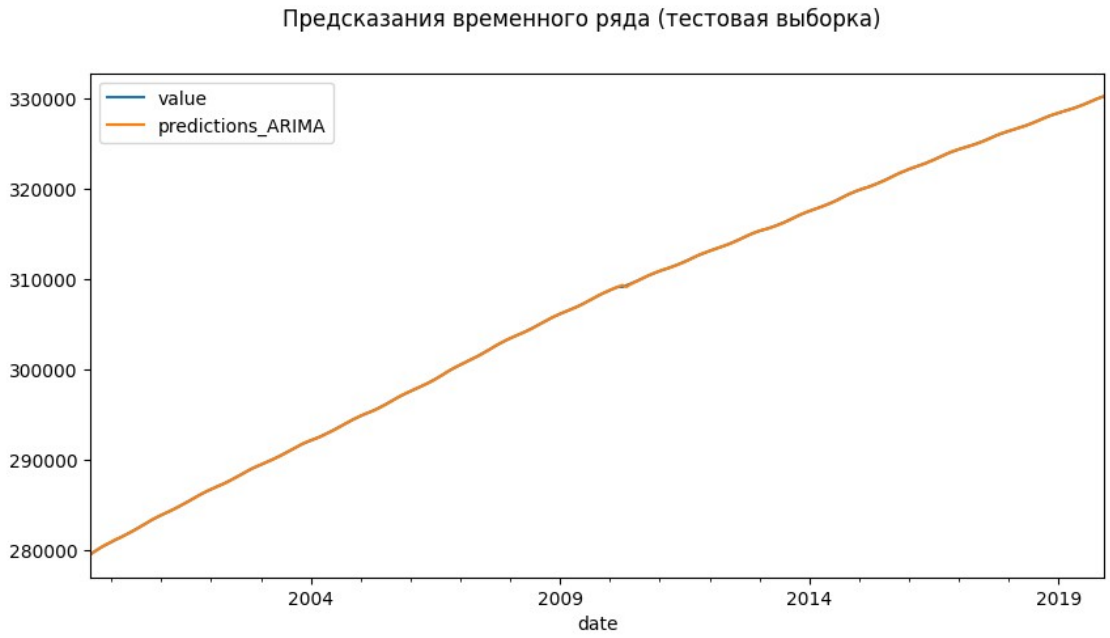
```
(243847.7678259804, 24.173499561405922)
```

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

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




```
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 регресии

```
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)

SR.fit(np.array(xnum_train).reshape(-1, 1), train.reshape(-1, 1))
```

```
C:\Users\Артемий\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\utils\validation.py:1143: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

    |   Population Average    |             Best Individual
|
---- -------------------------
------------------------------------------ ----------
 Gen   Length          Fitness   Length          Fitness      OOB
Fitness  Time Left
   0   263.65      2.43463e+63       23      7.14077e+09
N/A      1.77m
   1   130.36      5.77055e+16       43      6.06688e+09
N/A     35.50s
   2    53.10      4.58992e+15       34      3.54847e+09
N/A     21.01s
   3    34.28      1.99853e+19       13      1.42699e+09
N/A     18.09s
```

```
   4    35.05      2.10424e+16       38      1.04052e+09
N/A     17.51s
   5    30.47      2.56729e+16       36      4.29436e+08
N/A     16.71s
   6    31.30      3.00498e+16       50      6.39791e+07
N/A     16.85s
   7    38.37      8.59782e+15       35      1.51165e+07
N/A     16.84s
   8    43.37      5.29474e+15       47      4.76034e+06
N/A     16.39s
   9    37.70      8.42452e+15       35      4.14545e+06
N/A     16.02s
  10    40.68      5.69103e+15       32      3.65059e+06
N/A     16.41s
  11    45.38      5.71108e+15       29      3.65015e+06
N/A     16.88s
  12    41.36      5.72894e+15       29      3.65015e+06
N/A     15.28s
  13    35.07      3.58233e+15       29      3.65015e+06
N/A     14.79s
  14    33.33      8.46569e+15       35      3.53261e+06
N/A     14.52s
  15    31.43      3.14997e+19       35      3.53261e+06
N/A     13.50s
  16    30.19      1.42657e+16       35      3.53261e+06
N/A     13.20s
  17    30.81      2.81228e+15       35      3.53261e+06
N/A     13.94s
  18    33.31      5.72757e+15       35      3.53261e+06
N/A     12.93s
  19    33.71      1.26632e+16       35      3.50395e+06
N/A     12.50s
  20    34.95      1.70198e+16       35      3.50395e+06
N/A     12.79s
  21    42.21      6.70957e+15       35      3.50395e+06
N/A     12.97s
  22    54.68      6.78469e+15       35      3.50395e+06
N/A     13.55s
  23    50.99      6.47928e+18      102      3.50387e+06
N/A     12.99s
  24    42.69      8.57551e+15       71      3.50376e+06
N/A     12.20s
  25    59.07      6.73374e+21       85      3.49756e+06
N/A     12.76s
  26    89.07      1.51918e+25       85      3.49756e+06
N/A     14.62s
  27   100.70      2.98833e+18       91      3.48956e+06
N/A     15.42s
  28   120.58      7.92131e+23       91      3.48956e+06
N/A     16.57s
```

```
  29   142.26      1.91023e+18      127      3.48498e+06
N/A     17.12s
  30   116.37       6.9315e+21       54      3.46676e+06
N/A     15.17s
  31   103.96      2.33782e+22       54      3.46676e+06
N/A     14.04s
  32   107.16      2.82439e+18       54      3.46676e+06
N/A     14.03s
  33   110.56      4.95099e+26      112      3.45858e+06
N/A     16.46s
  34    94.20      1.96986e+18      114      3.45249e+06
N/A     12.10s
  35    77.71       6.0703e+15      133      3.43034e+06
N/A     10.79s
  36   111.25      5.62717e+15       79      3.42948e+06
N/A     12.14s
  37   142.44       1.4552e+18      246      3.41658e+06
N/A     13.28s
  38   171.28      3.11029e+19      187      3.36822e+06
N/A     15.01s
  39   197.58       2.8446e+16      187      3.36419e+06
N/A     16.16s
  40   213.08      1.12226e+16      212      3.35931e+06
N/A     15.76s
  41   193.33      7.07447e+17      181      3.35563e+06
N/A     14.02s
  42   200.58      9.48793e+19      308      3.25166e+06
N/A     13.96s
  43   203.16       6.9535e+17      308      3.24914e+06
N/A     12.84s
  44   271.65      2.48275e+15      434      3.17665e+06
N/A     14.93s
  45   340.95      1.45248e+18      434      3.17665e+06
N/A     17.53s
  46   407.23       2.9286e+14      874      3.13466e+06
N/A     18.52s
  47   475.59      8.20919e+13      857      3.13086e+06
N/A     19.41s
  48   698.39      6.58531e+17     1124       3.1245e+06
N/A     25.36s
  49   871.75      5.67064e+14     1140       3.1232e+06
N/A     28.84s
  50  1008.67      1.44739e+18     1126      3.11533e+06
N/A     31.04s
  51  1040.20      8.00984e+13     1337       3.1087e+06
N/A     33.62s
  52  1087.90       4.8939e+10     1352      3.10262e+06
N/A     31.57s
  53  1212.74      6.88053e+18     1338      3.09244e+06
N/A     31.48s
```

```
  54  1332.59      9.76027e+14     1324      3.09015e+06
N/A     31.34s
  55  1375.70       4.2908e+14     1361      3.08045e+06
N/A     29.90s
  56  1400.24      4.21109e+14     1622      3.07579e+06
N/A     27.95s
  57  1485.49      3.56926e+14     1361       3.0712e+06
N/A     27.59s
  58  1565.07      5.99454e+17     1379      3.06568e+06
N/A     26.42s
  59  1519.96      1.18494e+10     1452      3.05779e+06
N/A     23.41s
  60  1441.96      3.61367e+14     1452      3.05779e+06
N/A     20.24s
  61  1484.91      3.60553e+14     1441      3.04915e+06
N/A     18.26s
  62  1502.33      8.71965e+13     1441      3.04637e+06
N/A     16.17s
  63  1499.87      5.11657e+12     1470      3.04262e+06
N/A     13.94s
  64  1457.59      3.97956e+14     1453      3.03789e+06
N/A     11.39s
  65  1514.59      3.44098e+14     1735      3.03427e+06
N/A      9.42s
  66  1597.79      6.70466e+14     1728      3.02874e+06
N/A      7.36s
  67  1652.89      3.52673e+14     1753      3.02842e+06
N/A      5.02s
  68  1696.85      3.44312e+14     1728      3.02504e+06
N/A      2.59s
  69  1756.59      5.96419e+17     1817      3.01702e+06
N/A      0.00s

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

print(SR._program)

sub(mul(sub(-36.019, -77.644), add(add(add(X0, 51.302),
add(add(add(X0, 55.353), mul(55.353, 65.255)), X0)),
sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(su
b(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(add(X0,
X0), sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))),
sin(mul(sub(sub(sub(add(X0, X0), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), add(add(add(add(X0,
```

51.302), add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(add(X0, sub(mul(sub(-36.019, -
77.644), add(X0, X0)), sub(add(X0, X0), add(add(add(X0, X0), X0),
sin(X0))))), 51.302), add(X0, X0))))), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))))), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(add(X0, 51.302), 51.302), add(add(X0, X0), X0)),
add(X0, X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(X0, sub(mul(sub(-36.019, -77.644), add(X0, X0)),
sub(add(X0, X0), mul(55.353, mul(sub(-36.019, -77.644),
add(sub(sub(sub(sub(sub(sub(sub(sub(sub(add(X0, X0), sin(mul(sub(-
36.019, -77.644), add(add(add(add(X0, 51.302), add(add(X0, X0), X0)),
add(X0, X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), add(add(add(X0,
sub(mul(sub(-36.019, -77.644), add(X0, X0)), sub(add(X0, X0),
add(add(add(X0, X0), X0), sin(X0))))), 51.302), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(add(add(add(X0, 51.302), add(add(X0, X0), X0)),
add(X0, X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), mul(sub(-36.019, -
77.644), add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0,
X0)), add(X0, X0))))))), sin(mul(sub(-36.019, -77.644), add(add(add(X0,
sub(mul(sub(-36.019, -77.644), add(X0, X0)), sub(add(X0, X0),
add(add(add(X0, X0), X0), sin(X0))))), 51.302), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(add(add(add(X0, X0), add(add(X0, X0), X0)),
add(X0, X0)), add(X0, X0))))), sin(X0)))))))), 51.302), add(X0,
X0))))), sin(mul(sub(-36.019, -77.644), add(add(add(X0, sub(mul(sub(-
36.019, -77.644), add(X0, X0)), sub(add(X0, X0), mul(55.353, mul(sub(-
36.019, -77.644), add(sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(add(X0,
X0), sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(add(add(add(X0, 51.302), add(add(X0, X0), X0)),
add(X0, X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(X0, sub(mul(sub(-36.019, -77.644), add(X0, X0)),
sub(add(X0, X0), add(add(add(X0, X0), X0), sin(X0))))), 51.302),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), add(add(add(add(X0,
51.302), add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(add(add(add(X0, 51.302), add(add(X0, X0), X0)),
add(X0, X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302), add(add(X0,
X0), X0)), add(X0, X0)), add(X0, X0))))))), sin(mul(sub(-36.019, -
77.644), add(add(add(X0, sub(mul(sub(-36.019, -77.644), add(X0, X0)),

sub(add(X0, X0), add(add(add(X0, X0), X0), sin(X0))))), 51.302),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), add(add(add(add(X0,
51.302), add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, X0), add(add(X0,
X0), X0)), add(X0, X0)), add(X0, X0))))), sin(X0)))))))), 51.302),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), add(add(add(add(X0,
51.302), add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(sin(X0), add(X0, X0)), add(X0,
X0))))), sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(add(add(X0, sub(mul(sub(-36.019, -77.644), sub(-
36.019, -77.644)), sub(add(X0, X0), mul(55.353, mul(sub(-36.019, -
77.644), add(add(add(X0, add(add(add(X0, X0), X0), X0)),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))), sin(X0)))))))), 51.302), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(add(add(add(X0, 51.302), add(add(X0, X0), X0)),
add(X0, X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(X0, sub(mul(add(add(add(X0, 51.302), add(add(X0, X0),
X0)), add(X0, X0)), add(X0, X0)), sub(add(X0, X0), add(add(add(X0,
X0), X0), sin(X0))))), 51.302), add(X0, X0))))), sin(mul(sub(-36.019,
-77.644), add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0,
X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), add(add(sin(X0),
add(X0, X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, X0), add(X0, sub(mul(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0)), sub(add(X0, X0),
add(add(add(X0, X0), X0), sin(X0)))))), add(X0, X0)), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(add(X0, sub(mul(sub(-36.019, -
77.644), add(X0, X0)), sub(add(X0, X0), add(add(add(X0, X0), X0),
sin(X0))))), 51.302), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), add(add(add(add(X0,
51.302), add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(add(add(add(X0, 51.302), add(add(X0, X0), X0)),
add(X0, X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(X0, sub(mul(sub(-36.019, -77.644), add(X0, X0)),
sub(add(X0, X0), mul(55.353, mul(sub(-36.019, -77.644),
add(add(add(X0, add(add(add(X0, X0), X0), X0)), X0), sin(X0)))))))),
51.302), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(add(X0,
51.302), X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(X0, sub(mul(sub(-36.019, -77.644), add(X0, X0)),
sub(add(X0, X0), mul(55.353, mul(sub(-36.019, -77.644),
add(add(add(X0, add(add(add(X0, X0), X0), X0)), X0), sin(X0)))))))),
51.302), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(add(X0,
51.302), X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),

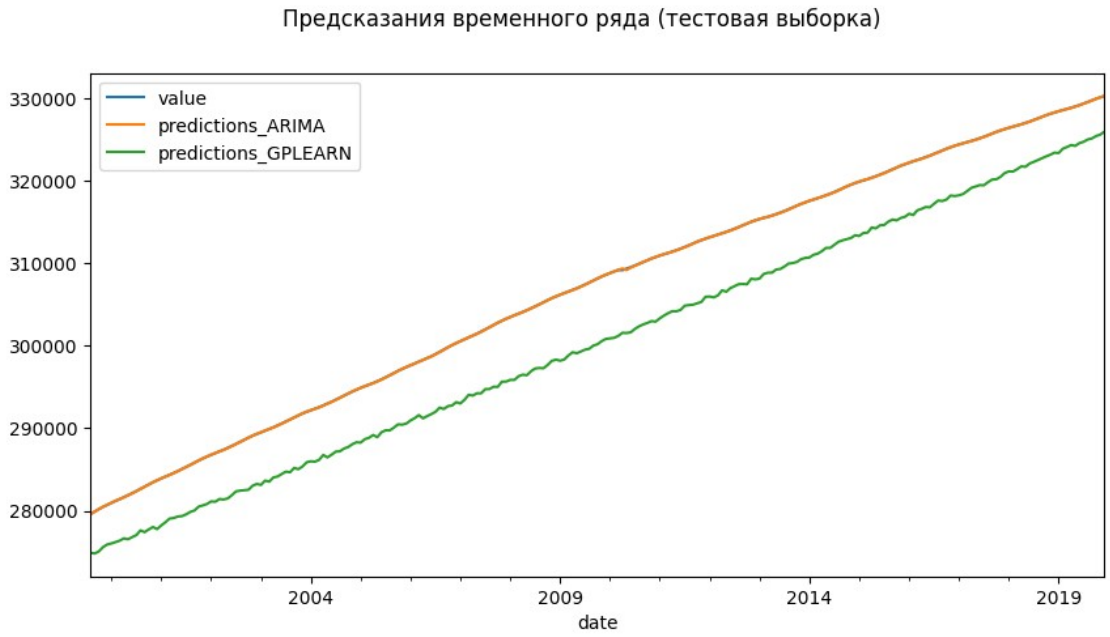
```
add(add(sin(X0), add(X0, X0)), add(X0, X0))))), sin(mul(sub(-36.019, -
77.644), add(add(X0, X0), X0)))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), add(add(add(add(X0,
51.302), add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(add(add(X0, sub(mul(sub(-36.019, -77.644),
add(X0, X0)), sub(sub(sub(sub(sub(sub(sub(sub(sub(sub(add(X0, X0),
sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))), add(add(X0, X0),
X0)), sin(mul(sub(-36.019, -77.644), add(add(add(X0, sub(mul(sub(-
36.019, -77.644), add(X0, X0)), sub(add(X0, X0), add(add(add(X0, X0),
X0), sin(X0))))), 51.302), add(X0, X0))))), sin(mul(sub(-36.019, -
77.644), add(sub(-36.019, -77.644), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(add(add(add(X0, 51.302), add(add(X0, X0), X0)),
add(X0, X0)), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), add(add(add(add(X0,
51.302), add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(add(X0, sub(mul(sub(-36.019, -
77.644), add(X0, X0)), sub(add(X0, X0), mul(55.353, mul(sub(-36.019, -
77.644), add(add(add(X0, add(add(add(X0, X0), X0), X0)),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))), sin(X0)))))))), 51.302), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0)))), add(X0, X0))))), sin(mul(sub(-36.019, -77.644),
add(add(add(add(X0, 51.302), add(add(X0, X0), X0)), add(X0, X0)),
add(X0, X0))))), sin(mul(sub(-36.019, -77.644), add(add(55.353,
51.302), add(X0, X0)))))))), 51.302), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(add(add(add(X0, 51.302), add(add(X0, X0), X0)),
add(sub(-36.019, -77.644), add(X0, X0))), add(X0, X0))))),
sin(mul(sub(-36.019, -77.644), add(add(add(add(X0, 51.302),
add(add(X0, X0), X0)), add(X0, X0)), add(X0, X0))))), sin(mul(sub(-
36.019, -77.644), add(-36.019, add(X0, X0)))))))), sub(add(add(X0, X0),
X0), mul(55.353, 65.255)))
```

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

```
array([274891.75793852, 274817.36307349, 275068.42349884,
275594.91553188,
       275909.57465535, 276033.9204471 , 276192.3291504 ,
276368.95663777,
       276651.56236873, 276542.01774132])
```

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

```
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()
```




```
error_SR = mean_squared_error(test, y_sr, squared=False)

# Ошибка прогноза
np.mean(Y), error_SR
```

```
(243847.7678259804, 6510.330169456957)
```

Качество прогноза моделей

```
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)}")

print("ARIMA")
print_metrics(test, predictions_arima)

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

```
ARIMA
R^2: 0.9999973075905816
MSE: 24.173499561405922
MAE: 16.034435650864996

GPLEARN
```

```
R^2: 0.8047153645391025
MSE: 6510.330169456957
MAE: 6443.710113418146
```

Вывод

Обе модели, ARIMA и GPLEARN, показали хороший результат. Лучшей по всем используемым метрикам оказалась модель ARIMA.