Московский государственный технический университет им. Н. Э. Баумана

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

«Анализ и прогнозирование временного ряда»

Выполнила:	Проверил:
Шимолина П.К.,	Нардид А.Н.,
группа ИУ5-61Б	каф. ИУ5
Дата:	Дата:
Подпись:	Подпись:

2023
Москва

lab-6

June 14, 2023

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

[6]:
```
df = pd.read_csv('births.csv', index_col = "Date", parse_dates = True)
```

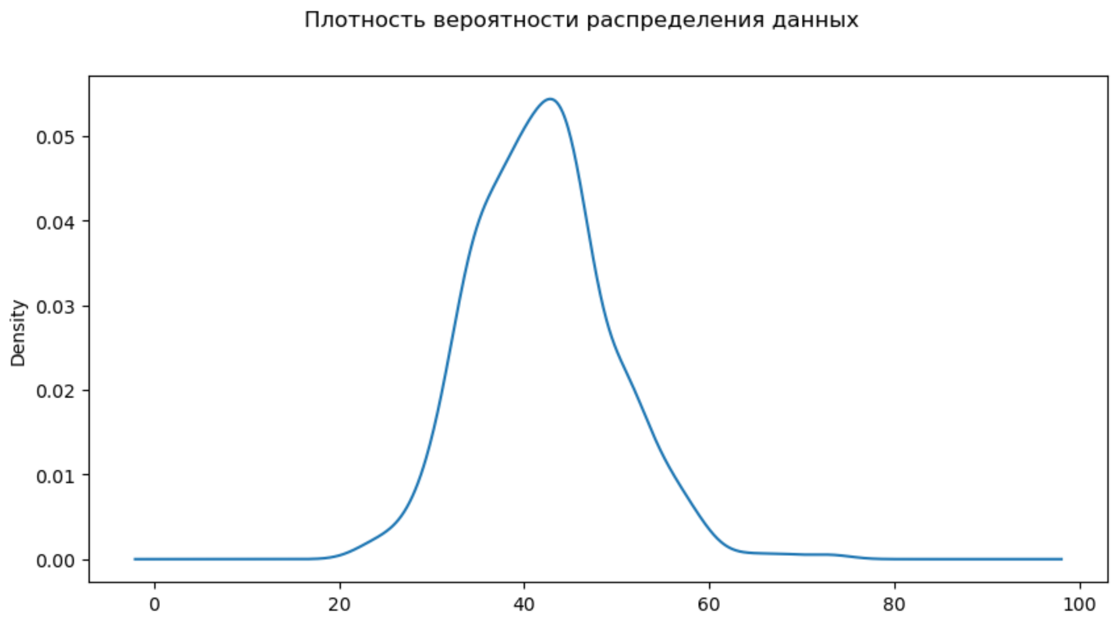
[7]:
```
df.head()
```

[7]:
```
            Births
Date
1959-01-01      35
1959-01-02      32
1959-01-03      30
1959-01-04      31
1959-01-05      44
```

[8]:
```
df.shape
```

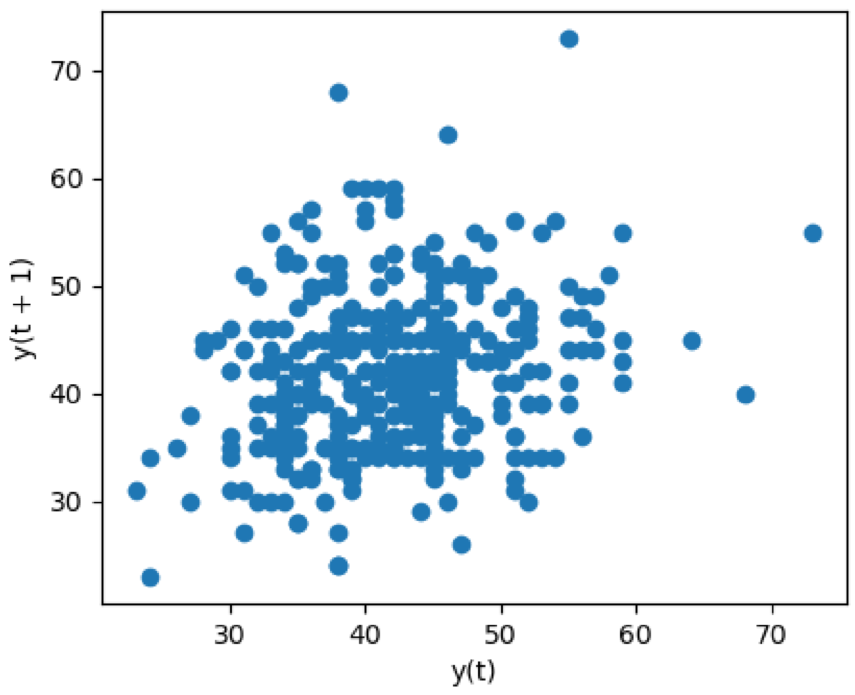
[8]:
```
(365, 1)
```

[9]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('                        ')
df.plot(ax=ax, kind='kde', legend=False)
pyplot.show()
```

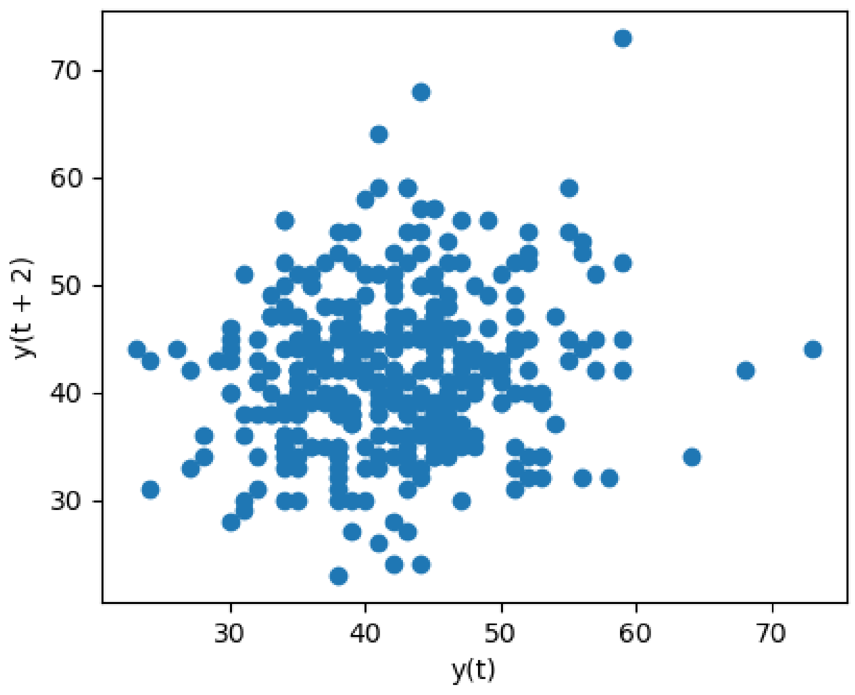


```
[10]: for i in range(1, 5):
          fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(5,4))
          fig.suptitle(f'       {i}')
          pd.plotting.lag_plot(df, lag=i, ax=ax)
          pyplot.show()
```

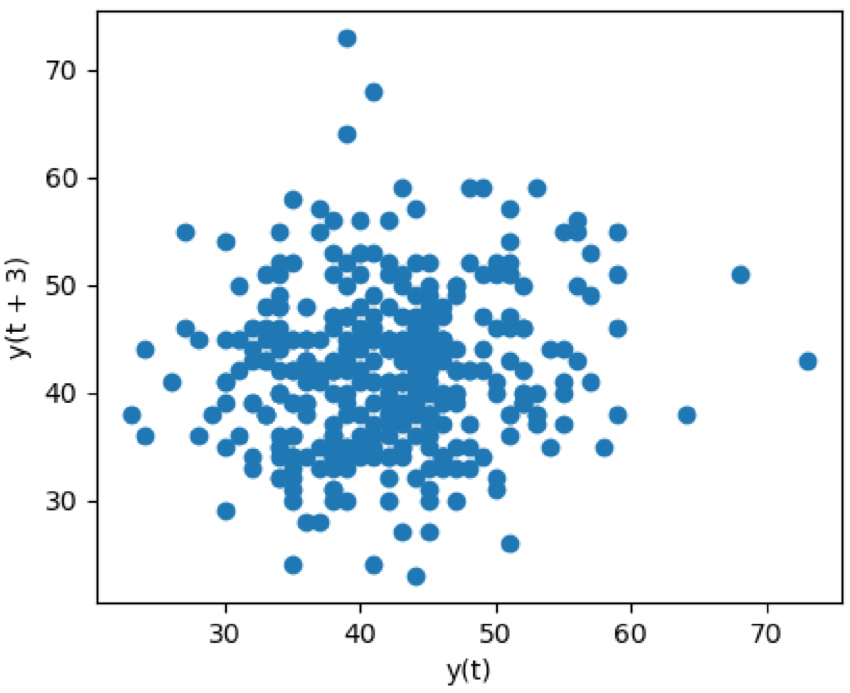
Лаг порядка 1

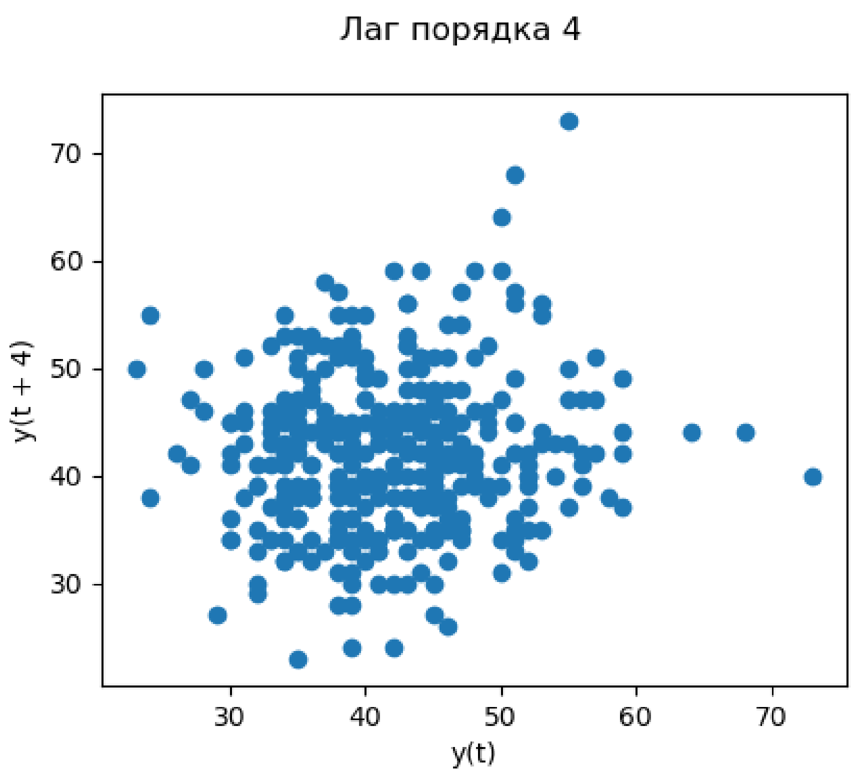


Лаг порядка 2

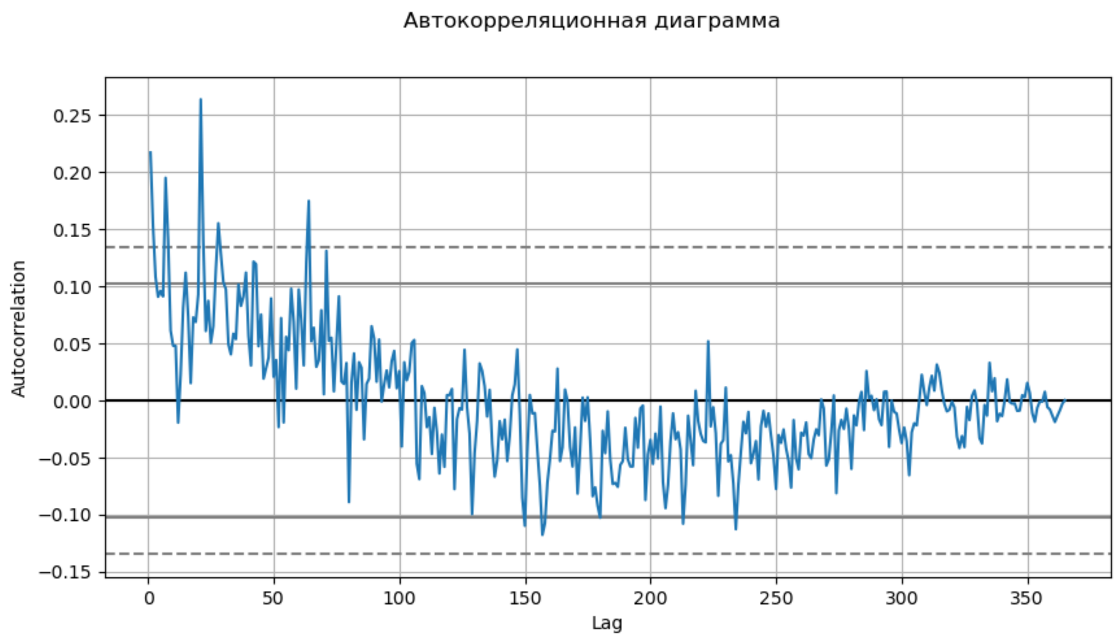


Лаг порядка 3

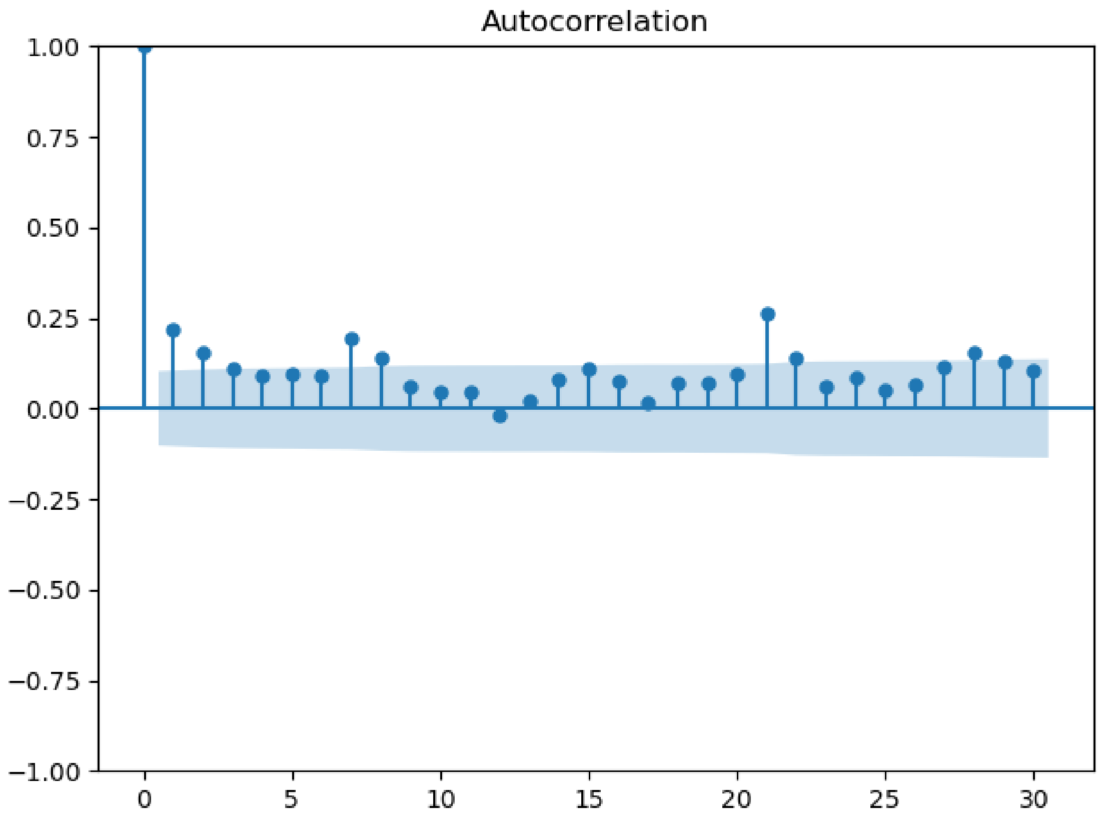




```
[11]: fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
      fig.suptitle('                    ')
      pd.plotting.autocorrelation_plot(df, ax=ax)
      pyplot.show()
```



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



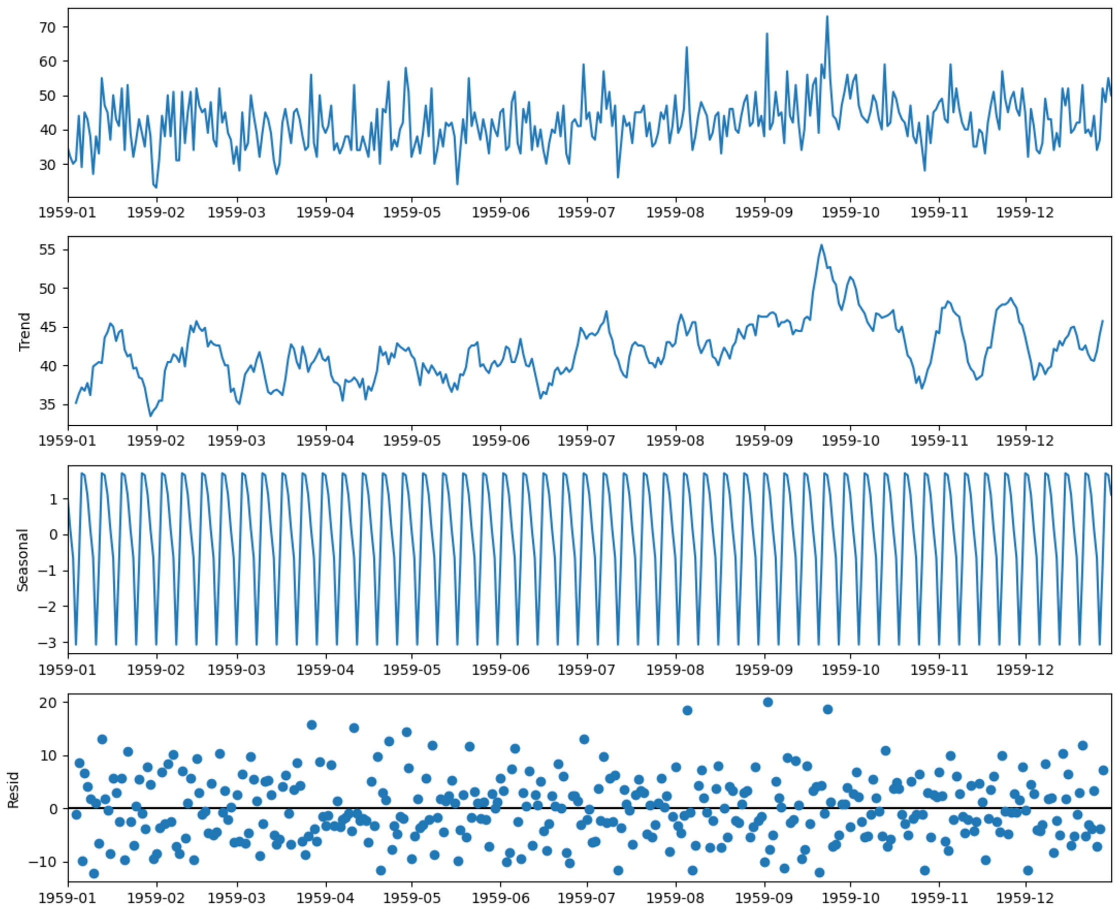
```
[13]: df.index = pd.to_datetime(df.index)
```

```
[14]: #            seasonal_decompose   statsmodels
      from statsmodels.tsa.seasonal import seasonal_decompose

      #
      from pylab import rcParams
      rcParams['figure.figsize'] = 11, 9

      #
      decompose = seasonal_decompose(df)
      decompose.plot()

      plt.show()
```



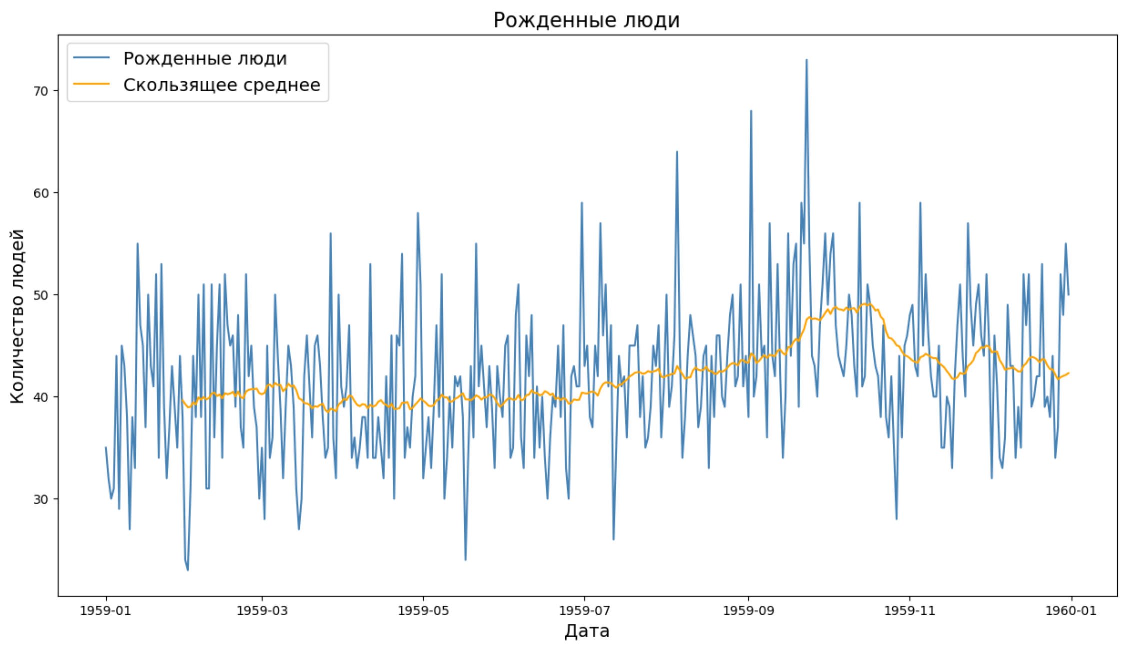
```
[17]: #
      plt.figure(figsize = (15,8))

      #                (                  )
      plt.plot(df, label = '        ', color = 'steelblue')
      plt.plot(df.rolling(window = 30).mean(), label = '          ', color = 'orange')

      #          ,
      plt.legend(title = '', loc = 'upper left', fontsize = 14)

      #
      plt.xlabel('   ', fontsize = 14)
      plt.ylabel('         ', fontsize = 14)
      plt.title('        ', fontsize = 16)

      #
      plt.show()
```



[19]:
```
#
from statsmodels.tsa.stattools import adfuller

#                                          adf_test
adf_test = adfuller(df['Births'])

#      p-value
print('p-value = ' + str(adf_test[1]))
```

```
p-value = 5.2434129901498554e-05
```

[29]:
```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.arima.model import ARIMA
```

[26]:
```
xnum = list(range(df.shape[0]))
Y = df['Births'].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:]
```

[31]:
```
history_arima = [x for x in train]
history_es = [x for x in train]
```

[32]:
```
arima_order = (6,1,0)
```

[33]:
```
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])
```

[34]:
```
error_arima = mean_squared_error(test, predictions_arima, squared=False)
```

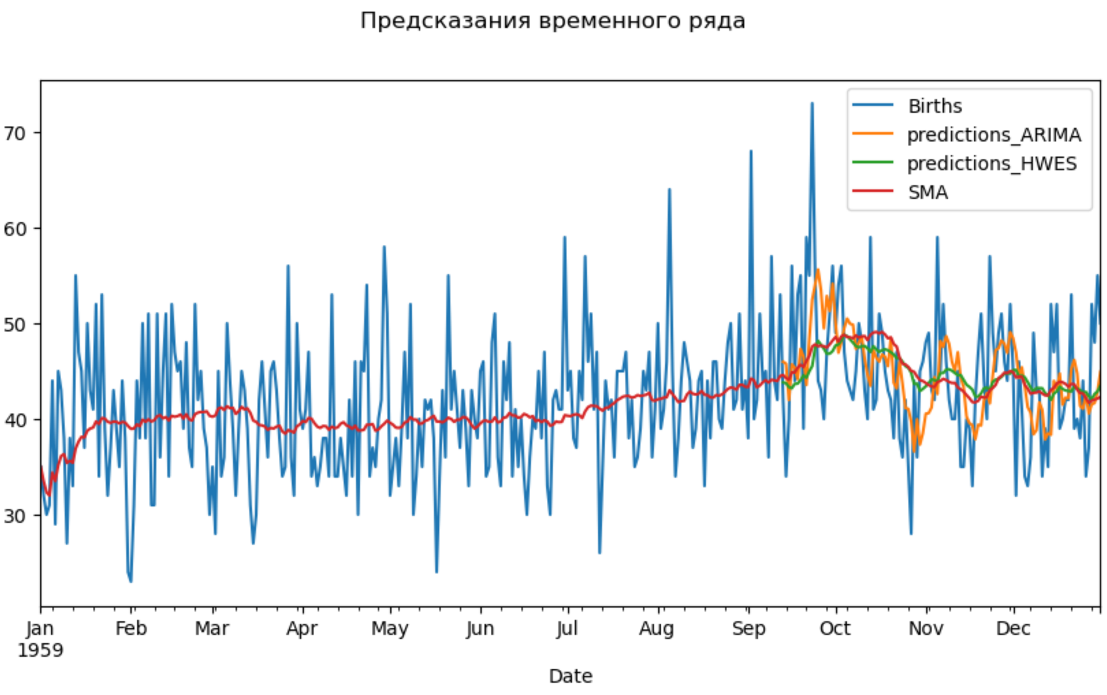
[35]:
```
predictions_es = list()
for t in range(len(test)):
    model_es = ExponentialSmoothing(history_es)
    model_es_fit = model_es.fit()
    yhat_es = model_es_fit.forecast()[0]
    predictions_es.append(yhat_es)
    history_es.append(test[t])
```

[36]:
```
error_es = mean_squared_error(test, predictions_es, squared=False)
```

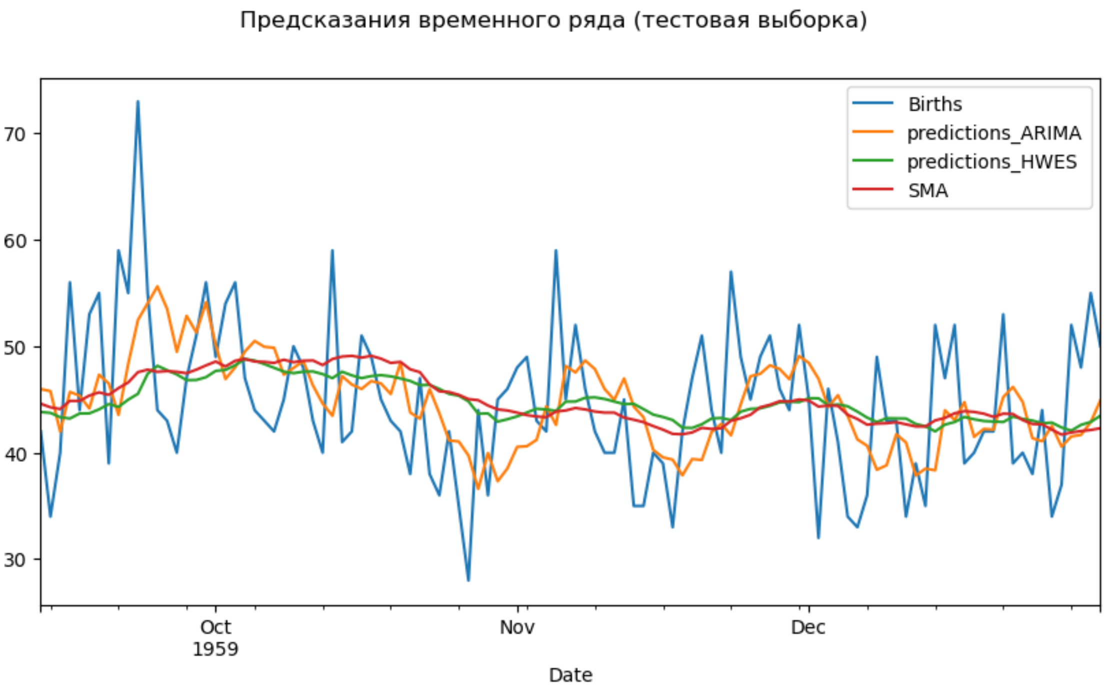
[37]:
```
df['predictions_ARIMA'] = (train_size * [np.NAN]) + list(predictions_arima)
df['predictions_HWES'] = (train_size * [np.NAN]) + list(predictions_es)
```

[40]:
```
df['SMA'] = df['Births'].rolling(30, min_periods=1).mean()
```

[41]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('                ')
df.plot(ax=ax, legend=True)
pyplot.show()
```



```
[42]: fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
      fig.suptitle('                    (          )')
      df[train_size:].plot(ax=ax, legend=True)
      pyplot.show()
```



ARIMA HWES

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

```
[46]: function_set = ['add', 'sub', 'mul', 'div', 'sin']
est_gp = 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)
```

```
[47]: est_gp.fit(np.array(xnum_train).reshape(-1, 1), train.reshape(-1, 1))
```

```
C:\anaconda3\lib\site-packages\sklearn\utils\validation.py:993:
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      7.25395e+55       26           368.24              N/A
1.93m
   1   168.80      3.08763e+11      190          137.664              N/A
57.31s
   2   187.17      1.23463e+10      190           137.64              N/A
54.87s
   3   126.69      3.36087e+22       14          63.3652              N/A
42.64s
   4   178.60      6.78824e+13       14          63.3685              N/A
53.37s
   5   123.38      1.01443e+14       10          58.1649              N/A
41.21s
   6    16.83        3.043e+14       25          57.1643              N/A
18.65s
   7    13.11      4.06308e+14        9          50.0916              N/A
19.34s
   8    15.99      4.05734e+14       22          50.0506              N/A
18.61s
   9    15.88      4.87469e+15       10          49.9905              N/A
20.16s
  10    15.11      2.53812e+15       41          49.8642              N/A
19.47s
```

```
  11    23.48      1.18829e+16       61          49.5823              N/A
19.32s
  12    26.29      2.84417e+15       19          49.2251              N/A
21.55s
  13    41.60      2.85776e+15       18          48.5365              N/A
22.46s
  14    52.15      7.12366e+15       29          48.4612              N/A
24.42s
  15    44.79      1.81606e+11      137          48.1081              N/A
22.79s
  16    38.99          37707.7      150          48.0293              N/A
21.36s
  17    57.33      5.89249e+06       59          47.2986              N/A
24.49s
  18   101.51      3.40558e+06       40          47.1072              N/A
34.02s
  19    97.39        1.701e+06       64          46.9361              N/A
28.80s
  20    67.71          30533.4      150          46.8244              N/A
24.79s
  21    73.77          59189.6       79          46.2824              N/A
25.28s
  22    85.94          37425.4      113          46.2358              N/A
26.47s
  23    97.07      1.30434e+06       61          45.8434              N/A
28.83s
  24    96.01      1.79859e+06      142          45.1803              N/A
26.86s
  25   108.38           686066      142          45.1803              N/A
28.56s
  26   130.18      1.06313e+06      154          44.8865              N/A
30.53s
  27   151.34           362193      152          44.7084              N/A
32.17s
  28   173.31          12847.7      256           44.563              N/A
37.23s
  29   170.85           708637      131          43.4877              N/A
34.96s
  30   164.65          18007.3      131          43.4877              N/A
32.21s
  31   159.97      7.12214e+06      204          43.0844              N/A
39.60s
  32   152.48           912883      204          43.0844              N/A
30.01s
  33   157.81           841393      296           42.874              N/A
32.40s
  34   202.25      2.11938e+06      321           42.627              N/A
33.46s
```

```
  35   247.06          951483       518         42.5981            N/A
36.35s
  36   252.69         22821.2       421         42.2247            N/A
36.23s
  37   276.74          245656       420         42.1337            N/A
39.97s
  38   347.55         11894.9       294          41.951            N/A
45.67s
  39   390.84      1.3304e+06       720         41.7752            N/A
48.72s
  40   410.14         254.802       762         41.7562            N/A
48.69s
  41   331.65         6379.31       466         41.0082            N/A
38.65s
  42   298.96          828984       552         40.4083            N/A
36.15s
  43   368.48          818223       551         39.8089            N/A
41.11s
  44   461.59          348617       692         39.3182            N/A
45.53s
  45   575.49         23406.9       626         39.2141            N/A
53.69s
  46   603.84         74068.7       693         38.3947            N/A
53.13s
  47   678.43         17712.6       716         38.1826            N/A
54.47s
  48   684.47         1933.23       755         37.9086            N/A
51.28s
  49   699.80          535678       755         37.7096            N/A
51.12s
  50   716.31         34874.9       741         37.7048            N/A
54.51s
  51   748.82         18785.5       808         37.4716            N/A
48.94s
  52   759.12         5933.17      1274         37.3378            N/A
47.11s
  53   747.69         34420.9      1193         36.9635            N/A
41.78s
  54   788.61          229922      1216         35.4431            N/A
40.50s
  55   845.38         11874.9      1236         35.4172            N/A
41.69s
  56  1159.01         11922.7      1216          34.917            N/A
52.82s
  57  1224.57         28908.2      1271         34.5268            N/A
51.30s
  58  1236.54          529802      1270          34.416            N/A
49.79s
```

```
  59   1233.49          214.684     1160          34.3678              N/A
41.83s
  60   1234.42           287984     1249          34.2335              N/A
39.51s
  61   1228.73          5863.85     1343          33.9161              N/A
35.58s
  62   1236.84          17811.7     1363          33.9147              N/A
31.66s
  63   1261.26          23379.8     1346           33.748              N/A
27.21s
  64   1277.07          16949.9     1346          33.2909              N/A
24.65s
  65   1274.63          17486.8     1335          33.2896              N/A
18.72s
  66   1289.95          119.186     1371          33.1259              N/A
13.63s
  67   1319.21          17531.2     2614          32.9963              N/A
9.42s
  68   1322.92          222.319     1437          33.0886              N/A
4.86s
  69   1329.79          295.903     1271          32.8986              N/A
0.00s
```

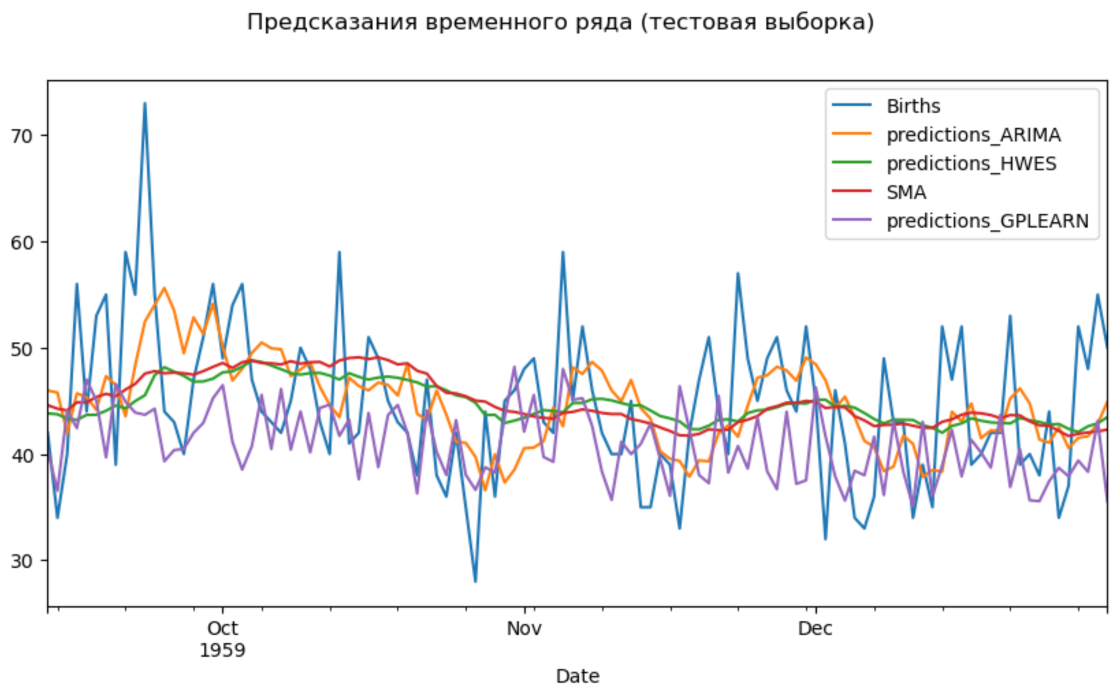
[47]:
```
SymbolicRegressor(const_range=(-100, 100),
                  function_set=['add', 'sub', 'mul', 'div', 'sin'],
                  generations=70, init_depth=(4, 10), metric='mse',
                  population_size=500, random_state=0, stopping_criteria=0.01,
                  verbose=1)
```

[48]:
```
#
y_gp = est_gp.predict(np.array(xnum_test).reshape(-1, 1))
y_gp[:10]
```

[48]:
```
array([40.28938386, 36.55290694, 44.0910671 , 42.43899561, 47.00219962,
       45.03593076, 39.69664985, 46.57498875, 44.92532119, 43.89997191])
```

[49]:
```
df['predictions_GPLEARN'] = (train_size * [np.NAN]) + list(y_gp)
```

[51]:
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('                 (         )')
df[train_size:].plot(ax=ax, legend=True)
pyplot.show()
```


Предсказания временного ряда (тестовая выборка)
Births
predictions_ARIMA
predictions_HWES
SMA
predictions_GPLEARN
70
60
50
40
30
Oct
1959
Nov
Dec
Date