Лабораторная работа 6

Задание

Выберите набор данных (датасет) для решения задачи прогнозирования временного ряда.

Визуализируйте временной ряд и его основные характеристики.

Разделите временной ряд на обучающую и тестовую выборку.

Произведите прогнозирование временного ряда с использованием как минимум двух методов.

Визуализируйте тестовую выборку и каждый из прогнозов.

Оцените качество прогноза в каждом случае с помощью метрик.

Выбор и загрузка данных

Возьмем датасет с данными о рождаемости мужчин и женщин в определенную дату.

year - год;

month - месяц;

day - день;

gender - пол;

births - количество родившихся.

Импортируем библиотеки

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import files
%matplotlib inline
sns.set(style="ticks")
```

Загружаем данные

```
files.upload()
```

Выбрать файлы births.csv

- **births.csv**(text/csv) - 264648 bytes, last modified: 26.09.2020 - 100% done

```
Saving births.csv to births (2).csv
{'births.csv': b'year,month,day,gender,births\n1969,1,1,F,4046\n1969,1,1,M,4440\n196
```

```
data = pd.read_csv('births.csv')
```

Первичный анализ

Первые 5 строк датасета:

```
data.head()
```

	year	month	day	gender	births
0	1969	1	1.0	F	4046
1	1969	1	1.0	M	4440
2	1969	1	2.0	F	4454
3	1969	1	2.0	M	4548
4	1969	1	3.0	F	4548

Определим размер датасета:

```
data.shape
```

```
(15547, 5)
```

Определим типы данных:

```
data.dtypes
```

```
year        int64
month       int64
day       float64
gender     object
births      int64
dtype: object
```

Удаление записей

Удалим несуществующий в реальности день

```
data = data.loc[data['day'] != 99]
```

Очистим записи с пустыми ячейками

```
data.isnull().sum()
```

```
year        0
month       0
day       480
gender      0
births      0
dtype: int64
```

```
data = data.dropna(axis=0, how='any')
```

```
data.isnull().sum()
```

```
year      0
month     0
day       0
gender    0
births    0
dtype: int64
```

Анализировать будем рождаемость мальчиков, поэтому уберем все строки с данными о рождаемости девочек.

```
data = data.loc[data['gender'] != 'F']
```

```
data.head()
```

	year	month	day	gender	births
1	1969	1	1.0	M	4440
3	1969	1	2.0	M	4548
5	1969	1	3.0	M	4994
7	1969	1	4.0	M	4520
9	1969	1	5.0	M	4198

```
data = data.astype({"day": "Int64"})
```

```
data.dtypes
```

```
year       int64
month      int64
day        Int64
gender    object
births     int64
dtype: object
```

Чтобы не было переизбытка данных, удалим некоторые так, чтобы осталось около тысячи записей. (даты удаляем, поскольку часто появляются несуществующие в месяце "февраль", а года - чтобы сократить количество записей и как следствие - время для анализа)

```
data = data.loc[data['day'] != 29]
```

```
data = data.loc[data['day'] != 30]
```

```
data = data.loc[data['day'] != 31]
```

```
data = data.loc[data['year'] != 1988]
```

```
data = data.loc[data['year'] != 1987]
```

```
data = data.loc[data['year'] != 1986]
```

```
data = data.loc[data['year'] != 1985]
```

```
data = data.loc[data['year'] != 1984]
```

```
data = data.loc[data['year'] != 1983]
```

```
data = data.loc[data['year'] != 1982]
```

```
data = data.loc[data['year'] != 1981]
```

```
data = data.loc[data['year'] != 1980]
```

```
data = data.loc[data['year'] != 1979]
```

```
data = data.loc[data['year'] != 1978]
```

```
data = data.loc[data['year'] != 1976]
```

```
data = data.loc[data['year'] != 1977]
```

```
data = data.loc[data['year'] != 1975]
```

```
data = data.loc[data['year'] != 1974]
```

```
data = data.loc[data['year'] != 1973]
```

```
data
```

	year	month	day	gender	births
1	1969	1	1	M	4440
3	1969	1	2	M	4548
5	1969	1	3	M	4994
7	1969	1	4	M	4520
9	1969	1	5	M	4198
...	...	...	...	...	...
3048	1972	12	24	M	3766
3050	1972	12	25	M	3655
3052	1972	12	26	M	4454
3054	1972	12	27	M	5130
3056	1972	12	28	M	5210

1344 rows × 5 columns

Столбец с датой

Создадим ячейку даты

```
data["date"] = data["year"].astype(str) + "/" + data["month"].astype(str) + '/' + data["da
```

Изменим формат

```
data['date'] = pd.to_datetime(data['date'])
```

```
data
```

	year	month	day	gender	births	date
1	1969	1	1	M	4440	1969-01-01
3	1969	1	2	M	4548	1969-01-02
5	1969	1	3	M	4994	1969-01-03
7	1969	1	4	M	4520	1969-01-04
9	1969	1	5	M	4198	1969-01-05
...	...	...	...	...	...	...
3048	1972	12	24	M	3766	1972-12-24
3050	1972	12	25	M	3655	1972-12-25
3052	1972	12	26	M	4454	1972-12-26
3054	1972	12	27	M	5130	1972-12-27
3056	1972	12	28	M	5210	1972-12-28

1344 rows × 6 columns

Удалим столбцы, которые не понадобятся для анализа

```
data = data.drop(['year','month', 'day', 'gender'],axis=1)
```

Сделаем дату индексом

```
data.set_index('date',inplace=True)
data.head()
```

	births
date	
1969-01-01	4440
1969-01-02	4548
1969-01-03	4994
1969-01-04	4520
1969-01-05	4198

Визуализация временного ряда

```
import matplotlib.pyplot as pyplot
```

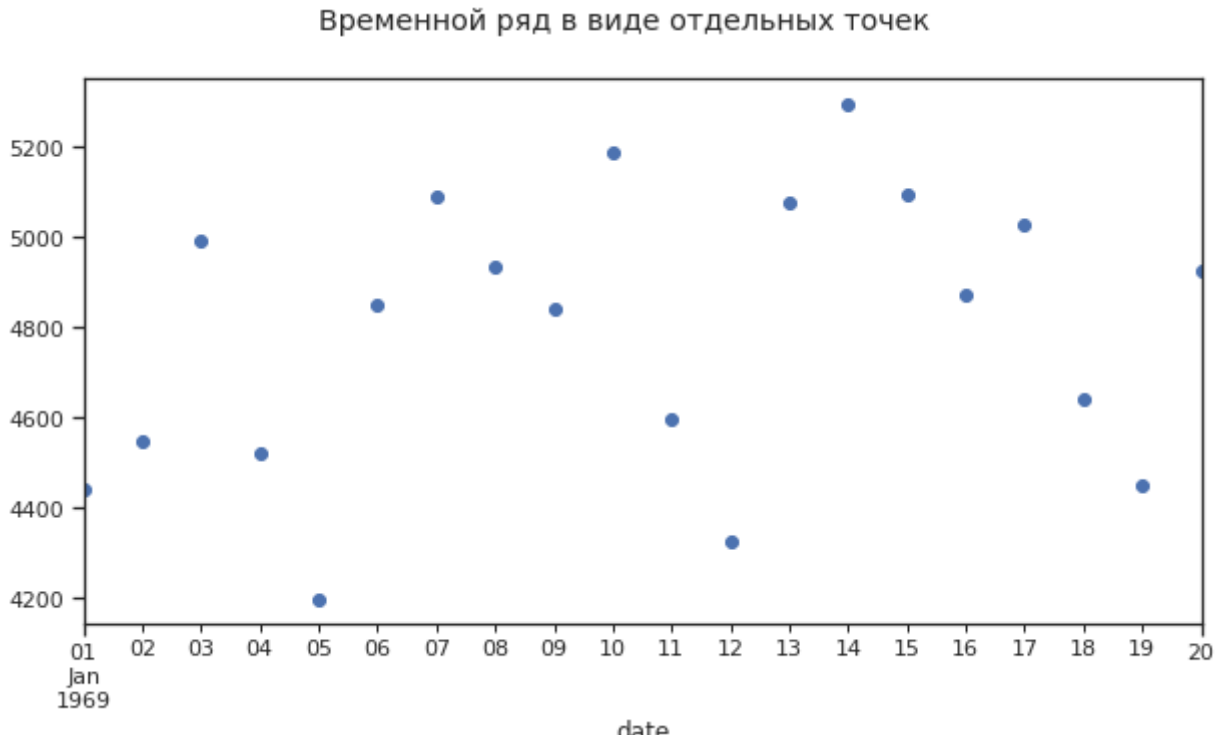
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Временной ряд в виде графика')
```

```
data.plot(ax=ax, legend=False)
pyplot.show()
```

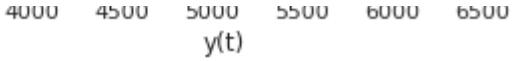

```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Первые 20 точек ряда')
data[:20].plot(ax=ax, legend=False)
pyplot.show()
```


```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Временной ряд в виде отдельных точек')
```

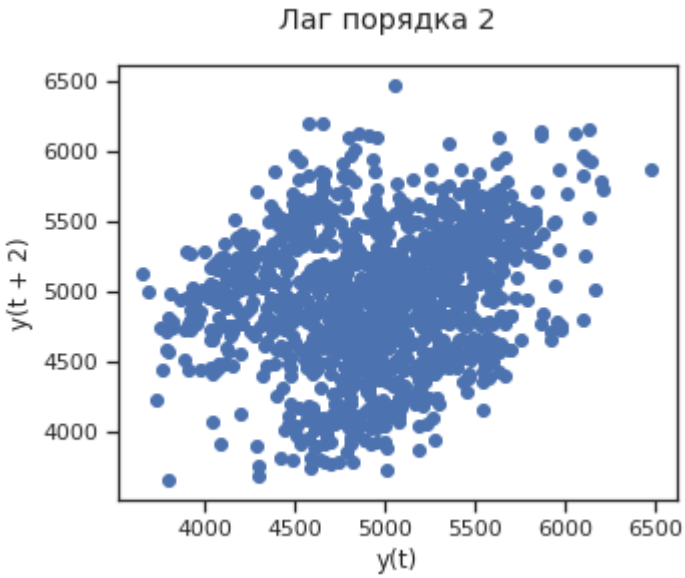
```
data[:20].plot(ax=ax, legend=False, style='bo')
pyplot.show()
```



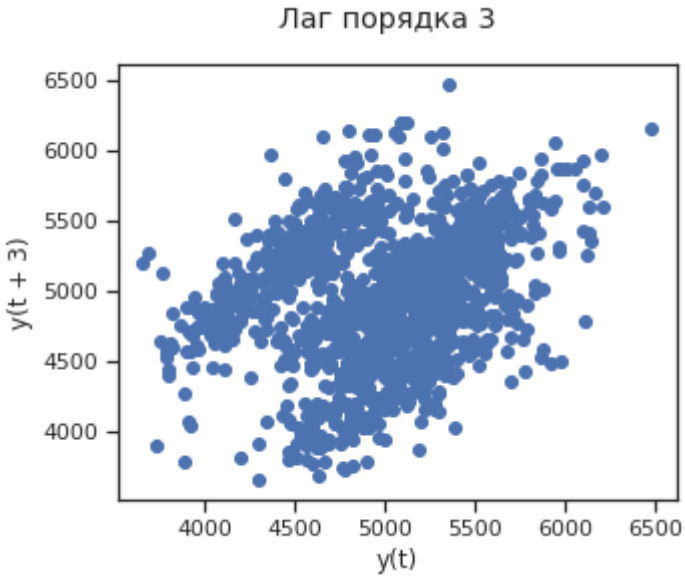

```
for i in range(1, 5):
    fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(5,4))
    fig.suptitle(f'Лаг порядка {i}')
    pd.plotting.lag_plot(data, lag=i, ax=ax)
    pyplot.show()
```




```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avo
```




```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avo
```

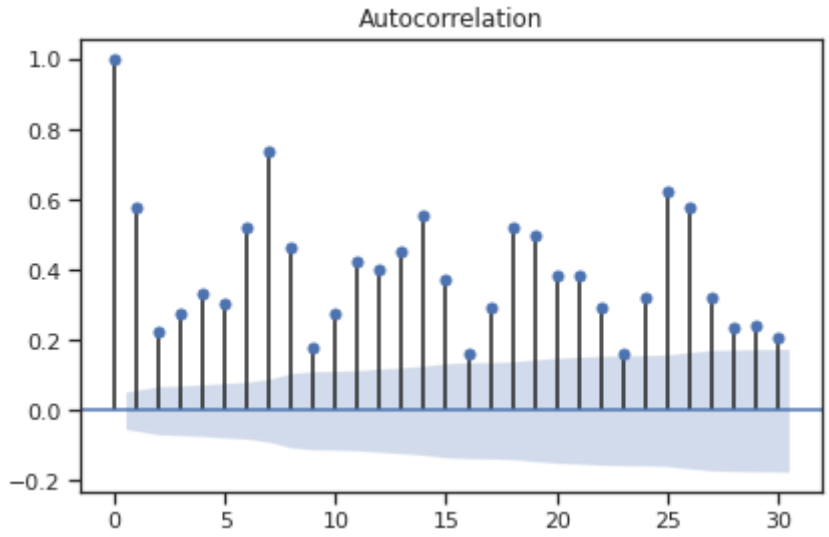



```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avo
```


```
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()
```

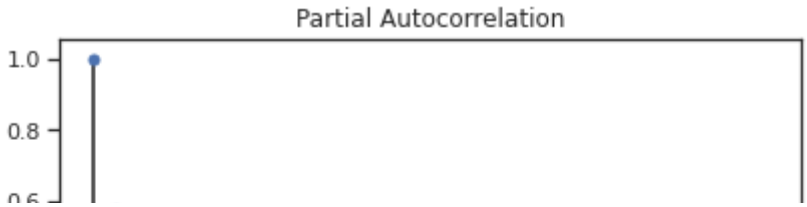
Автокорреляционная функция

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



Частичная автокорреляционная функция

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



Разделение временного ряда на обучающую и тестовую выборку


```
!pip install statsmodels
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scipy>=0.18 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from pa
```

```
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.holtwinters import ExponentialSmoothing


data_2 = data.copy()


# Целочисленная метка шкалы времени
xnum = list(range(data_2.shape[0]))
# Разделение выборки на обучающую и тестовую
Y = data_2['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:]
history_arima = [x for x in train]
```

```
## Прогнозирование временного ряда авторегре
```

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
```

```
(4970.4077380952385, 363.8965630721177)
```

```
# 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

```
!pip install gplearn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting gplearn
  Downloading gplearn-0.4.2-py3-none-any.whl (25 kB)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages
Installing collected packages: gplearn
Successfully installed gplearn-0.4.2
```

```
from gplearn.genetic import SymbolicRegressor
```

```
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))
```

```
       18    65.88      1.37349e+08       83          174437             N/A     33.5
       19    62.04      5.76339e+08       78          173154             N/A     29.6
       20    75.02      3.71029e+08       86          172818             N/A     31.9
       21    88.69      7.79251e+07       96          172566             N/A     35.9
       22    86.82       5.6928e+07       92          172355             N/A     38.7
       23    85.31      1.32844e+10       92          171749             N/A     33.0
       24    92.61      1.25936e+08      171          171686             N/A     34.7
       25   101.44      3.86237e+07      158          171360             N/A     36.9
       26   126.00      3.41738e+08      191          170930             N/A     37.6
```

```
   27   161.87      9.38867e+12      191           170617              N/A     42.7
   28   177.33      2.15541e+08      350           170603              N/A     44.3
   29   226.10      2.59214e+13      182           170526              N/A     54.4
   30   267.00      2.53826e+06      201           170526              N/A      1.0
   31   270.40       9.2933e+09      182           170482              N/A      1.0
   32   206.39        1.773e+07      206           170157              N/A     48.5
   33   199.63      6.47369e+07      216           170154              N/A     44.8
   34   201.44      2.00535e+07      204           170137              N/A     43.3
   35   209.67       7.7513e+07      321           170131              N/A     43.2
   36   228.91      1.23131e+07      365           170121              N/A     43.7
   37   251.17      8.87931e+07      222           170005              N/A     47.6
   38   302.73       1.8101e+07      325           169971              N/A     52.8
   39   326.55      2.96461e+08      353           169789              N/A     52.6
   40   282.70      1.40978e+10      471           169750              N/A     48.4
   41   308.36      5.80891e+06      278           169368              N/A     48.9
   42   343.99      3.09408e+07      276           169365              N/A     52.0
   43   442.51      4.31858e+10      463           168607              N/A      1.0
   44   473.86      2.79857e+08      465           168606              N/A      1.0
   45   387.01      4.18732e+11      569           168262              N/A     49.5
   46   502.40      1.38684e+07      535           168259              N/A     58.4
   47   621.78      3.45216e+08      606           168222              N/A      1.4
   48   627.17      4.82988e+12      537           167964              N/A      1.0
   49   612.68       1.2195e+10      623           167781              N/A      1.0
   50   674.52      1.19192e+07      623           167781              N/A      1.0
   51   582.23      1.11305e+07      631           167565              N/A     52.9
   52   634.52      7.99418e+11      693           167479              N/A     55.0
   53   665.36      1.71768e+11      696           167426              N/A     53.9
   54   670.71       2.7018e+07      782           167386              N/A     49.3
   55   720.23           843374      778           167376              N/A      1.2
   56   762.77      1.50982e+07      698           167242              N/A     49.1
   57   856.56      3.43511e+08     1122           166901              N/A     50.7
   58   895.91      3.26876e+08     1122           166869              N/A     56.2
   59  1215.53      2.91313e+12     1283           166798              N/A     55.8
   60  1143.84      2.69046e+11     1110           165978              N/A     48.9
   61  1181.80      7.38727e+06     1083           165674              N/A     44.0
   62  1208.54      1.00466e+10     1420           164788              N/A     39.1
   63  1273.34       8.0671e+14     1394           164719              N/A     35.1
   64  1403.98      5.83146e+08     1577           164523              N/A     32.6
   65  1510.84      4.39554e+06     1446           164485              N/A     28.1
   66  1572.59      1.46573e+08     1796           163547              N/A     22.3
   67  1524.68      9.67727e+08     1738           161605              N/A     14.0
   68  1646.03      4.53374e+14     1738           159599              N/A      7.6
   69  1750.08      9.09365e+11     1740           158394              N/A      0.0
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)
```

```
add(add(sub(sub(div(sub(sub(65.225, -79.669), sub(X0, 35.398)), div(add(mul(-12.995,
```

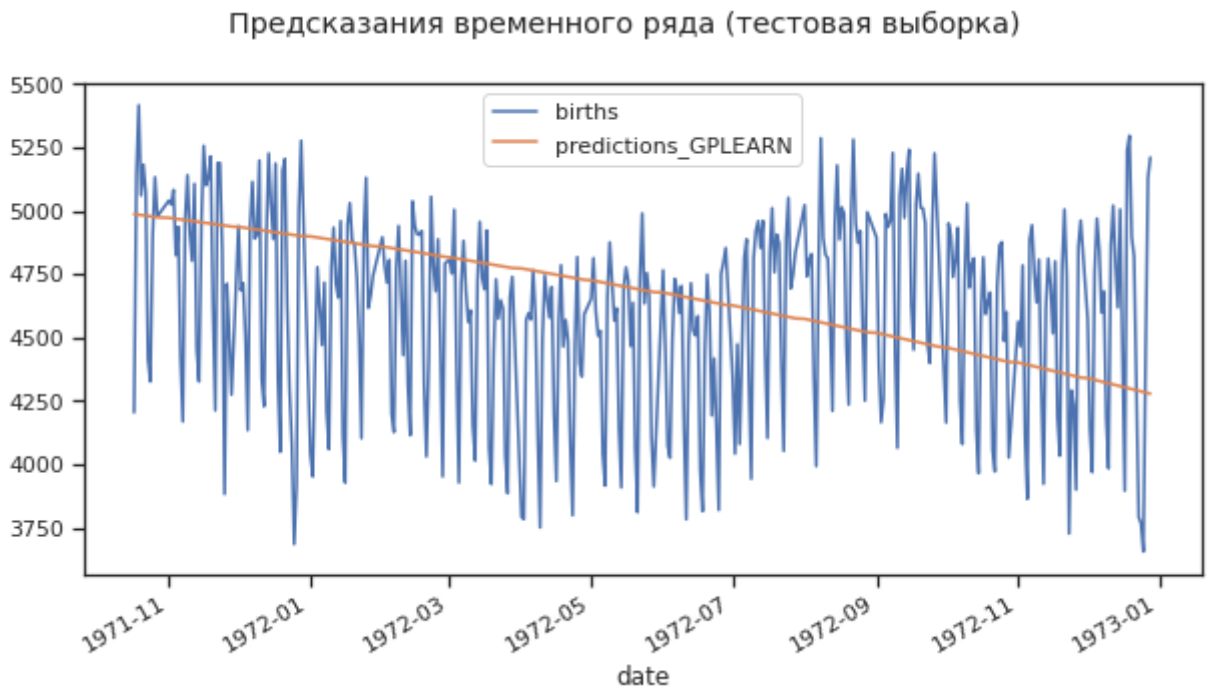
```
# Предсказания
y_sr = SR.predict(np.array(xnum_test).reshape(-1, 1))
```

```
y_sr[:10]
```

```
array([4986.45912116, 4985.25749585, 4984.05312687, 4982.84601335,
       4981.63615444, 4980.42354933, 4979.20819722, 4977.99009732,
       4976.7692489 , 4975.54565121])
```

```
# 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
```

```
(4970.4077380952385, 427.58662432085737)
```

Качество прогноза моделей

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


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: 0.10525026815303329
MSE: 363.8965630721177
MAE: 293.7380334344495

GPLEARN
R^2: -0.23536101464103076
MSE: 427.58662432085737
MAE: 339.6186106097561
```

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

✓ 0 сек. выполнено в 14:54