Лабораторная работа №6 Прогнозирование временных рядов

Импорт библиотек

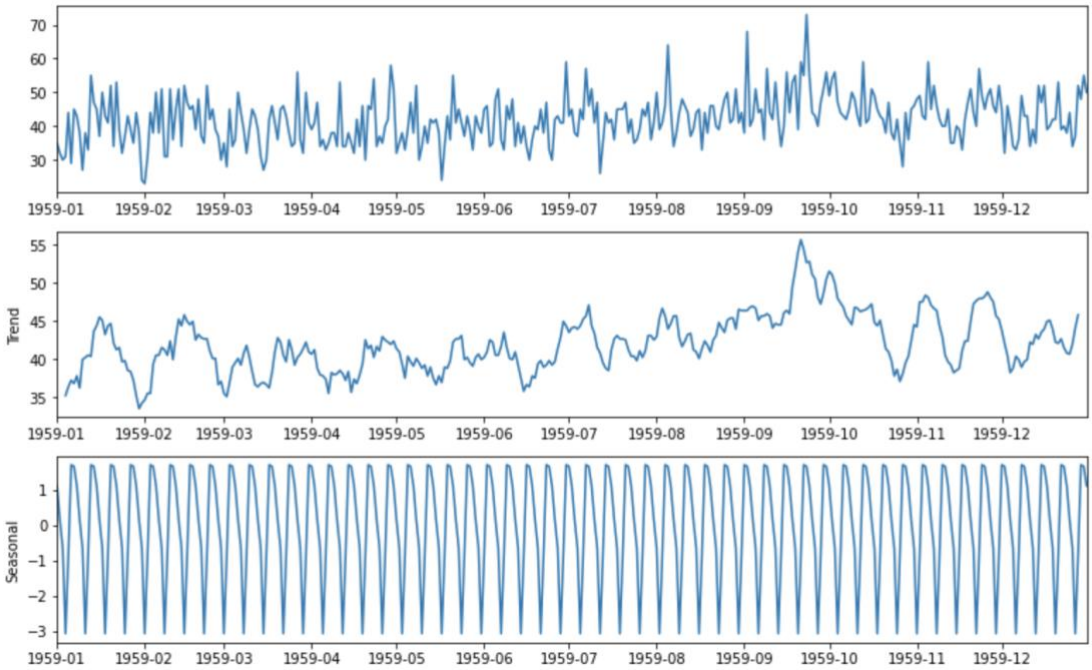
```
[2] # импортируем необходимые библиотеки
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
```

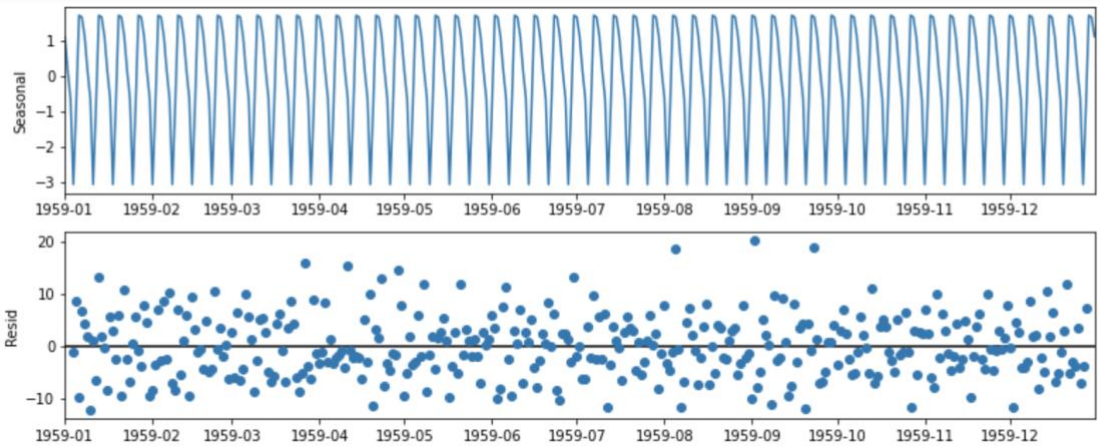
Разложение временного ряда на компоненты

```
[10]
    births = pd.read_csv("/content/births.csv", index_col = 'Date', parse_dates = True)
    births.head(3)
```

Date	Births
1959-01-01	35
1959-01-02	32
1959-01-03	30

```
decompose = seasonal_decompose(births)
decompose.plot()

plt.show()
```





Проверка временного ряда на стационарность

```
[ ] # проведем тест Дики-Фуллера (Dickey-Fuller test)

    adf_test = adfuller(births['Births'])

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

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

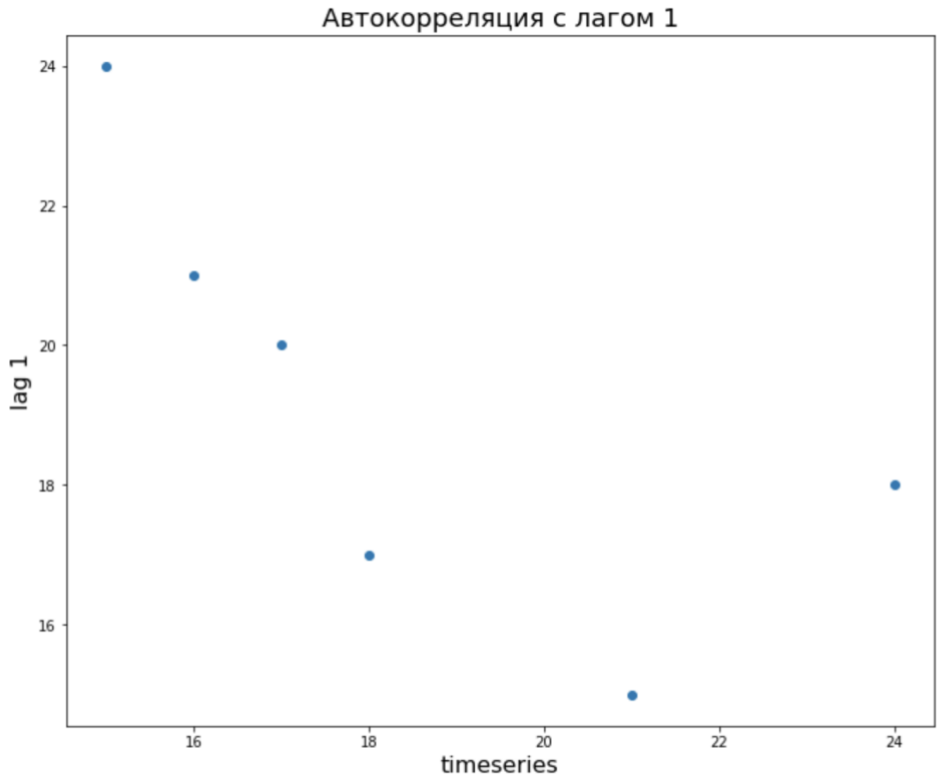
Автокорреляция

```
# для начала возьмем искусственные данные
data = np.array([16, 21, 15, 24, 18, 17, 20])

# для сдвига на одно значение достаточно взять этот ряд, начиная со второго элемента
lag_1 = data[1:]

# посчитаем корреляцию для лага 1 (у исходных данных мы убрали последний элемент)
# так как мы получим коррелационную матрицу, возьмем первую строку и второй столбец [0, 1]
np.round(np.corrcoef(data[:-1], lag_1)[0,1], 2)
```

```
-0.71
```

```
[ ] # построим точечную диаграмму
    plt.scatter(data[:-1], lag_1)

    # добавим подписи
    plt.xlabel('timeseries', fontsize = 16)
    plt.ylabel('lag 1', fontsize = 16)
    plt.title('Автокорреляция с лагом 1', fontsize = 18)
```

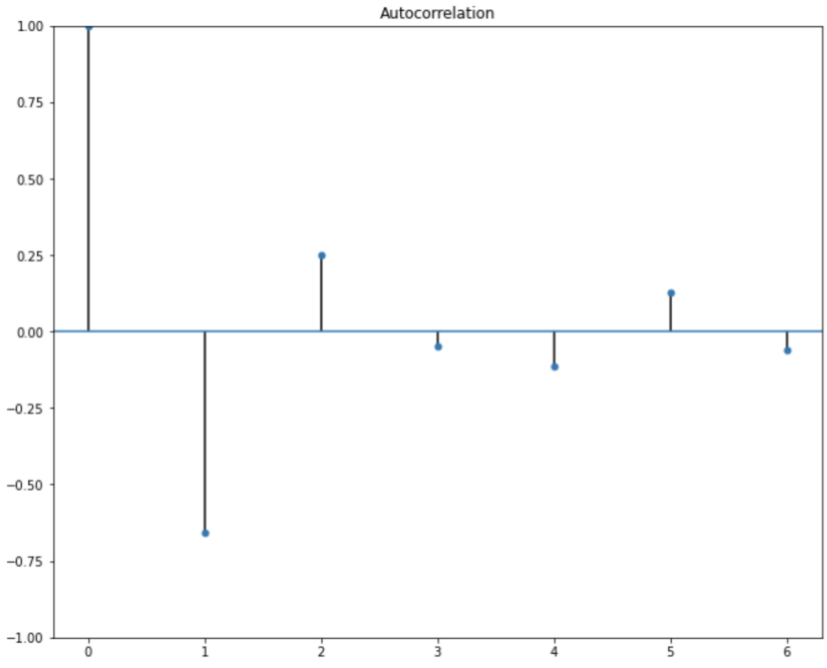
```
Text(0.5, 1.0, 'Автокорреляция с лагом 1')
```



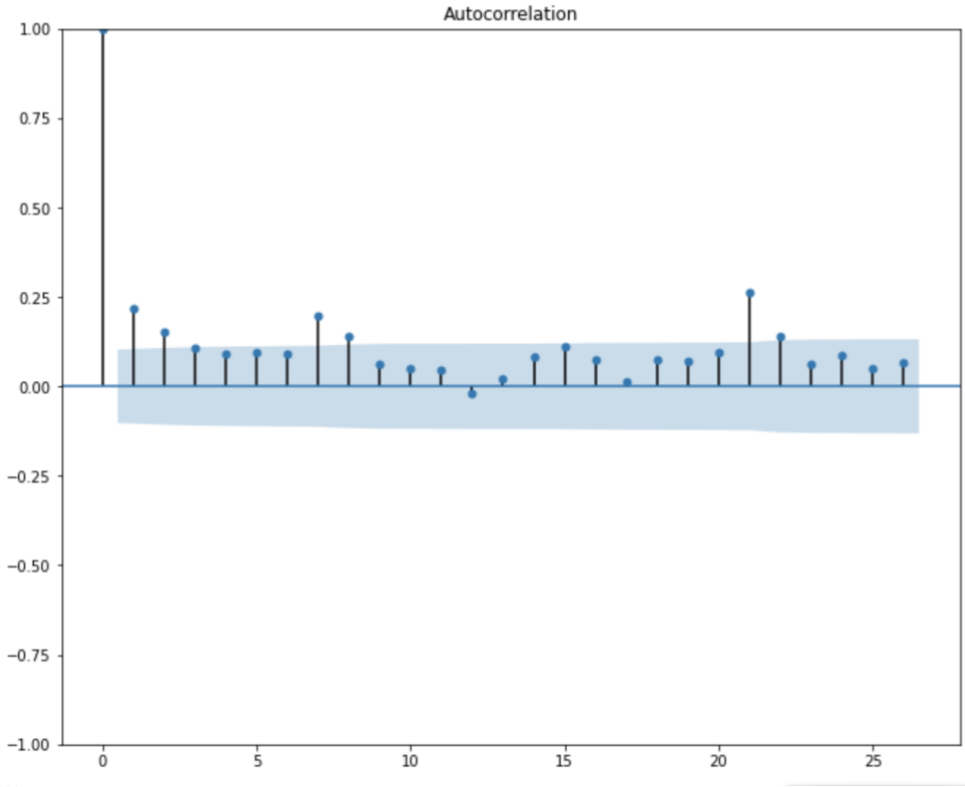

```
[ ]  lag = data[1:]
# посчитаем корреляцию для лага 1 (у исходных данных мы убрали последний элемент)
# так как мы получим коррелационную матрицу, возьмем первую строку и второй столбец [0, 1]
np.round(np.corrcoef(data[:-1], lag_1)[0,1], 2)
```

```
-0.71
```

```
# импортируем автокорреляционную функцию (ACF)
from statsmodels.graphics.tsaplots import plot_acf

# применим функцию к нашему набору данных
plot_acf(data, alpha = None)
plt.show()
```



```
plot_acf(births)
plt.show()
```



Моделирование и построение прогноза

Экспоненциальное сглаживание

```
[ ]  alpha = 0.2

     # первое значение совпадает со значением временного ряда
     exp_smoothing = [births['Births'][0]]


     for i in range(1, len(births['Births'])):
       exp_smoothing.append(alpha * births['Births'][i] + (1 - alpha) * exp_smoothing[i - 1])

     # выведем прогнозное значение для 366-го дня (1 января 1960 года)
     exp_smoothing[-1]
```

```
46.6051933602952
```

```
[ ]  # посмотрим на количество фактических и прогнозных значений
     len(births), len(exp_smoothing)
```

```
(365, 365)
```

```
[ ]  # добавим кривую сглаживаия в качестве столбца в датафрейм
     births['Exp_smoothing'] = exp_smoothing
     births.tail(3)
```

	Births	Exp_smoothing

Date		
1959-12-29	48	43.445615
1959-12-30	55	45.756492
1959-12-31	50	46.605193

[ ]
```
from datetime import timedelta

# возьмём последний индекс (31 декабря 1959 года)
last_date = births.iloc[[-1]].index


last_date = last_date + timedelta(days = 1)
last_date

# добавим его в датафрейм
births = births.append(pd.DataFrame(index = last_date))

# значения за этот день останутся пустыми
births.tail()
```

Date	Births	Exp_smoothing
1959-12-28	52.0	42.307018
1959-12-29	48.0	43.445615
1959-12-30	55.0	45.756492

Date	Births	Exp_smoothing
1959-12-28	52.0	42.307018
1959-12-29	48.0	43.445615
1959-12-30	55.0	45.756492
1959-12-31	50.0	46.605193
1960-01-01	NaN	NaN

[ ]
```
# сдвинем этот столбец на один день вперед
births['Exp_smoothing'] = births['Exp_smoothing'].shift(1)
```

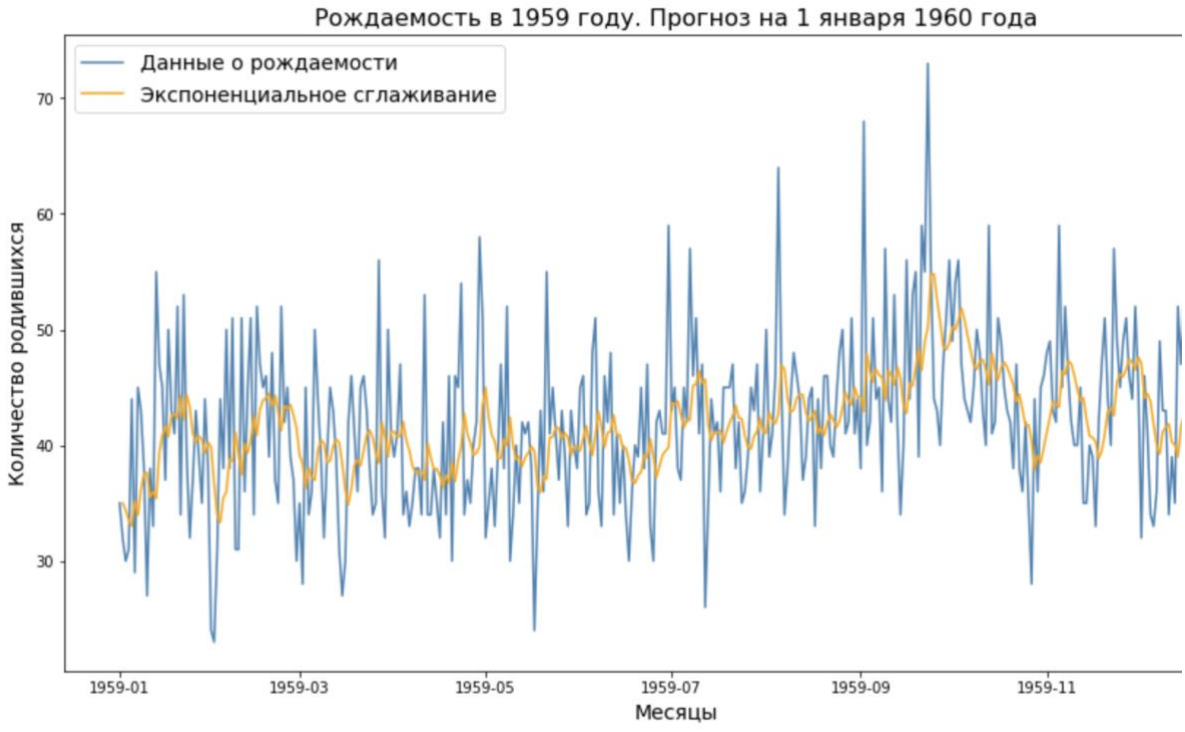
[ ]
```
# как и должно быть первое прогнозное значение совпадает с предыдущим фактическим
births.head()
```

Date	Births	Exp_smoothing
1959-01-01	35.0	NaN
1959-01-02	32.0	35.000
1959-01-03	30.0	34.400
1959-01-04	31.0	33.520
1959-01-05	44.0	33.016

```
[ ] # и у нас есть прогноз на один день вперед
    births.tail()
```

Date	Births	Exp_smoothing
1959-12-28	52.0	39.883773
1959-12-29	48.0	42.307018
1959-12-30	55.0	43.445615
1959-12-31	50.0	45.756492
1960-01-01	NaN	46.605193

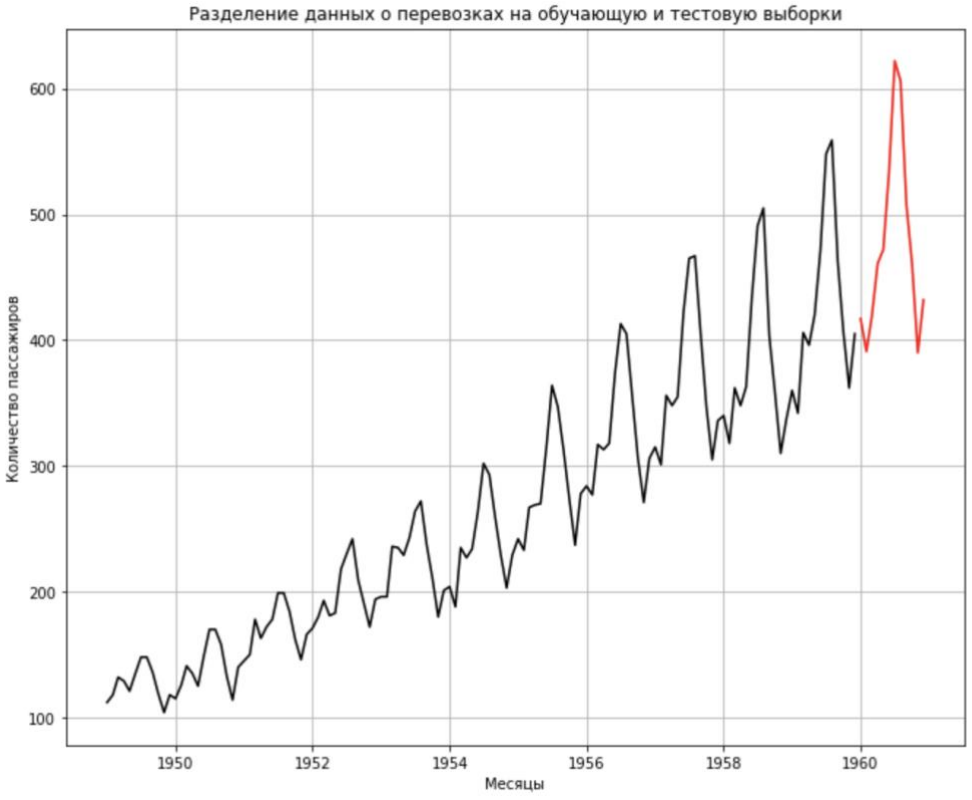
```
[ ] # посмотрим на результат на графике


    plt.figure(figsize = (15,8))

    # выведем данные о рождаемости и кривую экспоненциального сглаживания
    plt.plot(births['Births'], label = 'Данные о рождаемости', color = 'steelblue')
    plt.plot(births['Exp_smoothing'], label = 'Экспоненциальное сглаживание', color = 'orange')

    # добавим легенду, ее положение на графике и размер шрифта
    plt.legend(title = '', loc = 'upper left', fontsize = 14)


    plt.ylabel('Количество родившихся', fontsize = 14)
    plt.xlabel('Месяцы', fontsize = 14)
    plt.title('Рождаемость в 1959 году. Прогноз на 1 января 1960 года', fontsize = 16)
    plt.show()
```



Модель SARIMAX

```
[ ]  # разобьём данные на обучающую и тестовую выборки

     # обучающая выборка будет включать данные до декабря 1959 года включительно
     train = passengers[:'1959-12']

     # тестовая выборка начнется с января 1960 года (по сути, один год)
     test = passengers['1960-01':]
```

```
[ ]  # выведем эти данные на графике
     plt.plot(train, color = "black")
     plt.plot(test, color = "red")

     plt.title('Разделение данных о перевозках на обучающую и тестовую выборки')
     plt.ylabel('Количество пассажиров')
     plt.xlabel('Месяцы')


     plt.grid()

     plt.show()
```



```
import warnings
warnings.simplefilter(action = 'ignore', category = Warning)

# обучим модель с соответствующими параметрами, SARIMAX(3, 0, 0)x(0, 1, 0, 12)


from statsmodels.tsa.statespace.sarimax import SARIMAX

# создадим объект этой модели
model = SARIMAX(train,
                order = (3, 0, 0),
                seasonal_order = (0, 1, 0, 12))


result = model.fit()
```

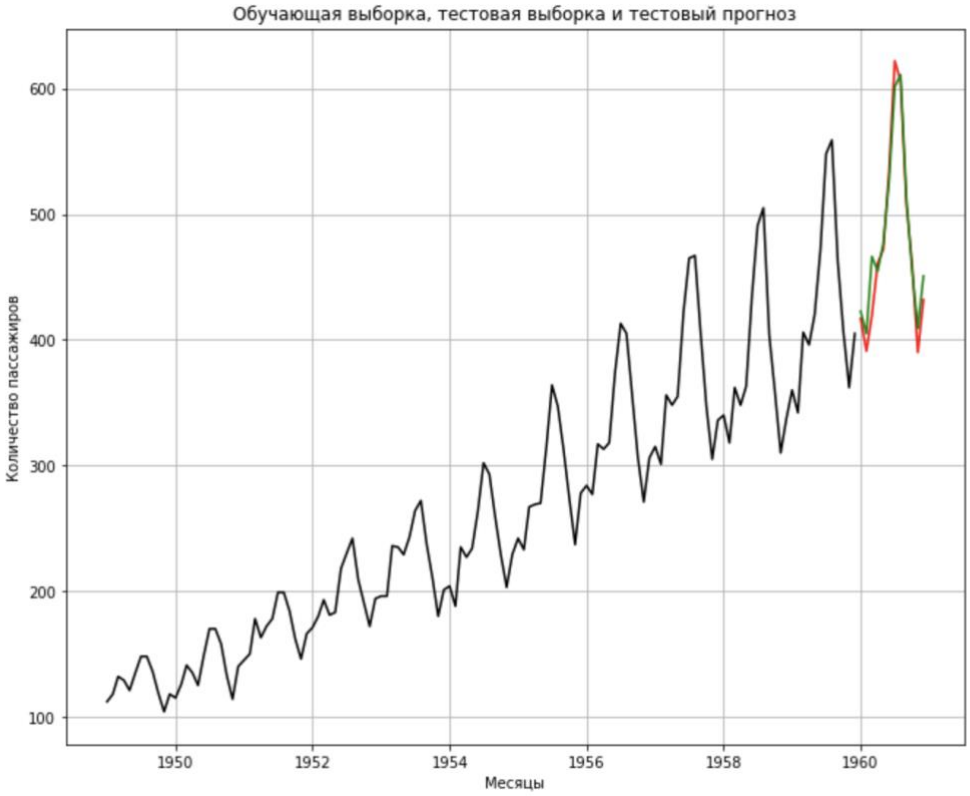
```
[ ] # мы можем посмотреть результат с помощью метода summary()
    print(result.summary())
```

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                      #Passengers   No. Observations:                  132
Model:             SARIMAX(3, 0, 0)x(0, 1, 0, 12)   Log Likelihood                -451.953
Date:                            Thu, 07 Apr 2022   AIC                            911.907
Time:                                    14:38:24   BIC                            923.056
Sample:                                01-01-1949   HQIC                           916.435
                                     - 12-01-1959
Covariance Type:                            opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7603      0.088      8.672      0.000       0.588       0.932
ar.L2          0.2875      0.133      2.164      0.030       0.0
ar.L3         -0.0823      0.109     -0.752      0.452      -0.297       0.132
sigma2       107.0022     13.170      8.125      0.000      81.190     132.814
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                 1.94
Prob(Q):                              0.94   Prob(JB):                         0.38
Heteroskedasticity (H):               1.44   Skew:                            -0.10
Prob(H) (two-sided):                  0.25   Kurtosis:                         3.59
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
[ ] # тестовый прогнозный период начнется с конца обучающего периода
    start = len(train)

    # и закончится в конце тестового
    end = len(train) + len(test) - 1


    predictions = result.predict(start, end)
    predictions
```

```
1960-01-01    422.703386
1960-02-01    404.947179
1960-03-01    466.293259
1960-04-01    454.781298
1960-05-01    476.848630
1960-06-01    527.162829
1960-07-01    601.449812
1960-08-01    610.821694
1960-09-01    513.229991
1960-10-01    455.692623
1960-11-01    409.200051
1960-12-01    450.754165
Freq: MS, Name: predicted_mean, dtype: float64
```

```
[ ]  # выведем три кривые (обучающая, тестовая выборка и тестовый прогноз)
     plt.plot(train, color = 'black')
     plt.plot(test, color = 'red')
     plt.plot(predictions, color = 'green')


     plt.title('Обучающая выборка, тестовая выборка и тестовый прогноз')
     plt.ylabel('Количество пассажиров')
     plt.xlabel('Месяцы')


     plt.grid()

     plt.show()
```



```
from sklearn.metrics import mean_squared_error

# рассчитаем MSE
print(mean_squared_error(test, predictions))

# и RMSE
print(np.sqrt(mean_squared_error(test, predictions)))
```

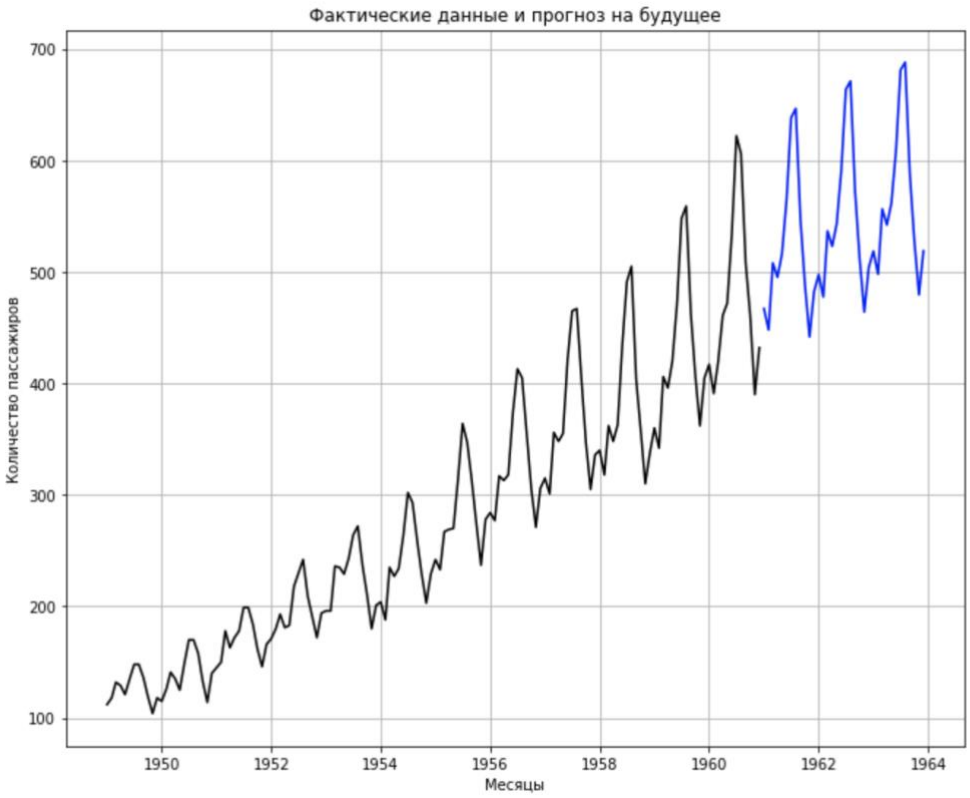
```
317.3956824782105
17.81560222047547
```

```
[ ] # прогнозный период начнется с конца имеющихся данных
start = len(passengers)

# и закончится 36 месяцев спустя
end = (len(passengers) - 1) + 3 * 12

# теперь построим прогноз на три года вперед
forecast = result.predict(start, end)

# посмотрим на весь 1963 год
forecast[-12:]
```

```
1963-01-01    518.603454
1963-02-01    497.909007
1963-03-01    556.406803
1963-04-01    542.133855
1963-05-01    561.524783
1963-06-01    609.244584
1963-07-01    681.016659
1963-08-01    687.950687
1963-09-01    587.995823
1963-10-01    528.167700
1963-11-01    479.454559
```

Снимок экрана

```
[ ] # выведем две кривые (фактические данные и прогноз на будущее)
plt.plot(passengers, color = 'black')
plt.plot(forecast, color = 'blue')


plt.title('Фактические данные и прогноз на будущее')
plt.ylabel('Количество пассажиров')
plt.xlabel('Месяцы')


plt.grid()

plt.show()
```


Фактические данные и прогноз на будущее
Количество пассажиров
Месяцы
100
200
300
400
500
600
700
1950
1952
1954
1956
1958
1960
1962
1964