**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 и искусственный интеллект

КАФЕДРА Системы обработки информации и управления

# ЛР №6

# По курсу
# «Технологии машинного обучения»

Подготовил:

Студент группы

**ИУ5-63Б Борисов А.М.**

08.04.2022

Проверил:

*2022 г.*

## 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.

## Решение:

- Временные ряды

- Анализ временных рядов

- Импорт данных и работа в библиотеке Pandas

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

```
[ ] # импортируем файл с данными о пассажирах
    passengers = pd.read_csv("/content/passengers.csv")
    passengers.head()
```

| | Month | #Passengers |
|---|---|---|
| 0 | 1949-01 | 112 |
| 1 | 1949-02 | 118 |
| 2 | 1949-03 | 132 |
| 3 | 1949-04 | 129 |
| 4 | 1949-05 | 121 |

```
[ ] # превратим дату в индекс и сделаем изменение постоянным
    passengers.set_index('Month', inplace = True)
    passengers.head()
```

```
[ ] # превратим дату (наш индекс) в объект datetime
    passengers.index = pd.to_datetime(passengers.index)

    # посмотрим на первые пять дат и на тип данных
    passengers.index[:5]
```

```
DatetimeIndex(['1949-01-01', '1949-02-01', '1949-03-01', '1949-04-01',
               '1949-05-01'],
              dtype='datetime64[ns]', name='Month', freq=None)
```

```
[ ] # все это можно сделать в одну строчку с помощью parse_dates = True
    passengers = pd.read_csv("/content/passengers.csv", index_col = 'Month', parse_dates = True)
    passengers.head()
```

| Month | #Passengers |
|---|---|
| 1949-01-01 | 112 |
| 1949-02-01 | 118 |
| 1949-03-01 | 132 |
| 1949-04-01 | 129 |
| 1949-05-01 | 121 |

```
[ ] # сделаем срез по дате, например, с августа 1949 по март 1950 года
    passengers['1949-08':'1950-03']
```

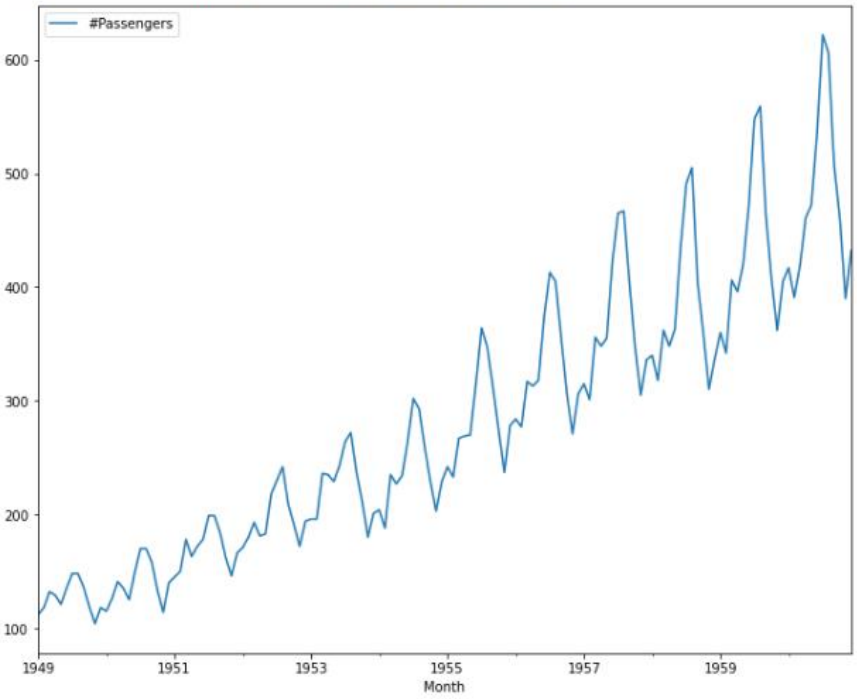
```
[ ] # рассчитаем скользящее среднее для трех предыдущих месяцев
    passengers.rolling(window = 3).mean().head()
```

| Month | #Passengers |
|---|---|
| 1949-01-01 | NaN |
| 1949-02-01 | NaN |
| 1949-03-01 | 120.666667 |
| 1949-04-01 | 126.333333 |
| 1949-05-01 | 127.333333 |

Построение графиков

```
[ ] # построим простой график изменения данных во времени прямо в библиотеке Pandas
    passengers.plot()
```

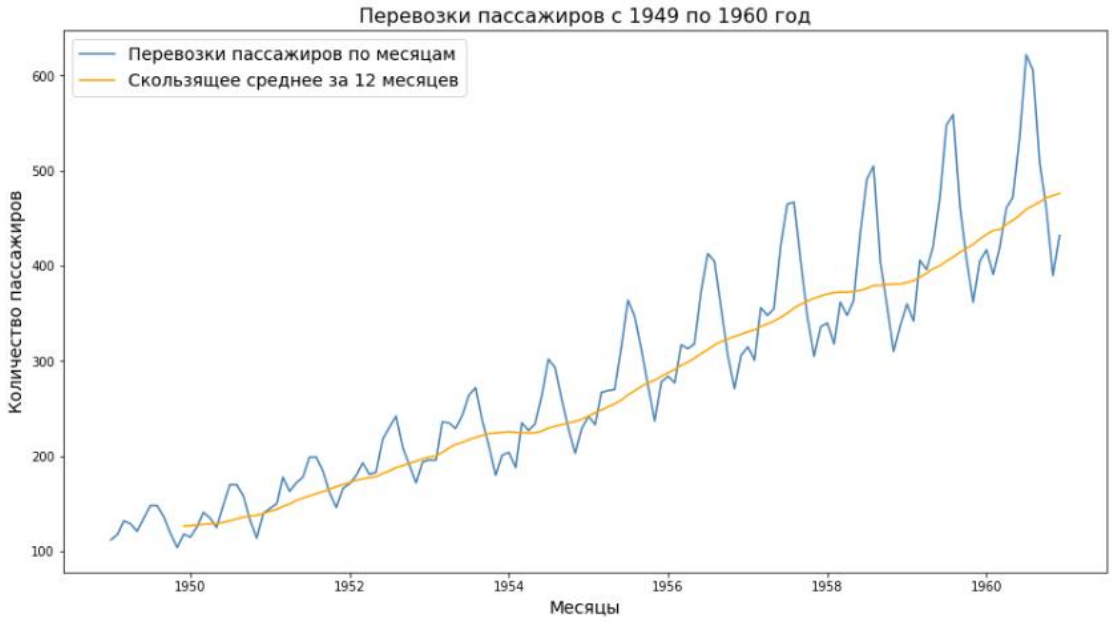
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbdb7d3cad0>
```




```
[ ] # изменим размер графика, уберем легенду и добавим подписи
    ax = passengers.plot(figsize = (12,6), legend = None)
    ax.set(title = 'Перевозки пассажиров с 1949 по 1960 год', xlabel = 'Месяцы', ylabel = 'Количество пассажиров')
```

```
[Text(0, 0.5, 'Количество пассажиров'),
 Text(0.5, 0, 'Месяцы'),
 Text(0.5, 1.0, 'Перевозки пассажиров с 1949 по 1960 год')]
```

```
[ ] # теперь воспользуемся библиотекой matplotlib для построения сразу двух графиков

    # зададим размер графика
    plt.figure(figsize = (15,8))

    # поочередно зададим кривые (перевозки и скользящее среднее) с подписями и цветом
    plt.plot(passengers, label = 'Перевозки пассажиров по месяцам', color = 'steelblue')
    plt.plot(passengers.rolling(window = 12).mean(), label = 'Скользящее среднее за 12 месяцев', color = 'orange')

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

    # добавим подписи к осям и заголовки
    plt.ylabel('Количество пассажиров', fontsize = 14)
    plt.xlabel('Месяцы', fontsize = 14)
    plt.title('Перевозки пассажиров с 1949 по 1960 год', fontsize = 16)

    # выведем обе кривые на одном графике
    plt.show()
```

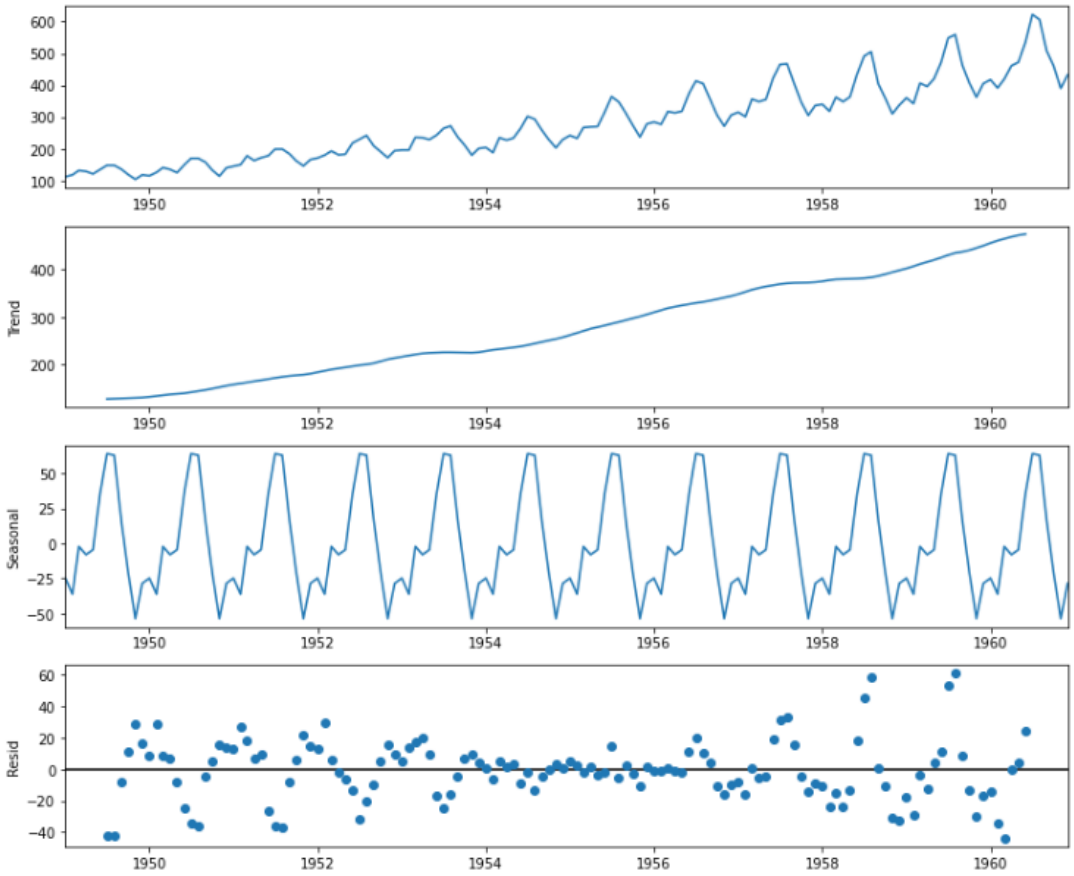


▾ Разложение временного ряда на компоненты

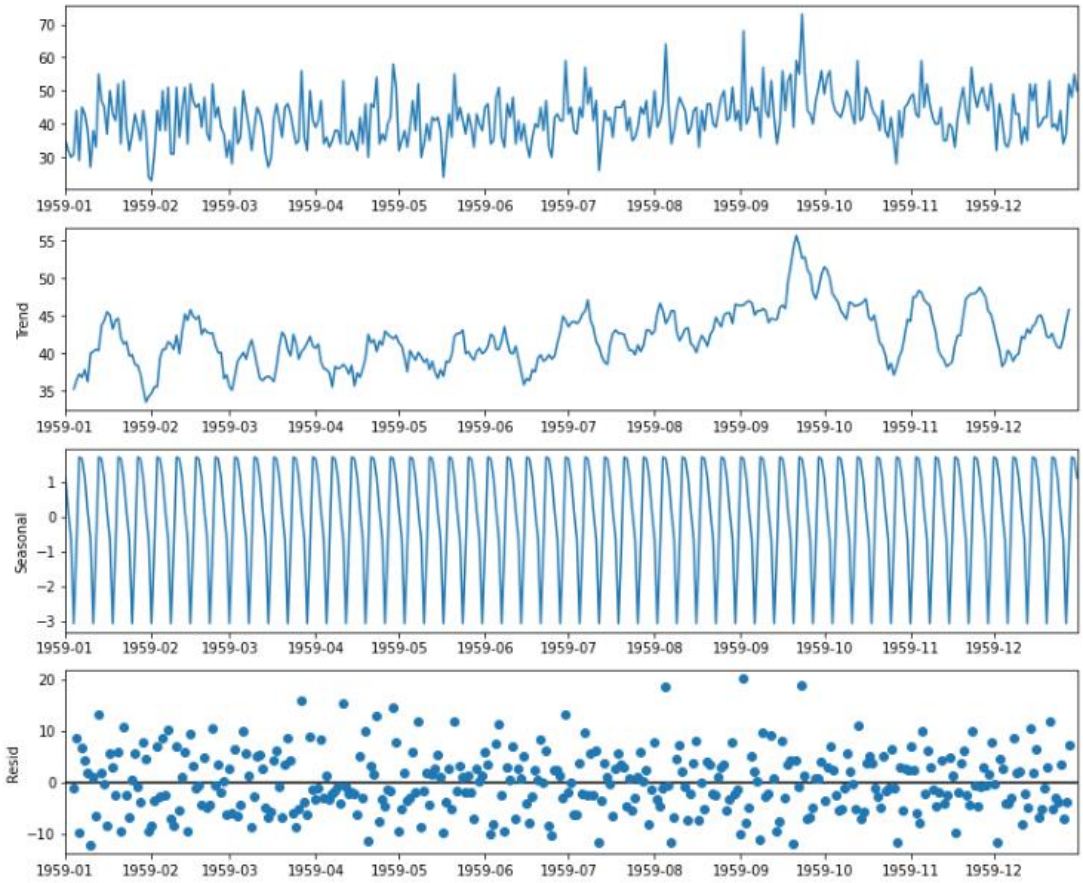
```
[ ] # для наглядности импортируем второй датасет
    # сразу превратим дату в индекс и преобразуем ее в объект datetime
    births = pd.read_csv("/content/births.csv", index_col = 'Date', parse_dates = True)
    births.head(3)
```

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

```
[ ] # импортируем функцию seasonal_decompose из statsmodels
    from statsmodels.tsa.seasonal import seasonal_decompose

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

    # применяем функцию к данным о перевозках
    decompose = seasonal_decompose(passengers)
    decompose.plot()

    plt.show()
```



```
[ ] # сделаем то же самое для данных о рождаемости
    decompose = seasonal_decompose(births)
    decompose.plot()

    plt.show()
```




## Проверка временного ряда на стационарность

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

    # импортируем необходимую функцию
    from statsmodels.tsa.stattools import adfuller

    # передадим ей столбец с данными о перевозках и поместим результат в adf_test
    adf_test = adfuller(passengers['#Passengers'])

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

```
p-value = 0.991880243437641
```

```
[ ] # теперь посмотрим на данные о рождаемости
    adf_test = adfuller(births['Births'])

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

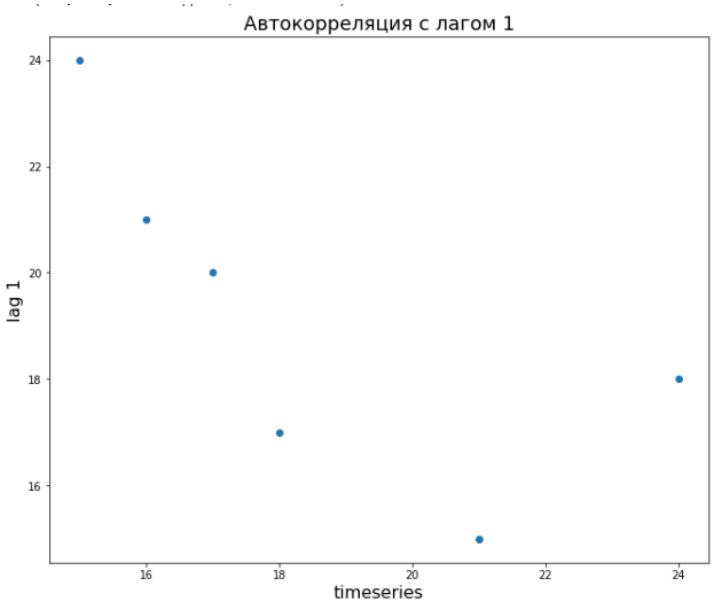
```
p-value = 5.2434129901498554e-05
```

## 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 корреляционную матрицу, возьмем первую строку и второй столбец [0, 1]
    np.round(np.corrcoef(data[:-1], lag_1)[0,1], 2)
```

```
-0.71
```

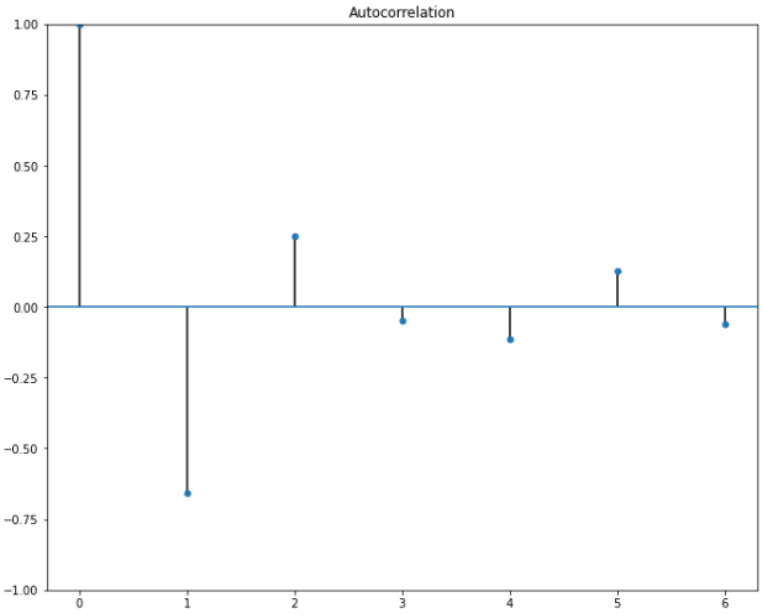
```
[ ] # построим точечную диаграмму
    plt.scatter(data[:-1], lag_1)

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



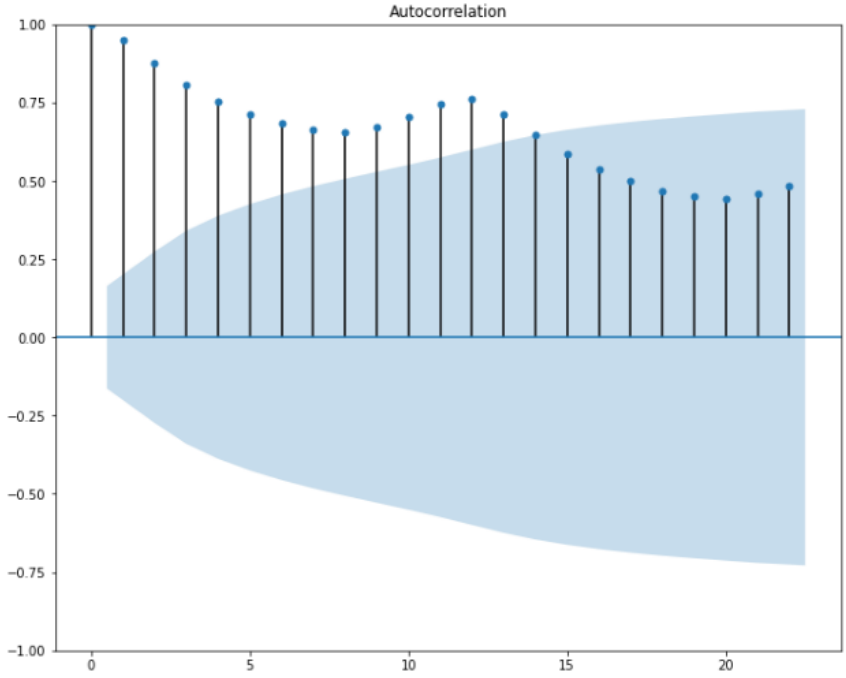

```
[ ] lag = data[1:]

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

```
-0.71
```

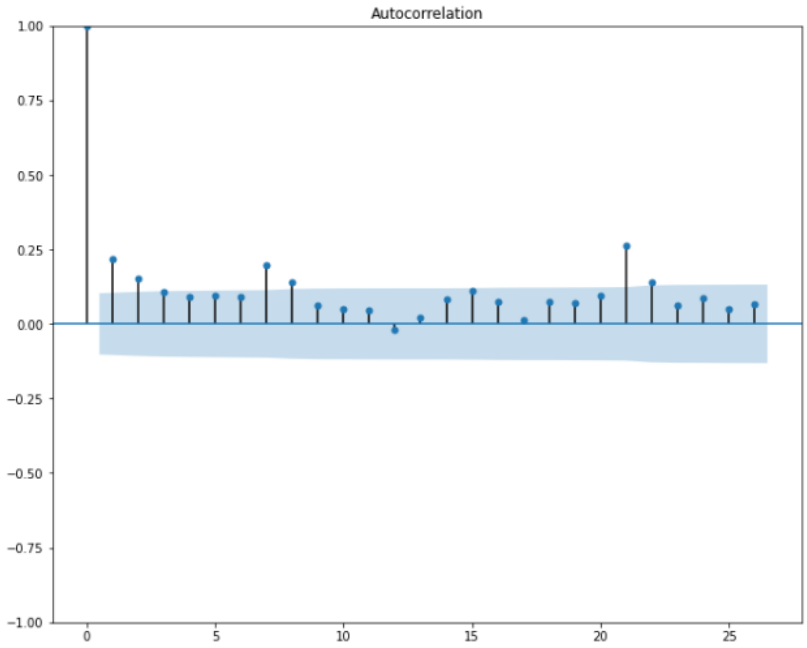
```
[ ] # импортируем автокорреляционную функцию (ACF)
    from statsmodels.graphics.tsaplots import plot_acf

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




```
[ ] # применим ее к данным о пассажирах
    plot_acf(passengers)
    plt.show()
```



```
[ ] # построим аналогичный график для данных о рождаемости
    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]]

    # в цикле for последовательно применяем формулу ко всем элементам ряда
    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)
```

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

```
[ ] # теперь нам нужно сдвинуть второй столбец на один день вперед (ведь это прогноз)

    # вначале создадим индекс за 1 января

    # для этого импортируем класс timedelta
    from datetime import timedelta

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

    # # "прибавим" один день
    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 |
| 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()
```

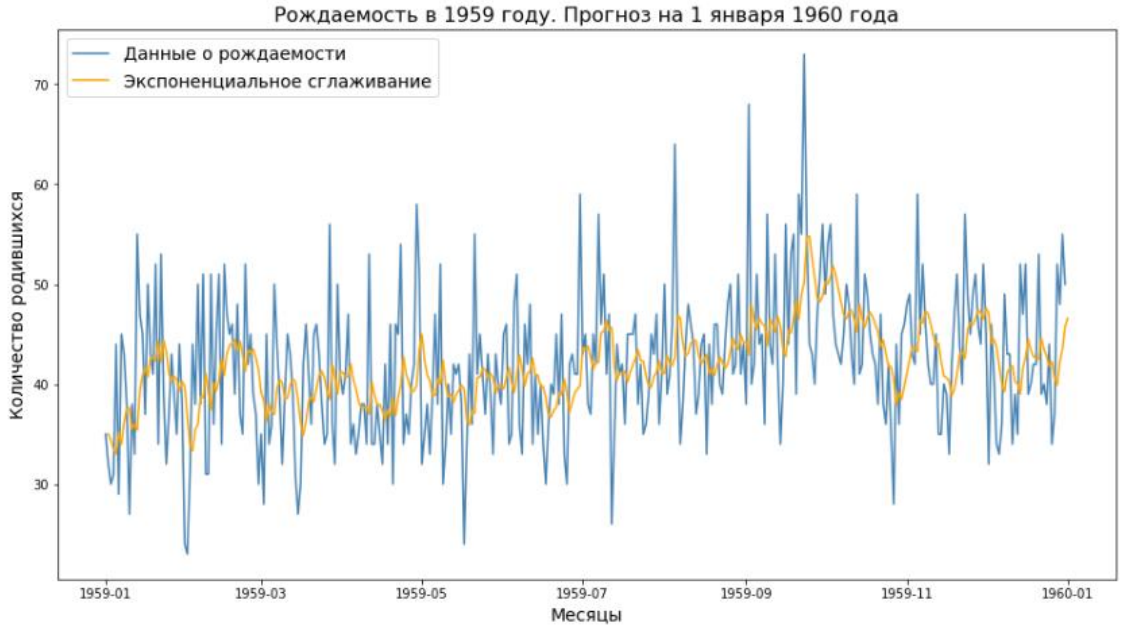
| Date | Births | Exp_smoothing |
|---|---|---|
| 1959-01-01 | 35.0 | NaN |
| 1959-01-02 | 32.0 | 35.000 |
| 1959-01-03 | 30.0 | 34.400 |
| 1959-01-04 | 31.0 | 33.520 |
| 1959-01-05 | 44.0 | 33.016 |

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

```
[ ] # 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 |

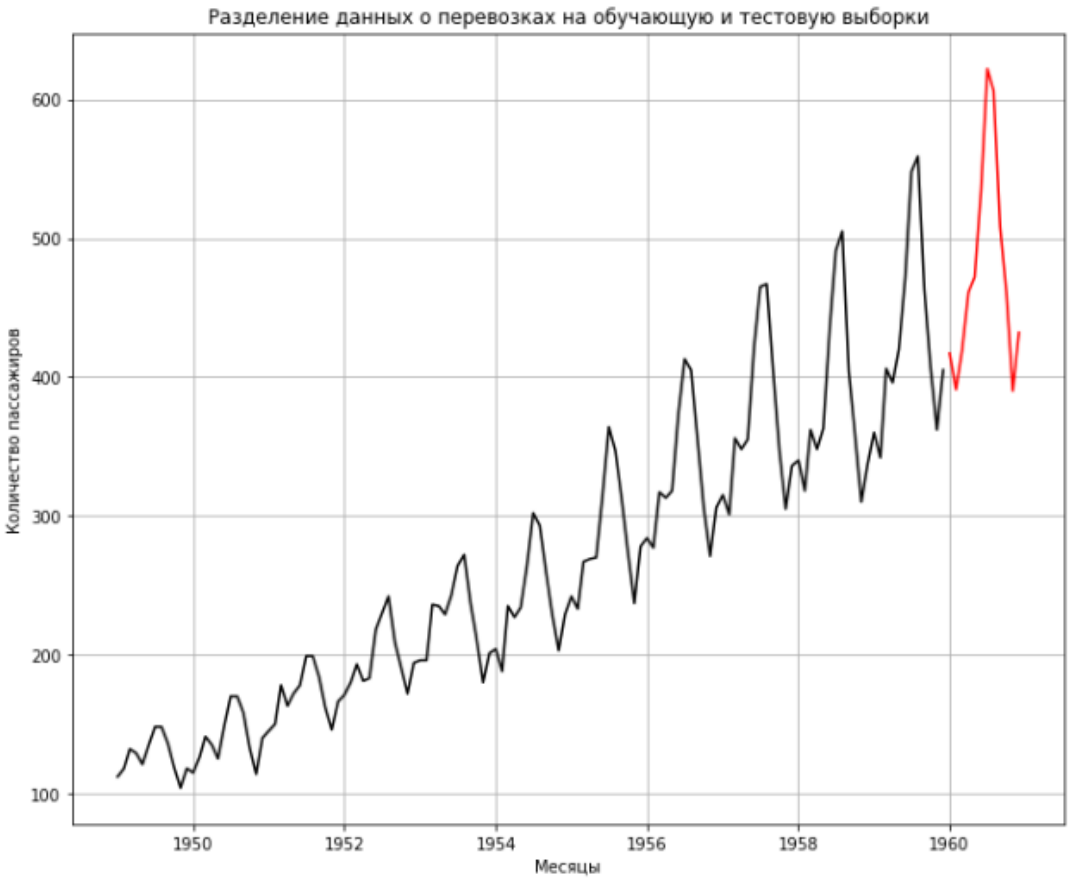
```
[ ] # посмотрим на результат на графике

    # зададим размер
    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)

    # добавим подписи к осям и заголовки
    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")

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

    # добавим сетку
    plt.grid()

    plt.show()
```



```
[ ] # принудительно отключим предупреждения системы
    import warnings
    warnings.simplefilter(action = 'ignore', category = Warning)

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

    # импортируем класс модели
    from statsmodels.tsa.statespace.sarimax import SARIMAX

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

    # применим метод fit
    result = model.fit()
```

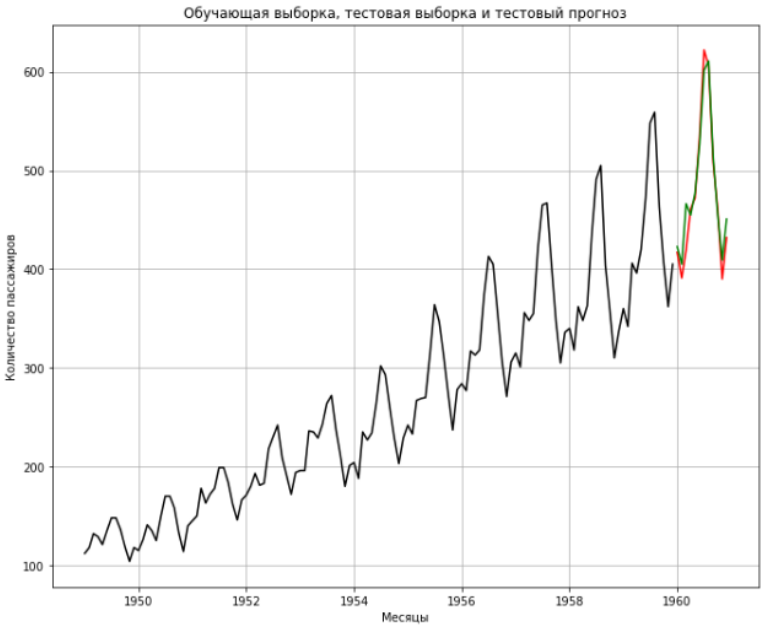
```
[ ] # мы можем посмотреть результат с помощью метода summary()
    print(result.summary())
```

```
                                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       0.027       0.548
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

    # применим метод predict
    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')

    # заголовок и подписи к осям
    plt.title('Обучающая выборка, тестовая выборка и тестовый прогноз')
    plt.ylabel('Количество пассажиров')
    plt.xlabel('Месяцы')

    # добавим сетку
    plt.grid()

    plt.show()
```



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

# заголовок и подписи к осям
plt.title('Фактические данные и прогноз на будущее')
plt.ylabel('Количество пассажиров')
plt.xlabel('Месяцы')

# добавим сетку
plt.grid()

plt.show()
```