```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.datasets import get_rdataset

# Загружаем данные
df = get_rdataset("AirPassengers", "datasets").data

# Преобразуем десятичное время в datetime
def decimal_year_to_datetime(dec_year):
    year = int(dec_year)
    month = int(round((dec_year - year) * 12)) + 1
    # Обработка граничного случая: если month == 13 (из-за округления)
    if month > 12:
        year += 1
        month = 1
    return pd.to_datetime(f"{year}-{month:02d}")

df['time'] = df['time'].apply(decimal_year_to_datetime)
df.set_index('time', inplace=True)
df.columns = ['passengers']

# Проверяем результат
print(df.head())
```

```
            passengers
time
1949-01-01         112
1949-02-01         118
1949-03-01         132
1949-04-01         129
1949-05-01         121
```

```
plt.figure(figsize=(12, 5))
plt.plot(df.index, df['passengers'], label='Пассажиры')
plt.title('Число авиапассажиров (1949–1960)')
plt.xlabel('Год')
plt.ylabel('Количество (тыс.)')
plt.grid(True)
plt.legend()
plt.show()
```


```
from statsmodels.tsa.stattools import adfuller

result = adfuller(df['passengers'])
print('ADF Statistic:', result[0])
print('p-value:', result[1])

if result[1] < 0.05:
    print("Ряд стационарен (отвергаем нулевую гипотезу)")
else:
    print("Ряд НЕ стационарен (принимаем нулевую гипотезу)")
```

```
ADF Statistic: 0.8153688792060498
p-value: 0.991880243437641
Ряд НЕ стационарен (принимаем нулевую гипотезу)
```

```
!pip install prophet
```

```
Requirement already satisfied: prophet in /usr/local/lib/python3.12/dist-packages (1.1.7)
Requirement already satisfied: cmdstanpy>=1.0.4 in /usr/local/lib/python3.12/dist-packages (from prophet) (1.2.5)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.12/dist-packages (from prophet) (2.0.2)
Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from prophet) (3.10.0)
Requirement already satisfied: pandas>=1.0.4 in /usr/local/lib/python3.12/dist-packages (from prophet) (2.2.2)
Requirement already satisfied: holidays<1,>=0.25 in /usr/local/lib/python3.12/dist-packages (from prophet) (0.81)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.12/dist-packages (from prophet) (4.67.1)
Requirement already satisfied: importlib_resources in /usr/local/lib/python3.12/dist-packages (from prophet) (6.5.2)
Requirement already satisfied: stanio<2.0.0,>=0.4.0 in /usr/local/lib/python3.12/dist-packages (from cmdstanpy>=1.0.4->prophet) (0.5.1)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.12/dist-packages (from holidays<1,>=0.25->prophet) (2.9.0.post0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=2.0.0->prophet) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=2.0.0->prophet) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=2.0.0->prophet) (4.60.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=2.0.0->prophet) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=2.0.0->prophet) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=2.0.0->prophet) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=2.0.0->prophet) (3.2.4)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.4->prophet) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.4->prophet) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil->holidays<1,>=0.25->prophet) (1.17.0)
```

```
from prophet import Prophet

# Prophet требует столбцы ds (дата) и y (значение)
df_prophet = df.reset_index()
df_prophet.columns = ['ds', 'y']

# Создаём и обучаем модель
```

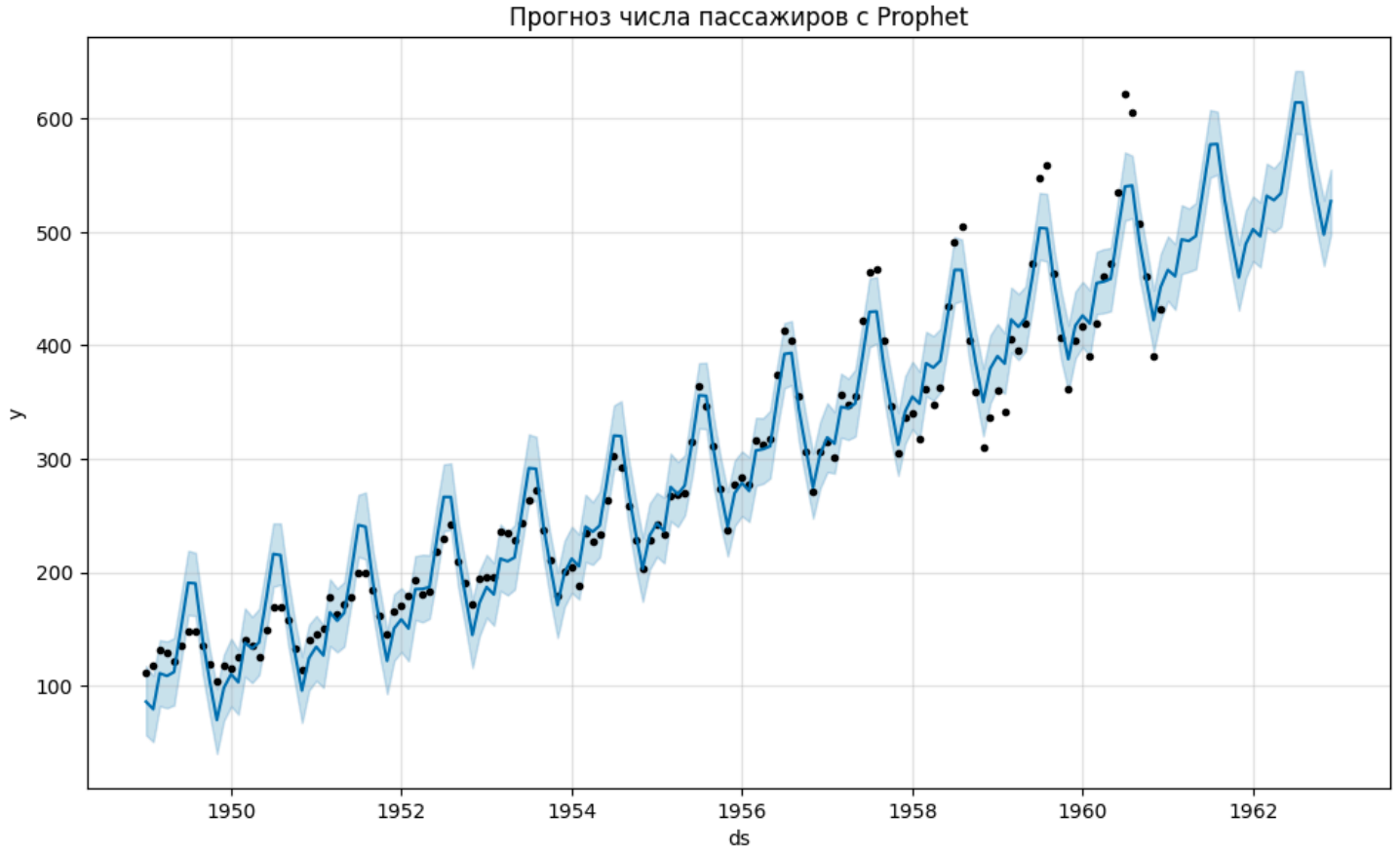
```
model = Prophet(yearly_seasonality=True, weekly_seasonality=False)
model.fit(df_prophet)

# Создаём будущие даты (прогноз на 24 месяца вперёд)
future = model.make_future_dataframe(periods=24, freq='MS')  # MS = начало месяца

# Делаем прогноз
forecast = model.predict(future)

# Визуализация
fig = model.plot(forecast)
plt.title('Прогноз числа пассажиров с Prophet')
plt.show()
```

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpc_1xwgn2/255uxeq9.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpc_1xwgn2/21vtmco9.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=28383', 'data', 'file=/tmp/tmpc_1xwgn2/255uxeq9.json', 'ini
09:38:16 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
09:38:16 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```



```
# Сравним реальные и предсказанные значения (только для прошлого)
forecast_past = forecast[forecast['ds'] <= df.index[-1]]
y_true = df['passengers'].values
y_pred = forecast_past['yhat'].values

from sklearn.metrics import mean_absolute_percentage_error
mape = mean_absolute_percentage_error(y_true, y_pred) * 100
print(f"MAPE (средняя абсолютная процентная ошибка): {mape:.2f}%")
```

```
MAPE (средняя абсолютная процентная ошибка): 7.22%
```

Уровень 1: Практика Построй прогноз на 12 месяцев (а не 24). Выведи компоненты модели Prophet: тренд, годовую сезонность.

```
model.plot_components(forecast)
```

Уровень 2: Эксперименты Попробуй модель SARIMA (сезонная ARIMA) из statsmodels. Подсказка: order=(1,1,1), seasonal_order=(1,1,1,12). Сравни MAPE у Prophet и SARIMA — какая точнее?

```
Напишите программный код или сгенерируйте его с помощью искусственного интеллекта.
```