```
def DA_pro_1():
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from sklearn.linear_model import LinearRegression

    # Generate synthetic engine gas path data
    np.random.seed(42)
    time = np.arange(0, 100, 0.5)  # Time axis [hours]
    N = len(time)
    PR = 12 + 0.6 * np.sin(0.12 * time) - 0.02 * time + np.random.normal(0, 0.18, N)
    PR[60:70] -= 1.5   # Simulate mild surge dip
    eff = 0.87 - 0.0008 * time + np.random.normal(0, 0.01, N)
    TET = 1320 + (40 * np.cos(0.15 * time)) + np.random.normal(0, 8, N)
    thrust = 105 + 10*(TET/1400) + 2*np.random.normal(0, 1, N)
    far = 0.037 + 0.003 * np.sin(0.3*time) + np.random.normal(0, 0.001, N)
    ab_on = ((time % 14) > 11).astype(int)
    TET[ab_on == 1] += 55
    thrust[ab_on == 1] += 10

    df = pd.DataFrame({
        'time': time,
        'PR': PR,
        'eff': eff,
        'TET': TET,
        'thrust': thrust,
        'far': far,
        'afterburner': ab_on
    })

    # Event & anomaly detection (rolling window and thresholds)
    df['PR_rolling'] = df['PR'].rolling(10, center=True).mean()
    df['eff_rolling'] = df['eff'].rolling(20, center=
```

```
33 True).mean()
34     df['TET_alert'] = df['TET'] > 1380  # Overheat
35     df['eff_alert'] = df['eff_rolling'] < 0.80  #
   Efficiency degradation
36     df['PR_dip_alert'] = (df['PR'] < df['PR_rolling'
   ] - 1.2)  # Surge warning
37
38     # Predictive maintenance: compressor efficiency
   RUL
39     reg = LinearRegression()
40     reg.fit(df['time'].values.reshape(-1, 1), df['eff
   '])
41     pred_end = (0.80 - df['eff'].iloc[-1]) / reg.
   coef_[0]
42     print(f"Predicted hours to minimum compressor
   efficiency: {abs(pred_end):.1f}")
43
44     # PLOTTING WITH LABELING, LEGENDS, and AXES
   DEFINED
45
46     plt.figure(figsize=(14, 8))
47
48     # 1. Compressor Pressure Ratio & Surge Warnings
49     plt.subplot(3, 1, 1)
50     plt.plot(df['time'], df['PR'], label='Compressor
   Pressure Ratio (PR)', color='blue')
51     plt.scatter(df['time'][df['PR_dip_alert']], df['
   PR'][df['PR_dip_alert']], color='red', label='Surge
   Warning')
52     plt.xlabel('Time (hours)')
53     plt.ylabel('Pressure Ratio (dimensionless)')
54     plt.title('Compressor PR & Surge Detection')
55     plt.grid(True)
56     plt.legend(loc='best')
57
58     # 2. Compressor Efficiency Trends
59     plt.subplot(3, 1, 2)
60     plt.plot(df['time'], df['eff'], label='Compressor
    Efficiency', color='blue')
61     plt.plot(df['time'], df['eff_rolling'], '--',
   label='Rolling Mean Efficiency', color='orange')
```

```
    plt.scatter(df['time'][df['eff_alert']], df['eff
'][df['eff_alert']], color='orange', label='
Efficiency Alert')
    plt.xlabel('Time (hours)')
    plt.ylabel('Efficiency (0-1)')
    plt.title('Compressor Efficiency Trends')
    plt.grid(True)
    plt.legend(loc='best')

    # 3. TET, Overheat Alerts, and Afterburner
Status
    plt.subplot(3, 1, 3)
    plt.plot(df['time'], df['TET'], label='Turbine
Entry Temperature (TET) [K]', color='blue')
    plt.scatter(df['time'][df['TET_alert']], df['TET
'][df['TET_alert']], color='purple', label='Overheat
 Alert')
    plt.xlabel('Time (hours)')
    plt.ylabel('Temperature (K)')
    plt.title('TET with Afterburner Cycles &
Overheat Detection')
    plt.grid(True)

    # Add secondary axis for afterburner on/off
    ax2 = plt.gca().twinx()
    ax2.plot(df['time'], df['afterburner'], 'k--',
alpha=0.5, label='Afterburner Status (On=1, Off=0)')
    ax2.set_ylabel('Afterburner Status (On=1, Off=0
)')
    ax2.set_ylim(-0.1, 1.1)

    # Combine legends from both y-axes for clarity
    lines_1, labels_1 = plt.gca().
get_legend_handles_labels()
    lines_2, labels_2 = ax2.
get_legend_handles_labels()
    plt.legend(lines_1 + lines_2, labels_1 +
labels_2, loc='upper right')

    plt.tight_layout()
    plt.show()
```

```
 91
 92
 93 DA_pro_1()
 94
 95
 96
 97
 98
 99
100
101
102
103
104
```