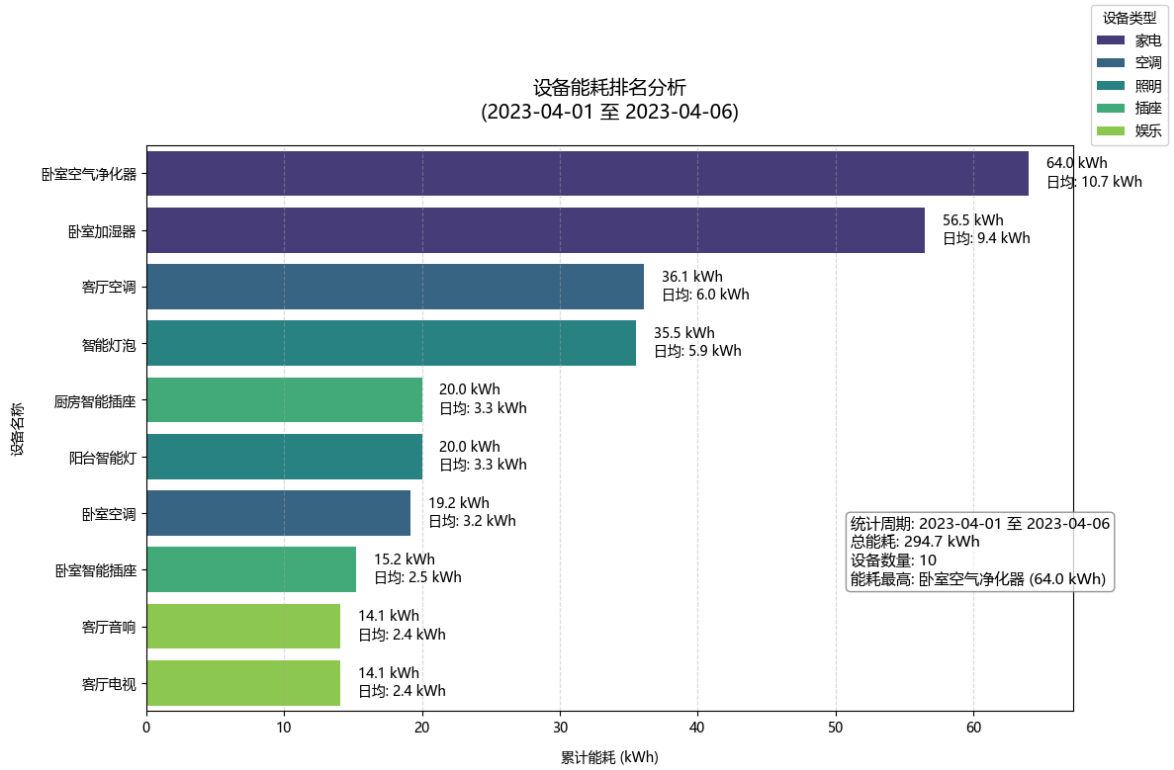
In [1]:
```
import requests
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from ipywidgets import interact, widgets

# FastAPI服务地址（确保服务已启动）
BASE_URL = "http://127.0.0.1:8000"
```

In [13]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import requests
from datetime import datetime

# 设置中文字体和负号显示
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.rcParams['axes.unicode_minus'] = False

def plot_device_energy_ranking(start_date='2023-04-01', end_date='2023-04-06'):
    """设备能耗排名分析"""
    # 调用API获取数据
    url = f"{BASE_URL}/analysis/device_energy_ranking"
    params = {
        "start_date": start_date,
        "end_date": end_date
    }

    try:
        response = requests.get(url, params=params, timeout=5)
        response.raise_for_status()
        data = response.json()
    except Exception as e:
        print(f"API请求失败: {str(e)}")
        return

    # 数据校验与处理
    if not data.get('summary'):
        print("无有效数据，请检查日期范围")
        return

    df = pd.DataFrame(data['summary'])

    # 创建可视化图表
    plt.figure(figsize=(12, 8))

    # 绘制水平条形图（按能耗排序）
    df = df.sort_values('total_energy', ascending=False)
    ax = sns.barplot(
        data=df,
        x='total_energy',
        y='name',
        hue='type',
        palette='viridis',
        dodge=False,
        saturation=0.8
    )
```

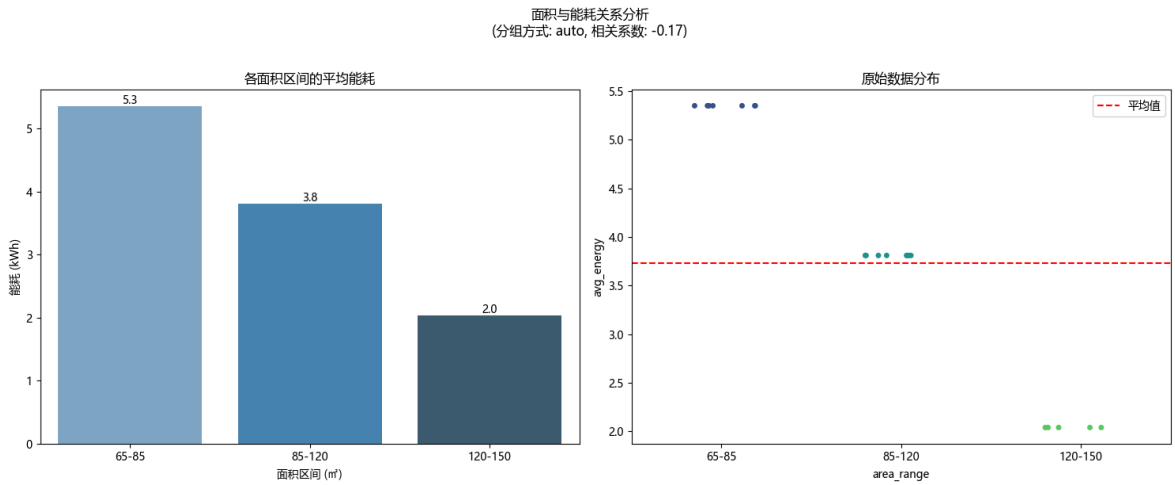
```
    # 计算日期差（字符串日期转换为datetime对象）
    date_diff = (datetime.strptime(end_date, '%Y-%m-%d') -
                 datetime.strptime(start_date, '%Y-%m-%d')).days + 1

    # 添加数据标签
    for i, (_, row) in enumerate(df.iterrows()):
        ax.text(
            row['total_energy'] + max(df['total_energy'])*0.02,
            i,
            f"{row['total_energy']:.1f} kWh\n日均: {row['total_energy']/date_dif
            va='center',
            ha='left',
            fontsize=10
        )

    # 添加统计信息
    total_energy = df['total_energy'].sum()
    stats_text = (
        f"统计周期: {start_date} 至 {end_date}\n"
        f"总能耗: {total_energy:.1f} kWh\n"
        f"设备数量: {len(df)}\n"
        f"能耗最高: {df.iloc[0]['name']} ({df.iloc[0]['total_energy']:.1f} kWh)"
    )

    plt.gcf().text(
        0.72, 0.25,
        stats_text,
        bbox=dict(facecolor='white', alpha=0.8, edgecolor='gray', boxstyle='roun
        fontsize=11
    )

    # 添加图表标注
    plt.title(f"设备能耗排名分析\n({start_date} 至 {end_date})", pad=20, fontsiz
    plt.xlabel("累计能耗 (kWh)", labelpad=10)
    plt.ylabel("设备名称", labelpad=10)
    plt.grid(axis='x', linestyle='--', alpha=0.4)
    plt.legend(title='设备类型', bbox_to_anchor=(1.02, 1), borderaxespad=0)

    # 调整布局并显示
    plt.tight_layout()
    plt.savefig('device_energy_ranking.png', dpi=300, bbox_inches='tight')  # 保
    plt.show()

# 直接调用函数
plot_device_energy_ranking(start_date='2023-04-01', end_date='2023-04-06')
```




In [15]:

```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import requests

def plot_area_energy(group_type='auto', custom_ranges="65-85,86-120,121-150"):
    """面积与能耗关系分析"""
    # 调用API
    url = f"{BASE_URL}/analysis/area_vs_energy"
    params = {"group_type": group_type, "custom_ranges": custom_ranges}

    try:
        response = requests.get(url, params=params)
        response.raise_for_status()
        data = response.json()
    except Exception as e:
        print(f"API请求失败: {str(e)}")
        return

    # 创建画布
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

    # 子图1: 分组能耗柱状图
    df_groups = pd.DataFrame(data['area_groups'])
    sns.barplot(data=df_groups, x='area_range', y='avg_energy',
                hue='area_range', palette="Blues_d", legend=False, ax=ax1)
    ax1.set_title("各面积区间的平均能耗")
    ax1.set_xlabel("面积区间 (㎡)")
    ax1.set_ylabel("能耗 (kWh)")

    # 添加数据标签
    for p in ax1.patches:
        ax1.annotate(f"{p.get_height():.1f}",
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha='center', va='center',
                     xytext=(0, 5),
```

```
                    textcoords='offset points')

    # 子图2：原始数据散点图
    df_points = pd.DataFrame([(g['area_range'], g['avg_energy'])
                              for g in data['area_groups']
                              for _ in range(g['user_count'])],
                             columns=['area_range', 'avg_energy'])

    sns.stripplot(data=df_points, x='area_range', y='avg_energy',
                  hue='area_range', palette="viridis", legend=False,
                  jitter=0.2, ax=ax2)
    ax2.set_title("原始数据分布")
    ax2.axhline(y=df_groups['avg_energy'].mean(), color='r', linestyle='--', lab
    ax2.legend()

    # 设置主标题
    plt.suptitle(f"面积与能耗关系分析\n(分组方式: {group_type}, 相关系数: {data['

    # 调整布局并保存
    plt.tight_layout()
    plt.savefig('area_energy_analysis.png', dpi=300, bbox_inches='tight')
    plt.show()

# 调用示例
plot_area_energy()
```




In [11]:

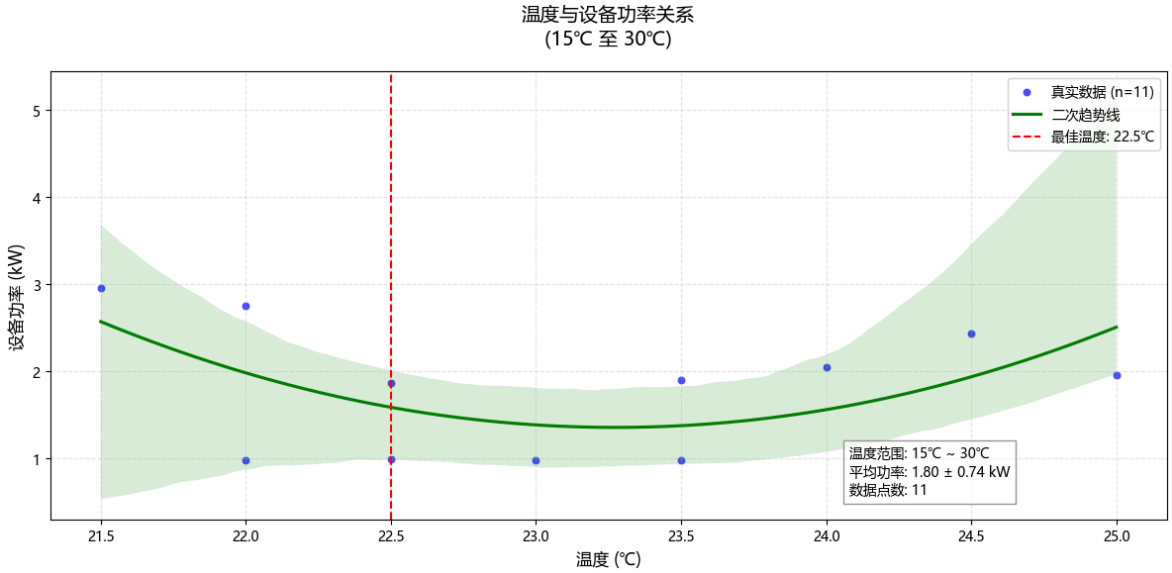
```
# 温度与功率分析可视化
import requests
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# 初始化设置
%matplotlib inline
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  # 中文字体
plt.rcParams['axes.unicode_minus'] = False  # 负号显示

# FastAPI服务地址
BASE_URL = "http://127.0.0.1:8000"

def fetch_power_data(temp_min, temp_max):
    try:
        response = requests.get(
```

```
            f"{BASE_URL}/analysis/temperature_vs_power",
            params={"temp_min": temp_min, "temp_max": temp_max},
            timeout=5
        )
        data = response.json()

        # 检查数据有效性
        if not data.get('data') or len(data['data']) < 3:
            print(f"数据不足（{len(data.get('data', []))}条），至少需要3条数据")
            return None, None

        df = pd.DataFrame(data['data'])

        # 数据清洗：移除极端异常值（保留±3σ内的数据）
        z_scores = stats.zscore(df['power_kw'])
        df = df[(np.abs(z_scores) < 3)]

        if len(df) < 3:
            print(f"有效数据不足（{len(df)}条），检查数据质量")
            return None, None

        return df, data.get('optimal_temp')

    except Exception as e:
        print(f"API请求失败: {str(e)}")
        return None, None

def plot_temperature_vs_power(temp_min=15, temp_max=30):
    """ 温度-功率关系可视化 """
    # 获取数据
    df, optimal_temp = fetch_power_data(temp_min, temp_max)
    if df is None:
        # 绘制空白图提示
        plt.figure(figsize=(10, 5))
        plt.text(0.5, 0.5,
                 "无足够有效数据\n请尝试扩大温度范围或检查数据源",
                 ha='center', va='center', fontsize=12)
        plt.axis('off')
        plt.show()
        return

    # 创建画布
    plt.figure(figsize=(12, 6))

    # 1. 散点图
    sns.scatterplot(
        data=df,
        x='temperature',
        y='power_kw',
        alpha=0.7,
        color='blue',
        label=f'真实数据 (n={len(df)})'
    )

    # 2. 趋势线（仅当数据量≥5时绘制）
    if len(df) >= 5:
        sns.regplot(
            data=df,
            x='temperature',
            y='power_kw',
```

```
            order=2,
            scatter=False,
            color='green',
            label='二次趋势线',
            ci=95
        )
    else:
        print("数据量不足（<5条），无法绘制趋势线")

    # 3. 标记最佳温度（如果存在）
    if optimal_temp:
        plt.axvline(
            x=optimal_temp,
            color='red',
            linestyle='--',
            linewidth=1.5,
            label=f"最佳温度: {optimal_temp}℃"
        )
    else:
        print("未计算出最佳温度点")

    # 4. 添加统计信息
    stats_text = (
        f"温度范围: {temp_min}℃ ~ {temp_max}℃\n"
        f"平均功率: {df['power_kw'].mean():.2f} ± {df['power_kw'].std():.2f} kW\
        f"数据点数: {len(df)}"
    )

    plt.gcf().text(
        0.72, 0.15,
        stats_text,
        bbox=dict(facecolor='white', alpha=0.8, edgecolor='gray'),
        fontsize=10
    )

    # 5. 添加图表标注
    plt.title(f"温度与设备功率关系\n({temp_min}℃ 至 {temp_max}℃)", pad=20, fon
    plt.xlabel("温度 (℃)", fontsize=12)
    plt.ylabel("设备功率 (kW)", fontsize=12)
    plt.grid(True, linestyle='--', alpha=0.3)
    plt.legend(loc='upper right', framealpha=0.8)

    plt.tight_layout()
    plt.show()

# 数据质量检查
def check_data_quality():
    """ 数据质量诊断 """
    print("正在检查数据质量...")
    test_ranges = [(15, 30), (20, 25), (10, 40)]  # 测试不同温度范围

    for t_min, t_max in test_ranges:
        response = requests.get(
            f"{BASE_URL}/analysis/temperature_vs_power",
            params={"temp_min": t_min, "temp_max": t_max}
        )
        data = response.json()
        n_data = len(data.get('data', []))

        print(f"温度 {t_min}-{t_max}℃: {n_data}条数据",
```

```
            f"(最佳温度: {data.get('optimal_temp', '无')}℃)")

# 直接调用函数，固定温度范围
print("温度与功率关系分析")
plot_temperature_vs_power(temp_min=15, temp_max=30)  # 设置默认温度范围
check_data_quality()
```

温度与功率关系分析




```
正在检查数据质量...
温度 15-30℃: 11条数据 (最佳温度: 22.5℃)
温度 20-25℃: 11条数据 (最佳温度: 21.5℃)
温度 10-40℃: 15条数据 (最佳温度: 22.5℃)
```

In [ ]: