在 [41]…
```
!pip install requests pandas matplotlib plotly wordcloud jieba seaborn textblob
```

```
Requirement already satisfied: requests in d:\aaa\lib\site-packages (2.32.2)
Requirement already satisfied: pandas in d:\aaa\lib\site-packages (2.2.2)
Requirement already satisfied: matplotlib in d:\aaa\lib\site-packages (3.8.4)
Requirement already satisfied: plotly in d:\aaa\lib\site-packages (5.22.0)
Requirement already satisfied: wordcloud in d:\aaa\lib\site-packages (1.9.4)
Requirement already satisfied: jieba in d:\aaa\lib\site-packages (0.42.1)
Requirement already satisfied: seaborn in d:\aaa\lib\site-packages (0.13.2)
Requirement already satisfied: textblob in d:\aaa\lib\site-packages (0.19.0)
Requirement already satisfied: charset-normalizer<4,>=2 in d:\aaa\lib\site-packages (from requests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in d:\aaa\lib\site-packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\aaa\lib\site-packages (from requests) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in d:\aaa\lib\site-packages (from requests) (2024.8.30)
Requirement already satisfied: numpy>=1.26.0 in d:\aaa\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in d:\aaa\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in d:\aaa\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in d:\aaa\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in d:\aaa\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in d:\aaa\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in d:\aaa\lib\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in d:\aaa\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in d:\aaa\lib\site-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in d:\aaa\lib\site-packages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in d:\aaa\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: tenacity>=6.2.0 in d:\aaa\lib\site-packages (from plotly) (8.2.2)
Requirement already satisfied: nltk>=3.9 in d:\aaa\lib\site-packages (from textblob) (3.9.1)
Requirement already satisfied: click in d:\aaa\lib\site-packages (from nltk>=3.9->textblob) (8.1.7)
Requirement already satisfied: joblib in d:\aaa\lib\site-packages (from nltk>=3.9->textblob) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in d:\aaa\lib\site-packages (from nltk>=3.9->textblob) (2023.10.3)
Requirement already satisfied: tqdm in d:\aaa\lib\site-packages (from nltk>=3.9->textblob) (4.66.4)
Requirement already satisfied: six>=1.5 in d:\aaa\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: colorama in d:\aaa\lib\site-packages (from click->nltk>=3.9->textblob) (0.4.6)
```

在 [1] …
```
# 载入包
import requests
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import jieba
from collections import Counter
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import seaborn as sns
from textblob import TextBlob
from ipywidgets import interact, widgets
from ipywidgets import Layout
from scipy import stats
import plotly.express as px


# 连接API
# 设置中文显示
plt.rcParams['font.sans-serif'] = ['SimHei']  # Windows中文
plt.rcParams['axes.unicode_minus'] = False

# API配置
BASE_URL = "http://localhost:8000"
```

在 [5] …
```
# 连接API
# 设置中文显示
plt.rcParams['font.sans-serif'] = ['SimHei']  # Windows中文
plt.rcParams['axes.unicode_minus'] = False

# API配置
BASE_URL = "http://localhost:8000"

def fetch_data(endpoint):
    """通用数据获取函数"""
    try:
        response = requests.get(f"{BASE_URL}{endpoint}")
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as e:
        print(f"获取数据失败: {e}")
        return None

# 获取所有需要的数据
print("正在从API获取数据...")
energy_data = fetch_data("/advanced_analysis/energy_breakdown")
feedback_texts = fetch_data("/advanced_analysis/sentiment_details")
device_feedbacks = fetch_data("/advanced_analysis/device_feedback_details")

if not all([energy_data, feedback_texts, device_feedbacks]):
```

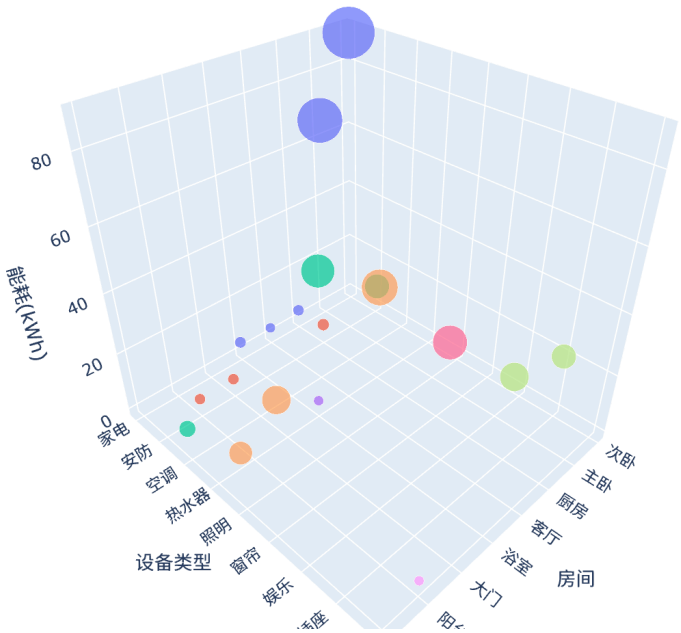
```
    print("数据获取不完整，请检查API服务")
else:
    # 转换为DataFrame方便处理
    energy_df = pd.DataFrame(energy_data)
    feedback_df = pd.DataFrame({"text": feedback_texts})
    device_feedback_df = pd.DataFrame(device_feedbacks)
    print("数据获取成功！")
```

正在从API获取数据...
数据获取成功！

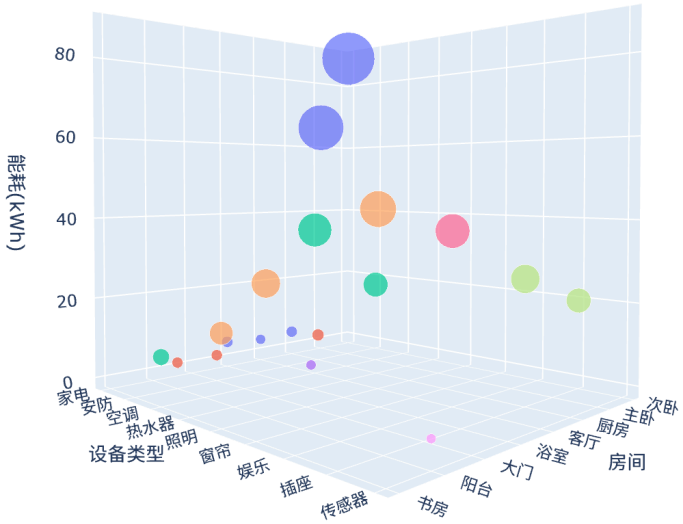
在 [33]…

```
# 不同房间的能耗
# 使用3D散点图模拟柱状图
fig = px.scatter_3d(energy_df,
                    x='room',
                    y='device_type',
                    z='total_energy',
                    color='device_type',
                    size='total_energy',
                    size_max=30,
                    title='<b>房间-设备类型-能耗分布</b>')

fig.update_layout(
    scene=dict(
        xaxis_title='房间',
        yaxis_title='设备类型',
        zaxis_title='能耗(kWh)'
    ),
    margin=dict(l=0, r=0, b=0, t=30)
)

fig.show()

fig.update_layout(
    scene_camera=dict(
        eye=dict(x=1.5, y=1.5, z=0.05),  # 视点位置（
        up=dict(x=0, y=0, z=1)          # 定义"上"方向
    )
)
```

房间-设备类型-能耗分布

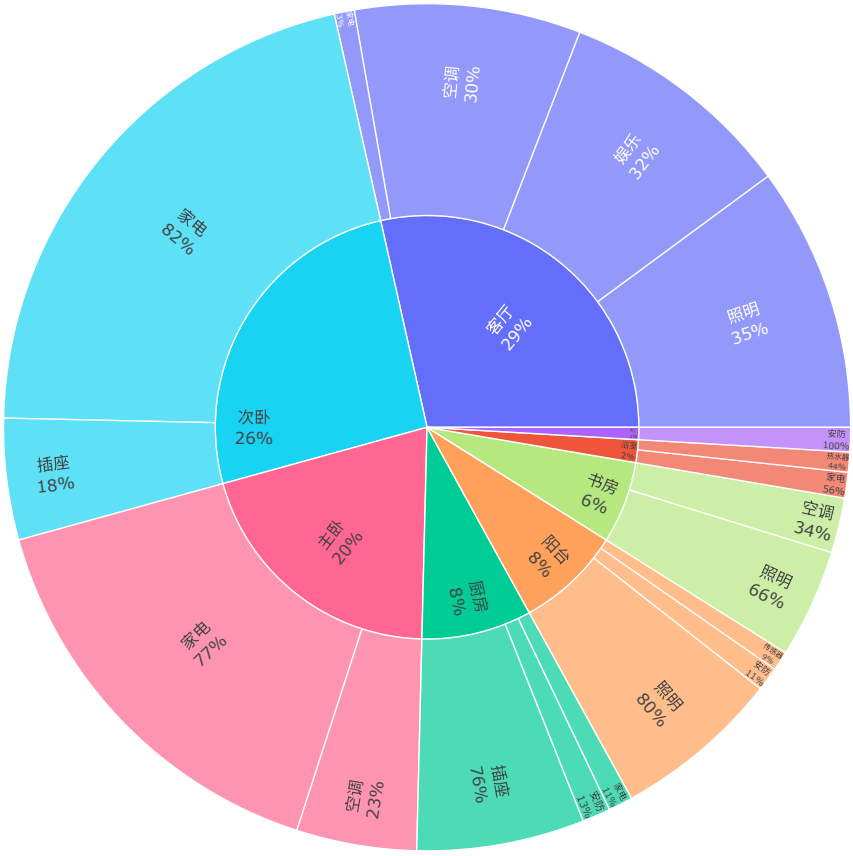


房间-设备类型-能耗分布



In [35]:
```
# 旭日图
fig = px.sunburst(
    energy_df,
    path=['room', 'device_type'],
    values='total_energy',
    title='<b>房间能耗分布旭日图</b>',
    color='room',
    width=1000,  # 调整宽度（像素）
    height=800   # 调整高度（像素）
)
fig.update_traces(
    insidetextorientation='radial',  # 文字沿径向排列
    textinfo='label+percent parent',  # 显示标签和父级占比
)

fig.show()
```

房间能耗分布旭日图

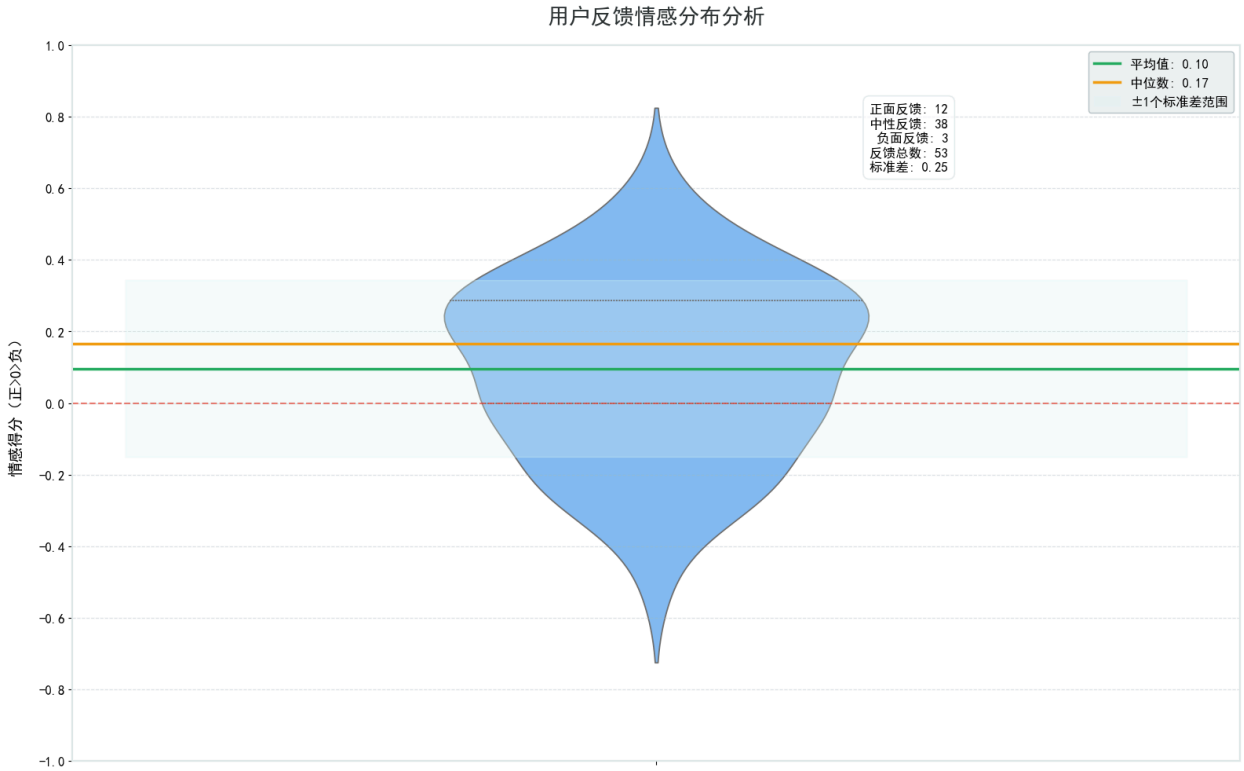


In [41]:
```
# 用户反馈
# 情感词典
positive_words = ["好", "满意", "方便", "智能", "快捷", "稳定", "安静", "节能",
                  "实用", "灵敏", "准确", "安全", "柔和", "干净", "合理", "改善",
                  "超出预期", "提升", "创意", "显著", "精准", "舒适", "清晰", "顺滑",
                  "柔和", "足够", "明显", "实用", "显著", "精准", "自然", "直观"]

negative_words = ["差", "不好", "慢", "卡顿", "故障", "噪音", "耗电", "复杂",
                  "不稳定", "太敏感", "一般", "连不上", "短", "不合理", "下降", "异响",
                  "失效", "不够自然", "不够直观", "下降明显", "丢失", "错误", "偏大",
                  "过小", "敏感", "误报", "耗电", "卡顿", "复杂", "异响", "偏大"]

# 扩展停用词列表
stop_words = ['的', '了', '是', '有', '在', '和', '就', '但', '很', '也', '又', '再',
              '这', '那', '与', '及', '或', '而', '且', '虽然', '但是', '然而', '因此',
              '所以', '因为', '如果', '功能', '模式', '效果', '速度', '时间', '工作',
              '后', '有时', '希望', '需要', '建议', '专区', '出现', '使用', '可以', '后',
              '有时', '但', '有点', '非常', '仍然', '希望', '需要', '建议', '一些', '不够',
              '出现', '使用', '可以', '功能', '模式', '效果', '速度', '时间', '工作', '后',
              '有时', '但', '有点', '非常', '仍然', 'hope', '需要', '建议', '一些', '不够','app',
              '一个']

# 计算情感分数的函数
def calculate_sentiment_scores(texts):
    scores = []
    for text in texts:
        words = [word for word in jieba.cut(text) if len(word) > 1 and word not in stop_words]
        if not words:  # 避免除以零
            scores.append(0)
            continue

        positive_count = sum(1 for word in words if word in positive_words)
        negative_count = sum(1 for word in words if word in negative_words)

        # 计算情感分数：(正面词数 - 负面词数) / 总词数
        sentiment_score = (positive_count - negative_count) / len(words)
        scores.append(sentiment_score)
```

```
    return scores

# 计算情感分数
sentiment_scores = calculate_sentiment_scores(feedback_texts)

# 准备文本数据
text = " ".join(feedback_texts)
words = [word for word in jieba.cut(text)
         if len(word) > 1 and word not in stop_words]
word_freq = Counter(words)

# 生成带情感颜色的词云
def color_func(word, font_size, position, orientation, random_state=None, **kwargs):
    if word in positive_words:
        return "#2ecc71"  # 绿色 - 正面
    elif word in negative_words:
        return "#e74c3c"  # 红色 - 负面
    else:
        return "#3498db"  # 蓝色 - 中性

wc = WordCloud(
    font_path="simhei.ttf",
    width=1200,
    height=600,
    background_color="#f8f9fa",  # 浅灰色背景
    max_words=50,  # 减少显示词数
    colormap=None,  # 禁用内置颜色映射
    contour_width=0,
    prefer_horizontal=0.8,  # 增加水平词比例
    relative_scaling=0.5
)
wc.generate_from_frequencies(word_freq)

# 创建词云图
plt.figure(figsize=(16, 10))
plt.imshow(wc.recolor(color_func=color_func), interpolation="bilinear")
plt.axis("off")

# 添加标题和图例
plt.title("用户反馈关键词情感分析", fontsize=18, pad=20, fontweight='bold')
plt.figtext(0.5, 0.05,
            "绿色=正面 | 红色=负面 | 蓝色=中性",
            ha="center", fontsize=12,
            bbox=dict(facecolor='white', alpha=0.7, edgecolor='lightgray'))

plt.tight_layout()
plt.show()
# 保存词云图
wc.to_file("sentiment_wordcloud.png")

# 情感分布图
def plot_sentiment_distribution(sentiment_scores, save_path=None):
    plt.figure(figsize=(16, 10))

    # 小提琴图
    sns.violinplot(
        y=sentiment_scores,
        inner="quartile",
        color="#74b9ff",  # 柔和的蓝色
        saturation=0.8,
        orient='v'  # 垂直方向
    )

    # 添加统计信息
    mean_score = np.mean(sentiment_scores)
    median_score = np.median(sentiment_scores)
    std_dev = np.std(sentiment_scores)

    # 添加中性线 (0点)
    plt.axhline(y=0, color='#e74c3c', linestyle='--', linewidth=1.5, alpha=0.7)

    # 添加均值和中位数线
    plt.axhline(y=mean_score, color='#27ae60', linestyle='-',
                linewidth=2.5, label=f'平均值: {mean_score:.2f}')
    plt.axhline(y=median_score, color='#f39c12', linestyle='-',
                linewidth=2.5, label=f'中位数: {median_score:.2f}')

    # 添加标准差区域
    std_plus = mean_score + std_dev
    std_minus = mean_score - std_dev
    plt.fill_betweenx([std_minus, std_plus], -1, 1,
                      color='#dff9fb', alpha=0.3, label='±1个标准差范围')

    # 添加分布信息
    plt.text(0.75, 0.92,
             f"正面反馈: {sum(1 for s in sentiment_scores if s > 0.3):,}\n"
             f"中性反馈: {sum(1 for s in sentiment_scores if -0.3 <= s <= 0.3):,}\n"
             f"负面反馈: {sum(1 for s in sentiment_scores if s < -0.3):,}\n"
```

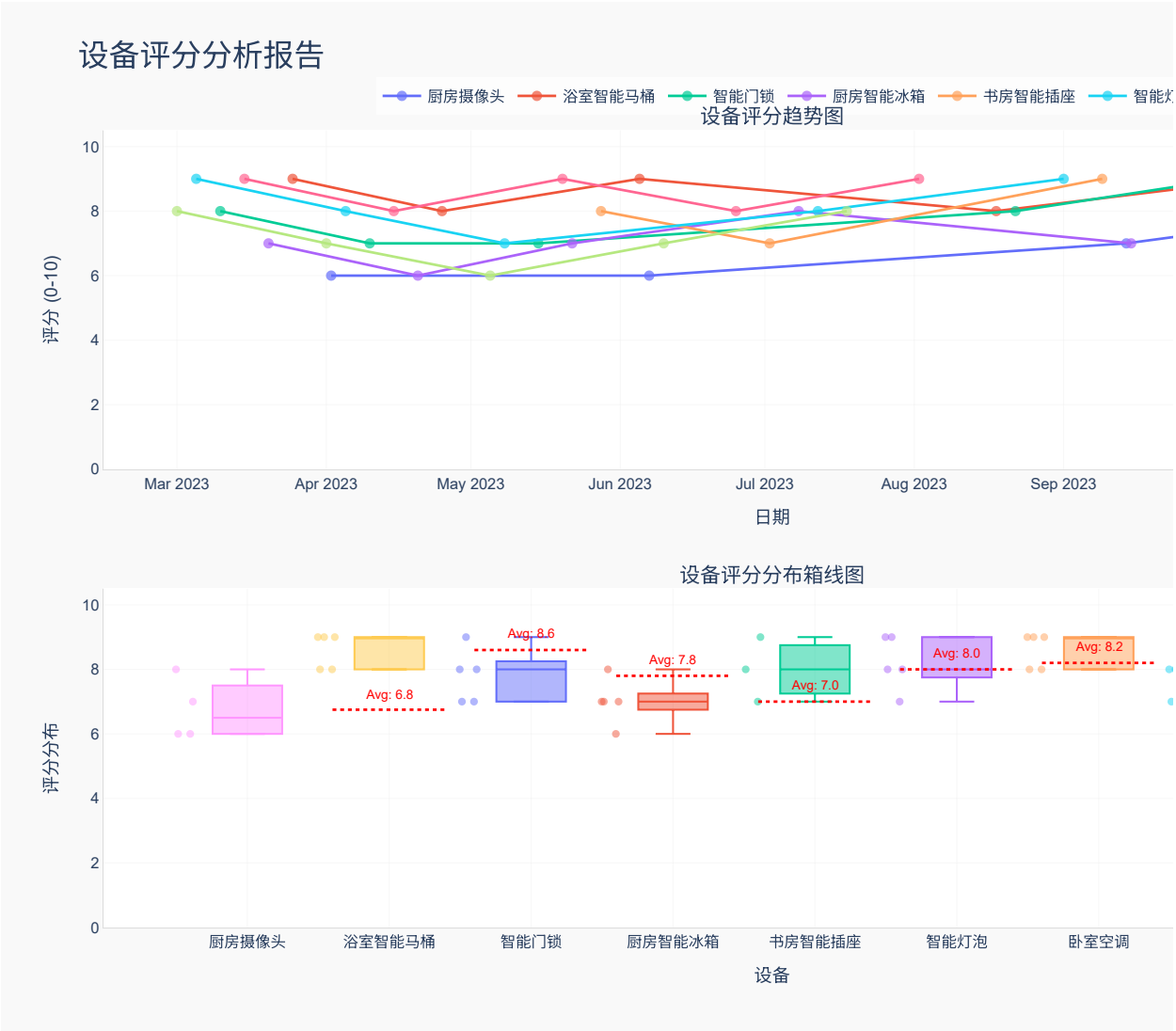
```
            f"反馈总数: {len(sentiment_scores):,}\n"
            f"标准差: {std_dev:.2f}",
            transform=plt.gca().transAxes,
            ha='right', va='top', fontsize=12,
            bbox=dict(facecolor='white', alpha=0.8, edgecolor='#dfe6e9', boxstyle="round,pad=0.5"))

    # 设置标题和标签
    plt.title("用户反馈情感分布分析", fontsize=20, pad=20, fontweight='bold', color='#2d3436')
    plt.ylabel("情感得分（正>0>负）", fontsize=14, labelpad=15)

    # 设置Y轴范围（1.1倍扩展）
    y_max = max(abs(min(sentiment_scores)), abs(max(sentiment_scores))) * 1.1
    plt.ylim(-y_max, y_max)

    # 添加背景网格和美化
    plt.grid(axis='y', linestyle='--', alpha=0.4, color='#b2bec3')

    # 自定义刻度
    plt.yticks(np.arange(-1.0, 1.1, 0.2), fontsize=12)

    # 添加图例
    plt.legend(loc='best', fontsize=12, framealpha=0.9,
               facecolor='#ecf0f1', edgecolor='#b2bec3')

    # 添加美观的边框
    for spine in plt.gca().spines.values():
        spine.set_color('#dfe6e9')
        spine.set_linewidth(1.5)

    plt.tight_layout()

    # 保存图形
    if save_path:
        plt.savefig(save_path, dpi=300, bbox_inches='tight', facecolor='white')

    plt.show()
    plt.close()

# 使用函数绘制情感分布并保存
plot_sentiment_distribution(sentiment_scores, save_path="sentiment.png")
```

用户反馈关键词情感分析



In [43]:
```
# 满意度评分
# 转换日期格式
device_feedback_df['feedback_time'] = pd.to_datetime(device_feedback_df['feedback_time'])
device_feedback_df['date'] = device_feedback_df['feedback_time'].dt.date
device_feedback_df['month'] = device_feedback_df['feedback_time'].dt.to_period('M')

# 创建包含两个子图的图表 (2行1列)
fig = make_subplots(
    rows=2,
    cols=1,
    shared_xaxes=False,
    vertical_spacing=0.15,
    subplot_titles=('设备评分趋势图', '设备评分分布箱线图'),
    specs=[[{"type": "xy"}], [{"type": "xy"}]]
)

# 时间序列折线图 (位于第1行)
for device in device_feedback_df['device_name'].unique():
    df_device = device_feedback_df[device_feedback_df['device_name'] == device]
    # 按日期排序以确保线图正确
    df_device = df_device.sort_values('date')

    fig.add_trace(
        go.Scatter(
            x=df_device['date'],
            y=df_device['rating'],
            mode='lines+markers',
            name=device,
            hovertemplate=f"设备: {device}<br>日期: %{{x}}<br>评分: %{{y}}",
            line=dict(width=2),
            marker=dict(size=8, opacity=0.7)
        ),
        row=1, col=1
    )

# 箱线图 (位于第2行)
for device in device_feedback_df['device_name'].unique():
    df_device = device_feedback_df[device_feedback_df['device_name'] == device]
    fig.add_trace(
        go.Box(
            y=df_device['rating'],
            name=device,
            boxpoints='all',  # 显示所有数据点
            jitter=0.3,       # 点的分散度
            pointpos=-1.8,    # 点相对于箱子的位置
            marker=dict(size=6, opacity=0.5),
            line=dict(width=1.5),
            showlegend=False  # 不在图例中重复显示
        ),
        row=2, col=1
    )

# 更新布局
```

```
fig.update_layout(
    height=800,  # 设置整个图表的高度
    width=1200,   # 设置整个图表的宽度
    title_text='设备评分分析报告',
    hovermode='closest',
    title_font=dict(size=24, family='Arial, sans-serif'),

    # 美化X轴
    xaxis=dict(
        title='日期',
        gridcolor='#f0f0f0',
        showline=True,
        linecolor='#bdbdbd'
    ),
    xaxis2=dict(
        title='设备',
        gridcolor='#f0f0f0',
        showline=True,
        linecolor='#bdbdbd'
    ),

    # 美化Y轴
    yaxis=dict(
        title='评分 (0-10)',
        range=[0, 10.5],  # 设置范围，稍微超出最大评分
        gridcolor='#f0f0f0',
        showline=True,
        linecolor='#bdbdbd'
    ),
    yaxis2=dict(
        title='评分分布',
        range=[0, 10.5],
        gridcolor='#f0f0f0',
        showline=True,
        linecolor='#bdbdbd'
    ),

    # 图例设置
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=1.02,
        xanchor="right",
        x=1,
        bgcolor='rgba(255,255,255,0.7)'
    ),

    # 整体样式
    plot_bgcolor='white',
    paper_bgcolor='#f9f9f9',
    font=dict(family='Arial, sans-serif', size=12)
)

# 为箱线图添加平均线
for i, device in enumerate(device_feedback_df['device_name'].unique(), start=1):
    df_device = device_feedback_df[device_feedback_df['device_name'] == device]
    mean_rating = df_device['rating'].mean()

    fig.add_shape(
        type='line',
        x0=i-0.4, x1=i+0.4,
        y0=mean_rating, y1=mean_rating,
        line=dict(color='red', width=2, dash='dot'),
        row=2, col=1
    )
    # 添加平均值标签
    fig.add_annotation(
        x=i,
        y=mean_rating + 0.5,
        text=f"Avg: {mean_rating:.1f}",
        showarrow=False,
        font=dict(color='red', size=10),
        row=2, col=1
    )

# 显示图表
fig.show()
```



In [45]:
```python
# 计算关键指标
total_energy = energy_df['total_energy'].sum()
avg_rating = device_feedback_df['rating'].mean()
positive_feedback = sum(1 for s in sentiment_scores if s > 0)

# 创建仪表板
fig = make_subplots(
    rows=2, cols=3,
    specs=[
        [{"type": "indicator"}, {"type": "indicator"}, {"type": "indicator"}],
        [{"type": "bar", "colspan": 2}, None, {"type": "pie"}]
    ],
    subplot_titles=("", "", "", "房间能耗对比", "情感分布")
)

# 指标卡
fig.add_trace(
    go.Indicator(
        mode="number",
        value=total_energy,
        number={"suffix": " kWh"},
        title={"text": "总能耗"}
    ), row=1, col=1)

fig.add_trace(
    go.Indicator(
        mode="number",
        value=avg_rating,
        number={"suffix": "/10"},
        title={"text": "平均满意度"}
    ), row=1, col=2)

fig.add_trace(
    go.Indicator(
        mode="number",
        value=positive_feedback,
        number={"suffix": f"/{len(feedback_texts)}"},
        title={"text": "正面反馈"}
```

```
    ), row=1, col=3)

# 房间能耗柱状图
room_energy = energy_df.groupby('room')['total_energy'].sum().sort_values()
fig.add_trace(
    go.Bar(
        x=room_energy.values,
        y=room_energy.index,
        orientation='h',
        marker_color=px.colors.sequential.Blues_r
    ), row=2, col=1)

# 情感分布饼图
sentiment_labels = ['正面', '中性', '负面']
sentiment_counts = [
    sum(1 for s in sentiment_scores if s > 0.1),
    sum(1 for s in sentiment_scores if -0.1 <= s <= 0.1),
    sum(1 for s in sentiment_scores if s < -0.1)
]
fig.add_trace(
    go.Pie(
        labels=sentiment_labels,
        values=sentiment_counts,
        hole=0.3,
        marker_colors=['green', 'gray', 'red']
    ), row=2, col=3)

fig.update_layout(
    title_text="<b>智能家居系统综合仪表板</b>",
    height=600,
    showlegend=True,
    template="plotly_white"
)
fig.show()
```

In [ ]: