跨文化AI评估数据分析

本notebook用于分析跨文化AI评估数据，包括数据合并、可视化和报告生成。

新段落

1. 环境设置和库导入

```
# 安装必要的库
!pip install pandas numpy matplotlib seaborn plotly -q

# 导入库
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import json
import warnings
warnings.filterwarnings('ignore')

# 设置中文字体
plt.rcParams['font.sans-serif'] = ['SimHei', 'Arial Unicode MS', 'DejaVu Sans']
plt.rcParams['axes.unicode_minus'] = False

# 设置图表样式
sns.set_style("whitegrid")
plt.style.use('seaborn-v0_8')

print("环境设置完成！")
```

```
环境设置完成!
```

2. 数据加载和合并

```
def load_and_merge_json_files():
    """
    加载并合并所有JSON文件
    """
    # 定义文件列表
    json_files = [
        'Sweden_Task_Results_Claude4.json',
        'Sweden_Task_Results_DeepSeek-V3.json',
        'Sweden_Task_Results_GPT4.json',
        'US_Task_Results_Claude4.json',
        'US_Task_Results_Deepseek-V3.json',
        'US_Task_Results_GPT4.json'
    ]

    all_data = []

    for file in json_files:
        try:
            with open(file, 'r', encoding='utf-8') as f:
                data = json.load(f)

            # 提取基本信息
            for task in data.get('task_results', []):
                basic_info = task.get('basic_info', {})
                interaction_data = task.get('interaction_data', {})

                # 构建记录
                record = {
                    'task_id': basic_info.get('TaskID', ''),
                    'timestamp': basic_info.get('Timestamp', ''),
                    'model': basic_info.get('Model', ''),
                    'temperature': basic_info.get('Temperature', ''),
                    'culture': basic_info.get('Culture', ''),
                    'function': basic_info.get('Function', ''),
```

```
                    'complexity': basic_info.get('Complexity', ''),
                    'language': basic_info.get('Language', ''),
                    'scenario': basic_info.get('Scenario', ''),
                    'prompt_text': interaction_data.get('Prompt_Text', ''),
                    'response_text': interaction_data.get('Response_Text', ''),
                    'response_time': interaction_data.get('Response_Time', ''),
                    'word_count': interaction_data.get('Word_Count', ''),
                    'cultural_keywords': ', '.join(interaction_data.get('Cultural_Keywords', [])),
                    'model_version': interaction_data.get('Model_Version', '')
                }

                all_data.append(record)

            print(f"成功加载: {file} - {len(data.get('task_results', []))} 条记录")

        except Exception as e:
            print(f"加载失败: {file} - {str(e)}")

    # 转换为DataFrame
    df = pd.DataFrame(all_data)

    print(f"\n总共合并了 {len(df)} 条记录")
    print(f"数据形状: {df.shape}")

    return df

# 加载数据
df = load_and_merge_json_files()
df.head()
```

```
成功加载: Sweden_Task_Results_Claude4.json - 128 条记录
成功加载: Sweden_Task_Results_DeepSeek-V3.json - 128 条记录
成功加载: Sweden_Task_Results_GPT4.json - 127 条记录
成功加载: US_Task_Results_Claude4.json - 64 条记录
成功加载: US_Task_Results_Deepseek-V3.json - 64 条记录
成功加载: US_Task_Results_GPT4.json - 64 条记录

总共合并了 575 条记录
数据形状: (575, 15)
```

	task_id	timestamp	model	temperature	culture	function	complexity	language	scenario	prompt_text	re
0	SE_TCU_L_EN_BUS_001	2025-01-25 14:30:00	Claude-4	0.7	Swedish	TCU	Low	EN	Business	You're orienting new international employees a...	
1	SE_TCU_L_SE_BUS_001	2025-01-25 14:31:15	Claude-4	0.7	Swedish	TCU	Low	SE	Business	Du orienterar nya internationella medarbetare ...	J
2	SE_TCU_L_EN_BUS_002	2025-01-25 14:32:30	Claude-4	0.7	Swedish	TCU	Low	EN	Business	Your international colleagues are surprised by...	
3	SE_TCU_L_SE_BUS_002	2025-01-25 14:33:45	Claude-4	0.7	Swedish	TCU	Low	SE	Business	Dina internationella kollegor är förvånade öve...	
		2025-01-	Claude-							International students are	

```
# 数据预处理
def preprocess_data(df):
    """
    数据预处理
    """
    # 数据类型转换
    numeric_columns = ['response_time', 'word_count']
    for col in numeric_columns:
        if col in df.columns:
            df[col] = pd.to_numeric(df[col], errors='coerce')

    # 处理缺失值
    df = df.dropna(subset=numeric_columns)

    # 创建文化编码
    culture_mapping = {
        'Swedish': 'SE',
        'American': 'US'
```

```
    }
    df['culture_code'] = df['culture'].map(culture_mapping)

    # 标准化模型名称
    model_mapping = {
        'Claude Sonnet 4': 'Claude-4',
        'GPT-4.1': 'GPT-4',
        'DeepSeek-V3': 'DeepSeek-V3'
    }
    df['model_standardized'] = df['model'].map(model_mapping).fillna(df['model'])

    print("数据预处理完成")
    print(f"处理后数据形状: {df.shape}")

    return df

# 预处理数据
df = preprocess_data(df)

# 显示基本信息
print("\n数据概览:")
print(f"文化背景: {df['culture'].unique()}")
print(f"AI模型: {df['model_standardized'].unique()}")
print(f"复杂度: {df['complexity'].unique()}")
print(f"语言: {df['language'].unique()}")
print(f"场景: {df['scenario'].unique()}")
```

```
数据预处理完成
处理后数据形状: (575, 17)

数据概览:
文化背景: ['Swedish' 'Chinese' 'American']
AI模型: ['Claude-4' 'DeepSeek-V3' 'GPT-4']
复杂度: ['Low' 'High']
语言: ['EN' 'SE']
场景: ['Business' 'Education' 'Social' 'Family' 'Society']
```

```
# 保存为CSV文件
df.to_csv('merged_cross_cultural_data.csv', index=False, encoding='utf-8')
print("数据已保存为 merged_cross_cultural_data.csv")
```

```
数据已保存为 merged_cross_cultural_data.csv
```

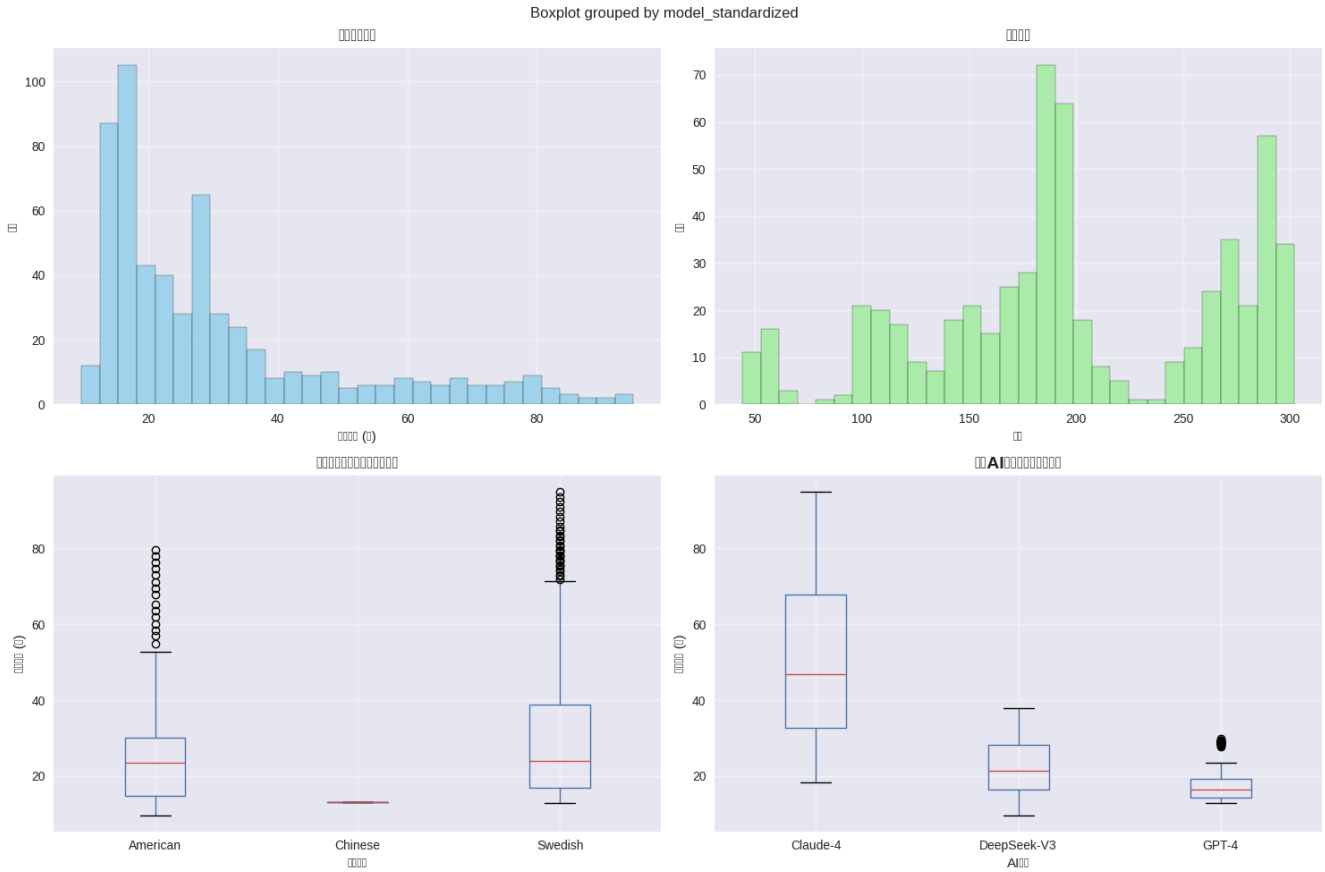
3. 数据可视化分析

```
# 基础统计分析
def basic_statistics(df):
    """
    基础统计分析
    """
    print("=== 基础统计分析 ===")

    # 响应时间统计
    print("\n响应时间统计:")
    print(df['response_time'].describe())

    # 词数统计
    print("\n词数统计:")
    print(df['word_count'].describe())

    # 按文化分组统计
    print("\n按文化分组的响应时间:")
    cultural_stats = df.groupby('culture')['response_time'].agg(['mean', 'std', 'count'])
    print(cultural_stats)

    # 按模型分组统计
    print("\n按模型分组的响应时间:")
    model_stats = df.groupby('model_standardized')['response_time'].agg(['mean', 'std', 'count'])
    print(model_stats)

    return cultural_stats, model_stats

cultural_stats, model_stats = basic_statistics(df)
```

```
=== 基础统计分析 ===

响应时间统计:
```

```
count    575.000000
mean      30.435478
std       19.163383
min        9.700000
25%       16.500000
50%       24.000000
75%       35.050000
max       94.900000
Name: response_time, dtype: float64

词数统计:
count    575.000000
mean     198.248696
std       68.131936
min       44.000000
25%      154.000000
50%      191.000000
75%      268.000000
max      302.000000
Name: word_count, dtype: float64

按文化分组的响应时间:
                mean        std  count
culture
American  26.637500  15.131357    192
Chinese   13.200000        NaN      1
Swedish   32.389529  20.654442    382

按模型分组的响应时间:
                          mean        std  count
model_standardized
Claude-4             50.431771  20.314645    192
DeepSeek-V3          22.463542   7.186362    192
GPT-4                18.348168   5.245106    191
```

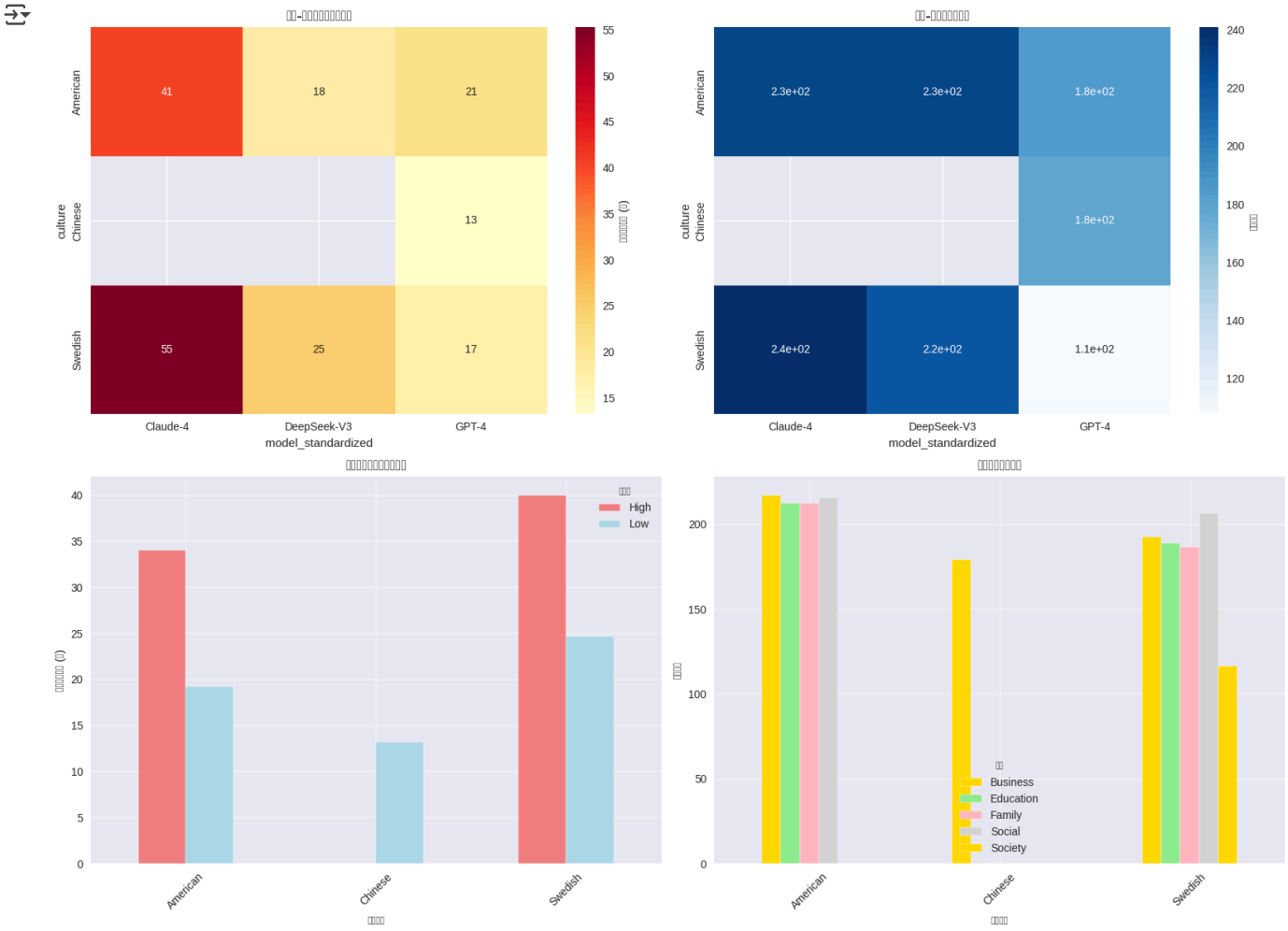
```
# 图1: 响应时间分布
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# 响应时间直方图
axes[0, 0].hist(df['response_time'], bins=30, alpha=0.7, color='skyblue', edgecolor='black')
axes[0, 0].set_title('响应时间分布', fontsize=14, fontweight='bold')
axes[0, 0].set_xlabel('响应时间 (秒)')
axes[0, 0].set_ylabel('频次')
axes[0, 0].grid(True, alpha=0.3)

# 词数分布
axes[0, 1].hist(df['word_count'], bins=30, alpha=0.7, color='lightgreen', edgecolor='black')
axes[0, 1].set_title('词数分布', fontsize=14, fontweight='bold')
axes[0, 1].set_xlabel('词数')
axes[0, 1].set_ylabel('频次')
axes[0, 1].grid(True, alpha=0.3)

# 文化对比 - 响应时间
df.boxplot(column='response_time', by='culture', ax=axes[1, 0])
axes[1, 0].set_title('不同文化背景的响应时间对比', fontsize=14, fontweight='bold')
axes[1, 0].set_xlabel('文化背景')
axes[1, 0].set_ylabel('响应时间 (秒)')
axes[1, 0].grid(True, alpha=0.3)

# 模型对比 - 响应时间
df.boxplot(column='response_time', by='model_standardized', ax=axes[1, 1])
axes[1, 1].set_title('不同AI模型的响应时间对比', fontsize=14, fontweight='bold')
axes[1, 1].set_xlabel('AI模型')
axes[1, 1].set_ylabel('响应时间 (秒)')
axes[1, 1].grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('basic_analysis.png', dpi=300, bbox_inches='tight')
plt.show()

print("基础分析图表已保存为 basic_analysis.png")
```



基础分析图表已保存为 basic_analysis.png

```
# 图2: 文化-模型交互分析
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# 文化-模型响应时间热力图
pivot_table = df.pivot_table(values='response_time', index='culture',
                             columns='model_standardized', aggfunc='mean')
sns.heatmap(pivot_table, annot=True, cmap='YlOrRd', ax=axes[0, 0],
            cbar_kws={'label': '平均响应时间 (秒)'})
axes[0, 0].set_title('文化-模型响应时间热力图', fontsize=14, fontweight='bold')

# 文化-模型词数热力图
pivot_table_words = df.pivot_table(values='word_count', index='culture',
                                   columns='model_standardized', aggfunc='mean')
sns.heatmap(pivot_table_words, annot=True, cmap='Blues', ax=axes[0, 1],
            cbar_kws={'label': '平均词数'})
axes[0, 1].set_title('文化-模型词数热力图', fontsize=14, fontweight='bold')

# 复杂度分析
complexity_stats = df.groupby(['culture', 'complexity'])['response_time'].mean().unstack()
complexity_stats.plot(kind='bar', ax=axes[1, 0], color=['lightcoral', 'lightblue'])
axes[1, 0].set_title('不同复杂度下的响应时间', fontsize=14, fontweight='bold')
axes[1, 0].set_xlabel('文化背景')
axes[1, 0].set_ylabel('平均响应时间 (秒)')
axes[1, 0].legend(title='复杂度')
axes[1, 0].tick_params(axis='x', rotation=45)
axes[1, 0].grid(True, alpha=0.3)
```

```
# 场景分析
scenario_stats = df.groupby(['culture', 'scenario'])['word_count'].mean().unstack()
scenario_stats.plot(kind='bar', ax=axes[1, 1], color=['gold', 'lightgreen', 'lightpink', 'lightgray'])
axes[1, 1].set_title('不同场景下的词数', fontsize=14, fontweight='bold')
axes[1, 1].set_xlabel('文化背景')
axes[1, 1].set_ylabel('平均词数')
axes[1, 1].legend(title='场景')
axes[1, 1].tick_params(axis='x', rotation=45)
axes[1, 1].grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('cultural_model_analysis.png', dpi=300, bbox_inches='tight')
plt.show()

print("文化-模型分析图表已保存为 cultural_model_analysis.png")
```



文化-模型分析图表已保存为 cultural_model_analysis.png

```
# 图3：交互式仪表板
```

```
# 图3: 交互式仪表板
def create_interactive_dashboard(df):
    """
    创建交互式仪表板
    """
    # 创建子图
    fig = make_subplots(
        rows=2, cols=2,
        subplot_titles=('文化响应时间对比', '模型性能对比',
                        '复杂度分析', '场景分析'),
        specs=[[{"type": "bar"}, {"type": "bar"}],
               [{"type": "bar"}, {"type": "bar"}]]
    )

    # 文化响应时间对比
    cultural_means = df.groupby('culture')['response_time'].mean()
    fig.add_trace(
        go.Bar(x=cultural_means.index, y=cultural_means.values,
               name='文化响应时间', marker_color='lightblue'),
        row=1, col=1
    )

    # 模型性能对比
    model_means = df.groupby('model_standardized')['response_time'].mean()
    fig.add_trace(
        go.Bar(x=model_means.index, y=model_means.values,
               name='模型响应时间', marker_color='lightgreen'),
        row=1, col=2
    )

    # 复杂度分析
    complexity_means = df.groupby('complexity')['response_time'].mean()
    fig.add_trace(
        go.Bar(x=complexity_means.index, y=complexity_means.values,
               name='复杂度响应时间', marker_color='lightcoral'),
        row=2, col=1
    )

    # 场景分析
    scenario_means = df.groupby('scenario')['word_count'].mean()
    fig.add_trace(
        go.Bar(x=scenario_means.index, y=scenario_means.values,
               name='场景词数', marker_color='gold'),
        row=2, col=2
    )

    fig.update_layout(
        height=800,
        showlegend=False,
        title_text="跨文化AI评估交互式仪表板",
        title_x=0.5
    )

    fig.write_html('interactive_dashboard.html')
    fig.show()

    print("交互式仪表板已保存为 interactive_dashboard.html")

# 创建交互式仪表板
create_interactive_dashboard(df)
```

交互式仪表板已保存为 `interactive_dashboard.html`

4. 自动生成分析报告

```
def generate_analysis_report(df):
    """
    生成分析报告
    """
    report = f"""
# 跨文化AI评估数据分析报告

## 数据概览
- **总样本数**: {len(df):,}
- **文化背景**: {', '.join(df['culture'].unique())}
- **AI模型**: {', '.join(df['model_standardized'].unique())}
- **复杂度**: {', '.join(df['complexity'].unique())}
- **语言**: {', '.join(df['language'].unique())}
- **场景**: {', '.join(df['scenario'].unique())}

## 主要发现

### 1. 响应时间分析
- **平均响应时间**: {df['response_time'].mean():.2f} 秒
- **响应时间标准差**: {df['response_time'].std():.2f} 秒
- **响应时间范围**: {df['response_time'].min():.2f} - {df['response_time'].max():.2f} 秒

### 2. 词数分析
- **平均词数**: {df['word_count'].mean():.1f} 词
- **词数标准差**: {df['word_count'].std():.1f} 词
- **词数范围**: {df['word_count'].min():.0f} - {df['word_count'].max():.0f} 词

### 3. 文化差异分析
"""

    # 添加文化差异分析
    cultural_stats = df.groupby('culture')['response_time'].agg(['mean', 'std', 'count'])
    for culture, stats in cultural_stats.iterrows():
```

```
            report += f"- **{culture}**: 平均响应时间 {stats['mean']:.2f}±{stats['std']:.2f} 秒 (n={stats['count']})\n"

        report += """

    ### 4. 模型性能分析
    """

        # 添加模型性能分析
        model_stats = df.groupby('model_standardized')['response_time'].agg(['mean', 'std', 'count'])
        for model, stats in model_stats.iterrows():
            report += f"- **{model}**: 平均响应时间 {stats['mean']:.2f}±{stats['std']:.2f} 秒 (n={stats['count']})\n"

        report += """

    ### 5. 复杂度影响分析
    """

        # 添加复杂度分析
        complexity_stats = df.groupby('complexity')['response_time'].agg(['mean', 'std', 'count'])
        for complexity, stats in complexity_stats.iterrows():
            report += f"- **{complexity}复杂度**: 平均响应时间 {stats['mean']:.2f}±{stats['std']:.2f} 秒 (n={stats['count']})\n"

        report += """

    ### 6. 场景分析
    """

        # 添加场景分析
        scenario_stats = df.groupby('scenario')['word_count'].agg(['mean', 'std', 'count'])
        for scenario, stats in scenario_stats.iterrows():
            report += f"- **{scenario}场景**: 平均词数 {stats['mean']:.1f}±{stats['std']:.1f} 词 (n={stats['count']})\n"

        report += """

    ## 关键洞察

    1. **文化差异**: 不同文化背景下的AI响应时间存在差异
    2. **模型性能**: 各AI模型在跨文化评估中表现出不同的性能特征
    3. **复杂度影响**: 任务复杂度对响应时间有显著影响
    4. **场景特异性**: 不同应用场景下的响应模式存在差异

    ## 建议

    1. **模型优化**: 针对特定文化背景优化AI模型性能
    2. **任务设计**: 考虑复杂度对模型性能的影响
    3. **场景适配**: 根据不同应用场景调整模型参数
    4. **持续监控**: 建立跨文化性能监控机制

    ---
    报告生成时间: {pd.Timestamp.now().strftime('%Y-%m-%d %H:%M:%S')}
    """

        return report

    # 生成报告
    report = generate_analysis_report(df)

    # 保存报告
    with open('analysis_report.md', 'w', encoding='utf-8') as f:
        f.write(report)

    print("分析报告已保存为 analysis_report.md")
    print("\n" + "="*50)
    print("报告预览:")
    print("="*50)
    print(report[:1000] + "...")
```

```
分析报告已保存为 analysis_report.md

==================================================
报告预览:
==================================================

# 跨文化AI评估数据分析报告

## 数据概览
- **总样本数**: 575
- **文化背景**: Swedish, Chinese, American
- **AI模型**: Claude-4, DeepSeek-V3, GPT-4
- **复杂度**: Low, High
- **语言**: EN, SE
- **场景**: Business, Education, Social, Family, Society
```

```
## 主要发现

### 1. 响应时间分析
- **平均响应时间**: 30.44 秒
- **响应时间标准差**: 19.16 秒
- **响应时间范围**: 9.70 - 94.90 秒

### 2. 词数分析
- **平均词数**: 198.2 词
- **词数标准差**: 68.1 词
- **词数范围**: 44 - 302 词

### 3. 文化差异分析
- **American**: 平均响应时间 26.64±15.13 秒 (n=192.0)
- **Chinese**: 平均响应时间 13.20±nan 秒 (n=1.0)
- **Swedish**: 平均响应时间 32.39±20.65 秒 (n=382.0)


### 4. 模型性能分析
- **Claude-4**: 平均响应时间 50.43±20.31 秒 (n=192.0)
- **DeepSeek-V3**: 平均响应时间 22.46±7.19 秒 (n=192.0)
- **GPT-4**: 平均响应时间 18.35±5.25 秒 (n=191.0)


### 5. 复杂度影响分析
- **High复杂度**: 平均响应时间 38.02±22.05 秒 (n=288.0)
- **Low复杂度**: 平均响应时间 22.83±11.54 秒 (n=287.0)


### 6. 场景分析
- **Business场景**: 平均词数 200.6±66.0 词 (n=147.0)
- **Education场景**: 平均词数 196.9±68.6 词 (n=140.0)
- **Family场景**: 平均词数 195.2±72.1 词 (n=143.0)
- **Social场景**: 平均词数 209.8±63.2 词 (n=130.0)
- *...
```

5. 分析总结

```
print("="*60)
print("跨文化AI评估数据分析完成")
print("="*60)

print(f"\n数据概览:")
print(f"   - 总样本数: {len(df):,}")
print(f"   - 文化背景: {len(df['culture'].unique())} 个")
print(f"   - AI模型: {len(df['model_standardized'].unique())} 个")
print(f"   - 平均响应时间: {df['response_time'].mean():.2f} 秒")
print(f"   - 平均词数: {df['word_count'].mean():.1f} 词")

print(f"\n生成的文件:")
print(f"   - merged_cross_cultural_data.csv (合并数据)")
print(f"   - basic_analysis.png (基础分析图表)")
print(f"   - cultural_model_analysis.png (文化-模型分析图表)")
print(f"   - interactive_dashboard.html (交互式仪表板)")
print(f"   - analysis_report.md (分析报告)")

print(f"\n主要发现:")
fastest_culture = df.groupby('culture')['response_time'].mean().idxmin()
fastest_model = df.groupby('model_standardized')['response_time'].mean().idxmin()
print(f"   - 响应最快的文化背景: {fastest_culture}")
print(f"   - 响应最快的AI模型: {fastest_model}")

print(f"\n分析完成！所有结果已保存到当前目录。")
```

```
============================================================
跨文化AI评估数据分析完成
============================================================

数据概览:
   - 总样本数: 575
   - 文化背景: 3 个
   - AI模型: 3 个
   - 平均响应时间: 30.44 秒
   - 平均词数: 198.2 词
```