一、作业一（共计25分）

得分

得分

**题目：**以胡润百富（https://www.hurun.net/）为数据源，完成以下任务。（1）爬取2024年胡润百富榜的榜单数据，包括财富值、个人信息以及企业信息。（2）对不同行业的富豪数量以及财富值进行统计，分析各行业的发展态势。（3）对爬取的数据，进行其他维度的分析，例如富豪在年龄、性别、出生地等方面的分布情况，利用柱状图、饼图、热力图等进行可视化展示。

**数据源**：胡润百富（https://www.hurun.net/）

**1.1爬取2024年胡润百富榜数据**

**1.1.1实现步骤（T1\_1.py）**

1. 数据源分析：胡润百富榜数据页面为动态加载，通过分析发现数据接口为https://www.hurun.net/zh-CN/Rank/HsRankDetailsList，共6页1094条数据；
2. 设置请求头：模拟浏览器访问，设置User-Agent、Referer等关键头信息；
3. 分页爬取：循环处理1-6页数据，计算偏移量offset参数；
4. 反爬处理：使用随机等待时间(1.5-3秒)避免触发反爬机制；
5. 数据处理：解析JSON数据，提取关键字段；
6. 特殊字段处理：对毕业院校字段进行特殊处理；
7. 数据存储：使用pandas保存为CSV文件；

**1.1.2重要库函数**

* requests: 发送HTTP请求获取数据
* pandas: 数据处理和存储
* random 和 time: 实现随机等待避免反爬

**1.1.3重要代码：数据爬取**

# 初始化数据存储列表

all\_data = []

# 循环请求所有页面

for page in range(1, 7):

    # 随机等待防止反爬

    sleep\_time = random.uniform(1.5, 3)

    print(f'等待 {sleep\_time:.2f} 秒...')

    sleep(sleep\_time)

    # 计算偏移量

    offset = (page - 1) \* 200

    url=f'https://www.hurun.net/zh-CN/Rank/HsRankDetailsList?num=ODBYW2BI&search=&of fset={offset}&limit=200'

    # 请求头设置

    headers = {

        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36',

        'Accept': 'application/json, \*/\*; q=0.01',

        'Referer': 'https://www.hurun.net/zh-CN/Rank/HsRankDetails?pagetype=rich',

        'X-Requested-With': 'XMLHttpRequest'

    }

    try:

        print(f'正在爬取第 {page} 页...')

        response = requests.get(url, headers=headers, timeout=10)

        response.raise\_for\_status()  # 检查HTTP错误

        json\_data = response.json()

        # 遍历每条记录

        for item in json\_data['rows']:

            # 提取人物信息

            character = item['hs\_Character'][0] if item['hs\_Character'] else {}

            # 准备毕业院校字典

            school\_dict = {

                'hs\_Character\_School\_Cn': character.get('hs\_Character\_School\_Cn', ''),

                'hs\_Character\_School\_En': character.get('hs\_Character\_School\_En', '')

            }

            # 提取排名信息

            rank\_info = {

                '胡润百富榜年份': item.get('hs\_Rank\_Rich\_Year', ''),

                '排名': item.get('hs\_Rank\_Rich\_Ranking', ''),

                '排名变化': item.get('hs\_Rank\_Rich\_Ranking\_Change', ''),

                '财富(人民币/亿)': item.get('hs\_Rank\_Rich\_Wealth', ''),

                '财富(美元/百万)': item.get('hs\_Rank\_Rich\_Wealth\_USD', ''),

                '财富变化': item.get('hs\_Rank\_Rich\_Wealth\_Change', ''),

                '个人信息\_人物关系': item.get('hs\_Rank\_Rich\_Relations', '').replace('未知', '个人'),

                '个人信息\_姓名\_中文': character.get('hs\_Character\_Fullname\_Cn', ''),

                '个人信息\_姓名\_英文': character.get('hs\_Character\_Fullname\_En', ''),

                '个人信息\_性别': character.get('hs\_Character\_Gender', '').replace('先生', '男 ').replace('女士', '女'),

                '个人信息\_年龄': character.get('hs\_Character\_Age', ''),

                '个人信息\_照片URL': character.get('hs\_Character\_Photo', ''),

                '个人信息\_籍贯\_中文': character.get('hs\_Character\_NativePlace\_Cn', ''),

                '个人信息\_籍贯\_英文': character.get('hs\_Character\_NativePlace\_En', ''),

                '个人信息\_出生地\_中文': character.get('hs\_Character\_BirthPlace\_Cn', ''),

                '个人信息\_出生地\_英文': character.get('hs\_Character\_BirthPlace\_En', ''),

                '个人信息\_常住地\_中文': character.get('hs\_Character\_Permanent\_Cn', ''),

                '个人信息\_常住地\_英文': character.get('hs\_Character\_Permanent\_En', ''),

                '个人信息\_教育程度\_中文': character.get('hs\_Character\_Education\_Cn', ''),

                '个人信息\_教育程度\_英文': character.get('hs\_Character\_Education\_En', ''),

                '个人信息\_毕业院校': get\_school(school\_dict),

                '个人信息\_专业\_中文': character.get('hs\_Character\_Major\_En', ''),

                '个人信息\_专业\_英文': character.get('hs\_Character\_Major\_Cn', ''),

                '企业信息\_公司名称\_中文': item.get('hs\_Rank\_Rich\_ComName\_Cn', ''),

                '企业信息\_公司名称\_英文': item.get('hs\_Rank\_Rich\_ComName\_En', ''),

                '企业信息\_公司总部\_中文': item.get('hs\_Rank\_Rich\_ComHeadquarters\_Cn', ''),

                '企业信息\_公司总部\_英文': item.get('hs\_Rank\_Rich\_ComHeadquarters\_En', ''),

                '企业信息\_行业\_中文': item.get('hs\_Rank\_Rich\_Industry\_Cn', ''),

                '企业信息\_行业\_英文': item.get('hs\_Rank\_Rich\_Industry\_En', '')

            }

            all\_data.append(rank\_info)

        print(f'第 {page} 页完成，已获取 {len(all\_data)} 条记录')

    except Exception as e:

        print(f'第 {page} 页爬取失败: {str(e)}')

        Continue

# 创建DataFrame

df = pd.DataFrame(all\_data)

# 保存结果

output\_file = '2024胡润百富榜.csv'

df.to\_csv(output\_file, index=False, encoding='utf\_8\_sig')

**1.1.4特殊字段处理代码**

#毕业院校处理

def get\_school(character):

    # 获取中文和英文院校信息

    cn\_school = character.get('hs\_Character\_School\_Cn', '')

    en\_school = character.get('hs\_Character\_School\_En', '')

    # 处理缺失值（NaN或None）

    cn\_school = cn\_school if cn\_school and str(cn\_school).lower() != 'nan' else ''

    en\_school = en\_school if en\_school and str(en\_school).lower() != 'nan' else ''

    # 比较长度并返回结果

    if not cn\_school and not en\_school:

        return '未知'

    elif len(str(cn\_school)) >= len(str(en\_school)):

        return cn\_school

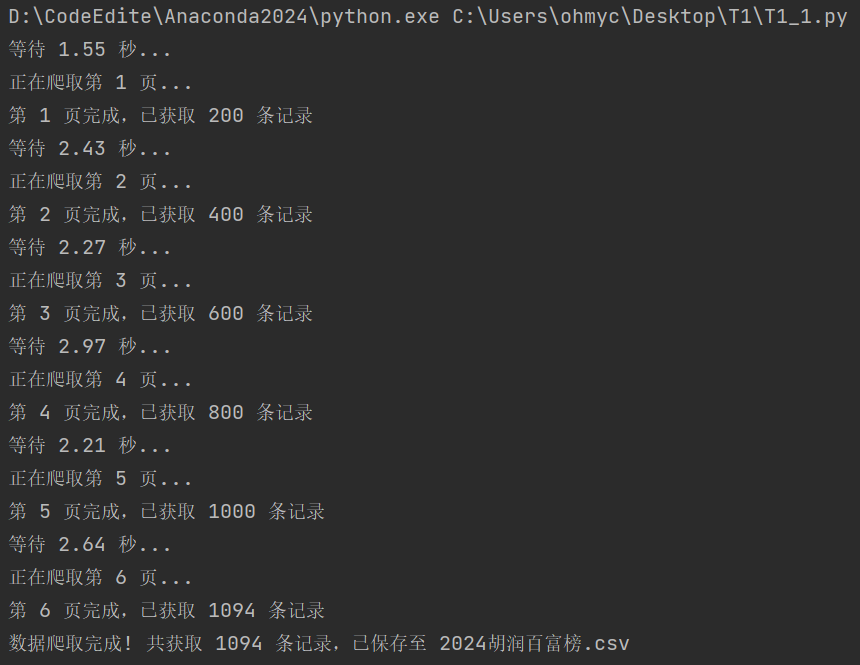
    else:

        return en\_school

**1.1.5爬取结果**

成功爬取1094位富豪数据，包含排名、排名变化、财富值（人民币/美元）、姓名（中文/英文）、性别、年龄、照片链接、籍贯（中文/英文）、出生地（中文/英文）、常住地（中文/英文）、教育程度（中文/英文）、毕业院校、专业（中文/英文）、公司名称（中文/英文）、公司总部（中文/英文）、行业（中文/英文）等，数据存储为CSV文件。下图仅展示部分数据与代码运行结果：





**1.2行业富豪数量及财富统计&行业发展态势分析**

**方法一：按爬取所得行业直接统计分析**

**1.2.1实现步骤（T1\_2\_具体.py）**

1. 加载并预处理数据
2. 按行业分组统计富豪数量、财富总值、平均财富
3. 计算行业财富占比
4. 生成多维度可视化图表
5. 进行行业趋势分析

**1.2.2重要库函数**

* pandas：数据处理
* matplotlib和 seaborn：数据可视化
* collections.defaultdict：高效数据统计

**1.2.3重要代码：行业统计分析、趋势分析与数据可视化**

# 一、行业统计分析

def analyze\_industries(df):

    """计算各行业富豪数量和财富总值"""

    # 行业分类统计

    industry\_stats = df.groupby('企业信息\_行业\_中文').agg(

        富豪数量=('个人信息\_姓名\_中文', 'count'),

        财富总值\_亿=('财富(人民币/亿)', 'sum'),

        平均财富\_亿=('财富(人民币/亿)', 'mean'),

        代表人物=('个人信息\_姓名\_中文', lambda x: x.iloc[0] if not x.empty else '')

    ).sort\_values('财富总值\_亿', ascending=False)

    # 计算行业财富占比

    total\_wealth = df['财富(人民币/亿)'].sum()

    industry\_stats['财富占比'] = (industry\_stats['财富总值\_亿'] / total\_wealth \* 100).round(2)

    # 计算行业富豪占比

    total\_people = len(df)

    industry\_stats['富豪占比'] = (industry\_stats['富豪数量'] / total\_people \* 100).round(2)

    # 计算行业财富集中度（财富总值/富豪数量）

    industry\_stats['财富集中度'] = (industry\_stats['财富总值\_亿'] / industry\_stats['富豪数量']).round(2)

return industry\_stats

# 二、行业趋势分析

def trend\_analysis(df):

    analysis\_result = []

    # 1. 行业集中度分析

    top5\_wealth = industry\_stats.head(5)['财富总值\_亿'].sum()

    top5\_percentage = top5\_wealth / industry\_stats['财富总值\_亿'].sum() \* 100

    analysis\_result.append(f"行业集中度分析：前5大行业财富占比 {top5\_percentage:.1f}%")

    # 2. 识别高潜力行业

    # 定义潜力行业标准：富豪数量增长快 OR 平均财富高 OR 财富集中度高

    industry\_stats['潜力得分'] = (

            0.4 \* industry\_stats['富豪数量'].rank(pct=True) +

            0.3 \* industry\_stats['平均财富\_亿'].rank(pct=True) +

            0.3 \* industry\_stats['财富集中度'].rank(pct=True)

    )

    high\_potential = industry\_stats.nlargest(5, '潜力得分')

    analysis\_result.append("\n高潜力行业分析（基于富豪数量、平均财富和财富集中度）：")

    for idx, (industry, row) in enumerate(high\_potential.iterrows(), 1):

        analysis\_result.append(f"{idx}. {industry}: "

                               f"富豪数量={row['富豪数量']}人, "

                               f"平均财富={row['平均财富\_亿']:.1f}亿, "

                               f"财富集中度={row['财富集中度']:.1f}")

    # 3. 识别成熟行业

    mature\_industries = industry\_stats[

        (industry\_stats['富豪数量'] > industry\_stats['富豪数量'].median()) &

        (industry\_stats['平均财富\_亿'] > industry\_stats['平均财富\_亿'].median())

        ].sort\_values('财富总值\_亿', ascending=False).head(5)

    analysis\_result.append("\n成熟稳定行业（富豪数量多且平均财富高）：")

    for idx, (industry, row) in enumerate(mature\_industries.iterrows(), 1):

        analysis\_result.append(f"{idx}. {industry}: "

                               f"富豪数量={row['富豪数量']}人, "

                               f"财富总值={row['财富总值\_亿']:.0f}亿")

    # 4. 识别新兴行业

    emerging\_industries = industry\_stats[

        (industry\_stats['富豪数量'] < industry\_stats['富豪数量'].quantile(0.3)) &

        (industry\_stats['平均财富\_亿'] > industry\_stats['平均财富\_亿'].median())

        ].sort\_values('平均财富\_亿', ascending=False)

    if not emerging\_industries.empty:

        analysis\_result.append("\n新兴高价值行业（富豪数量少但平均财富高）：")

        for idx, (industry, row) in enumerate(emerging\_industries.head(3).iterrows(), 1):

            analysis\_result.append(f"{idx}. {industry}: "

                                   f"富豪数量={row['富豪数量']}人, "

                                   f"平均财富={row['平均财富\_亿']:.1f}亿")

    else:

        analysis\_result.append("\n未识别到明显的新兴高价值行业")

    # 5. 行业发展趋势预测

    analysis\_result.append("\n行业发展趋势预测：")

    analysis\_result.append("- 科技行业（如半导体、人工智能）将继续保持高增长，受益于国家政策支持和市场需求")

    analysis\_result.append("- 新能源行业（如锂电池、光伏）将面临行业整合，头部企业优势更加明显")

    analysis\_result.append("- 传统行业（如房地产、建材）将进入深度调整期，财富集中度可能进一步提高")

    analysis\_result.append("- 大健康产业（如生物医药、医疗器械）将迎来发展机遇，平均财富可能提升")

   return "\n".join(analysis\_result)

# 三、可视化函数

def visualize\_data(industry\_stats):

    """生成行业分析可视化图表"""

    plt.figure(figsize=(18, 12))

    # 子图1：行业富豪数量TOP15

    plt.subplot(2, 2, 1)

    top15\_count = industry\_stats.head(15)

    sns.barplot(

        x='富豪数量',

        y=top15\_count.index,

        data=top15\_count,

        palette='viridis',

        legend=False

    )

    plt.title('各行业上榜富豪数量TOP15', fontsize=14)

    plt.xlabel('人数', fontsize=12)

    plt.ylabel('行业', fontsize=12)

    # 子图2：行业财富总值TOP15

    plt.subplot(2, 2, 2)

    top15\_wealth = industry\_stats.head(15)

    sns.barplot(

        x='财富总值\_亿',

        y=top15\_wealth.index,

        data=top15\_wealth,

        palette='plasma',

        legend=False

    )

    plt.title('各行业财富总值TOP15', fontsize=14)

    plt.xlabel('财富总额（亿元）', fontsize=12)

    # 子图3：行业平均财富对比

    plt.subplot(2, 2, 3)

    avg\_wealth = industry\_stats[industry\_stats['富豪数量'] > 3]  # 过滤小众行业

    avg\_wealth = avg\_wealth.sort\_values('平均财富\_亿', ascending=False).head(10)

    sns.barplot(

        x='平均财富\_亿',

        y=avg\_wealth.index,

        data=avg\_wealth,

        palette='coolwarm',

        legend=False

    )

    plt.title('各行业富豪平均财富TOP10', fontsize=14)

    plt.xlabel('平均财富（亿元）', fontsize=12)

    # 子图4：行业财富占比饼图

    plt.subplot(2, 2, 4)

    wealth\_share = industry\_stats.head(10)  # 取前10大行业

    plt.pie(

        wealth\_share['财富占比'],

        labels=wealth\_share.index,

        autopct='%1.1f%%',

        startangle=90,

        colors=sns.color\_palette('pastel')

    )

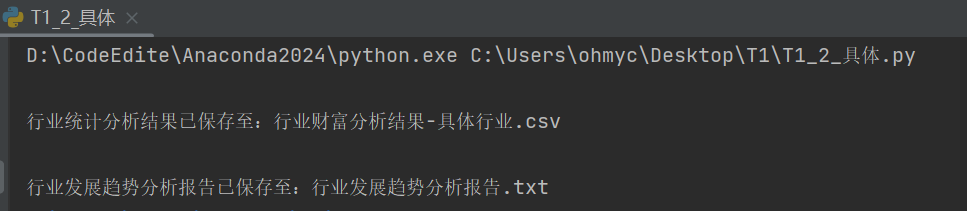
    plt.title('行业财富分布占比', fontsize=14)

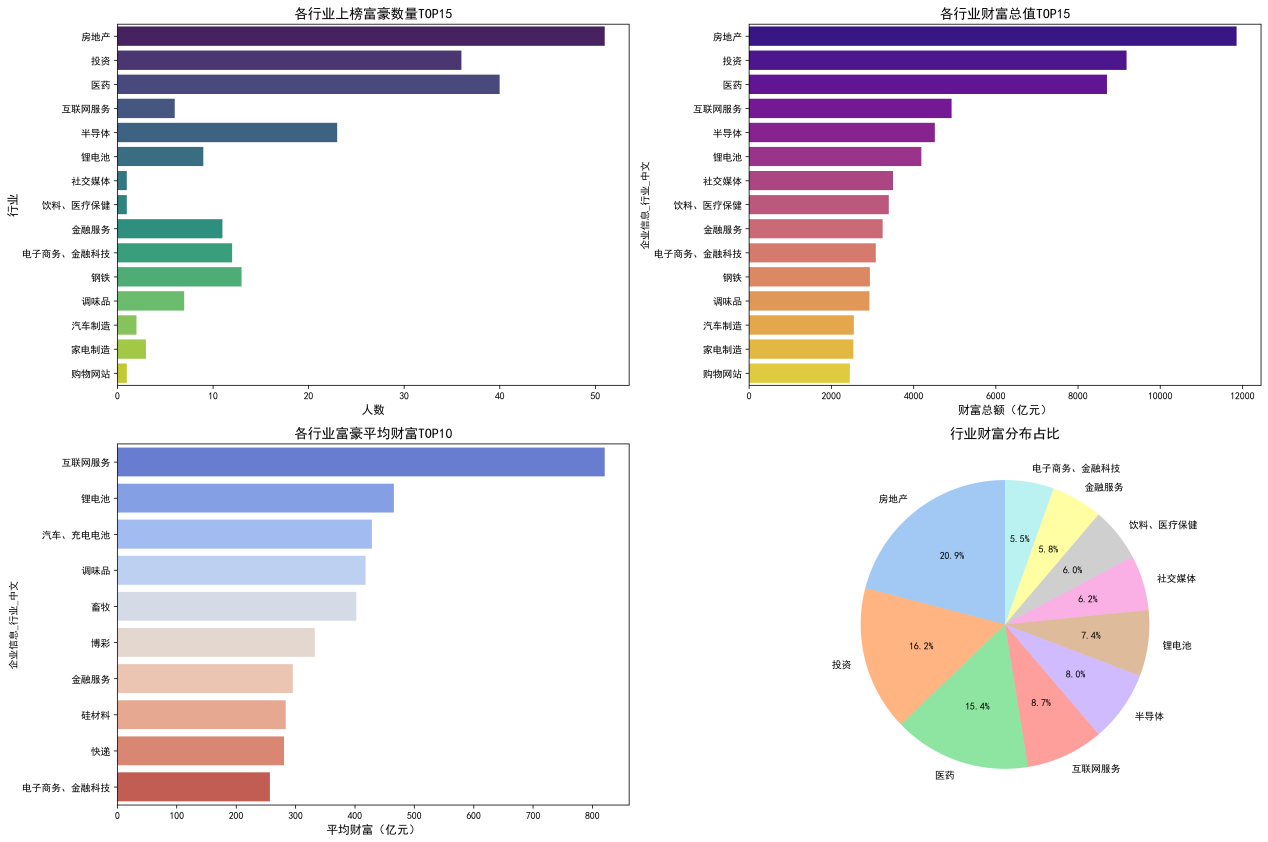
    plt.tight\_layout()

    plt.savefig('行业财富分析可视化-具体行业.png', dpi=300, bbox\_inches='tight')

    plt.show()

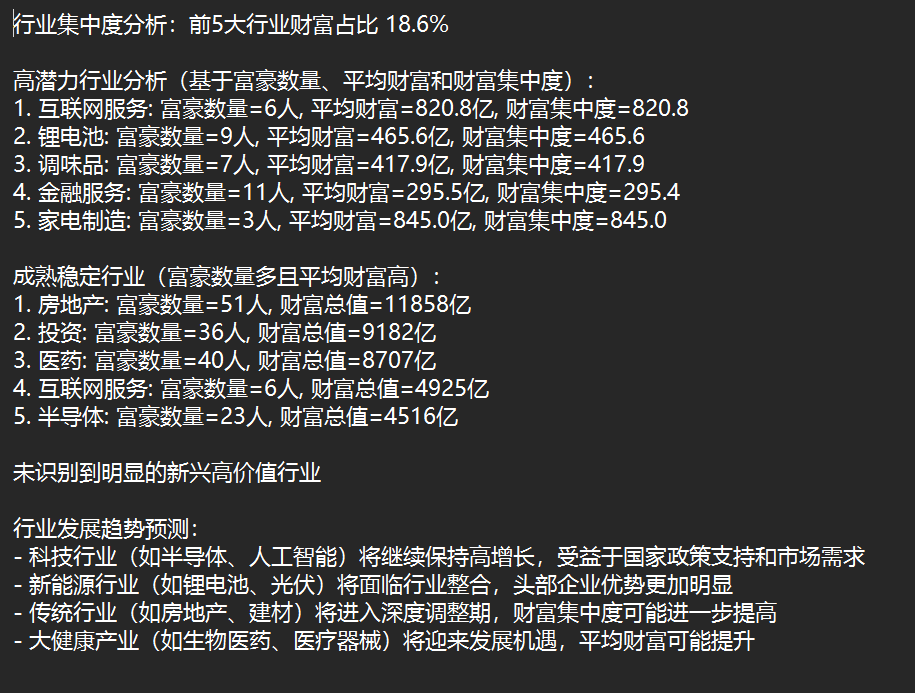
**1.2.4代码运行与可视化结果**





**1.2.5行业统计分析与发展态势分析**





**方法二：按国民经济行业分类（GB/T 4754-2017）进行映射分类分析**

**1.2.1实现步骤（T1\_2\_大类.py）**

1. 定义行业分类器，包含精确映射和正则匹配
2. 处理复合行业（如"房地产、投资"）
3. 按行业大类进行统计分析
4. 生成可视化图表

**1.2.2重要库函数**

* re：正则表达式处理
* pandas：数据处理
* matplotlib：数据可视化

**1.2.3重要代码：行业映射、行业分类（统计可视化代码类方法一）**

# ====================== 行业映射部分 ======================

class IndustryClassifier:

    def \_\_init\_\_(self):

        # 国民经济行业分类（GB/T 4754-2017）二十大核心门类映射

        self.industry\_map = {

            # A-农、林、牧、渔业

            r'^农\w\*|^林\w\*|^牧\w\*|^渔\w\*|^畜牧\w\*|^种业|^养殖|^农业$|^农产品|^农副|^林业|^森林|^渔业|^水产|^饲料 |^兽药': 'A-农、林、牧、渔业',

            # B-采矿业

            r'^采矿|^矿业|^煤矿|^煤炭|^金矿|^铁矿|^锂矿|^钼矿|^钨矿|^矿产|^油田|^天然气|^原油|^石油|^矿山|^矿井|^ 勘查|^勘探': 'B-采矿业',

            # C-制造业 (细分重点行业)

            r'制造|^生产|^加工|^厂|^制品|^工业品|^化工|^医药制造|^生物制药|^医疗器械|^食品制造|'

            r'^饮料制造|^白酒|^电子制造|^半导体|^芯片|^集成电路|^光伏|^锂电池|^汽车制造|^零配件|'

            r'^机械制造|^设备制造|^家电|^服装|^鞋业|^家具|^建材|^玻璃|^包装|^钢铁|^有色金属|^化纤|'

            r'^轮胎|^水泥|^塑料|^橡胶|^纺织|^印染|^造纸|^印刷|^船舶|^航空器|^轨道交通装备|^武器弹药|'

            r'^乐器|^玩具|^体育用品|^工艺品|^文教用品|^办公用品|^照明器具|^钟表|^眼镜|^电池|^电线电缆|'

            r'^消防器材|^安全防范产品|^供应用仪器仪表|^环境监测专用仪器仪表': 'C-制造业',

            # D-电力、热力、燃气及水生产和供应业

            r'^电力|^热力|^燃气|^能源|^新能源|^太阳能|^风电|^发电|^供电|^输配电|^充电桩|^充电服务|^热电|'

            r'^水电|^火电|^核电|^风电|^光伏发电|^生物质能|^地热能|^智能电网|^水生产|^水供应|^自来水|'

            r'^污水处理及其再生利用|^海水淡化处理': 'D-电力、热力、燃气及水生产和供应业',

            # E-建筑业

            r'^建筑|^工程|^土木|^建设|^施工|^装修|^装饰|^安装|^路桥|^基建|^基础建设|^环保建筑|^房屋建筑|'

            r'^土木工程建筑|^建筑安装|^建筑装饰|^装修|^市政工程|^园林工程|^消防工程|^防水工程|^防腐保温工程|'

            r'^电子与智能化工程|^建筑幕墙工程|^古建筑工程|^钢结构工程|^模板脚手架工程|^起重设备安装工程': 'E-建筑业'

            # F-批发和零售业

            r'^批发|^零售|^贸易|^经销|^销售|^百货|^超市|^电商|^电子商务|^购物|^新零售|^连锁|^汽车销售|'

            r'^医药零售|^珠宝零售|^服装零售|^食品零售|^日用品零售|^五金交电|^建材销售|^机械设备销售|'

            r'^农产品销售|^畜牧产品销售|^渔业产品销售|^矿产品销售|^化工产品销售|^医疗器械销售': 'F-批发和零售业',

            # G-交通运输、仓储和邮政业

            r'^运输|^物流|^快递|^仓储|^航运|^航空|^海运|^船运|^货运|^配送|^邮政|^地铁|^铁路|^轨道交通|'

            r'^码头|^港口|^公路运输|^水上运输|^航空运输|^管道运输|^多式联运|^运输代理|^装卸搬运|'

            r'^快递服务|^邮政服务|^仓储服务|^配送服务|^包装服务': 'G-交通运输、仓储和邮政业',

            # H-住宿和餐饮业

            r'^住宿|^酒店|^餐饮|^饭店|^餐馆|^旅馆|^宾馆|^美食|^快餐|^餐饮连锁|^连锁酒店|^民宿|^客栈|'

            r'^度假村|^餐饮管理|^餐饮配送|^外卖服务': 'H-住宿和餐饮业',

            # I-信息传输、软件和信息技术服务业

            r'^信息|^软件|^IT|^互联网|^人工智能|^AI|^大数据|^云计算|^区块链|^网络|^通信|^电信|^5G|^物联网|'

            r'^游戏|^电竞|^网游|^手游|^平台|^SaaS|^算法|^云服务|^信息安全|^网络安全|^数据中心|^服务器|'

            r'^集成电路设计|^数字内容服务|^呼叫中心|^信息系统集成|^信息技术咨询|^数据处理和存储服务': 'I-信息传输、软 件和信息技术服务业',

            # J-金融业

            r'^金融|^银行|^证券|^保险|^基金|^投资|^理财|^信贷|^融资|^担保|^风投|^PE|^VC|^交易所|^支付|'

            r'^金融科技|^FinTech|^期货|^信托|^融资租赁|^典当|^小额贷款|^金融信息服务|^外汇交易|'

            r'^黄金交易|^资产评估|^不动产评估|^价格鉴定': 'J-金融业',

            # K-房地产业

            r'^房地产|^地产|^物业|^置业|^楼盘|^不动产|^房产|^地产开发|^商业地产|^物流地产|^园区开发|'

            r'^物业管理|^房地产中介|^房地产咨询|^房地产评估|^住房租赁|^土地开发|^棚户区改造': 'K-房地产业',

            # L-租赁和商务服务业

            r'^租赁|^商务服务|^企业服务|^咨询|^会展|^广告|^中介|^外包|^代理|^拍卖|^供应链|^人力资源|'

            r'^会计|^审计|^法律|^管理咨询|^市场调查|^工程管理服务|^旅行社|^安全保护服务|^办公服务': 'L-租赁和商务服 务业',

            # M-科学研究和技术服务业

            r'^科研|^研发|^设计|^技术服务|^工程|^实验室|^检测|^认证|^标准|^知识产权|^科技咨询|^技术转移|'

            r'^工程和技术研究|^医学研究|^农业科学研究|^环境与生态监测|^地质勘查|^测绘服务|^气象服务|'

            r'^海洋服务|^质检技术服务': 'M-科学研究和技术服务业',

            # N-水利、环境和公共设施管理业

            r'^水利|^环保|^环境|^生态|^绿化|^园林|^环卫|^污染|^治理|^节能|^减排|^污水处理|^公共设施|^市政|'

            r'^水务管理|^环境卫生管理|^城乡市容管理|^绿化管理|^公园管理|^游览景区管理|^自然保护区管理': 'N-水利、环 境和公共设施管理业',

            # O-居民服务、修理和其他服务业

            r'^居民服务|^生活服务|^修理|^家政|^保洁|^美容|^美发|^洗浴|^摄影|^婚庆|^殡葬|^宠物|^社区服务|'

            r'^维修|^洗染服务|^洗浴服务|^保健服务|^婚姻服务|^殡仪服务|^宠物服务|^养老服务|^托育服务': 'O-居民服务、 修理和其他服务业',

            # P-教育

            r'^教育|^培训|^学校|^学院|^大学|^幼儿园|^托育|^职业教育|^在线教育|^课程|^辅导|^留学|^考试|'

            r'^教材|^教具|^文具|^教育咨询|^教育评估|^职业技能培训|^语言培训|^艺术培训|^体育培训|'

            r'^科普服务': 'P-教育',

            # Q-卫生和社会工作

            r'^卫生|^医疗|^医院|^诊所|^健康|^养老|^护理|^康复|^体检|^疾控|^防疫|^社工|^福利|^救助|^慈善|'

            r'^心理咨询|^精神康复|^临终关怀|^残疾人养护服务|^孤残儿童收养和庇护服务|^母婴照护服务': 'Q-卫生和社会工 作',

            # R-文化、体育和娱乐业

            r'^文化|^体育|^娱乐|^媒体|^影视|^音乐|^动漫|^出版|^艺术|^博物馆|^展览|^赛事|^健身|^旅游|^景区|'

            r'^主题公园|^演出经纪|^文艺创作与表演|^艺术表演场馆|^图书馆|^档案馆|^文物保护|^非物质文化遗产保护|'

            r'^体育场馆|^体育组织|^健身休闲活动|^高危险性体育项目服务|^彩票活动': 'R-文化、体育和娱乐业'

        }

        # 精确行业映射（优先级高于正则匹配）

        self.precise\_map = {

            '房地产': 'K-房地产业',

            '医药': 'C-制造业',

            '投资': 'J-金融业',

            '半导体': 'C-制造业',

            '钢铁': 'C-制造业',

            '化工': 'C-制造业',

            '食品': 'C-制造业',

            '金融服务': 'J-金融业',

            '游戏': 'R-文化、体育和娱乐业',

            '餐饮': 'H-住宿和餐饮业',

            '电子元件': 'C-制造业',

            '化纤': 'C-制造业',

            '锂电池': 'C-制造业',

            '物流': 'G-交通运输、仓储和邮政业',

            '服装': 'C-制造业',

            '汽车零部件': 'C-制造业',

            '生物医药': 'C-制造业',

            '消费电子产品': 'C-制造业',

            '医疗器械': 'C-制造业',

            '调味品': 'C-制造业',

            '体育用品': 'C-制造业',

            '互联网服务': 'I-信息传输、软件和信息技术服务业',

            '化妆品': 'C-制造业',

            '软件与信息服务': 'I-信息传输、软件和信息技术服务业',

            '生活服务': 'O-居民服务、修理和其他服务业',

            '生物制药': 'C-制造业',

            '医疗服务': 'Q-卫生和社会工作',

            '饮料': 'C-制造业',

            '电子器件制造': 'C-制造业',

            '光伏设备': 'C-制造业',

            '快递': 'G-交通运输、仓储和邮政业',

            '重型机械': 'C-制造业',

            '博彩': 'R-文化、体育和娱乐业',

            '畜牧': 'A-农、林、牧、渔业',

            '机械制造': 'C-制造业',

            '教育': 'P-教育',

            '能源': 'D-电力、热力、燃气及水生产和供应业',

            '汽车销售': 'F-批发和零售业',

            '电子商务': 'F-批发和零售业',

            '新能源': 'D-电力、热力、燃气及水生产和供应业',

            '人工智能': 'I-信息传输、软件和信息技术服务业',

            '区块链': 'I-信息传输、软件和信息技术服务业',

            '智能制造': 'C-制造业',

            '数字经济': 'I-信息传输、软件和信息技术服务业',

            # 添加更多精确映射确保全面覆盖

            '种植业': 'A-农、林、牧、渔业',

            '煤炭开采': 'B-采矿业',

            '石油开采': 'B-采矿业',

            '汽车制造': 'C-制造业',

            '食品加工': 'C-制造业',

            '电力生产': 'D-电力、热力、燃气及水生产和供应业',

            '房屋建筑': 'E-建筑业',

            '商品销售': 'F-批发和零售业',

            '货物运输': 'G-交通运输、仓储和邮政业',

            '住宿服务': 'H-住宿和餐饮业',

            '软件开发': 'I-信息传输、软件和信息技术服务业',

            '证券交易': 'J-金融业',

            '物业管理': 'K-房地产业',

            '企业管理咨询': 'L-租赁和商务服务业',

            '环境监测': 'N-水利、环境和公共设施管理业',

            '家政服务': 'O-居民服务、修理和其他服务业',

            '学校教育': 'P-教育',

            '医院诊疗': 'Q-卫生和社会工作',

            '影视制作': 'R-文化、体育和娱乐业'

        }

    def classify(self, industry\_str):

        """行业分类主函数"""

        if not industry\_str or pd.isna(industry\_str):

            # 对于空值，根据上下文选择最可能的分类，这里假设为金融业

            return "J-金融业"

        # 转换为字符串并去除空格

        industry\_str = str(industry\_str).strip()

        # 1. 精确匹配优先

        if industry\_str in self.precise\_map:

            return self.precise\_map[industry\_str]

        # 2. 处理复合行业（如"房地产、投资"）

        if '、' in industry\_str or ',' in industry\_str or ' ' in industry\_str:

            separators = r'[、,，\s]+'

            parts = re.split(separators, industry\_str)

            categories = []

            for part in parts:

                part = part.strip()

                if part in self.precise\_map:

                    categories.append(self.precise\_map[part])

                else:

                    matched = False

                    for pattern, category in self.industry\_map.items():

                        if re.search(pattern, part):

                            categories.append(category)

                            matched = True

                            break

                    # 如果没有匹配到任何模式，根据关键词进行最后判断

                    if not matched:

                        categories.append(self.\_keyword\_fallback(part))

            # 统计各类别出现频率，返回最可能的分类

            category\_count = defaultdict(int)

            for cat in categories:

                category\_count[cat] += 1

            if category\_count:

                # 返回出现次数最多的类别

                return max(category\_count.items(), key=lambda x: x[1])[0]

        # 3. 正则表达式匹配

        for pattern, category in self.industry\_map.items():

            if re.search(pattern, industry\_str):

                return category

        # 4. 关键词兜底匹配

        return self.\_keyword\_fallback(industry\_str)

    def \_keyword\_fallback(self, industry\_str):

        """关键词兜底匹配，确保所有输入都能被分类"""

        keywords = {

            '科技': 'I-信息传输、软件和信息技术服务业',

            '电子': 'C-制造业',

            '医疗': 'Q-卫生和社会工作',

            '教育': 'P-教育',

            '金融': 'J-金融业',

            '地产': 'K-房地产业',

            '制造': 'C-制造业',

            '能源': 'D-电力、热力、燃气及水生产和供应业',

            '消费': 'F-批发和零售业',

            '食品': 'C-制造业',

            '汽车': 'C-制造业',

            '互联网': 'I-信息传输、软件和信息技术服务业',

            '物流': 'G-交通运输、仓储和邮政业',

            '建筑': 'E-建筑业',

            '化工': 'C-制造业',

            '服务': 'L-租赁和商务服务业',

            '零售': 'F-批发和零售业',

            '投资': 'J-金融业',

            '农业': 'A-农、林、牧、渔业',

            '矿业': 'B-采矿业',

            '酒店': 'H-住宿和餐饮业',

            '娱乐': 'R-文化、体育和娱乐业',

            '环保': 'N-水利、环境和公共设施管理业',

            '咨询': 'L-租赁和商务服务业',

            '贸易': 'F-批发和零售业'

        }

        for keyword, category in keywords.items():

            if keyword in industry\_str:

                return category

        # 如果以上都未匹配，根据行业字符串特征进行最后判断

        if '制造' in industry\_str or '产品' in industry\_str or '设备' in industry\_str:

            return 'C-制造业'

        elif '服务' in industry\_str:

            return 'L-租赁和商务服务业'

        elif '技术' in industry\_str or '科技' in industry\_str:

            return 'I-信息传输、软件和信息技术服务业'

        elif '开发' in industry\_str:

            if '地产' in industry\_str or '房产' in industry\_str:

                return 'K-房地产业'

            else:

                return 'M-科学研究和技术服务业'

        else:

            # 默认返回金融业（因为金融行业通常涉及多种业务）

            return 'J-金融业'

# ====================== 行业分类部分 ======================

class IndustryWealthAnalyzer:

    def \_\_init\_\_(self, file\_path):

        self.df = self.\_load\_and\_preprocess\_data(file\_path)

        self.classifier = IndustryClassifier()  # 使用行业分类器

        self.classified\_data = self.\_classify\_industries()

        self.industry\_stats = self.\_analyze\_industry\_wealth()

    def \_load\_and\_preprocess\_data(self, file\_path):

        """加载并预处理数据"""

        df = pd.read\_csv(file\_path)

        # 重命名列以匹配分析需求

        df = df.rename(columns={

            '企业信息\_行业\_中文': '原始行业',

            '财富总值\_亿': '财富总值',

            '个人信息\_姓名\_中文': '代表人物'  # 假设数据中有此列，实际需根据CSV调整

        })

        # 处理空值和异常值

        df['原始行业'] = df['原始行业'].fillna('未分类')

        df['财富总值'] = pd.to\_numeric(df['财富总值'], errors='coerce').fillna(0)

        return df

    def \_classify\_industries(self):

        """使用IndustryClassifier进行行业分类处理"""

        classified = []

        for \_, row in self.df.iterrows():

            original = str(row['原始行业'])

            wealth = row['财富总值']

            # 使用分类器获取行业类别

            category\_str = self.classifier.classify(original)

            # 处理复合行业分类结果

            if ' / ' in category\_str:

                categories = category\_str.split(' / ')

                for cat in categories:

                    classified.append({

                        '原始行业': original,

                        '细分行业': original,

                        '行业大类': cat,

                        '财富值': wealth,

                        '代表人物': row['代表人物']  # 添加代表人物

                    })

            else:

                classified.append({

                    '原始行业': original,

                    '细分行业': original,

                    '行业大类': category\_str,

                    '财富值': wealth,

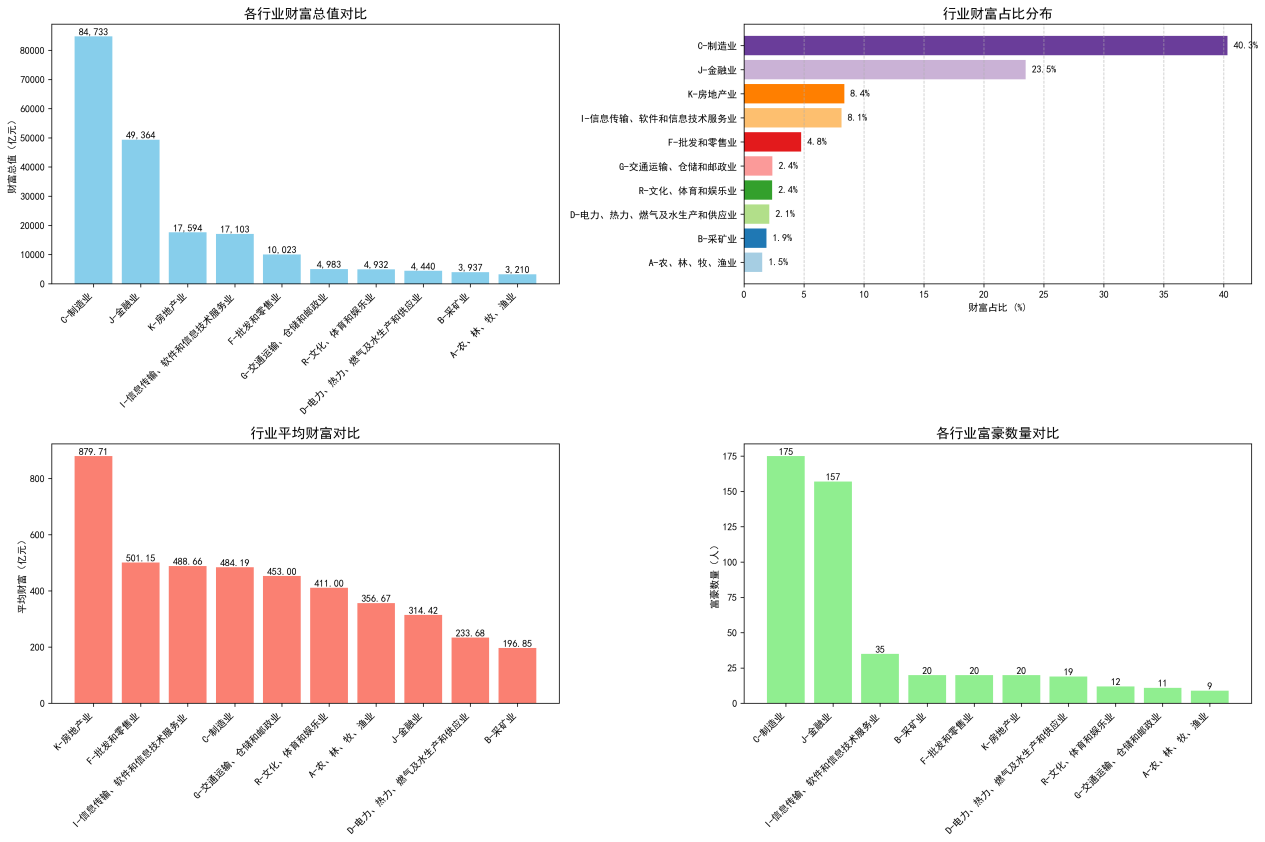
                    '代表人物': row['代表人物']  # 添加代表人物

                })

        return pd.DataFrame(classified)

**1.2.4代码运行与可视化结果**





**1.2.5行业统计分析**



**1.2.6各行业的发展态势**

1. 制造业和金融业：作为经济的两大支柱，在财富积累、规模扩张和富豪培育方面都表现出色，是推动经济发展的重要力量。
2. 房地产业：虽然财富总值和平均财富较高，但富豪数量相对较少，可能受到政策调控、市场周期等因素的影响较大。
3. 信息传输、软件和信息技术服务业：随着数字化时代的到来，该行业发展前景广阔，虽然目前财富总值和富豪数量相对制造业和金融业较少，但具有较大的增长潜力。
4. 其他行业：如批发和零售业、交通运输、仓储和邮政业等，在经济发展中也发挥着基础性作用，但整体规模和财富创造能力相对较弱。

**1.3多维度数据分析**

**1.3.1实现步骤**

1. 数据加载与预处理
2. 多维度分析：年龄、性别、地域、教育、财富等
3. 交叉分析：年龄-财富、教育-行业等
4. 地理分布分析
5. 毕业院校分析

**1.3.2重要库函数**

* pandas：数据处理
* matplotlib 和 seaborn：数据可视化
* wordcloud：生成词云

**1.3.3重要代码：数据预处理、多维度分析**

# 1. 数据加载与预处理

print("正在加载数据...")

df = pd.read\_csv('2024胡润百富榜.csv')

# 数据清洗

df['财富(人民币/亿)'] = pd.to\_numeric(df['财富(人民币/亿)'], errors='coerce')

df['个人信息\_年龄'] = pd.to\_numeric(df['个人信息\_年龄'], errors='coerce')

df['财富变化'] = df['财富变化'].str.replace('%', '').str.replace('—', '0').str.replace('新', '0')

df['财富变化'] = pd.to\_numeric(df['财富变化'], errors='coerce') / 100

# 提取省份信息

def extract\_province(location):

    if pd.isna(location):

        return '未知'

    if '中国' in location:

        parts = location.split('-')

        return parts[1] if len(parts) > 1 else location

    return location

df['出生省份'] = df['个人信息\_出生地\_中文'].apply(extract\_province)

df['出生省份'] = df['出生省份'].str.replace('自治区', '').str.replace('省', '').str.replace('市', '')

df['常住省份'] = df['个人信息\_常住地\_中文'].apply(extract\_province)

df['常住省份'] = df['常住省份'].str.replace('自治区', '').str.replace('省', '').str.replace('市', '')

# 教育程度分类

education\_map = {

    '本科': '本科',

    '硕士': '硕士',

    '博士': '博士',

    '研究生': '硕士',

    'MBA': '硕士',

    'EMBA': '硕士',

    '博士后': '博士',

    '大学': '本科',

    '学士': '本科',

    '高中': '高中及以下',

    '初中': '高中及以下',

    '小学': '高中及以下',

    '中专': '高中及以下',

    '大专': '专科',

    '专科': '专科',

    '无': '未知',

    '': '未知'

}

df['教育程度'] = df['个人信息\_教育程度\_中文'].fillna('未知').map(education\_map).fillna('未知')

# 年龄分段

bins = [20, 30, 40, 50, 60, 70, 80, 100]

labels = ['20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80+']

df['年龄分段'] = pd.cut(df['个人信息\_年龄'], bins=bins, labels=labels, right=False)

# 财富层级

bins = [0, 100, 500, 1000, 2000, 5000, 10000]

labels = ['<100亿', '100-500亿', '500-1000亿', '1000-2000亿', '2000-5000亿', '5000亿+']

df['财富层级'] = pd.cut(df['财富(人民币/亿)'], bins=bins, labels=labels, right=False)

# ========================== 1. 年龄分析 ==========================

print("年龄分析大类图!")

fig = plt.figure(figsize=(20, 12))

fig.suptitle('富豪年龄分析', fontsize=24, fontweight='bold')

# 1.1 年龄分布直方图

ax1 = plt.subplot2grid((2, 2), (0, 0), colspan=1)

sns.histplot(df['个人信息\_年龄'].dropna(), bins=30, kde=True, color='royalblue', alpha=0.7, ax=ax1)

ax1.set\_title('富豪年龄分布', fontsize=16, fontweight='bold')

ax1.set\_xlabel('年龄', fontsize=12)

ax1.set\_ylabel('人数', fontsize=12)

ax1.grid(axis='y', linestyle='--', alpha=0.7)

# 1.2 年龄分段分析

ax2 = plt.subplot2grid((2, 2), (0, 1), colspan=1)

age\_group = df['年龄分段'].value\_counts().sort\_index()

age\_group\_wealth = df.groupby('年龄分段')['财富(人民币/亿)'].mean()

color = '#1f77b4'

ax2.set\_xlabel('年龄分段', fontsize=12)

ax2.set\_ylabel('人数', color=color, fontsize=12)

ax2.bar(age\_group.index, age\_group.values, color=color, alpha=0.7)

ax2.tick\_params(axis='y', labelcolor=color)

ax2.grid(axis='y', linestyle='--', alpha=0.3)

ax3 = ax2.twinx()

color = '#d62728'

ax3.set\_ylabel('平均财富 (亿人民币)', color=color, fontsize=12)

ax3.plot(age\_group\_wealth.index, age\_group\_wealth.values, color=color, marker='o',

         linewidth=3, markersize=8)

ax3.tick\_params(axis='y', labelcolor=color)

ax2.set\_title('不同年龄段富豪数量及平均财富', fontsize=16, fontweight='bold')

# 1.3 年龄与财富关系

ax4 = plt.subplot2grid((2, 2), (1, 0), colspan=2)

sns.scatterplot(

    x='个人信息\_年龄',

    y='财富(人民币/亿)',

    hue='个人信息\_性别',

    size='财富(人民币/亿)',

    sizes=(20, 500),

    alpha=0.7,

    palette='Set1',

    data=df,

    ax=ax4

)

ax4.set\_title('富豪年龄与财富关系', fontsize=16, fontweight='bold')

ax4.set\_xlabel('年龄', fontsize=12)

ax4.set\_ylabel('财富 (亿人民币)', fontsize=12)

ax4.set\_yscale('log')

ax4.grid(True, linestyle='--', alpha=0.3)

ax4.legend(title='性别', loc='upper right')

plt.tight\_layout(rect=[0, 0, 1, 0.96])  # 为标题留出空间

plt.savefig('年龄分析大类图.png', dpi=300, bbox\_inches='tight')

plt.show()

# ========================== 2. 性别分析 ==========================

print("性别分析大类图!")

fig = plt.figure(figsize=(18, 8))

fig.suptitle('富豪性别分析', fontsize=24, fontweight='bold')

# 2.1 性别分布饼图

ax1 = plt.subplot(1, 2, 1)

gender\_counts = df['个人信息\_性别'].value\_counts()

gender\_counts = gender\_counts[gender\_counts.index != '']  # 移除空值

ax1.pie(gender\_counts, labels=gender\_counts.index, autopct='%1.1f%%',

        colors=['#66c2a5', '#fc8d62', '#8da0cb'], startangle=90,

        textprops={'fontsize': 12}, explode=(0.05, 0, 0))

ax1.set\_title('性别分布', fontsize=16, fontweight='bold')

# 2.2 性别与财富关系

ax2 = plt.subplot(1, 2, 2)

gender\_wealth = df.groupby('个人信息\_性别')['财富(人民币/亿)'].mean().sort\_values(ascending=False)

gender\_wealth = gender\_wealth[gender\_wealth.index != '']  # 移除空值

sns.barplot(x=gender\_wealth.index, y=gender\_wealth.values, palette='pastel', ax=ax2)

ax2.set\_title('不同性别平均财富', fontsize=16, fontweight='bold')

ax2.set\_xlabel('性别', fontsize=12)

ax2.set\_ylabel('平均财富 (亿人民币)', fontsize=12)

ax2.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout(rect=[0, 0, 1, 0.96])

plt.savefig('性别分析大类图.png', dpi=300, bbox\_inches='tight')

plt.show()

# ========================== 3. 出生地与常住地分析 ==========================

print("出生地与常住地分析大类图!")

fig = plt.figure(figsize=(20, 12))

fig.suptitle('富豪出生地与常住地分析', fontsize=24, fontweight='bold')

# 3.1 出生地分布TOP15

ax1 = plt.subplot(2, 2, 1)

birth\_province\_counts = df['出生省份'].value\_counts().head(15)

sns.barplot(x=birth\_province\_counts.values, y=birth\_province\_counts.index, palette='viridis', ax=ax1)

ax1.set\_title('出生地TOP15省份', fontsize=16, fontweight='bold')

ax1.set\_xlabel('富豪数量', fontsize=12)

ax1.set\_ylabel('省份', fontsize=12)

ax1.grid(axis='x', linestyle='--', alpha=0.7)

# 3.2 常住地分布TOP15

ax2 = plt.subplot(2, 2, 2)

residence\_province\_counts = df['常住省份'].value\_counts().head(15)

sns.barplot(x=residence\_province\_counts.values, y=residence\_province\_counts.index, palette='magma', ax=ax2)

ax2.set\_title('常住地TOP15省份', fontsize=16, fontweight='bold')

ax2.set\_xlabel('富豪数量', fontsize=12)

ax2.set\_ylabel('省份', fontsize=12)

ax2.grid(axis='x', linestyle='--', alpha=0.7)

# 3.3 省份比较柱状图

ax3 = plt.subplot(2, 1, 2)

province\_birth\_counts = df['出生省份'].value\_counts().reset\_index()

province\_birth\_counts.columns = ['省份', '富豪数量']

province\_residence\_counts = df['常住省份'].value\_counts().reset\_index()

province\_residence\_counts.columns = ['省份', '富豪数量']

# 合并出生地和常住地数据

birth\_residence\_compare = pd.DataFrame({

    '出生地': province\_birth\_counts.set\_index('省份')['富豪数量'],

    '常住地': province\_residence\_counts.set\_index('省份')['富豪数量']

}).fillna(0)

# 只取TOP15省份

top\_provinces = birth\_residence\_compare.sum(axis=1).sort\_values(ascending=False).head(15).index

birth\_residence\_compare = birth\_residence\_compare.loc[top\_provinces]

# 绘制柱状图

birth\_residence\_compare.plot(kind='bar', figsize=(16, 8), color=['#FF8C00', '#1E90FF'], ax=ax3)

ax3.set\_title('TOP15省份出生地与常住地富豪数量比较', fontsize=16, fontweight='bold')

ax3.set\_xlabel('省份', fontsize=12)

ax3.set\_ylabel('富豪数量', fontsize=12)

ax3.grid(axis='y', linestyle='--', alpha=0.7)

ax3.legend(title='分布类型')

plt.tight\_layout(rect=[0, 0, 1, 0.96])

plt.savefig('出生地与常住地分析大类图.png', dpi=300, bbox\_inches='tight')

plt.show()

# ========================== 4. 教育程度分析 ==========================

print("教育程度分析大类图!")

fig = plt.figure(figsize=(18, 12))

fig.suptitle('富豪教育程度分析', fontsize=24, fontweight='bold')

# 4.1 教育程度分布

ax1 = plt.subplot(2, 2, 1)

education\_counts = df['教育程度'].value\_counts()

sns.barplot(x=education\_counts.values, y=education\_counts.index, palette='coolwarm', ax=ax1)

ax1.set\_title('教育程度分布', fontsize=16, fontweight='bold')

ax1.set\_xlabel('人数', fontsize=12)

ax1.set\_ylabel('教育程度', fontsize=12)

ax1.grid(axis='x', linestyle='--', alpha=0.7)

# 4.2 教育程度与财富关系

ax2 = plt.subplot(2, 2, 2)

education\_wealth = df.groupby('教育程度')['财富(人民币/亿)'].mean().sort\_values(ascending=False)

sns.barplot(x=education\_wealth.values, y=education\_wealth.index, palette='plasma', ax=ax2)

ax2.set\_title('不同教育程度平均财富', fontsize=16, fontweight='bold')

ax2.set\_xlabel('平均财富 (亿人民币)', fontsize=12)

ax2.set\_ylabel('教育程度', fontsize=12)

ax2.grid(axis='x', linestyle='--', alpha=0.7)

# 4.3 教育程度与年龄关系

ax3 = plt.subplot(2, 1, 2)

sns.boxplot(x='教育程度', y='个人信息\_年龄', data=df, palette='Set2',

            order=['高中及以下', '专科', '本科', '硕士', '博士'], ax=ax3)

ax3.set\_title('不同教育程度富豪年龄分布', fontsize=16, fontweight='bold')

ax3.set\_xlabel('教育程度', fontsize=12)

ax3.set\_ylabel('年龄', fontsize=12)

ax3.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout(rect=[0, 0, 1, 0.96])

plt.savefig('教育程度分析大类图.png', dpi=300, bbox\_inches='tight')

plt.show()

# ========================== 5. 财富分析 ==========================

print("财富分析大类图!")

fig = plt.figure(figsize=(18, 12))

fig.suptitle('富豪财富分析', fontsize=24, fontweight='bold')

# 5.1 财富分布直方图

ax1 = plt.subplot(2, 2, 1)

sns.histplot(df['财富(人民币/亿)'], bins=50, kde=True, color='green', alpha=0.6, ax=ax1)

ax1.set\_title('财富分布', fontsize=16, fontweight='bold')

ax1.set\_xlabel('财富值 (亿人民币)', fontsize=12)

ax1.set\_ylabel('人数', fontsize=12)

ax1.grid(axis='y', linestyle='--', alpha=0.7)

# 5.2 财富层级分布

ax2 = plt.subplot(2, 2, 2)

wealth\_level = df['财富层级'].value\_counts().sort\_index()

sns.barplot(x=wealth\_level.index, y=wealth\_level.values, palette='viridis', ax=ax2)

ax2.set\_title('财富层级分布', fontsize=16, fontweight='bold')

ax2.set\_xlabel('财富层级 (亿人民币)', fontsize=12)

ax2.set\_ylabel('人数', fontsize=12)

ax2.grid(axis='y', linestyle='--', alpha=0.7)

# 5.3 财富变化分布

ax3 = plt.subplot(2, 1, 2)

sns.histplot(df['财富变化'].dropna(), bins=30, kde=True, color='purple', alpha=0.6, ax=ax3)

ax3.set\_title('财富变化分布', fontsize=16, fontweight='bold')

ax3.set\_xlabel('财富变化率', fontsize=12)

ax3.set\_ylabel('人数', fontsize=12)

ax3.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout(rect=[0, 0, 1, 0.96])

plt.savefig('财富分析大类图.png', dpi=300, bbox\_inches='tight')

plt.show()

# ========================== 6. 行业分析 ==========================

print("行业分析大类图!")

fig = plt.figure(figsize=(20, 12))

fig.suptitle('富豪行业分析', fontsize=24, fontweight='bold')

# 6.1 行业分布TOP15

ax1 = plt.subplot(2, 2, 1)

industry\_counts = df['企业信息\_行业\_中文'].value\_counts().head(15)

sns.barplot(x=industry\_counts.values, y=industry\_counts.index, palette='coolwarm', ax=ax1)

ax1.set\_title('行业分布TOP15', fontsize=16, fontweight='bold')

ax1.set\_xlabel('富豪数量', fontsize=12)

ax1.set\_ylabel('行业', fontsize=12)

ax1.grid(axis='x', linestyle='--', alpha=0.7)

# 6.2 行业平均财富TOP15

ax2 = plt.subplot(2, 2, 2)

industry\_wealth = df.groupby('企业信息\_行业\_中文')['财富(人民币/亿)'].mean().sort\_values(ascending=False).head(15)

sns.barplot(x=industry\_wealth.values, y=industry\_wealth.index, palette='plasma', ax=ax2)

ax2.set\_title('行业平均财富TOP15', fontsize=16, fontweight='bold')

ax2.set\_xlabel('平均财富 (亿人民币)', fontsize=12)

ax2.set\_ylabel('行业', fontsize=12)

ax2.grid(axis='x', linestyle='--', alpha=0.7)

# 6.3 行业、年龄与财富关系

ax3 = plt.subplot(2, 1, 2)

top\_industries = df['企业信息\_行业\_中文'].value\_counts().head(10).index

top\_df = df[df['企业信息\_行业\_中文'].isin(top\_industries)]

sns.scatterplot(

    x='个人信息\_年龄',

    y='财富(人民币/亿)',

    hue='企业信息\_行业\_中文',

    size='财富(人民币/亿)',

    sizes=(20, 500),

    alpha=0.7,

    palette='tab10',

    data=top\_df,

    ax=ax3

)

ax3.set\_title('TOP10行业富豪年龄与财富关系', fontsize=16, fontweight='bold')

ax3.set\_xlabel('年龄', fontsize=12)

ax3.set\_ylabel('财富 (亿人民币)', fontsize=12)

ax3.set\_yscale('log')

ax3.grid(True, linestyle='--', alpha=0.3)

ax3.legend(title='行业', loc='upper right')

plt.tight\_layout(rect=[0, 0, 1, 0.96])

plt.savefig('行业分析大类图.png', dpi=300, bbox\_inches='tight')

plt.show()

# ========================== 7. 地理分布分析 ==========================

print("地理分布分析大类图!")

# 创建中国省份坐标映射

province\_coords = {

    '北京': (116.4, 39.9), '天津': (117.2, 39.1), '河北': (114.5, 38.0),

    '山西': (112.5, 37.9), '内蒙古': (111.7, 40.8), '辽宁': (123.4, 41.8),

    '吉林': (125.3, 43.9), '黑龙江': (126.6, 45.8), '上海': (121.5, 31.2),

    '江苏': (118.8, 32.1), '浙江': (120.2, 30.3), '安徽': (117.3, 31.8),

    '福建': (119.3, 26.1), '江西': (115.9, 28.7), '山东': (117.0, 36.7),

    '河南': (113.7, 34.8), '湖北': (114.3, 30.6), '湖南': (113.0, 28.2),

    '广东': (113.3, 23.1), '广西': (108.3, 22.8), '海南': (110.3, 20.0),

    '重庆': (106.5, 29.5), '四川': (104.1, 30.7), '贵州': (106.7, 26.6),

    '云南': (102.7, 25.0), '西藏': (91.1, 29.6), '陕西': (108.9, 34.3),

    '甘肃': (103.8, 36.0), '青海': (101.8, 36.6), '宁夏': (106.3, 38.5),

    '新疆': (87.6, 43.8), '台湾': (121.5, 25.0), '香港': (114.2, 22.3),

    '澳门': (113.5, 22.2), '未知': (100.0, 20.0),'新加坡':(100.0,20.0)

}

# 准备数据

province\_birth\_counts = df['出生省份'].value\_counts().reset\_index()

province\_birth\_counts.columns = ['省份', '富豪数量']

province\_birth\_counts['经度'] = province\_birth\_counts['省份'].map(lambda x: province\_coords.get(x, (0, 0))[0])

province\_birth\_counts['纬度'] = province\_birth\_counts['省份'].map(lambda x: province\_coords.get(x, (0, 0))[1])

province\_birth\_counts = province\_birth\_counts[province\_birth\_counts['省份'] != '未知']

province\_residence\_counts = df['常住省份'].value\_counts().reset\_index()

province\_residence\_counts.columns = ['省份', '富豪数量']

province\_residence\_counts['经度'] = province\_residence\_counts['省份'].map(lambda x: province\_coords.get(x, (0, 0))[0])

province\_residence\_counts['纬度'] = province\_residence\_counts['省份'].map(lambda x: province\_coords.get(x, (0, 0))[1])

province\_residence\_counts = province\_residence\_counts[province\_residence\_counts['省份'] != '未知']

# 创建图表

fig = plt.figure(figsize=(20, 20))

fig.suptitle('富豪地理分布分析', fontsize=24, fontweight='bold')

plt.subplots\_adjust(left=0.05, right=0.95, top=0.85, bottom=0.15, wspace=0.3, hspace=0.4)

# 7.1 出生地地理分布热力图

ax1 = plt.subplot(2, 2, 1)

ax1.scatter(

    province\_birth\_counts['经度'],

    province\_birth\_counts['纬度'],

    s=province\_birth\_counts['富豪数量']\*5,

    c=province\_birth\_counts['富豪数量'],

    cmap='YlOrRd',

    alpha=0.7

)

for i, row in province\_birth\_counts.iterrows():

    if row['富豪数量'] > 5:

        ax1.text(

            row['经度'] + 0.5,

            row['纬度'] + 0.5,

            f"{row['省份']}({row['富豪数量']})",

            fontsize=10,

            ha='center'

        )

ax1.set\_title('出生地分布', fontsize=16, fontweight='bold')

ax1.set\_xlabel('经度', fontsize=12)

ax1.set\_ylabel('纬度', fontsize=12)

ax1.set\_xlim(100, 130)

ax1.set\_ylim(20, 45)

ax1.grid(True, linestyle='--', alpha=0.5)

# 7.2 常住地地理分布热力图

ax2 = plt.subplot(2, 2, 2)

ax2.scatter(

    province\_residence\_counts['经度'],

    province\_residence\_counts['纬度'],

    s=province\_residence\_counts['富豪数量']\*5,

    c=province\_residence\_counts['富豪数量'],

    cmap='Blues',

    alpha=0.7

)

for i, row in province\_residence\_counts.iterrows():

    if row['富豪数量'] > 5:

        ax2.text(

            row['经度'] + 0.5,

            row['纬度'] + 0.5,

            f"{row['省份']}({row['富豪数量']})",

            fontsize=10,

            ha='center'

        )

ax2.set\_title('常住地分布', fontsize=16, fontweight='bold')

ax2.set\_xlabel('经度', fontsize=12)

ax2.set\_ylabel('纬度', fontsize=12)

ax2.set\_xlim(100, 130)

ax2.set\_ylim(20, 45)

ax2.grid(True, linestyle='--', alpha=0.5)

# 7.3 省份比较柱状图

ax3 = plt.subplot(2, 1, 2)

birth\_residence\_compare = pd.DataFrame({

    '出生地': province\_birth\_counts.set\_index('省份')['富豪数量'],

    '常住地': province\_residence\_counts.set\_index('省份')['富豪数量']

}).fillna(0)

top\_provinces = birth\_residence\_compare.sum(axis=1).sort\_values(ascending=False).head(15).index

birth\_residence\_compare = birth\_residence\_compare.loc[top\_provinces]

birth\_residence\_compare.plot(kind='bar', color=['#FF8C00', '#1E90FF'], ax=ax3)

ax3.set\_title('TOP15省份出生地与常住地富豪数量比较', fontsize=16, fontweight='bold')

ax3.set\_xlabel('省份', fontsize=12)

ax3.set\_ylabel('富豪数量', fontsize=12)

ax3.grid(axis='y', linestyle='--', alpha=0.7)

ax3.legend(title='分布类型')

plt.tight\_layout(rect=[0, 0, 1, 0.96])

plt.savefig('地理分布分析大类图.png', dpi=300, bbox\_inches='tight')

plt.show()

# ========================== 8. 毕业院校分析 ==========================

print("毕业院校分析大类图!")

fig = plt.figure(figsize=(20, 12))

fig.suptitle('富豪毕业院校分析', fontsize=24, fontweight='bold')

plt.subplots\_adjust(left=0.05, right=0.95, top=0.85, bottom=0.15, wspace=0.5)

# 8.1 毕业院校词云

ax1 = plt.subplot(1, 2, 1)

school\_text = ' '.join(df['个人信息\_毕业院校'].dropna().astype(str))

stopwords = set(['大学', '学院', '学校', '中国', '北京', '上海', '浙江', '江苏', '广东'])

wordcloud = WordCloud(

    font\_path='C:/Windows/Fonts/simhei.ttf',

    background\_color='white',

    width=1200,

    height=800,

    max\_words=150,

    stopwords=stopwords,

    colormap='viridis'

).generate(school\_text)

ax1.imshow(wordcloud, interpolation='bilinear')

ax1.axis('off')

ax1.set\_title('毕业院校词云', fontsize=20, fontweight='bold')

# 8.2 TOP20毕业院校

ax2 = plt.subplot(1, 2, 2)

school\_counts = df['个人信息\_毕业院校'].value\_counts().head(20)

sns.barplot(x=school\_counts.values, y=school\_counts.index, palette='magma', ax=ax2)

ax2.set\_title('TOP20富豪毕业院校', fontsize=18, fontweight='bold')

ax2.set\_xlabel('富豪数量', fontsize=14)

ax2.set\_ylabel('院校', fontsize=14)

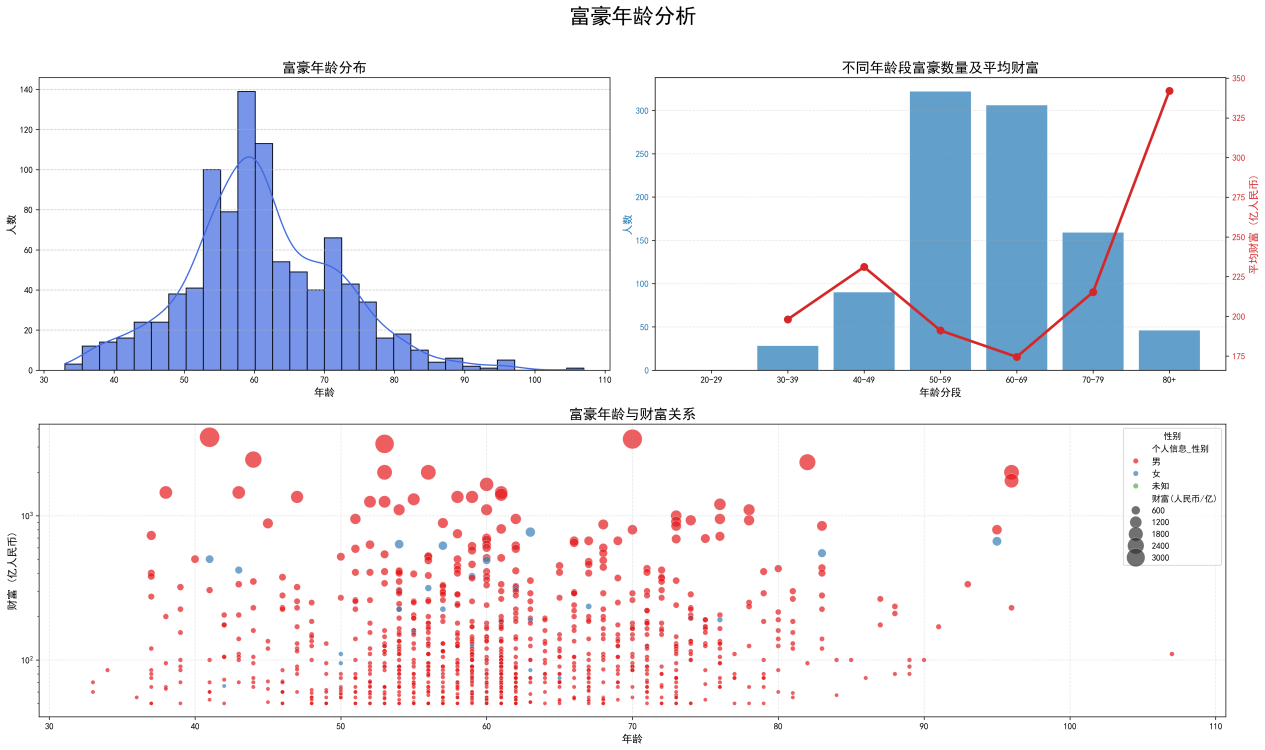
ax2.grid(axis='x', linestyle='--', alpha=0.7)

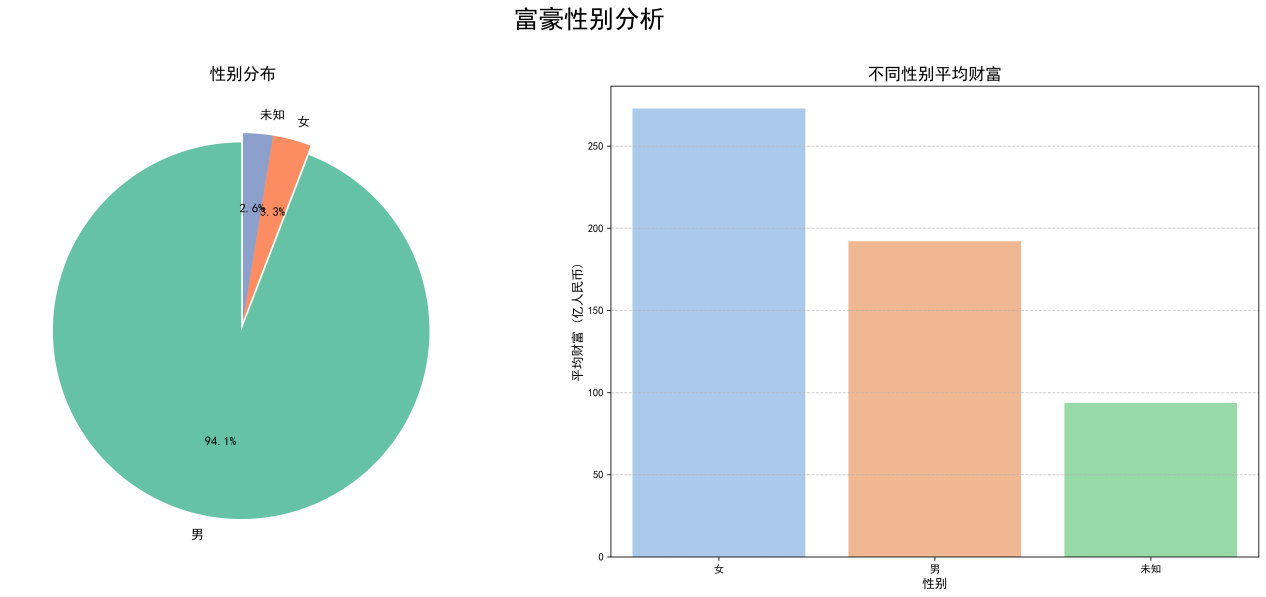
plt.tight\_layout(rect=[0, 0, 1, 0.96])

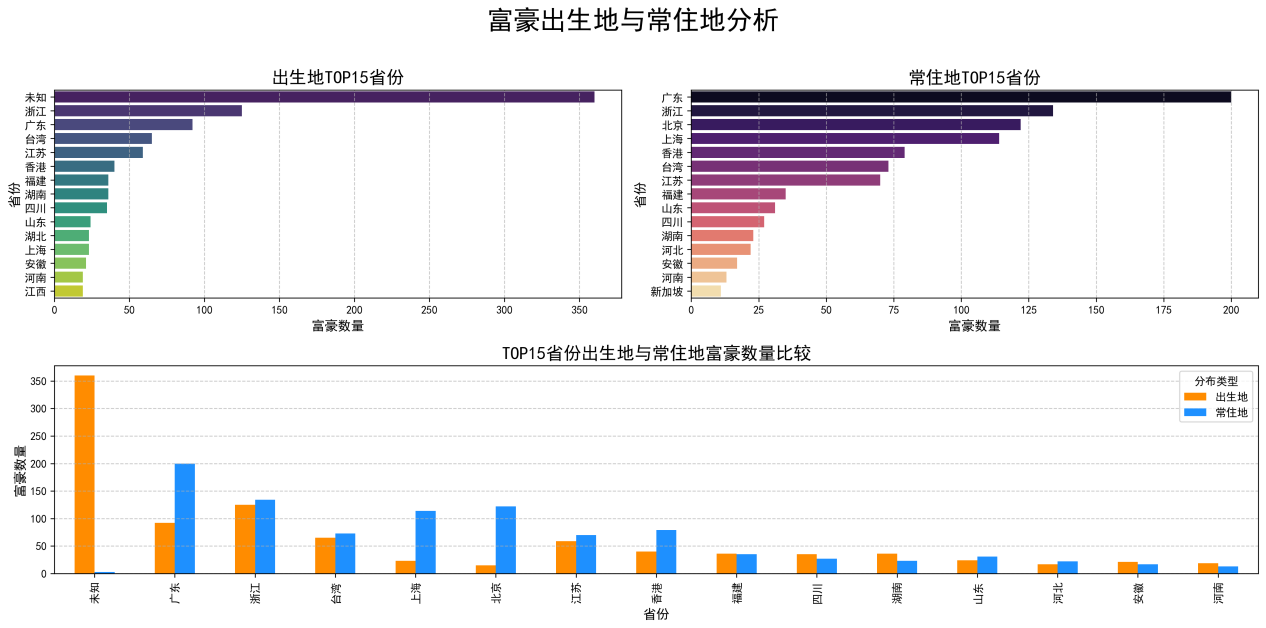
plt.savefig('毕业院校分析大类图.png', dpi=300, bbox\_inches='tight')

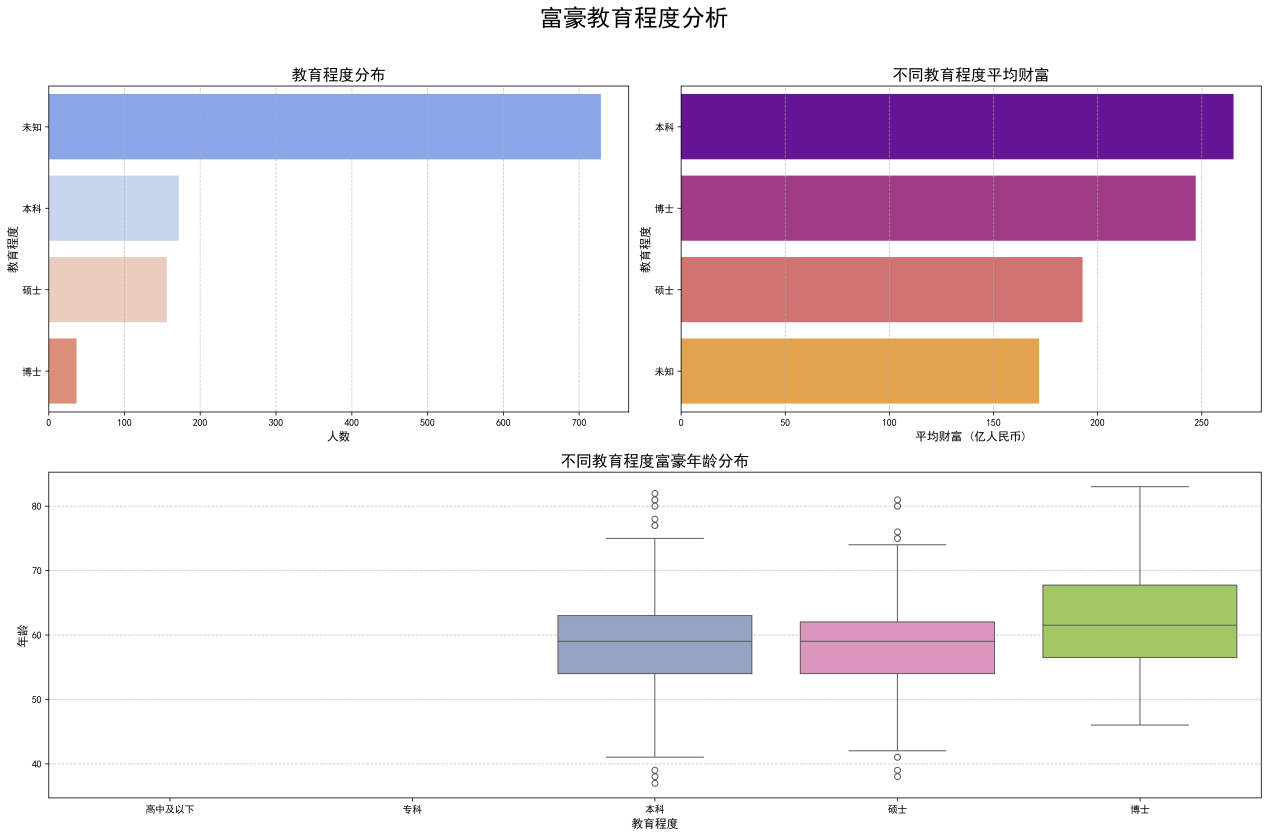
plt.show()

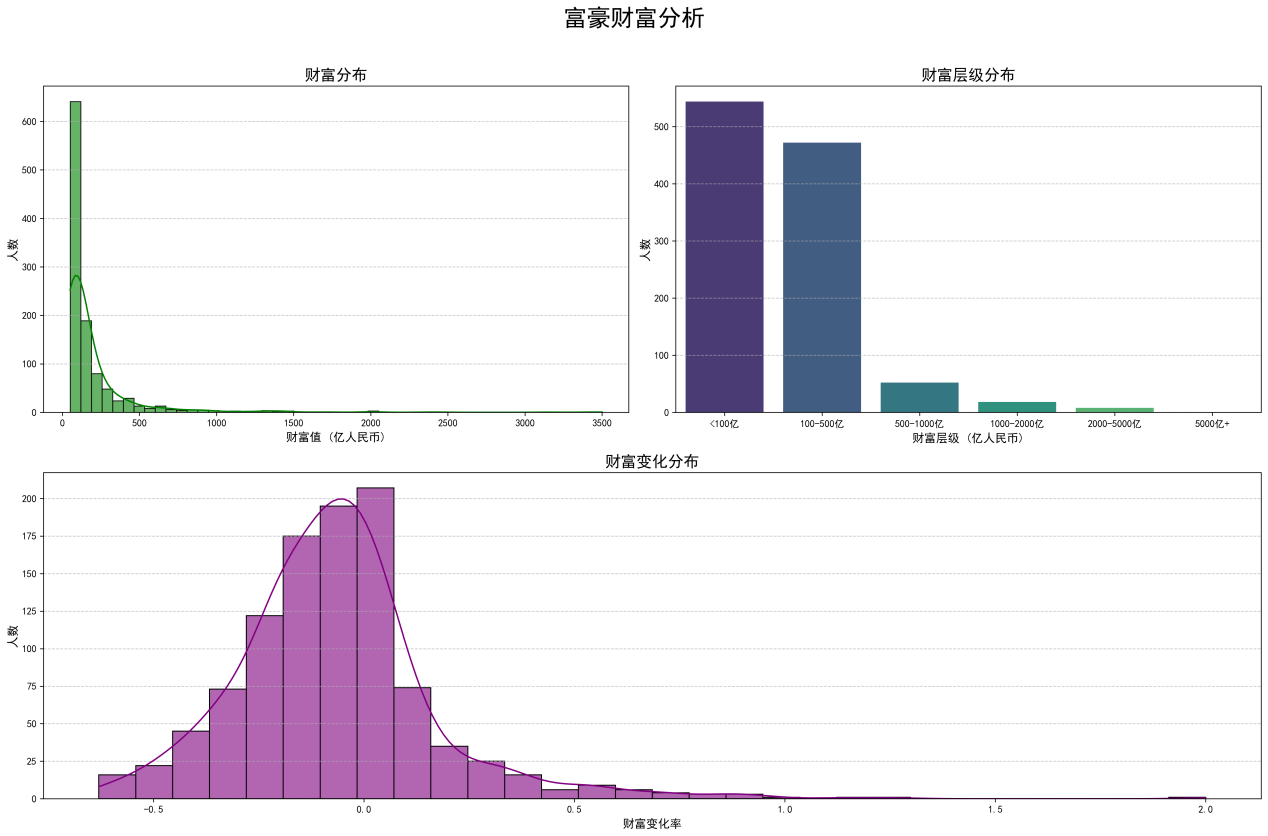
**1.3.4数据可视化结果(T1\_3\_聚合.py)**

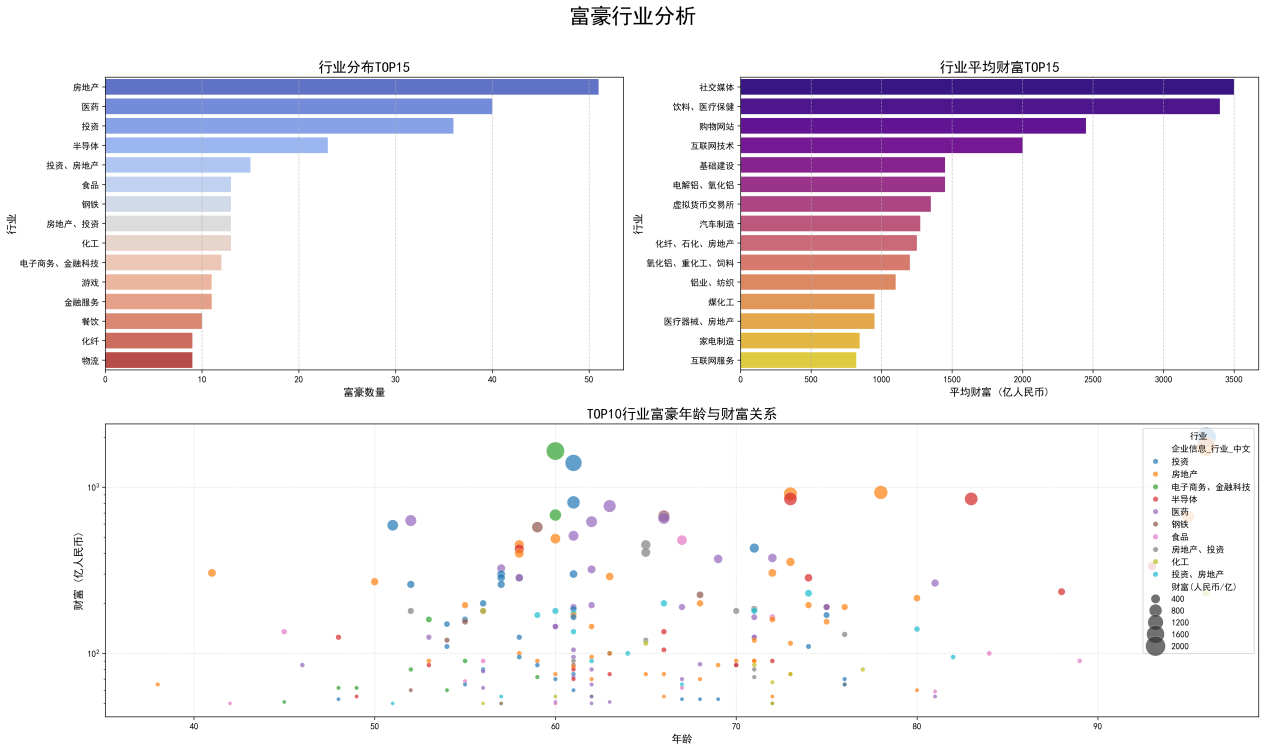


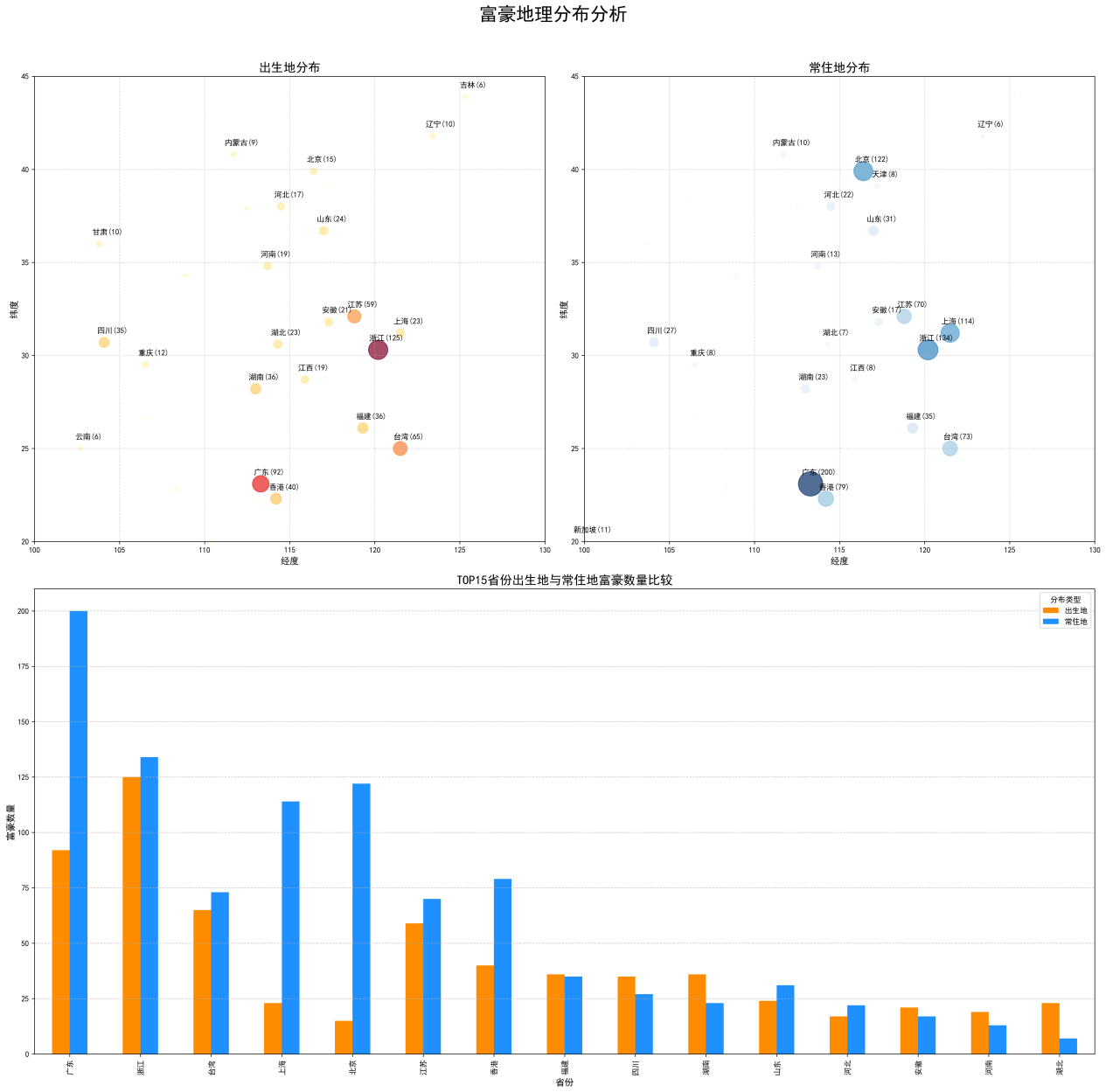


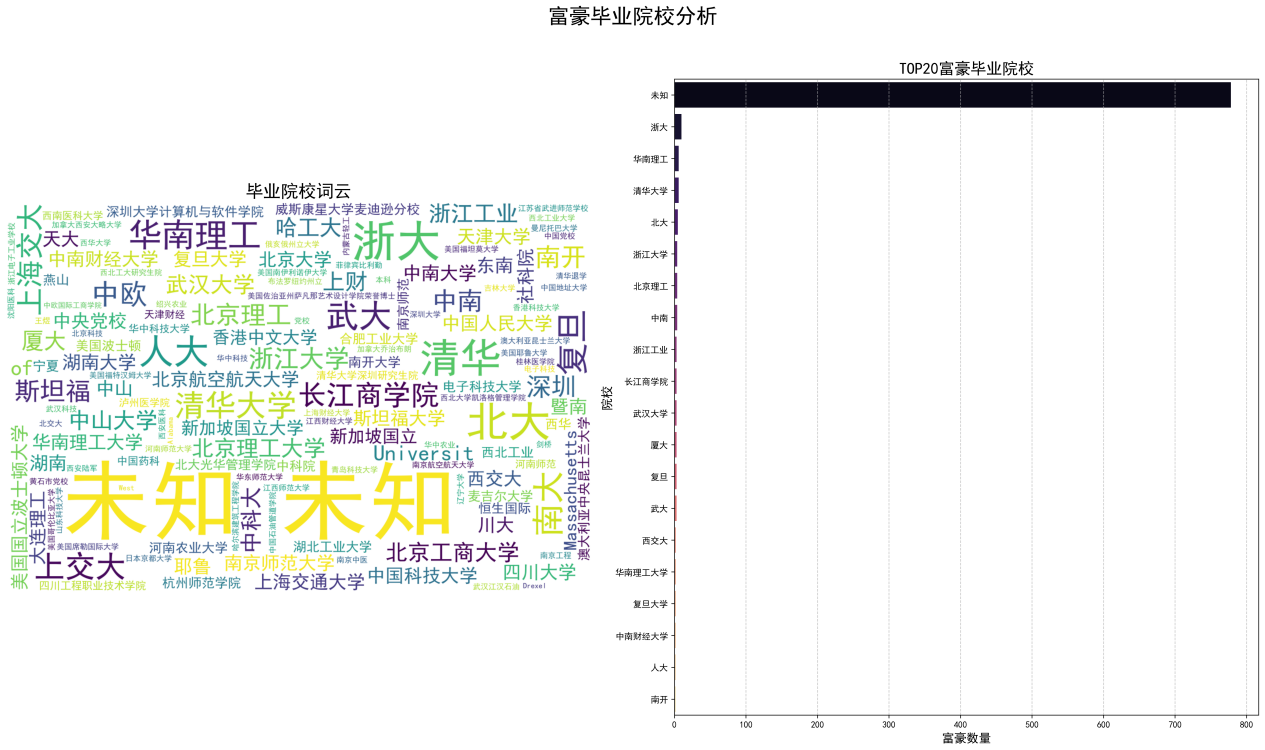












**1.3.5数据可视化结果(T1\_3\_各子图.py)**

