

数据探索性分析与预处理过程报告

项目地址：https://github.com/bazhouyilian/data_mining1/tree/master

姓名：李文博 学号：3220241370

1.数据集分析及可视化

- 数据集分析：首先将数据读取进来，查看数据集的基本信息，包括数据集的形状、列名、数据类型、数据集的前5行等，代码如下所示：

```
df = pd.read_parquet(path)

# 查看前几行
print(df.head())

# 查看性别具体的值
print(df['purchase_history'].value_counts())

# 查看基本信息
print(df.info())

def extract_purchase_info(purchase):
    try:
        record = json.loads(purchase)
        avg_price = record.get('average_price', 0)
        item_count = len(record.get('items', []))
        category = record.get('category', '未知')
        return pd.Series([avg_price, item_count, category])
    except:
        print(f"解析失败: {purchase}, 错误信息: {e}")
        return pd.Series([0, 0, '未知'])

df[['purchase_avg_price', 'purchase_item_count', 'purchase_category']] =
df['purchase_history'].apply(extract_purchase_info)

dig_data = ['age', 'income', 'credit_score', 'purchase_avg_price',
'purchase_item_count']

print(df[dig_data].describe())
```

结果如下所示：

(EDA_preprocessing) D:\code\python\EDA_preprocessing\python EDA.py														
	id	timestamp	user_name	chinese_name	email	age	...	chinese_address	purchase_history	is_active	registration_date	credit_score	phone_number	
0	1	2025-01-15T02:06:51+00:00	DFPVPK	冯娜	xlbvqpk@126.com	77	...	青海省绍兴青年路90号1单元241	{"average_price":361.45,"category":"食品","items...	False	2024-10-16		693	924-519-5
030														
1	2	2023-01-05T14:43:16+00:00	wzobou	马晓	pqkfypth@gmail.com	96	...	江西省保定北京路123号3单元1336	{"average_price":406.99,"category":"家居","items...	False	2020-01-11		823	371-868-4
475														
2	3	2024-11-17T23:59:54+00:00	ftotzea	傅佳	delhadue@outlook.com	30	...	四川省温州平安路152号5单元2424	{"average_price":152.56,"category":"服装","items...	False	2024-06-01		381	655-419-5
633														
3	4	2023-10-01T16:58:51+00:00	wwsbbu	马雨	dlwFlftg@gmail.com	53	...	江西省南通长江路97号2单元659	{"average_price":712.9,"category":"电子产品","item...	False	2021-10-05		532	644-588
-9573														
4	5	2024-02-23T13:10:07+00:00	utbca	杨玲	bxaureys@163.com	88	...	江苏省南昌人民路150号2单元784	{"average_price":489.15999999999997,"category":...	False	2022-09-10		769	971-685-513

图1 默认前5行信息

Data columns (total 15 columns):

#	Column	Non-Null Count	Dtype
0	id	1250000 non-null	int64
1	timestamp	1250000 non-null	object
2	user_name	1250000 non-null	object
3	chinese_name	1250000 non-null	object
4	email	1250000 non-null	object
5	age	1250000 non-null	int64
6	income	1250000 non-null	float64
7	gender	1250000 non-null	object
8	country	1250000 non-null	object
9	chinese_address	1250000 non-null	object
10	purchase_history	1250000 non-null	object
11	is_active	1250000 non-null	bool
12	registration_date	1250000 non-null	object
13	credit_score	1250000 non-null	int64
14	phone_number	1250000 non-null	object

dtypes: bool(1), float64(1), int64(3), object(10)

图2 数据集列数及类型

[5 rows x 15 columns]

purchase_history															
{ "average_price":187.20999999999998,"category":"书籍","items":[{"id":230}]}															26
{ "average_price":651.52,"category":"书籍","items":[{"id":325}]}															26
{ "average_price":842.5899999999999,"category":"书籍","items":[{"id":714}]}															26
{ "average_price":575.29,"category":"电子产品","items":[{"id":614}]}															26
{ "average_price":318.88,"category":"书籍","items":[{"id":5}]}															25
..															
{ "average_price":508.96,"category":"食品","items":[{"id":923}, {"id":681}, {"id":549}, {"id":172}, {"id":405}]}															12
{ "average_price":609.9399999999999,"category":"食品","items":[{"id":866}, {"id":836}, {"id":568}, {"id":284}, {"id":956}, {"id":780}]}															12
{ "average_price":993.07,"category":"食品","items":[{"id":117}, {"id":901}, {"id":464}, {"id":938}]}															12
{ "average_price":985.15,"category":"电子产品","items":[{"id":198}, {"id":777}, {"id":673}, {"id":489}, {"id":374}]}															12
{ "average_price":561.4300000000001,"category":"服装","items":[{"id":917}, {"id":763}]}															12

图3 对购物记录的查看

	age	income	credit_score	purchase_avg_price	purchase_item_count
count	1.250000e+06	1.250000e+06	1.250000e+06	1.250000e+06	1.250000e+06
mean	5.888587e+01	4.989553e+05	5.742449e+02	5.047035e+02	5.485817e+00
std	2.400122e+01	2.890911e+05	1.593347e+02	2.859578e+02	2.876076e+00
min	1.800000e+01	0.000000e+00	3.000000e+02	1.000000e+01	1.000000e+00
25%	3.800000e+01	2.490000e+05	4.360000e+02	2.565100e+02	3.000000e+00
50%	5.900000e+01	4.970000e+05	5.730000e+02	5.059900e+02	5.000000e+00
75%	8.000000e+01	7.500000e+05	7.120000e+02	7.515100e+02	8.000000e+00
max	1.000000e+02	1.000000e+06	8.500000e+02	1.000000e+03	1.000000e+01

图4 数值型数据的描述

- 数据可视化：对数据进行可视化，分别从三方面来进行可视化，包括用户的基本属性、用户的行为模式以及各种属性之间的关系，代码如下所示：
 - 用户基本属性：对用户基本属性进行可视化，包括性别、年龄、收入、信用得分、国家等，代码如下所示：

```
#绘制基本属性相关的图
def draw_basic_attributes_distribution(df):

    # 创建子图：2 行 3 列
    fig, axes = plt.subplots(2, 3, figsize=(18, 10))
    fig.suptitle("用户基础属性分布", fontsize=16, fontproperties=font)

    # 1. 年龄分布
    sns.histplot(df['age'], bins=30, kde=True, ax=axes[0, 0],
```

```

color='skyblue')
axes[0, 0].set_title("年龄分布", fontproperties=font)

# 2. 收入分布
sns.histplot(df['income'], bins=30, kde=True, ax=axes[0, 1],
color='salmon')
axes[0, 1].set_title("收入分布", fontproperties=font)

# 3. 性别分布
sns.countplot(x='gender', data=df, ax=axes[0, 2], palette='Set2')
axes[0, 2].set_title("性别分布", fontproperties=font)
axes[0, 2].set_xticklabels(axes[0, 2].get_xticklabels(),
fontproperties=font)

# 4. 国家分布 (取前10国家)
top_countries = df['country'].value_counts().nlargest(10)
sns.barplot(x=top_countries.values, y=top_countries.index, ax=axes[1,
0], palette='viridis')
axes[1, 0].set_title("国家分布 (前十)", fontproperties=font)
axes[1, 0].set_yticklabels(axes[1, 0].get_yticklabels(),
fontproperties=font)

# 5. 信用分分布
sns.histplot(df['credit_score'], bins=30, kde=True, ax=axes[1, 1],
color='mediumseagreen')
axes[1, 1].set_title("信用分分布", fontproperties=font)

# 第六个子图隐藏
axes[1, 2].axis('off')

# 自动调整布局
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()

draw_basic_attributes_distribution(df)

```

- 用户行为模式：对用户行为模式进行可视化，包括购买记录、购买的平均价格、购买物品的数量、购买物品的种类等、收入、信用分数以及收入和信用分数的关系等，代码如下所示：

```

def draw_user_behavior_distribution(df):
# 提取用户行为字段
    def extract_purchase_info(purchase):
        try:
            record = json.loads(purchase)
            avg_price = record.get('average_price', 0)
            item_count = len(record.get('items', []))
            category = record.get('category', '未知')
            return pd.Series([avg_price, item_count, category])
        except:
            print(f"解析失败: {purchase}, 错误信息: {e}")
            return pd.Series([0, 0, '未知'])

```

```

    df[['purchase_avg_price', 'purchase_item_count', 'purchase_category']]
= df['purchase_history'].apply(extract_purchase_info)

# 创建图形
fig, axes = plt.subplots(2, 3, figsize=(16, 12))
fig.suptitle("用户行为可视化", fontsize=16, fontproperties=font)

# 1. 平均购买价格 - 箱线图
sns.boxplot(y='purchase_avg_price', data=df, ax=axes[0, 0],
color='lightblue')
axes[0, 0].set_title("平均购买价格分布 (箱线图)", fontproperties=font)

# 2. 购买商品数量 - 直方图
sns.histplot(df['purchase_item_count'], bins=20, kde=False, ax=axes[0,
1], color='salmon')
axes[0, 1].set_title("每个用户的购买数量分布", fontproperties=font)

# 3. 主要消费品类 - 柱状图
top_categories = df['purchase_category'].value_counts().nlargest(10)
sns.barplot(x=top_categories.index, y=top_categories.values, ax=axes[1,
0], palette='Set3')
axes[1, 0].set_title("用户主要消费品类分布", fontproperties=font)
axes[1, 0].tick_params(axis='x', rotation=30)
axes[1, 0].set_xticklabels(axes[1, 0].get_xticklabels(),
fontproperties=font)

# 4. 收入 vs 信用分数 - 散点图
sns.scatterplot(x='income', y='credit_score', data=df, alpha=0.4,
ax=axes[1, 1])
axes[1, 1].set_title("财务状况 (收入 vs 信用分)", fontproperties=font)
correlation = df[['income',
'credit_score']].corr(method='spearman').iloc[0, 1]
print(f"收入与信用分数的斯皮尔曼相关系数为: {correlation:.2f}")

# 5. 收入直方图
sns.histplot(df['income'], bins=30, kde=False, ax=axes[1, 2],
color='salmon')
axes[1, 2].set_title("收入分布", fontproperties=font)

axes[0, 2].axis('off')

# 自动调整布局
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
return df

df = draw_user_behavior_distribution(df)

```

- 用户属性与行为模式的关系：对用户属性与行为模式的关系进行可视化，包括用户属性与购买记录的关系、用户属性与收入关系以及用户属性与信用分数关系等，代码如下所示：

```
# #绘制相关性热力图
def draw_correlation_heatmap(df):
    selected_cols = ['income', 'credit_score', 'purchase_avg_price',
                    'purchase_item_count']
    df_selected = df[selected_cols].dropna() # 去掉缺失值行

# 计算斯皮尔曼相关性矩阵
    corr_matrix = df_selected.corr(method='spearman')

# 绘制热力图
    plt.figure(figsize=(8, 6))
    sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm',
square=True)
    plt.title("用户关键指标相关性热力图", fontproperties=font)
    plt.tight_layout()
    plt.show()

draw_correlation_heatmap(df)
```

- 使用上述方法对10G数据集进行可视化，部分结果如下所示：

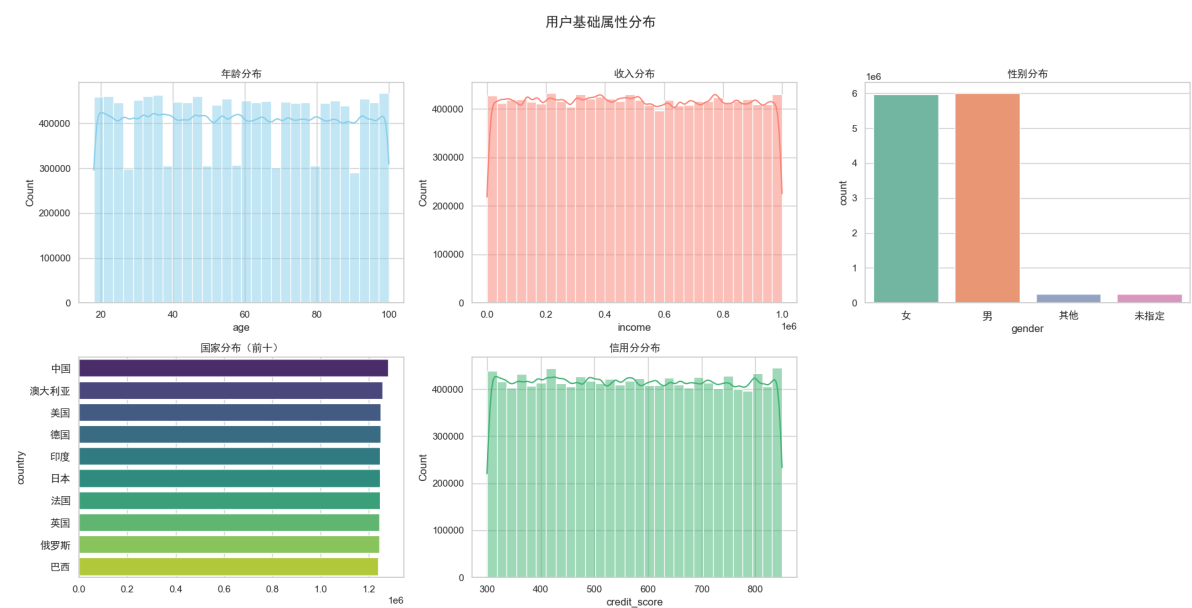


图5 10G数据集用户基本属性的可视化

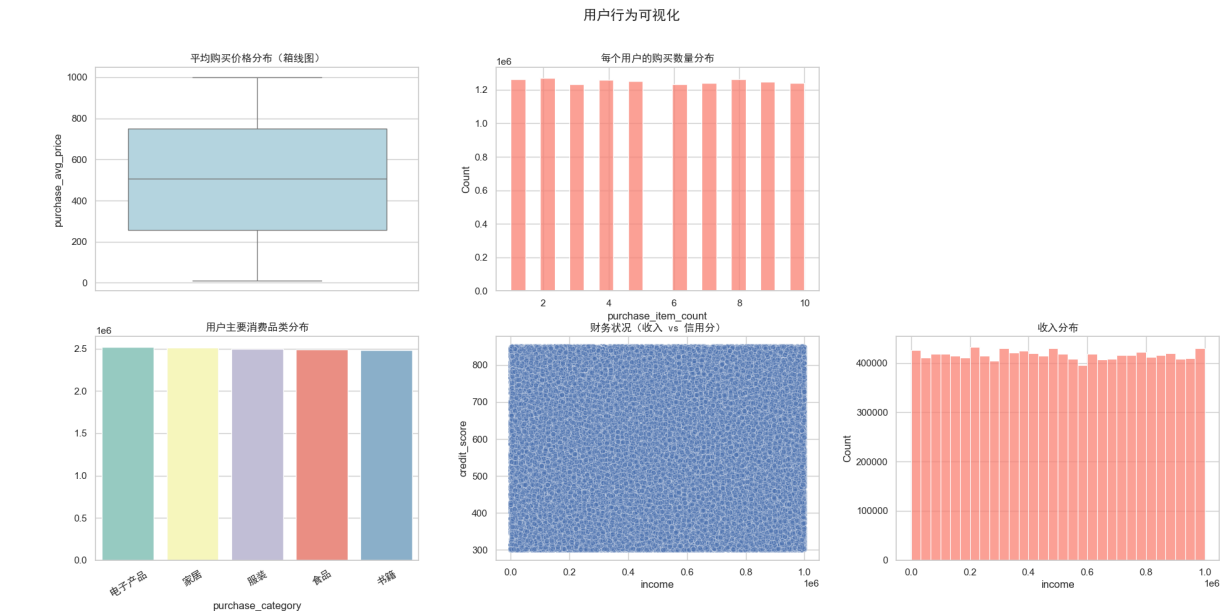


图6 10G数据集用户行为的可视化

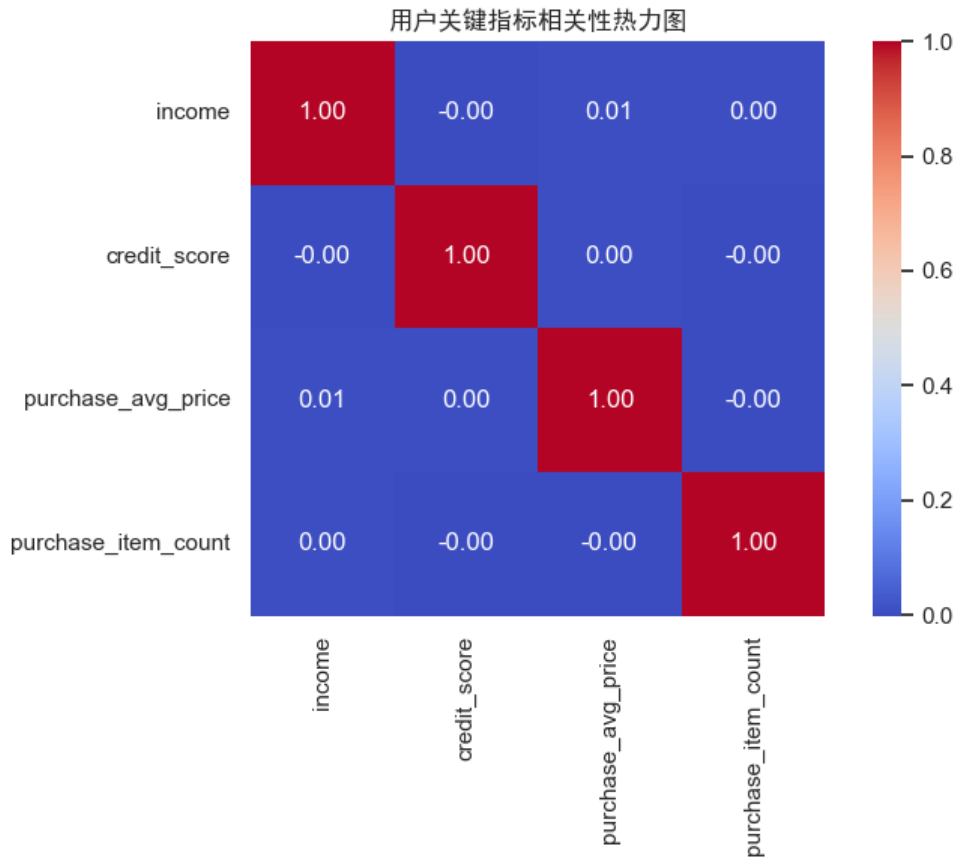


图7 10G数据集可视化各属性关系

- 对30G数据集进行可视化，部分结果如下所示：

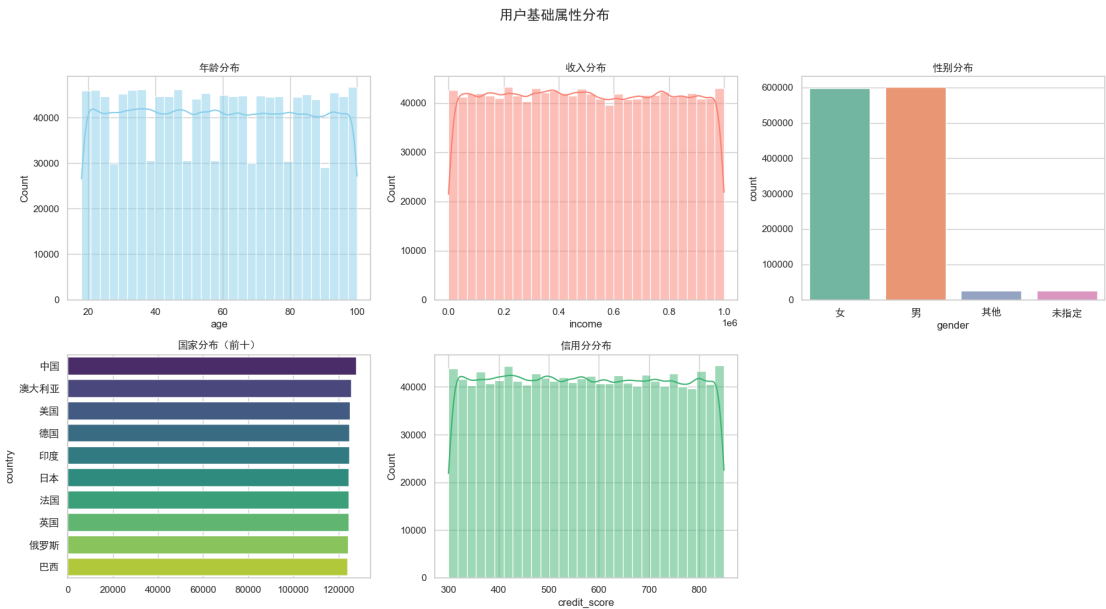


图8 30G数据集用户基本属性的可视化

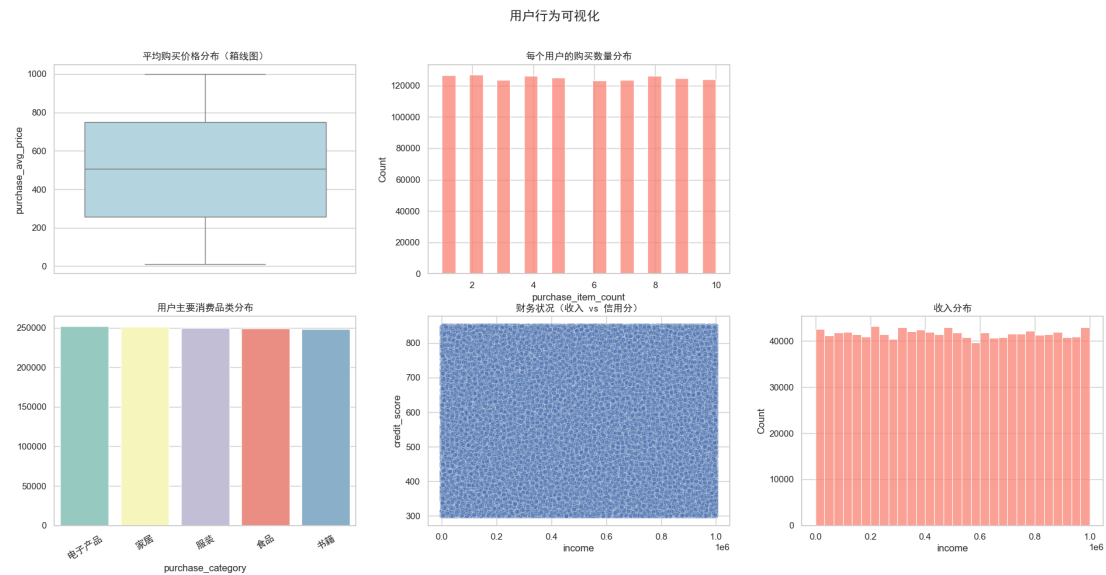


图9 30G数据集用户行为的可视化

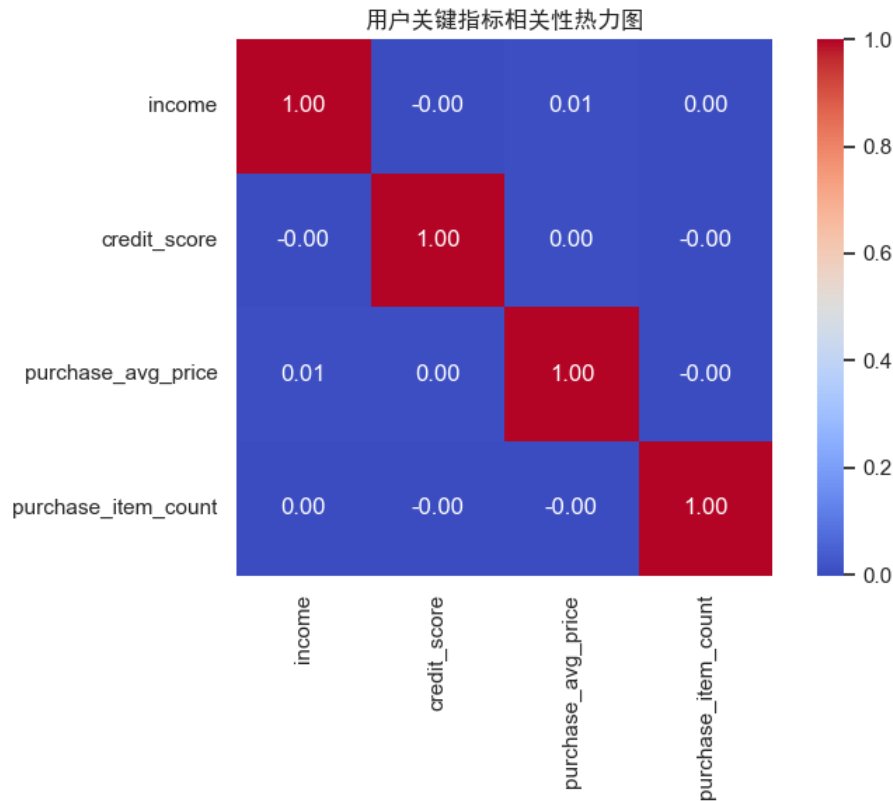


图10 30G数据集可视化各属性关系

2.数据预处理

- 数据清理
 - 处理缺失值：先检查数据集里面的数据是否存在缺失值，如果存在缺失值，则将存在缺失值的元组删除。缺失值的检查代码如下：

```
#缺失值处理
def handle_missing_values(df_clean):
    missing = df_clean.isnull().sum()
    # 计算缺失值比例
    missing_ratio = missing / len(df_clean) * 100

    # 创建一个 DataFrame 来存储缺失值信息
    missing_info = pd.DataFrame({
        '缺失值数量': missing,
        '缺失值比例 (%)': missing_ratio
    })

    # 打印缺失值信息表格
    print("缺失值统计：")
    print(missing_info)

    # 处理空缺值
    df_clean = df_clean.dropna()
    print(f"处理空缺值后剩余记录数：{len(df_clean)}\n")
    return df_clean
```


- 处理异常值：异常值是指与数据集中的大部分数据差异显著的点。处理方法为删除异常值，直接删除异常数据点。
- 使用四分位间距 (IQR)对数值型数据进行异常值检查，包括年龄、收入、信用得分，代码如下所示：

```
def remove_outliers_iqr(df, column):  
    """使用 IQR 方法去除某列的异常值"""  
    Q1 = df[column].quantile(0.25)  
    Q3 = df[column].quantile(0.75)  
    IQR = Q3 - Q1  
    lower_bound = max(Q1 - 1.5 * IQR, 0)  
    upper_bound = Q3 + 1.5 * IQR  
    print(f"{column} 过滤范围: {lower_bound:.2f} ~  
{upper_bound:.2f}")  
    return df[(df[column] >= lower_bound) & (df[column] <=  
upper_bound)]
```

使用IQR进行计算以后，可以画出箱线图，对异常值进行可视化，代码如下所示：

```
def draw_boxplot(df, col, file_id, info):  
    plt.figure(figsize=(6, 4))  
    sns.boxplot(y=df[col])  
    plt.title(f"{col} 箱线图 - {file_id}", fontproperties=font)  
    plt.tight_layout()  
  
    # 构建保存路径  
    save_path =  
    f"./result_images/{info}boxplot_{col}_{file_id}.png"  
    save_dir = os.path.dirname(save_path)  
  
    # 如果目录不存在，则创建目录  
    if not os.path.exists(save_dir):  
        os.makedirs(save_dir)  
  
    plt.savefig(save_path) # 保存为图片  
    plt.close()
```

- 处理类别型数据：对类别型数据进行异常值检查，包括性别、邮箱、电话号码，代码如下所示：

```
def is_valid_phone(phone):  
    return bool(re.match(r"\d{3}-\d{3}-\d{4}$", phone))  
def is_valid_email(email):  
    return bool(re.match(r"^[^@]+@[^@]+\.[^@]+$", email))  
  
expected_cols = ['gender', 'email', 'phone_number']  
existing = set(df_clean.columns).intersection(expected_cols)
```

```

if 'gender' in existing:
    t1 = len(df_clean)
    df_clean = df_clean[df_clean['gender'].isin(['男', '女'])]
    t2 = len(df_clean)
    print(f"删除性别异常值后: {t1 - t2} 行被移除")

if 'email' in existing:
    t1 = len(df_clean)
    df_clean = df_clean[df_clean['email'].apply(is_valid_email)]
    t2 = len(df_clean)
    print(f"删除邮箱异常值后: {t1 - t2} 行被移除")

if 'phone_number' in existing:
    t1 = len(df_clean)
    df_clean = df_clean[df_clean['phone_number'].apply(is_valid_phone)]
    t2 = len(df_clean)
    print(f"删除电话号码异常值后: {t1 - t2} 行被移除")

```

- 对数值型数据（包括购买记录中的数据）进行归一化处理，方便后续任务，代码如下所示：

```

def normalize_numeric_columns(df, columns):
    scaler = MinMaxScaler()
    df[columns] = scaler.fit_transform(df[columns])
    return df

def extract_purchase_features(df):
    """从 purchase_history 中提取 average_price 和 item_count"""
    avg_prices = []
    item_counts = []

    for record in df['purchase_history']:
        try:
            data = json.loads(record)
            avg_prices.append(data.get('average_price', np.nan))
            item_counts.append(len(data.get('items', [])))
        except Exception:
            avg_prices.append(np.nan)
            item_counts.append(np.nan)

    df['purchase_avg_price'] = avg_prices
    df['purchase_item_count'] = item_counts
    return df

# 处理登记时间
# 确保 registration_date 是 datetime 类型
df_clean['registration_date'] = pd.to_datetime(df_clean['registration_date'])
# 获取当前日期
now = pd.to_datetime("today")
# 计算注册天数（注册到现在经过了多少天）
df_clean['registration_days'] = (now -

```

```
df_clean['registration_date']).dt.days

# 选择要归一化的字段
numeric_columns = ['registration_days', 'age', 'income', 'credit_score',
'purchase_avg_price', 'purchase_item_count']
# 归一化数值字段
df_clean = normalize_numeric_columns(df_clean, numeric_columns)
print(f"✅ 数据已归一化")

df_clean['is_active_num'] = df_clean['is_active'].astype(int)
```

- 对10G数据集进行预处理的结果如下：

缺失值统计：

	字段	缺失值数量	缺失值比例 (%)	文件
0	id	0	0.0	part-00000
1	timestamp	0	0.0	part-00000
2	user_name	0	0.0	part-00000
3	chinese_name	0	0.0	part-00000
4	email	0	0.0	part-00000
5	age	0	0.0	part-00000
6	income	0	0.0	part-00000
7	gender	0	0.0	part-00000
8	country	0	0.0	part-00000
9	chinese_address	0	0.0	part-00000
10	purchase_history	0	0.0	part-00000
11	is_active	0	0.0	part-00000
12	registration_date	0	0.0	part-00000
13	credit_score	0	0.0	part-00000
14	phone_number	0	0.0	part-00000

处理空缺值后剩余记录数：12500000

图11 遍历处理10G数据集的每个子文件的缺失值展示

```
data > 🟢 10G_dataresult_missing_summary.csv
1  字段,缺失值数量,缺失值比例 (%),文件
2  |
```

图12 整个10G数据集的缺失值展示

```
处理字段: age
age 过滤范围: 0.00 ~ 143.00
删除异常值后: 0 行被移除
异常值比例: 0.0%

处理字段: income
income 过滤范围: 0.00 ~ 1500500.00
删除异常值后: 0 行被移除
异常值比例: 0.0%

处理字段: credit_score
credit_score 过滤范围: 23.00 ~ 1127.00
删除异常值后: 0 行被移除
异常值比例: 0.0%
删除性别异常值后: 492750 行被移除
删除邮箱异常值后: 0 行被移除
删除电话号码异常值后: 0 行被移除
```

图13 遍历处理10G数据集的每个子文件的异常值展示

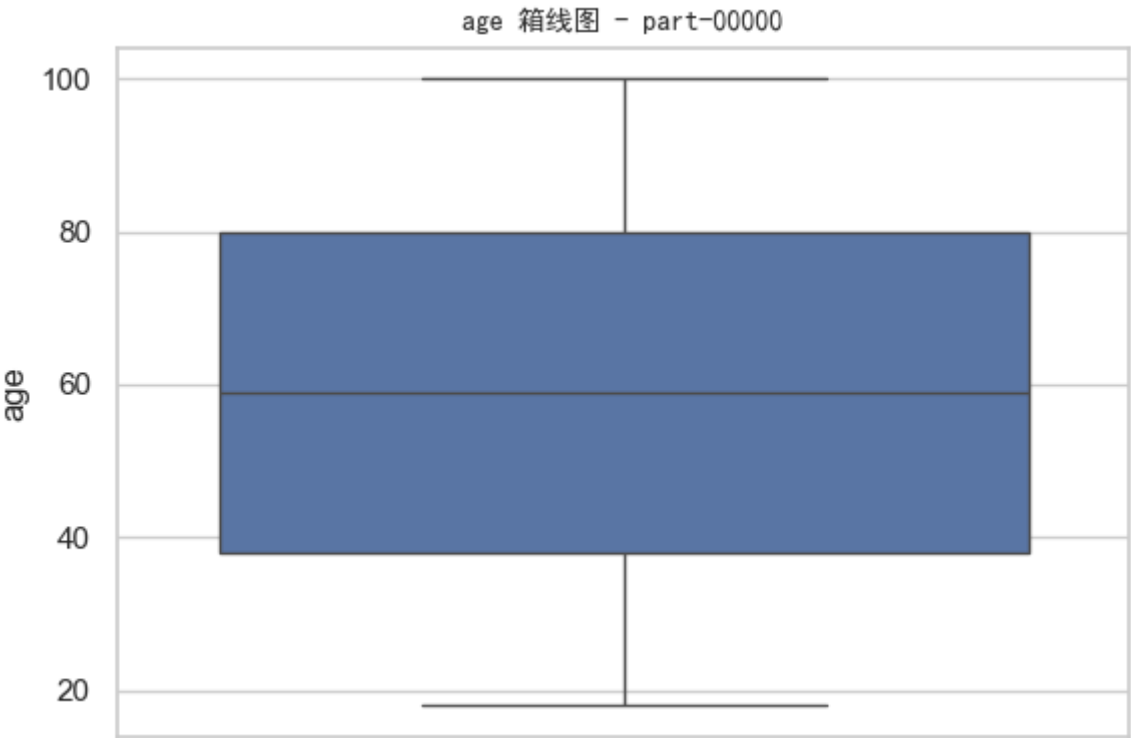


图14 遍历处理10G数据集的每个子文件的箱线图展示--年龄

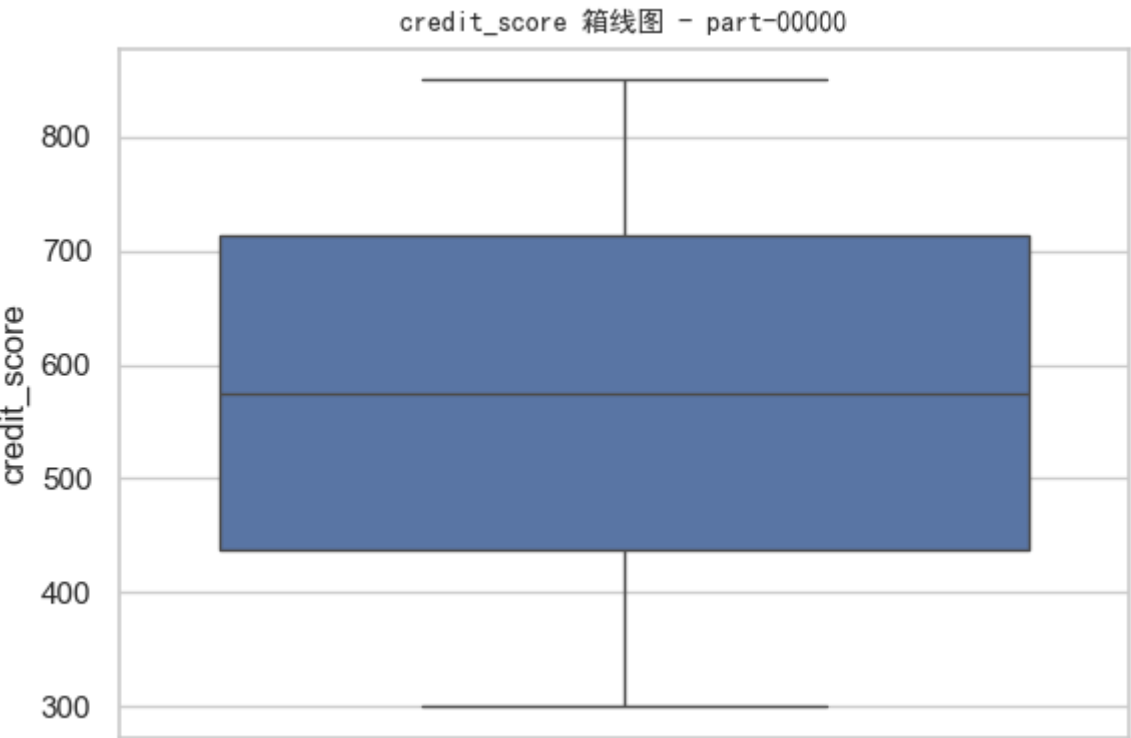


图15 遍历处理10G数据集的每个子文件的箱线图展示--信用得分

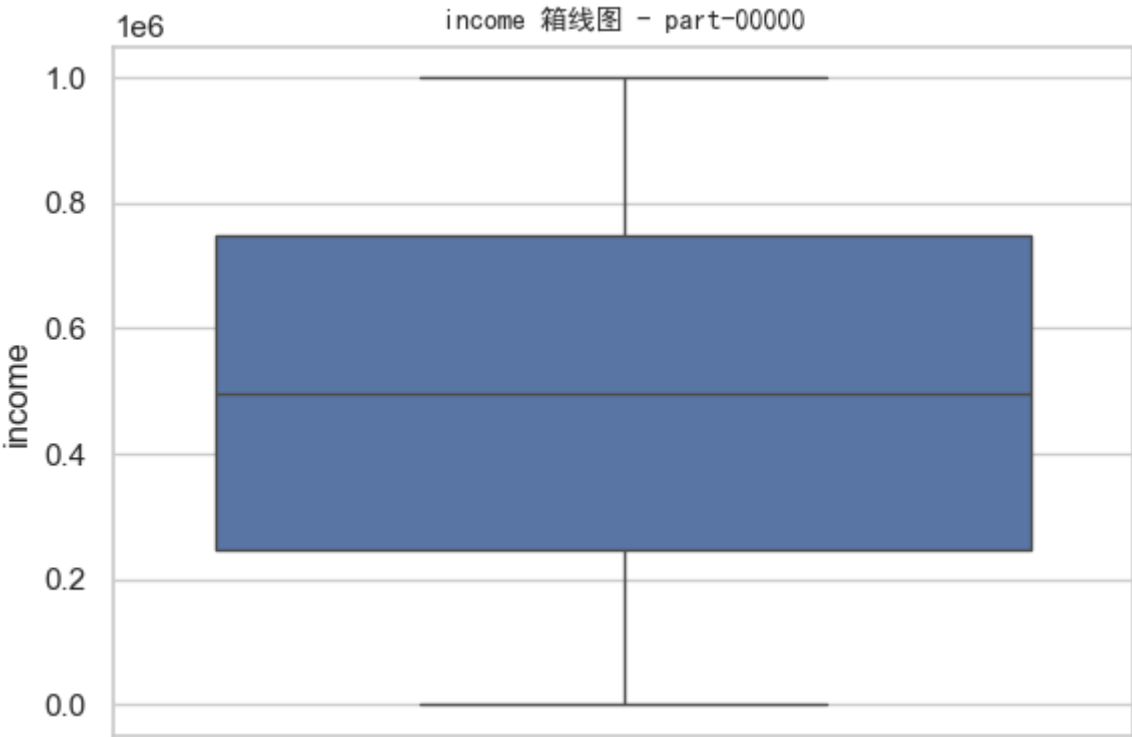


图16 遍历处理10G数据集的每个子文件的箱线图展示--收入

```
data > 10G_dataresult_outlier_summary.csv
```

1	文件, 字段, 异常值数量, 异常值比例 (%)
2	part-00000, gender, 492750, 3.941999999999997
3	part-00001, gender, 507500, 4.06
4	part-00002, gender, 494750, 3.957999999999997
5	part-00003, gender, 491250, 3.93
6	part-00004, gender, 494000, 3.952
7	part-00005, gender, 505875, 4.047
8	part-00006, gender, 500625, 4.005
9	part-00007, gender, 516250, 4.130000000000001

图17 整个10G数据集的异常值展示

结束时间: 2025-04-10 16:47:24
总耗时: 21.686404418945312 分钟

图18 运行时间截图

- 对30G数据集进行预处理的结果如下：

缺失值统计：

	字段	缺失值数量	缺失值比例 (%)	文件
0	id	0	0.0	part-00000
1	timestamp	0	0.0	part-00000
2	user_name	0	0.0	part-00000
3	chinese_name	0	0.0	part-00000
4	email	0	0.0	part-00000
5	age	0	0.0	part-00000
6	income	0	0.0	part-00000
7	gender	0	0.0	part-00000
8	country	0	0.0	part-00000
9	chinese_address	0	0.0	part-00000
10	purchase_history	0	0.0	part-00000
11	is_active	0	0.0	part-00000
12	registration_date	0	0.0	part-00000
13	credit_score	0	0.0	part-00000
14	phone_number	0	0.0	part-00000

处理空缺值后剩余记录数：18750000

图19 遍历处理30G数据集的每个子文件的缺失值展示

```
data > 30G_dataresult_missing_summary.csv
1  字段,缺失值数量,缺失值比例 (%),文件
2
```

图20 整个30G数据集的缺失值展示

处理字段: age
age 过滤范围: 0.00 ~ 143.00
删除异常值后: 0 行被移除
异常值比例: 0.0%

处理字段: income
income 过滤范围: 0.00 ~ 1500500.00
删除异常值后: 0 行被移除
异常值比例: 0.0%

处理字段: credit_score
credit_score 过滤范围: 23.00 ~ 1127.00
删除异常值后: 0 行被移除
异常值比例: 0.0%
删除性别异常值后: 739211 行被移除
删除邮箱异常值后: 0 行被移除
删除电话号码异常值后: 0 行被移除

图21 遍历处理30G数据集的每个子文件的异常值展示

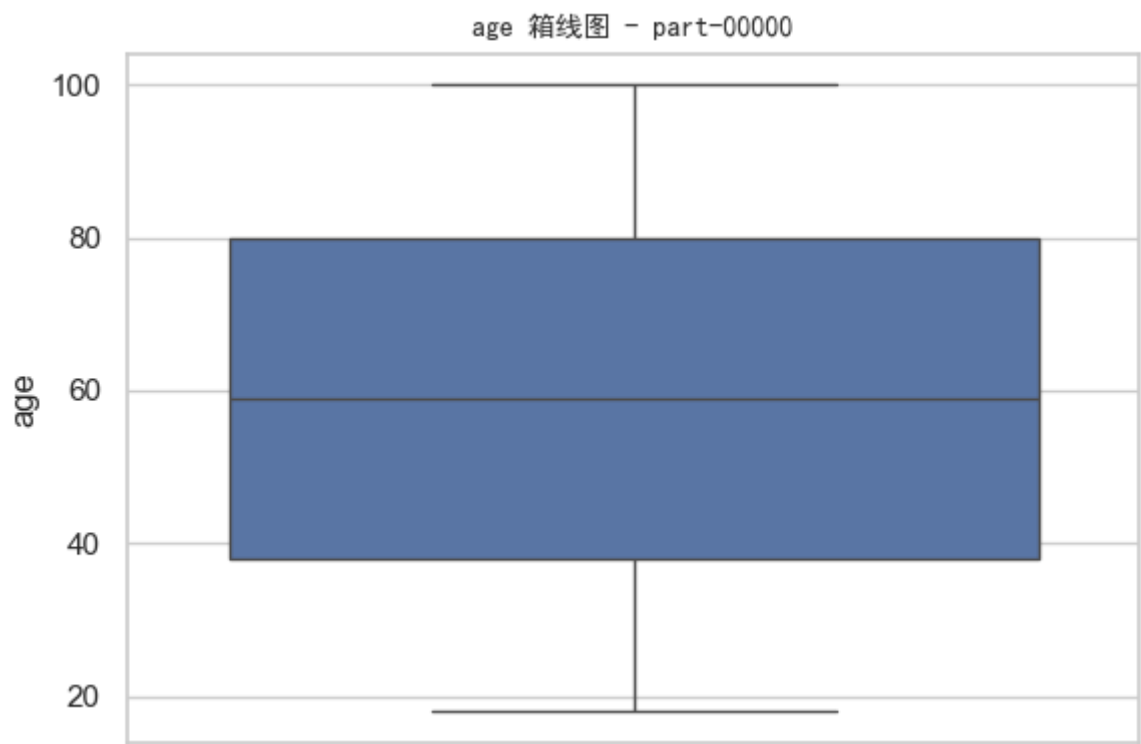


图22 遍历处理30G数据集的每个子文件的箱线图展示--年龄

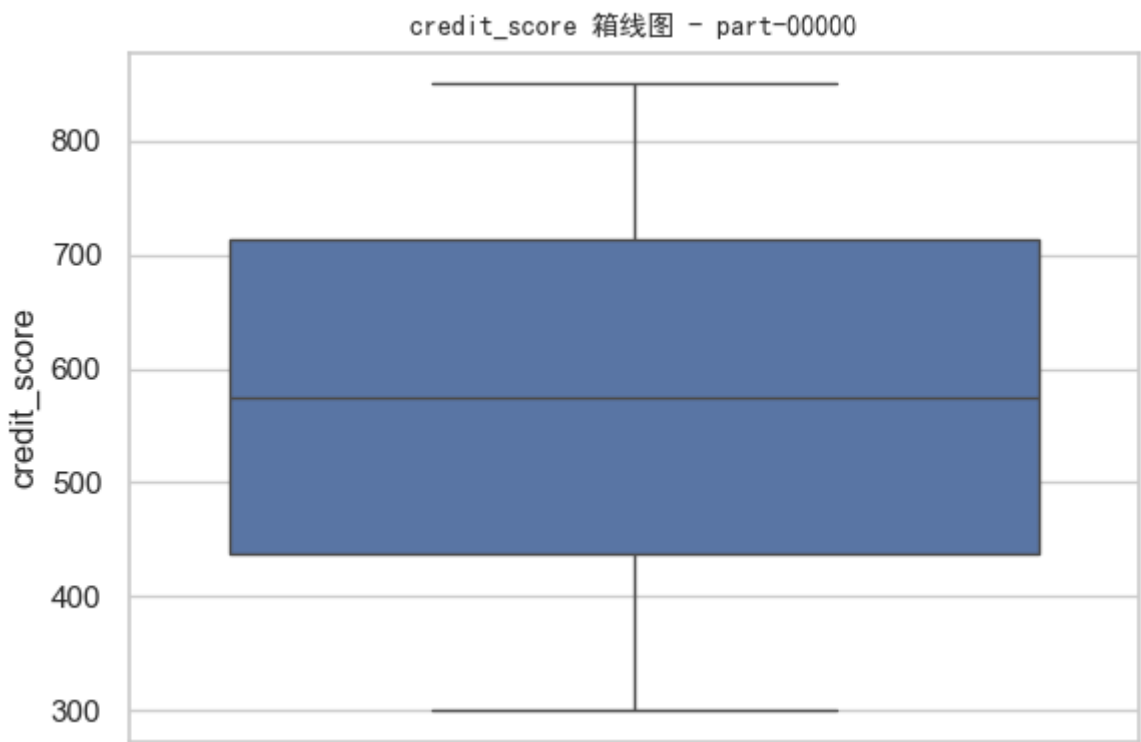


图23 遍历处理30G数据集的每个子文件的箱线图展示--信用得分

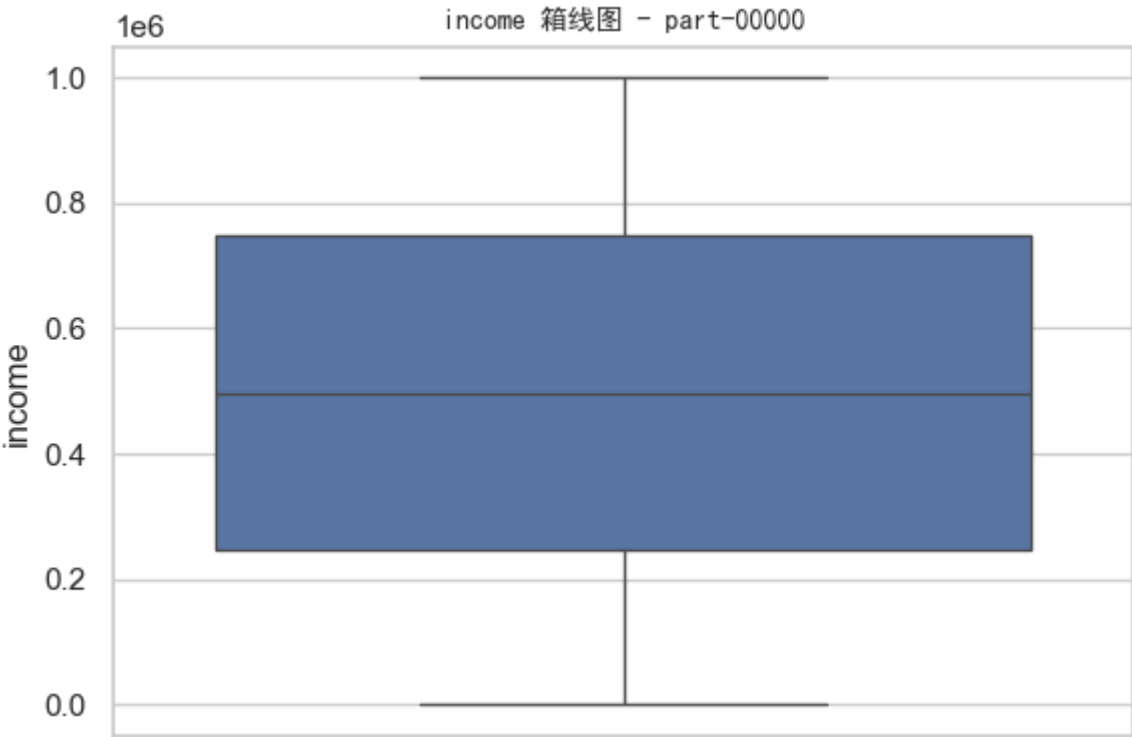


图24 遍历处理30G数据集的每个子文件的箱线图展示--收入

```
data > 30G_dataresult_outlier_summary.csv
1 文件,字段,异常值数量,异常值比例 (%)
2 part-00000,gender,739211,3.9424586666666666
3 part-00001,gender,761260,4.060053333333333
4 part-00002,gender,742132,3.9580373333333334
5 part-00003,gender,736875,3.93
6 part-00004,gender,740990,3.9519466666666667
7 part-00005,gender,758791,4.046885333333333
8 part-00006,gender,750849,4.0045280000000005
9 part-00007,gender,774428,4.1302826666666667
10 part-00008,gender,769870,4.105973333333333
11 part-00009,gender,759177,4.0489440000000005
12 part-00010,gender,738970,3.941173333333333
13 part-00011,gender,741216,3.953152
14 part-00012,gender,766905,4.09016
15 part-00013,gender,743259,3.964048
16 part-00014,gender,736495,3.927973333333333
17 part-00015,gender,737790,3.93488
```

图25 整个30G数据集的异常值展示

```
结束时间: 2025-04-10 18:02:12
总耗时: 64.34711194038391 分钟
```

图26 运行时间截图

3.识别潜在高价值用户

- 经过了数据可视化和预处理，数据具有较好的质量，可以进行后续任务——识别潜在高价值用户。为了识别潜在高价值用户，我们可以直接计算综合得分或者使用机器学习算法，如决策树、随机森林、支持向量机等。在这里，我选择直接计算综合得分的方法来识别潜在高价值用户。具体的计算方法如下面的代码所示：


```

#计算综合得分
df_clean['user_value_score'] = (
    0.25 * df_clean['income'] +
    0.20 * df_clean['credit_score'] +
    0.20 * df_clean['purchase_avg_price'] +
    0.15 * df_clean['purchase_item_count'] +
    0.10 * df_clean['is_active_num'] +
    0.10 * (1 - df_clean['registration_days']) # 注册越早分数越高
)

threshold = df_clean['user_value_score'].quantile(0.90)
high_value_users = df_clean[df_clean['user_value_score'] >=
threshold]
print(f"🌀 高价值用户数量: {len(high_value_users)}")

# 加入列表
all_high_value_users.append(high_value_users)

# 拼接所有高价值用户
final_high_value_users = pd.concat(all_high_value_users,
ignore_index=True)

# 保存到文件
output_path = 'data/' + info + 'high_value_users.parquet'
final_high_value_users.to_parquet(output_path, engine='pyarrow',
index=False)
print(f"✅ 所有高价值用户数据已保存到: {output_path}")

```

这段代码可以计算每个用户的综合得分，处理数据集中的每个文件，提取每个文件里面的高价值用户，并将所有高价值用户拼接在一起保存起来，得到整个数据集的高价值用户。其中，计算综合得分的参数也可以微调。

- 处理10G数据集的结果如下：




图27 归一化处理10G数据集的每个子文件以及查找高价值用户

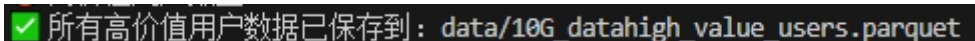


图28 10G数据集的所有高价值用户

- 处理30G数据集的结果如下：



图29 归一化处理30G数据集的每个子文件以及查找高价值用户

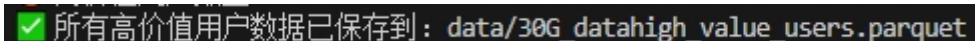


图30 30G数据集的所有高价值用户