

## 第四章 数据预处理

### 1. 计算题

用list、set、drop\_duplicate、corr、dataframe.equals 方法去除数据集文件meal的重复值

#### 方法一：使用 list 方法去重

```
In [ ]: import pandas as pd
import numpy as np

meal = pd.read_csv('meal.csv')
print('原始数据:')
print(meal)
print(f'\n原始数据行数: {len(meal)}')
```

```
In [ ]: new_list = []
for i in range(len(meal)):
    row = meal.iloc[i, 1:].tolist()
    if row not in new_list:
        new_list.append(row)

meal_list = pd.DataFrame(new_list, columns=meal.columns[1:])
print('list方法去重结果:')
print(meal_list)
print(f'去重后行数: {len(meal_list)}')
```

#### 方法二：使用 set 方法去重

```
In [ ]: meal_set = set()
index_list = []
for i in range(len(meal)):
    row = tuple(meal.iloc[i, 1:].tolist())
    if row not in meal_set:
        meal_set.add(row)
        index_list.append(i)

meal_set_df = meal.iloc[index_list].reset_index(drop=True)
print('set方法去重结果:')
print(meal_set_df)
print(f'去重后行数: {len(meal_set_df)}')
```

#### 方法三：使用 drop\_duplicates 方法去重

```
In [ ]: meal_drop = meal.drop_duplicates(subset=['meal_name', 'price', 'category', 'calorie'])
print('drop_duplicates方法去重结果:')
```

```
print(meal_drop)
print(f'去重后行数: {len(meal_drop)}')
```

## 方法四：使用 corr 相关性检测重复列

```
In [ ]: corr_matrix = meal[['meal_id', 'price', 'calories']].corr()
print('相关性矩阵:')
print(corr_matrix)
print('\n相关系数接近1的列可能存在重复信息')
```

## 方法五：使用 dataframe.equals 判断去重前后是否一致

```
In [ ]: meal_dedup = meal.drop_duplicates(subset=['meal_name', 'price', 'category', 'calori
print('原始数据与去重后数据是否相同:', meal.equals(meal_dedup))
print(f'原始数据行数: {len(meal)}, 去重后行数: {len(meal_dedup)}')
print('\n说明数据中存在重复值, 去重操作有效。')
```

## 2. 用拉格朗日方法对第三章习题数据「9月各部门的产品销售数量」进行插值处理

```
In [ ]: from scipy.interpolate import lagrange

sales = pd.read_excel('9月份各部门产品销售数量.xlsx')
print('原始数据:')
print(sales.head(10))
print(f'\n数据形状: {sales.shape}')
print('\n缺失值统计:')
print(sales.isnull().sum())
```

```
In [ ]: sales_missing = sales.copy()
sales_missing.loc[2, '销售部'] = np.nan
sales_missing.loc[5, '产品部'] = np.nan
sales_missing.loc[8, '市场部'] = np.nan
sales_missing.loc[12, '销售部'] = np.nan
sales_missing.loc[15, '产品部'] = np.nan

print('制造缺失值后的数据:')
print(sales_missing.head(16))
print('\n缺失值统计:')
print(sales_missing.isnull().sum())
```

```
In [ ]: def lagrange_interp(s, n, k=5):
    y = s[list(range(max(0, n-k), n)) + list(range(n+1, min(len(s), n+1+k)))]
    y = y[y.notnull()]
    return lagrange(y.index, y.values)(n)

for col in ['销售部', '产品部', '市场部']:
    for i in range(len(sales_missing)):
        if sales_missing[col].isnull().iloc[i]:
            sales_missing.loc[i, col] = lagrange_interp(sales_missing[col], i)
```

```
print('插值处理后的数据:')
print(sales_missing.head(16))
print('\n缺失值统计:')
print(sales_missing.isnull().sum())
```

```
In [ ]: print('原始值 vs 插值结果对比:')
print(f'销售部 index=2: 原始值={sales.loc[2, "销售部"]}, 插值={sales_missing.loc[2, "销售部"]}')
print(f'产品部 index=5: 原始值={sales.loc[5, "产品部"]}, 插值={sales_missing.loc[5, "产品部"]}')
print(f'市场部 index=8: 原始值={sales.loc[8, "市场部"]}, 插值={sales_missing.loc[8, "市场部"]}')
print(f'销售部 index=12: 原始值={sales.loc[12, "销售部"]}, 插值={sales_missing.loc[12, "销售部"]}')
print(f'产品部 index=15: 原始值={sales.loc[15, "产品部"]}, 插值={sales_missing.loc[15, "产品部"]}')
```

## 3. 完成第四章课后习题的操作题

### 3.1 数据清洗 - 处理缺失值

```
In [ ]: data = pd.DataFrame({
    '姓名': ['张三', '李四', '王五', '赵六', '钱七'],
    '语文': [85, np.nan, 92, 78, 88],
    '数学': [90, 85, np.nan, 82, 95],
    '英语': [78, 82, 88, np.nan, 90]
})
print('原始数据:')
print(data)
print('\n缺失值统计:')
print(data.isnull().sum())

data_filled = data.copy()
for col in ['语文', '数学', '英语']:
    data_filled[col].fillna(data_filled[col].mean(), inplace=True)

print('\n均值填充后的数据:')
print(data_filled)
```

### 3.2 数据清洗 - 处理异常值

```
In [ ]: np.random.seed(42)
data_outlier = pd.DataFrame({
    '销售额': np.append(np.random.normal(100, 10, 50), [200, 5, 250])
})

mean = data_outlier['销售额'].mean()
std = data_outlier['销售额'].std()

print(f'均值: {mean:.2f}')
print(f'标准差: {std:.2f}')
print(f'上限( $\mu+3\sigma$ ): {mean + 3*std:.2f}')
print(f'下限( $\mu-3\sigma$ ): {mean - 3*std:.2f}')

outliers = data_outlier[(data_outlier['销售额'] > mean + 3*std) | (data_outlier['销售额'] < mean - 3*std)]
print(f'\n异常值个数: {len(outliers)}')
```

```
print('异常值:')  
print(outliers)
```

### 3.3 数据标准化

```
In [ ]: sales_data = pd.read_excel('9月份各家电产品销售额.xlsx')  
print('原始销售额数据:')  
print(sales_data.head())  
  
min_val = sales_data['销售额'].min()  
max_val = sales_data['销售额'].max()  
sales_data['Min-Max标准化'] = (sales_data['销售额'] - min_val) / (max_val - min_val)  
  
mean_val = sales_data['销售额'].mean()  
std_val = sales_data['销售额'].std()  
sales_data['Z-Score标准化'] = (sales_data['销售额'] - mean_val) / std_val  
  
print('\n标准化后的数据:')  
print(sales_data[['家电名称', '销售额', 'Min-Max标准化', 'Z-Score标准化']])
```