实验 3-1 数据预处理

建议课时：30 分钟

一、 实验目的

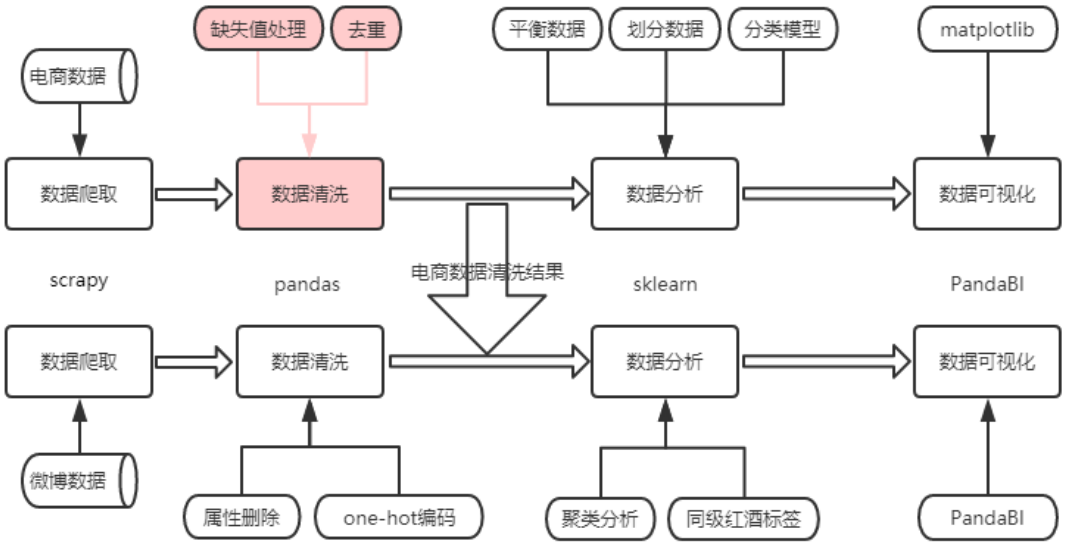
- 了解 pandas 的常用函数
- 了解数据清洗的基础方法

二、 实验环境

Python3 开发环境，第三方包有 pandas

三、 实验步骤

本节处理的内容有：



此小节主要讲述的是数据预处理。主要包含以下四个部分：

- 去重（去除重复数据）；
- 据观察，搜索结果中仍含有红酒杯，讲解红酒的书籍等其他商品数据，此部分将利用商品属性来进行判别，红酒有酒精度，特性，品类等特征，非红酒商品则没有那些属性，以此作为判断依据进行数据的删选；
- 据观察，平台上红酒的销售有瓶装，礼盒装，箱装等不同包装，即需要调整价格，且容量和包装这两个属性也并不可信，有些是总数的显示，有些是见瓶身，见包装的字样，唯一较为可靠的是销售平台的标题，此处将采

用自然语言处理中的词性分析解析得到红酒的数量。为简化后续同级红酒的比较，经统计，750ml 的红酒占多数，此处选择删选其他容量数据，仅提取 750ml 的红酒相关数据。

- 对无用的列或是数据较为繁杂不准确的列进行删选。

1. 处理数据格式

首先利用 json.loads 解决爬取数据中的中文编码问题：

```
import csv
import json
csvfile = open('电商红酒.csv', 'r')
reader = csv.reader(csvfile)#读取到的数据是将每行数据当做列表返回的
rows = []#用来存储解析后的没条数据
for row in reader:
    row_str = ",".join(row)#row为list类型需转为str，该数据变为字典型字符串
    row_dict = json.loads(row_str)#
    #将每行数据中嵌套字典拆开存储到列表中
    newdict = {}
    for k in row_dict:
        if type(row_dict[k]) == str:#将键值对赋给新字典
            newdict[k] = row_dict[k]
        elif type(row_dict[k]) == dict:#若存在嵌套字典，将该字典中的key和value作为属性和属性值
            newdict.update(row_dict[k])
    rows.append(newdict)
```

然后利用 pandas 将数据转化为 DataFrame 形式，并保存为 csv 文件

```
import pandas as pd
df = pd.DataFrame(rows)#将字典型数组转为DataFrame形式
#将DataFrame 写入到 csv 文件
df.to_csv("wine_df_shop.csv",encoding="utf-8_sig", index = False)
```

最后查看数据量和数据维度

```
print(df.shape)
print(df.columns.values)
```

运行结果如下：

```
(33734, 266)
['CT数' 'ISBN' 'id' 'keyword' 'name' 'price' 'shop_id' 'shop_name' 'sku_id'
 'url' '丛书名' '中图法分类号' '主体材质' '主色调' '主袋' '主要材质成分' '主题词' '书写' '产区' '产品产地'
 '产品净重(kg)' '产品净重（kg）' '产品包装' '产品包装尺寸' '产品包装尺寸（cm）' '产品包装重量（kg）' '产品品牌'
 '产品型号' '产品尺寸(mm)' '产品尺寸（cm）' '产品尺寸（mm）' '产品尺寸（长*宽*高）' '产品尺寸（长*宽）CM：'
 '产品尺寸（长x宽x高mm）' '产品展开尺寸（cm）' '产品展开尺寸（cm）' '产品展示尺寸（cm）' '产品展示尺寸（cm）'
 '产品执行标准' '产品承重量（kg）' '产品承重（kg）' '产品标准号' '产品类别' '产品重量（kg）' '产地' '人群' '介质'
 '作者' '使用尺寸（cm）' '保温时间' '保质期' '保质期（年）' '储存方法' '光源个数' '光源类型' '冷冻容积（L）'
 '冷冻能力(kg/24h)' '冷藏容积（L）' '净含量' '净含量（kg）' '净含量（mL/g）' '净含量（mL）' '净尺寸(mm)'
 '净重(KG)' '净重量（kg）' '出版时间' '出版社' '分类' '制冷方式' '功效' '功率(W)' '功能' '功能位'
 '功能特性' '包装' '包装尺寸（cm）' '包装礼盒' '包装规格' '单片净含量（mL）' '印刷时间' '印次' '厚度' '原产国'
 '原产地' '原料' '原材料产地' '原麦汁浓度' '口味' '口感' '口述' '品牌' '品相' '品类' '商品包装'
 '商品承重量（kg）' '商品承重（kg）' '固定方式' '国产/进口' '国别' '图案' '图片形式' '地区' '场合' '场景'
 '型号' '填充物' '壁饰题材' '外壳材质' '外文名' '头数' '套装数量' '妆效' '字幕语言' '字数' '存储方法' '安装方式'
 '审图号' '容积（L）' '容量' '容量(L)' '容量（ml）' '导演' '尺寸' '尺寸（cm）' '尺寸（mm）'
 '尺寸（长宽高、杯口、杯底直径、容量）' '层位' '层数' '层架材质' '工艺' '幅数' '年代' '年份' '度数' '开关类型'
 '开口方式' '开本' '开门方式' '形状' '成色' '手机袋' '拉链暗袋' '指示灯' '接口类型' '控制方式' '摄影' '摆放位置'
 '摆放方式' '收纳尺寸（cm）' '整理' '斜挎带' '方案' '是否360°出水' '是否为特殊用途化妆品' '是否冷热水' '是否双面'
 '是否可折叠' '是否含糖' '是否含龙头' '是否带光源' '是否有扶手' '是否有盖' '是否有茶格' '最大写入速度(MB/s)'
 '最大读取速度(MB/s)' '有无钢圈' '朗读' '材质' '材质（杯体、盖子、密封圈、杯口环、茶网）' '材质（锅体、锅盖、锅底）'
 '校对' '款式' '正文语言' '毯子尺寸' '水槽样式' '注释' '温区' '演员' '灯口' '灯罩材质' '灯身材质'
 '照射面积(㎡)' '照明功率(W)' '片长' '版次' '特性' '独特设计' '甜度' '生产许可证' '生产许可证号' '用纸' '用途'
 '电压' '电压(V)' '电脑插袋' '画芯材质' '省份' '碟数' '笔尖材质' '笔杆材质' '等级' '类别' '类型' '系列'
 '组合' '组合形式' '组套件数' '组立尺寸（cm）' '绘者' '编撰' '编者' '肩带长度（cm）' '能否放A4文件' '能效等级'
 '色号颜色' '节日' '花器风格' '花型' '著者' '葡萄品种' '表盘材质' '表面处理' '表面工艺' '袖长' '装裱形式' '规格'
 '规格（cm）' '规格（直径、内径、每个隔层之间的距离）' '规格（长宽高）' '认证型号' '证书' '证件袋' '译者' '读者对象'
```

可见，总共有 33734 条数据，266 维属性；查看一下保存的文件，观察数据发现有很多属性（如 CT 数、ISBN、面料、页数、题材等）不属于本主题“红酒”研究属性，可能是以关键词爬取数据时获取的有关红酒书籍、醒酒器、酒具等方面的数据，这些噪声数据应予以删除，后续会介绍具体讲解。

目前关键词字段存储的是"品牌 红酒"（如“拉菲 红酒”，“LAFITE 红酒”），处理流程如下：

- 首先去掉关键词中的“红酒”字符,只保留红酒品牌
- 其次对于红酒品牌存在中英文两种表述的，统一归为形如“拉菲/LAFITE”形式

```
# 处理红酒名称（统一规范中文/英文的格式）
#去keyword中"红酒"字符
df['keyword']=df['keyword'].str.split('红酒').str[0]

#整理红酒品牌
brand = pd.read_csv("./data/wine_brand.csv",header=None,encoding="utf-8")
brands =[]
for k1,k2 in zip(list(brand[0]),list(brand[1])):
    if pd.isnull(k2):
        brands.append(k1)
    else:
        brands.append(k1+"/"+k2)

# 品牌替换
def modify_keywords(w, lists):
    for b in lists:
        if w.strip() in b:
            return b
    return w.strip()


df['keyword'] = df['keyword'].apply(modify_keywords, lists =brands)
df['keyword']=df['keyword'].str.lower()#转化为小写

print(df["keyword"].head(10))
```

运行结果如下：

```
0                泸州老窖
1                  美绚
2          通化/tonghua
3        长城/greatwall
4        长城/greatwall
5    干露/concha y toro
6           拉菲/lafite
7           山图/shantu
8           山图/shantu
9           山图/shantu
Name: keyword, dtype: object
```

2. 去重（去除重复数据）

由于红酒品牌搜索时有中文和英文，可能会存在相同数据，根据 url 属性来判断两条数据是否重复。

```python
#若爬取数据有相同的url，则认为是相同数据，只保留一条

#数据是否有相同行，若有返回true，否则False
if df.duplicated(subset=["url"], keep=False).any():
    print("存在重复数据")
    df = df.drop_duplicates(subset=["url"], keep="first")
else:
    print("不存在重复数据")
print(df.shape)
```

```
不存在重复数据
(33734, 266)
```

可见，当前数据中无重复数据，接下来处理删除非红酒类商品。

3. 过滤非红酒类商品

观察数据，发现数据中还存在红酒杯、醒酒器黄酒等噪声数据，而“甜度”是红酒的一般特性，若无该属性则很大程度上是非红酒数据，可能酒具以及黄酒白酒等数据

```python
df = df.dropna(subset=["甜度"])
print(df.shape)
```

```
(18734, 266)
```

当前还剩余 18734 条数据

4. 过滤非 750ml 红酒数据并处理价格

将删除“产品重量”属性，因为包装不同，礼盒或者整箱产品包装中带有赠品，酒杯酒具等物品，质量变化数据不可靠，难以清洗。

统一红酒容量为 750ml，并计算单瓶价格；继而可以删除标题，包装和容量三个属性列。

据观察，包装和容量的形容多样，单位也多样，标题的可信度反而相对较高，所以我们从标题中匹配 750ml 的红酒数据。

操作步骤如下：

- 保留标题中含有 750 字样的数据；
- 含有特殊的瓶数转换成相应的数字，转第 4 步；

  若匹配到两次及两次以上，则排除该数据。
- 利用 jieba 分词工具提取标题中的数字，仅保留限定数值之内的数据：

  若 750 后面的数字存在，则认为是瓶数，转第 4 步；

  若 750 前面的数字存在，则认为是瓶数，转第 4 步；

  若前后都没有限定数值之内的数字出现，判断是单瓶，转第 4 步；

  若 750 前后面的数字都存在，则排除该数据。
- 重新计算价格。

具体代码如下：

```
# 提取瓶数和重新计算价格
#红酒商品描述中有含有的数字主要有三种年份、容量和瓶数，容量固定为750，主要识别瓶数

# 抽出含750字样的数据
df = df[df["name"].str.contains("750")]
print(df.shape)

#处理价格
import jieba.posseg as pseg

def get_numbers(words, num_type = "int"):
    """返回string中的数字
    Args:
        words: 字符串
        num_type: 获取字符串中float型数字还是int型数字
    """
    nums = []
    for w,p in pseg.cut(words): #jieba词性标注不能将"750ml*6"中的6识别为数字，换一种方式
        if num_type == "int":
            try: nums.append(int(w))
            except:continue
        if num_type == "float":
            try: nums.append(float(w))
            except:continue
    return nums
```

此处函数考虑 float 类型是为了函数复用，因为之后的处理过程仍需要该函数。

```
for i, name in zip(df_index, df["name"]):
    bottles = [int(wine_bottles[k])for k in wine_bottles if name.find(k) != -1]    #
    if len(bottles) == 1:
        df['price'].loc[i] = float('%.2f' % (df['price'].loc[i]/bottles[0]))
        continue
    elif len(bottles)>1:
        df = df.drop(index = i,axis = 0)
        continue

    numbers = get_numbers(name,"int")        #bottles==0，用jieba分词提取红酒标题中的数字
    if 750 not in numbers:        #选取是含有750的样本，但结巴分词可能将750切成其他组合词，识别不出750这个数字，删除
        df = df.drop(index = i,axis = 0)
        continue
    numbers = [n for n in get_numbers(name,"int") if n in [750,1,2,3,4,5,6,8,12]]        #提取标题中特定数字
    if len(numbers) == numbers.count(750):         #只有750数字,认为是单瓶不处理
        continue
    if numbers.count(750) > 1:        #存在多个数字且750个数有多个，删除
            df = df.drop(index = i,axis = 0)
            continue
    if numbers.index(750) == 0:        #存在多个数字，750只有一个，且750后面有数字,更改
        df['price'].loc[i] = float('%.2f' % (df['price'].loc[i]/numbers[1]))
    elif numbers.index(750) == -1:        #存在多个数字，750只有一个，且750前面有数字，更改
        df['price'].loc[i] = float('%.2f' % (df['price'].loc[i]/numbers[-2]))
    else:
         df = df.drop(index = i,axis = 0)    #存在多个数字，750只有一个，且750前面后面都有数字，删除
```

5. 过滤无用属性

删除缺失值比较多的属性列，查看每列缺失值的数目

```
print("每列缺失值个数:")
print(df.isnull().sum())#统计每列的缺失值个数
```

观察每列的缺失值数目，采用删除缺失值超过 80%的列的策略

```
#删除列数据，thresh作用：保留至少有2329（11648*20%）个非 NA 数的列
df = df.dropna(axis=1,thresh=2329)
# 再删除["id","shop_id","shop_name","sku_id","url"]
df.drop(["id","shop_id","shop_name","sku_id","url"],axis=1,inplace=True)
```