传统文本分析实现网购评论的对象分类

In [1]:

```
import re
import jieba
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import f1_score,accuracy_score,recall_score
```

数据读取与预处理

In [2]:

```
df = pd.read_csv("online_shopping_10_cats.csv")[:60000]
df.head()
```

Out[2]:

	cat	label	review
0	书籍	1	做父母一定要有刘墉这样的心态，不断地学习，不断地进步，不断地给自己补充新鲜血液，让自己保持...
1	书籍	1	作者真有英国人严谨的风格，提出观点、进行论述论证，尽管本人对物理学了解不深，但是仍然能感受到...
2	书籍	1	作者长篇大论借用详细报告数据处理工作和计算结果支持其新观点。为什么荷兰曾经县有欧洲最高的生产...
3	书籍	1	作者在战几时之前用了＂拥抱＂令人叫绝．日本如果没有战败，就有会有美军的占领，没胡官僚主义的延...
4	书籍	1	作者在少年时即喜阅读，能看出他精读了无数经典，因而他有一个庞大的内心世界。他的作品最难能可贵...

In [3]:

```
#使用re正则提取中文并用jieba分词提取词语语料
extract_chinese = re.compile(r'[\u4e00-\u9fa5]+')
chinese_corpus_raw = df['review'].tolist()
chinese_corpus_raw
df['chinese_corpus']=[jieba.lcut("".join(extract_chinese.findall(str(corpus)))) for corpus in
df.head()
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\23176\AppData\Local\Temp\jieba.cache
Loading model cost 0.606 seconds.
Prefix dict has been built successfully.
```

Out[3]:

	cat	label	review	chinese_corpus
0	书籍	1	做父母一定要有刘墉这样的心态，不断地学习，不断地进步，不断地给自己补充新鲜血液，让自己保持...	[做, 父母, 一定, 要, 有, 刘墉, 这样, 的, 心态, 不断, 地, 学习, 不断...
1	书籍	1	作者真有英国人严谨的风格，提出观点、进行论述论证，尽管本人对物理学了解不深，但是仍然能感受到...	[作者, 真有, 英国人, 严谨, 的, 风格, 提出, 观点, 进行, 论述, 论证, 尽...
2	书籍	1	作者长篇大论借用详细报告数据处理工作和计算结果支持其新观点。为什么荷兰曾经县有欧洲最高的生产...	[作者, 长篇大论, 借用, 详细, 报告, 数据处理, 工作, 和, 计算结果, 支持, ...
3	书籍	1	作者在战几时之前用了＂拥抱＂令人叫绝．日本如果没有战败，就有会有美军的占领，没胡官僚主义的延...	[作者, 在, 战, 几时, 之前, 用, 了, 拥抱, 令人, 叫绝, 日本, 如果, 没...
4	书籍	1	作者在少年时即喜阅读，能看出他精读了无数经典，因而他有一个庞大的内心世界。他的作品最难能可贵...	[作者, 在, 少年, 时即, 喜, 阅读, 能, 看出, 他, 精读, 了, 无数, 经典...

In [9]:

```
#将每条评论分词后整合到一个列表中，将每个词用空格隔开放入一个列表中
words_list = []
corpus = []
for corpu in df['chinese_corpus'].tolist():
    words_list.append(corpu)
    corpus.append(' '.join(corpu))
words_list[0]
```

Out[9]:

```
['做',
 '父母',
 '一定',
 '要',
 '有',
 '刘墉',
 '这样',
 '的',
 '心态',
 '不断',
 '地',
 '学习',
 '不断',
 '地',
 '进步',
 '不断',
 '地',
 '给',
```

In [5]:

```
#每个词用空格分割
corpus
```

Out[5]:

```
['做 父母 一定 要 有 刘墉 这样 的 心态 不断 地 学习 不断 地 进步 不断 地 给 自己 补充 新鲜血液 让 自己 保持 一颗 年轻 的 心 我 想 这 是 他 能 很 好 的 和 孩子 沟通 的 一个 重要 因素 读 刘墉 的 文章 总能 让 我 看到 一个 快乐 的 平易近人 的 父亲 他 始终 站 在 和 孩子 同样 的 高度 给 孩子 创造 着 一个 充满 爱 和 自由 的 生活 环境 很 喜欢 刘墉 在 字里行间 流露出 的 做 父母 的 那种 小 狡黠 让 人 总是 忍俊不禁 父母 和 子女 之间 有时候 也 是 一种 战斗 武力 争斗 过于 低级 了 智力 较量 才 更 有 趣味 所以 做 父母 的 得 加把劲 了 老 思想 老 观念 注定 会 一败涂地 生命 不息 学习 不止 家庭教育 真的 是 乐在其中',
 '作者 真有 英国人 严谨 的 风格 提出 观点 进行 论述 论证 尽管 本人 对 物理学 了解 不深 但是 仍然 能 感受 到 真理 的 火花 整本书 的 结构 颇 有 特点 从 当时 本 书写于 八十年代 流行 的 计算机 话题 引入 再用 数学 物理学 宇宙学 做 必要 的 铺垫 这些 内容 占据 了 大部分 篇幅 最后 回到 关键问题 电脑 能 不能 代替 人脑 和 现在 流行 的 观点 相反 作者 认为 人 的 某种 洞察 是 不能 被 算法 模拟 的 也许 作者 想 说 人 的 灵魂 是 无可取代 的',
 '作者 长篇大论 借用 详细 报告 数据处理 工作 和 计算结果 支持 其新 观点 为什么 荷兰 曾经 县有 欧洲 最高 的 生产率 为什么 在 文化 上 有着 深刻 纽带 关系 的 中国 和 日本 却 在 经济 发展 上 有着 极大 的 差异 为什么 英国 的 北美 殖民地 造就 了 经济 强大 的 美国 而 西班牙 的 北美 殖民 却 造就 了 落后 的 墨西哥 很 有 价值 但 不 包
```

In [6]:

```
#构建类别与编号的转换字典，并将类别转成编号
class2idx ={'书籍':0, '平板':1, '手机':2, '水果':3, '洗发水':4, '热水器':5, '蒙牛':6, '衣服':7,
idx2class = {idx:class_ for class_,idx in class2idx.items()}
class_idx =[class2idx[calss_] for calss_ in df['cat'].values]
class2idx
```

Out[6]:

```
{'书籍': 0,
 '平板': 1,
 '手机': 2,
 '水果': 3,
 '洗发水': 4,
 '热水器': 5,
 '蒙牛': 6,
 '衣服': 7,
 '计算机': 8,
 '酒店': 9}
```

TF-IDF和Bag-of-words实现向量化

In [7]:

```
#sklearn实现TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vec=TfidfVectorizer(max_features=1000)
tfidf_matrix=tfidf_vec.fit_transform(corpus)
```

In [8]:

```
#sklearn实现bagofword
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=1000)
tfidf_matrix1 = vectorizer.fit_transform(corpus)
```

In [9]:

```
#划分训练集验证集
train_x1, valid_x1, train_y1, valid_y1 = train_test_split(tfidf_matrix, class_idx, random_state
```

In [10]:

```
#划分训练集验证集
train_x2, valid_x2, train_y2, valid_y2 = train_test_split(tfidf_matrix1, class_idx, random_stat
```

使用SVM实现网购评论的对象分类

In [11]:

```
#定义SVM分类器并进行分类，评价指标设为准确率，召回率和F1值
def SVM(train_x,train_y,valid_x,valid_y):
    svm = SVC(max_iter=500)
    svm.fit(train_x, train_y)
    val_pred = svm.predict(valid_x)
    f1 = f1_score(valid_y, val_pred, average='macro')
    Accuracy_score = accuracy_score(valid_y, val_pred)
    Recall_score = recall_score(valid_y, val_pred,average='macro')
    print(f'Accuracy_score:{Accuracy_score}')
    print(f'Recall_score:{Recall_score}')
    print(f'f1_score:{f1}')
```

In [12]:

```
#使用SVM分类器并进行分类(TF-IDF)
SVM(train_x1,train_y1, valid_x1, valid_y1)
```

```
D:\Anaconda3\envs\bd\lib\site-packages\sklearn\svm\_base.py:289: ConvergenceWarning:
Solver terminated early (max_iter=500).  Consider pre-processing your data with Stan
dardScaler or MinMaxScaler.
  ConvergenceWarning,

Accuracy_score:0.7315833333333334
Recall_score:0.7237494564062985
f1_score:0.7507279936063987
```

In [13]:

```
#使用SVM分类器并进行分类(Bag-of-words)
SVM(train_x2,train_y2, valid_x2, valid_y2)
```

```
D:\Anaconda3\envs\bd\lib\site-packages\sklearn\svm\_base.py:289: ConvergenceWarning:
Solver terminated early (max_iter=500).  Consider pre-processing your data with Stan
dardScaler or MinMaxScaler.
  ConvergenceWarning,

Accuracy_score:0.6163333333333333
Recall_score:0.6450487458510076
f1_score:0.6550754410041175
```