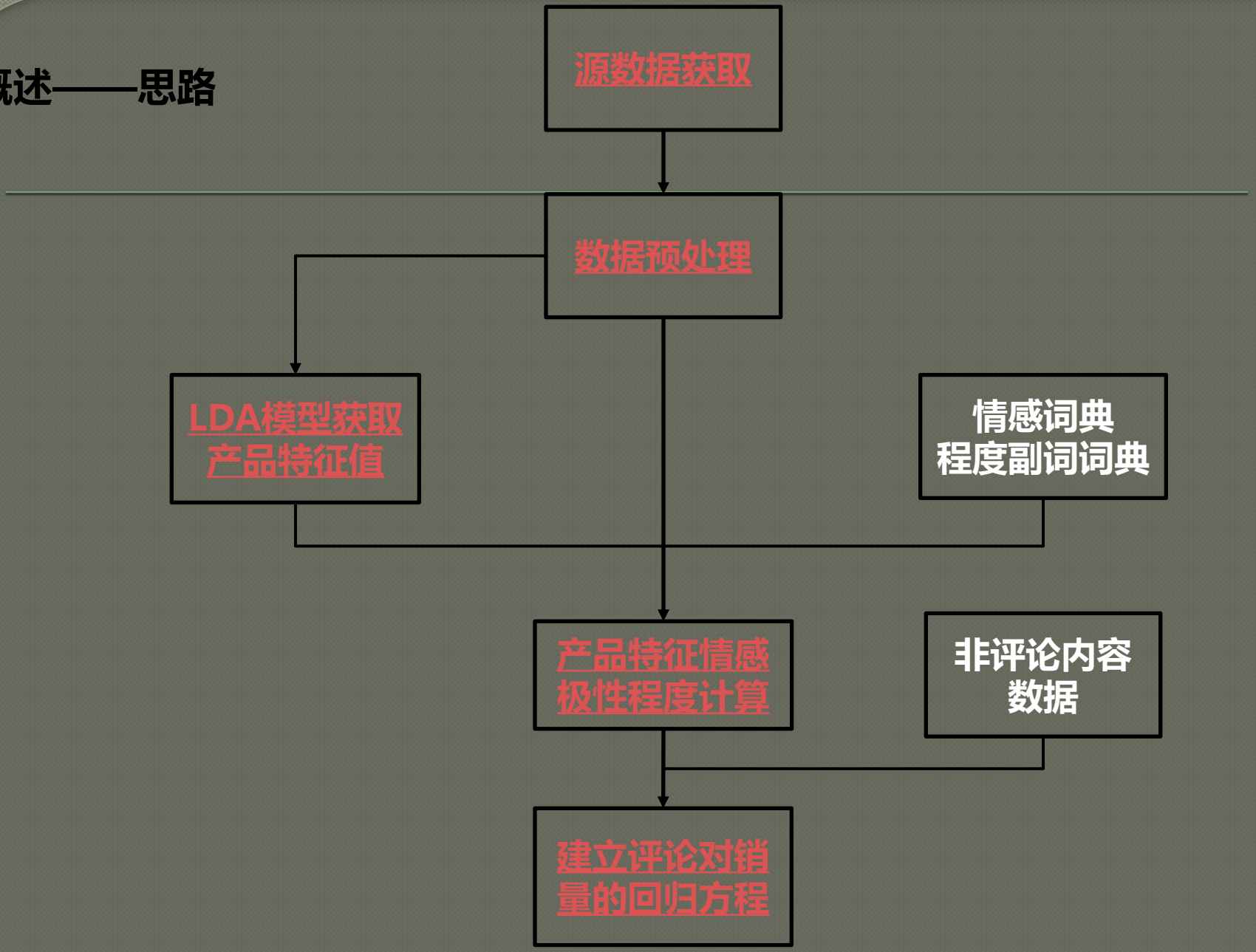
基于评论挖掘对商品销量的影响

——以某电商手机销售为例

概述——思路


源数据获取
数据预处理
LDA模型获取
产品特征值
情感词典
程度副词词典
产品特征情感
极性程度计算
非评论内容
数据
建立评论对销
量的回归方程

源数据获取

利用爬虫软件于2015年4月3日在亚马逊网（Amazon.com）上爬取了手机销售排行榜上前100名的产品信息。

主要包括：销售排名、产品名称、产品市场价格、产品现价、折扣、评论数量、评论平均星级以及每条评论的评论内容。

源数据获取

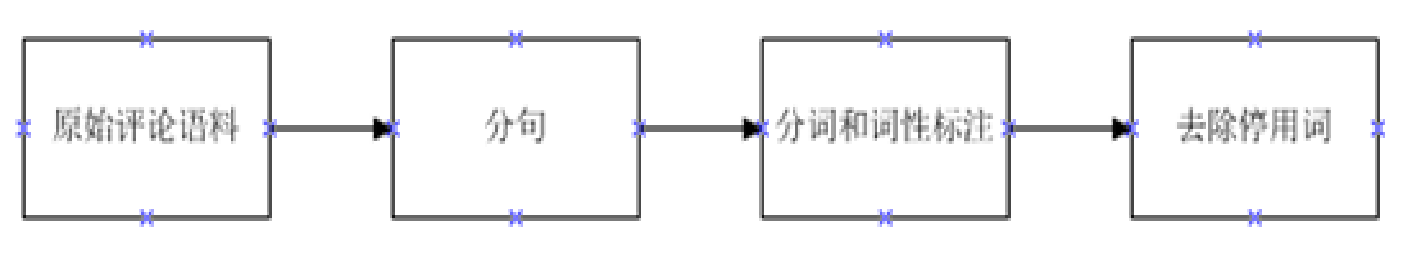
对采集的数据进行了初步的清理，删除了评论内容小于10的数据，最后获得91款手机数据，并按照原排名排序，下图是相应的描述性统计：

	平均	中位数	最小值	最大值
平均星数	4.130769	4.1	3.2	4.7
评论数量	328.9121	208	12	2326
市场价	1868.67	1299	169	5999
现价	1137.227	807	99	4983
优惠价格	761.0551	440.8	1	3279
折扣	0.383242	0.4	0	0.8

评论总数量约为3万条，足够我们进行下一步的特征抽取工作。

返回

评论数据预处理



分句：每条评论可能由多个句子组成，每一句话所谈到的内容或者产品特征均不相同。如果以每条评论为单位来进行产品特征评论语句来分类容易产生混淆。分句用Python语言及其扩展包实现。

例：“这款手机质量好，就是价钱有点贵。”
对每一条评论语句进行分句处理，这样上面例句就将分为“这款手机质量好，”和“就是价钱有点贵。”两个句子，分别可以很好的表示“质量”与“价钱”这两个特征。

评论数据预处理

分词与词性标注：无论是产品的特征词还是情感观点词都需要通过分词从连续的句子中分离出来，而这些往往都是名词和形容词，所以分词之后对词性的标注将有利于我们识别这些词，这为之后的文本处理工作奠定了数据基础。

与英文分词不同，中文词与词之间不能够严格的按照空格来区分，所以中文的分词工作需要利用某些方法来进行。本文采用python语言的jieba扩展包来分词。同时导入用户自定义词典，即存在于自定义词典中的词语不会被错误的分割，增加分词的准确性。

例：“这款手机质量好，”

通过jieba分词和词性标注过后得到结果为：“这/r 款/q 手机/n 质量/n 好/a ，/w” 像“手机”“质量”都被标注为名词，“好”被标注为形容词。

评论数据预处理

去除停用词：文本中一些介词、量词、助词、标点符号等对文本研究无意义的词，需要剔除，所以我们还需要对这些评论语料进行停用词过滤和标点符号过滤。停用词和标点符号的过滤可以采用根据停用词表，用Python语言编写过滤程序，取出停用词。

例：“这款手机质量好，”

在经过分词和词性标注后，对其进行停用词过滤，其结果为：“手机/n 质量/n 好/a”。指示代词“这”量词“款”和标点符号“，”就被过滤掉，留下的是可能成为产品特征和情感观点的词。

返回

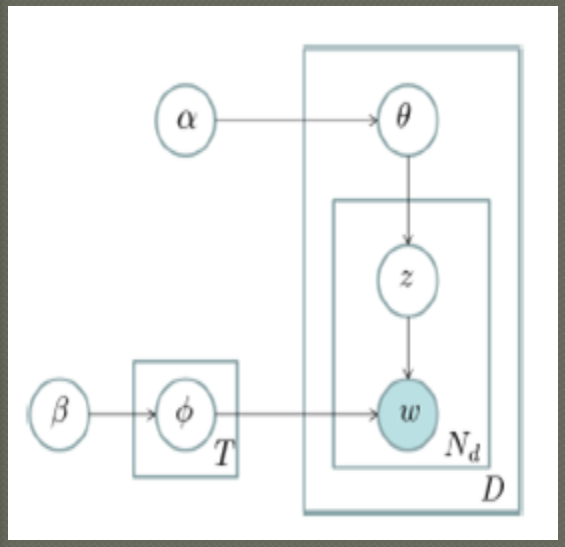
LDA模型获取产品特征值

LDA模型：

- ✓ 一种非监督机器学习技术，可以用来识别大规模文档集（document collection）或语料库（corpus）中潜藏的主题信息。
- ✓ 采用了词袋（bag of words）的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。
- ✓ 每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

LDA模型获取产品特征值

对于每一篇文档，LDA模型的生成步骤如下：



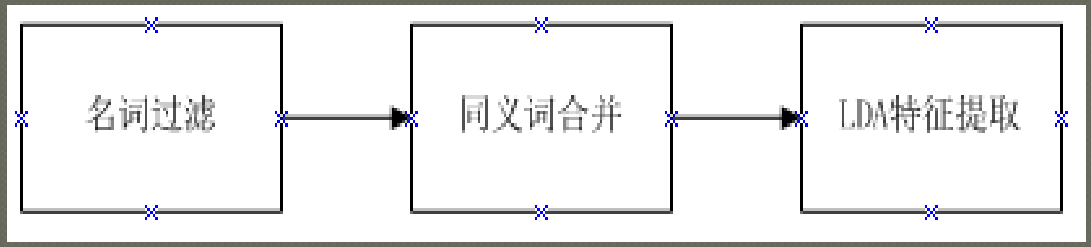
1. 首先选择一个N，N代表的是这篇文档的单词总数，N服从泊松分布，即N~Poisson(ξ)。
2. 选择θ，θ是一个列向量，代表的是这篇文档中每个主题发生的概率。θ服从狄利特雷分布，即θ~Dirichlet(α)分布，α是Dirichlet分布的参数。
3. 对于N各单词中的每一个：
4. 选择该单词属于的主题Zn，Zn服从Multinomial(θ)多项分布。
5. 选择Wn，Wn是根据有条件的概率分布p(Wn| Zn, β)产生的。β是一个K X V维的矩阵，其中β_{ij}= P(w_i= 1 | z_j= 1)（I = 1, 2, 3, …,K; j = 1,2,3,…,V）表示的是在主题z_j下单词w_i的生成概率。

$$p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta)$$

LDA模型获取产品特征值

通过对评论语料进行预处理，使得语料信息不再是杂乱无章、无规律可循的，因而为我们接下来用91款手机所有的评论信息作为评论语料，找到消费者在购买手机时所重点关注的产品特征。

基于LDA模型的产品特征提取的基本流程：



LDA模型获取产品特征值

名词过滤：产品特征大多数为名词，所以还需要在预处理语料的基础上剔除掉其他词，只留下名词，这样再一次缩小产品特征词的提取范围。

并不是评论语料中所有的名词都可以作为产品的特征词，一些名词往往不是产品特征，例如“时间”、“地点”、“人物”、“东西”等。为了得到能够较为准确的表达产品特征的候选词，我们需要对上述名词语料进行名词过滤处理。使用过滤规则对名词进行名词过滤处理是常用的方法。

LDA模型获取产品特征值

过滤规则	例子
（1）专有名词（包括时间、地点、人名等）	三月、九点、北京、上海、爸爸、妈妈
（2）常见的商品品牌名词	Iphone、三星、诺基亚

过滤规则（1）和（2）中的专有名词可以通过词性标签过滤掉，使用jieba分词二级词性标注后，可以对一些特有名词进行特定的标注。这样通过名词过滤规则可以将大部分出现频率较高但并不是产品特征的名词过滤掉。

例：人名使用标签nr表示，方位词使用标签f表示，地名使用标签ns表示，机构团体使用标签nt表示，字母专名使用nx表示，其他专属名词用nz表示。

LDA模型获取产品特征值

同义词合并：中文对某一特征可以用多种方式表达，例如“价钱”、“价格”、“价位”表达的都是一个意思。在网络评论中，消费者往往都有自己的语言习惯，不会使用同一的表达方式，从而导致某些表达方式因为出现频率过高被选出来而其他的则不能被选出来。为了解决这个问题，在LDA模型提取特征词之前将表达同一个意思的不同表达词汇统一用一个词来表示。

LDA模型获取产品特征值

```
FeatureCombine()
begin
    for word in reviews
        if word in (for syn in synonym)
            word = syn[0]
end
```

reviews：处理过的评论语料

synonym：整个同义词词林

syn表示某个词汇的所有表达方式。

遍历reviews中所有的词汇，对每一个词汇word在synonym中每一个syn中查询，找到相同的则用该syn中第一个表达方式替换掉原word。这样大部分的同义词都用一个表达方式表示出来了。

LDA模型获取产品特征值

LDA模型特征提取：我们需要通过得到每个主题下词汇的概率分布P（w| z = j），w表示词汇，z表示主题；并且通过设定一个阀值ε，只有大于这个阀值的词才能成为产品特征，即I = {Wji|Pwji≥ε}。

所以应用LDA模型进行特征词的提取，我们需要确定的参数有：

①LDA模型中的超参数，即主题分布和词汇分布的Dirichlet先验分布中的超参数α和β；

②LDA模型中的主题数目K；

③每个主题下特征词的候选阀值ε。

本文取α = 50 / K，β = 0.01，K = 8，阀值ε= 0.05

LDA模型获取产品特征值

使用python拓展包中的LDA模块进行特征词提取。具体步骤如下：

1）以评论语料中所有词汇最为LDA模型的词典

2）使用上述词典对所有评论语料转换为LDA模型语料库

3）使用LDA模型对语料库进行训练

4）得到每个主题下词汇的概率分布

5）使用阀值筛选出合适的词作为产品特征词

```
FeatureExtract()
begin
    dic=corpora.Dictionary(reviews)//step1
    corpus=[dic.doc2bow(text)for text in posWords]//step2
    lda = models.LdaModel(corpus, id2word=dic, num_topics=8, α= 50 / 8,β=
0.01)//step3
    ldaout=lda.print_topics(8,0.05)//step4,5
end
```

LDA模型获取产品特征值

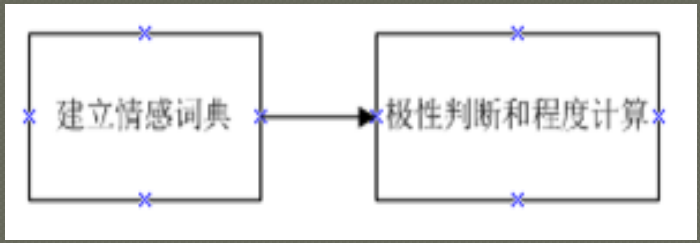
最后得到的产品特征如下表所示，总计42个产品特征词：

0.6847437	屏幕	0.1866161	像素	0.0774262	充电器
0.5175504	电池	0.1821743	手感	0.0714475	内存
0.5114914	价格	0.1641411	功能	0.0673808	听筒
0.4594722	性价比	0.1535693	电话	0.0648542	贴膜
0.3977095	质量	0.1485616	摄像头	0.063997	电量
0.3126718	声音	0.1396821	发货	0.0623619	流量
0.2889515	包装	0.1227769	分辨率	0.0608212	机身
0.283428	软件	0.1204811	待机时间	0.0601843	性能
0.2743404	耳机	0.1118809	拍照	0.0589477	运行速度
0.2607854	物流	0.1054465	发票	0.0558255	操作
0.236151	系统	0.1040438	信号	0.0555499	网速
0.2201349	速度	0.1013437	售后	0.0545191	字体
0.2193151	游戏	0.0871869	配置		
0.2123876	客服	0.0819272	开机		
0.2071563	外观	0.0784527	视频		

返回

情感极性判断与程度计算

利用情感词典进行情感极性和极性程度的计算大致步骤如下图：



建立情感词典：首先我们需要建立情感极性词典，也就是正面情感词典和负面情感词典，其中包含了一些表示情感极性的词，例如“好”、“漂亮”、“差”、“烂”等词。

情感极性判断与程度计算

在得到了情感极性词典之后只能对语句进行情感极性的判断，要计算极性程度还需要另外一些词典，其中最重要的就是副词词典。

副词词典包含的主要是否定副词和程度副词，否定副词可以在一定程度上提高情感词典对语句极性判断的准确率。

例：“质量不好”，在使用了否定副词词典后结果就为负向；
另外通过程度副词比如“较”、“极”、“稍微”、“有点”，并给不同程度的副词赋予不同的权值，可以在情感极性基础上计算极性程度。

情感极性判断与程度计算

完整的情感词典：

词典名称	词典内容	例子
Positive.txt	正向情感词	好、漂亮
Negative.txt	负向情感词	差、烂
Inversedict.txt	否定副词	不、没
mostdict.txt	“最”级别程度词	绝对、之极
Verydict.txt	“很”级别程度词	分外、很是
Moredict.txt	“较”级别程度词	较为、越发
Ishdict.txt	“稍”级别程度词	略为、稍微
Insufficientdict.txt	“欠”级别程度词	不怎么、半点

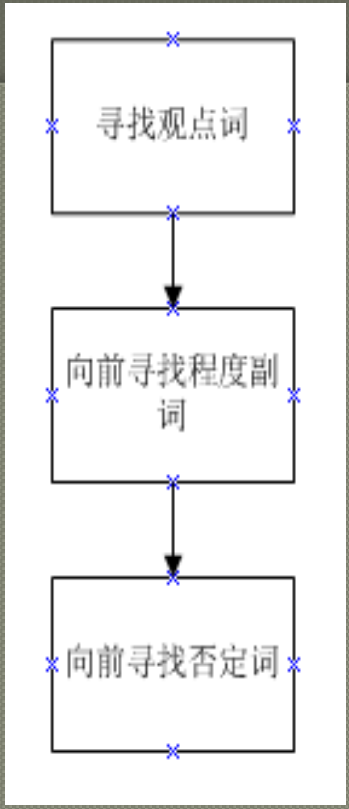
情感极性判断与程度计算

对情感词典赋权值：有了情感词典和程度词词典，在进行程度计算之前还需要对不同程度词赋予不同的权值。

情感及程度	权值
Positive	1
Negative	-1
Inversedict	-1
mostdict	2
Verydict	1.5
Moredict	1.25
Ishdict	0.5
Insufficientdict	0.25

情感极性判断与程度计算

极性程度计算：对含有特征词的语句进行以下极性程度计算，并以均值代表该产品特征的极性程度



正向：score = 1
负向：score = -1

Score = score * weight

偶数：score=score
奇数：score=-score

情感极性判断与程度计算

对抓取的91款手机评论语料分别进行上述操作，得到消费者对每款手机每个产品特征的情感极性和极性程度（正数表示正向情感程度，负数表示负向情感程度）。以其中一款手机为例，我们最后得到的数据为：

屏幕评分	-0.0416667	像素评分	0.0534091	充电器评分	0.0357143
电池评分	0.0714286	手感评分	0.7142857	内存评分	0.1029412
价格评分	0.3525641	功能评分	0.2692308	听筒评分	0
性价比评分	0.4407895	电话评分	0.625	贴膜评分	-0.4027778
质量评分	0.47	摄像头评分	0.4642857	电量评分	-0.1
声音评分	-0.1428571	发货评分	0.45	流量评分	-0.5
包装评分	0.3095238	分辨率评分	0.55	机身评分	0.1
软件评分	-0.2380952	待机时间评分	1.1666667	性能评分	0.5882353
耳机评分	-0.6206897	拍照评分	0.3409091	运行速度评分	0.9166667
物流评分	0.5555556	发票评分	-0.3072917	操作评分	0.3333333
系统评分	0.8796296	信号评分	0.2	网速评分	0.7727273
速度评分	0.7659574	售后评分	-0.25	字体评分	-0.5
游戏评分	0.0277778	配置评分	0.15		
客服评分	0.5	开机评分	-0.25		
外观评分	0.8	视频评分	0.1111111		

返回

建立评论对销量的回归方程

模型设定： $Y = \alpha_0 + \alpha_1 x_{1i} + \alpha_2 x_{2i} + \alpha_3 x_{3i} + \ldots + \alpha_{46} x_{46i} + \mu_i$，
其中μ表示随机变量，下标i = 1,2… N，表示不同款手机

Y	ln(排名)	X_{16}	速度评分	X_{32}	配置评分	
X_1	ln(评论数量)	X_{17}	游戏评分	X_{33}	开机评分	
X_2	ln(平均星级)	X_{18}	客服评分	X_{34}	视频评分	
X_3	ln(现价)	X_{19}	外观评分	X_{35}	充电器评分	
X_4	折扣	X_{20}	像素评分	X_{36}	内存评分	
X_5	屏幕评分	X_{21}	手感评分	X_{37}	听筒评分	
X_6	电池评分	X_{22}	功能评分	X_{38}	贴膜评分	
X_7	价格评分	X_{23}	电话评分	X_{39}	电量评分	
X_8	性价比评分	X_{24}	摄像头评分	X_{40}	流量评分	
X_9	质量评分	X_{25}	发货评分	X_{41}	机身评分	
X_{10}	声音评分	X_{26}	分辨率评分	X_{42}	性能评分	
X_{11}	包装评分	X_{27}	待机时间评分	X_{43}	运行速度评分	
X_{12}	软件评分	X_{28}	拍照评分	X_{44}	操作评分	
X_{13}	耳机评分	X_{29}	发票评分	X_{45}	网速评分	
X_{14}	物流评分	X_{30}	信号评分	X_{46}	字体评分	
X_{15}	系统评分	X_{31}	售后评分			

模型结果

逐步回归：因为并不是所有的解释变量即产品特征都对销量有显著的影响，所以我们需要去掉不显著的解释变量。因此本文对该模型采用逐步回归方法，此方法可以选取对被解释变量显著影响的解释变量，剔除不显著的变量，并且可以在一定程度上解决解释变量之间的多重共线性问题。

模型结果

R-Square	0.7341			
变量名	Parameter Estimate		P	VIF
截距	Intercept	5.37585	<0.001	0
ln(评论数量	X_1	-0.18325	0.0323	2.15357
折扣	X_4	-0.88034	0.0683	1.29115
价格评分	X_7	-0.66208	0.0043	1.43542
质量评分	X_9	-1.41087	0.0487	1.22883
声音评分	X_{10}	-0.74448	0.0018	1.30495
软件评分	X_{12}	-1.0409	0.0019	1.63341
物流评分	X_{14}	-0.41591	0.0672	1.62505
客服评分	X_{18}	-0.41957	0.011	1.55188
功能评分	X_{22}	-0.70012	0.0005	1.85554
待机时间评分	X_{27}	-1.28309	<0.0001	1.57185
拍照评分	X_{28}	-0.35642	0.0147	1.45809
发票评分	X_{29}	-0.48806	0.0095	1.40921
贴膜评分	X_{38}	-0.15567	0.0851	1.49744
操作评分	X_{44}	-0.4332	0.0243	1.20253

R-Square=0.7341：模型拟合程度效果良好；

P值<0.1：最后筛选出来的解释变量对被解释变量显著影响；

方差膨胀因子（VIF）<10：解释变量之间没有存在明显的多重共线性。

模型结果

Ln(排名) = 5.37585 + -0.18325*ln(评论数量) + -0.88034*折扣 + -0.66208*价格评分 + -1.41087*质量评分 + -0.74448*声音评分 + -1.0409*软件评分 + -0.41592*物流评分 + 0.41957*客服评分 + -0.70012*功能评分 + -1.28309*待机时间评分 + -0.35642*拍照评分 + -0.48806*发票评分 + -0.15567*贴膜评分 + -0.4332*操作评分

- ✓ 各个解释变量的系数都为负数 ⟶ 评价越高排名越靠前，即销量越高。
- ✓ 能够显著影响销量的产品特征只有“价格”、“质量”、“声音”等12个特征
- ✓ 根据其系数大小可以判断出各个特征对销量影响的程度

返回

Python语言及其扩展包

- ✓ 一种面向对象、解释型的计算机程序设计语言，在1989年由Guido van Rossum发明。
- ✓ 语法明确并且很少出现有歧义的语法，使得python源代码具有更好的可读性。
- ✓ 完全面向对象的语言，任何的函数、字符串、数字、模块都是对象，并且完全支持重载、继承、多态等，增强了代码的重复利用率。
- ✓ 拓展性很强，提供了大量的API和工具方便程序员使用其他语言如C、C++来编写拓展模块，因此常被称为一种“胶水语言”。
- ✓ 开源性，拥有强大的外援包，这些扩展包往往提供一些特制的功能方便其他程序员使用，从而增加了开发效率，减少重复性劳动。

返回

jieba扩展包

这个拓展包可以实现对中文文本的分词以及词性标注工作，并提供三种模式：精确模式、全模式和搜索引擎模式。Jieba扩展包在处理中文文本时效果显著。

返回

谢谢！