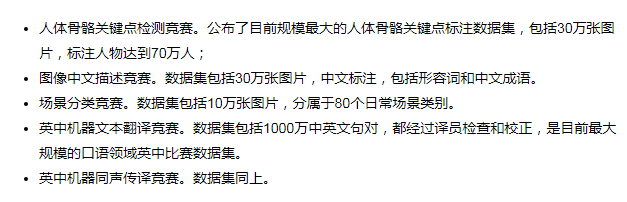
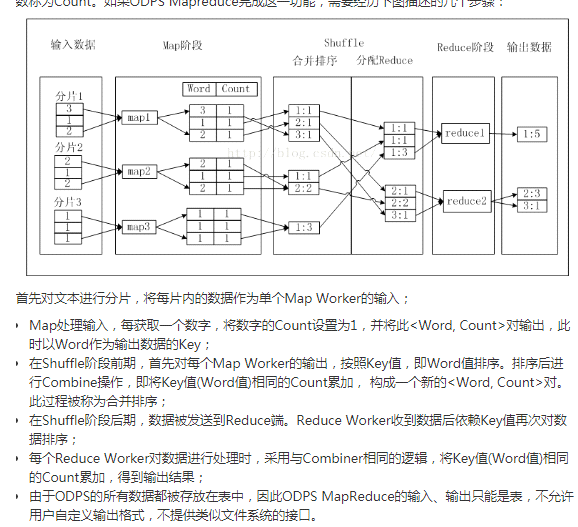
TIANCHI\_4362@aliyun-tianchi

4)8iWEhG



textrank



create table if not exists wcout1 (key string,cnt bigint);

打开主账户控制台，查看accesskey与value

Acecess **”**LTAIEDmUartK5Oak**”** **”MrKgfna9qs0bLQUJlOMm8COIiMbWK7”**

<https://yq.aliyun.com/articles/68227>问题解答

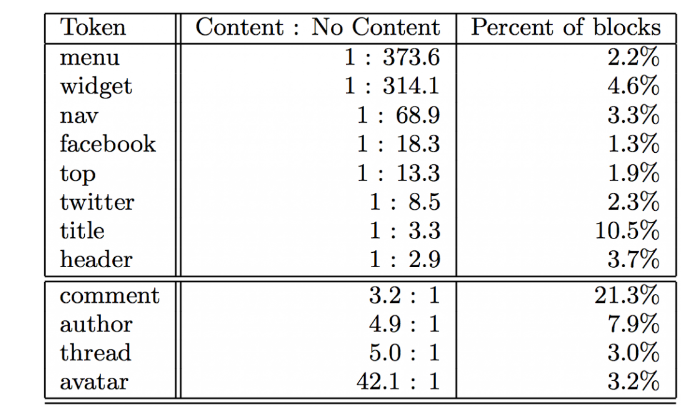
沙箱

使用自己的库产生故障

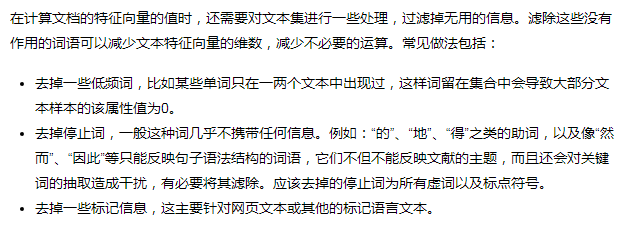
a).adl\_tianchi\_content\_risk\_training\_part1 b).adl\_tianchi\_content\_risk\_training\_part2 c).adl\_tianchi\_content\_risk\_training\_part3 d).adl\_tianchi\_content\_risk\_training\_part4 e).adl\_tianchi\_content\_risk\_training\_part5 f).adl\_tianchi\_content\_risk\_training\_part6 g).adl\_tianchi\_content\_risk\_training\_part7 h).adl\_tianchi\_content\_risk\_training\_part8 i).adl\_tianchi\_content\_risk\_training\_part9

adl\_tianchi\_content\_risk\_training\_part1 adl\_tianchi\_content\_risk\_training\_part2 adl\_tianchi\_content\_risk\_training\_part3 adl\_tianchi\_content\_risk\_training\_part4 adl\_tianchi\_content\_risk\_training\_part5 adl\_tianchi\_content\_risk\_training\_part6 adl\_tianchi\_content\_risk\_training\_part7 adl\_tianchi\_content\_risk\_training\_part8 adl\_tianchi\_content\_risk\_training\_part9

We also extract some additional structured information at this point: the title tag, the meta description tag, and H1/H2 tags.



<http://www.infoq.com/cn/articles/machine-learning-automatic-classification-of-text-data>



以文档频率为例，在特征选择过程中由于某些关键的词语低于了人为设定的阈值，所以会被直接忽视掉，而很多情况这部分词汇能包含较多的信息，对于分类的重要性比较大。怎么能够进一步理解这部分的信息，是急需要解决的问题。一个想法是找到这些使用频率比较低的词语相似的高频词，譬如在讨论“月亮”的古诗词中，包含了很多低频的同义词，如“玉兔”，“婵娟”等，如果我们能把这些低频的词语合并到一个维度，无疑是能够增强分类系统对文档的理解深度的。词向量这一概念能够有效地表示词语之间的相似性，适用于这种方法。

先介绍一下词向量的定义。一种最简单的词向量是one-hot representation，就是用一个很长的向量来表示一个词，向量的长度是词典D的大小N，向量的分量只有一个为1，其他全为0，1的位置对应该词在词典中的索引。这种词向量表示有一些缺点：容易受维数灾难的困扰。另一种词向量是Distributed Representation，它最早是Hinton于1986年提出来的，可以克服one-hot representation的上述缺点。其基本想法是：通过训练将某种语言中的每个词映射成一个固定长度的短向量。所有这些向量构成一个词向量空间，每个向量是该空间中的一个点，在这个空间上引入距离，就可以根据词之间的距离来判断它们之间的（词法、语义上的）相似性了。如何获取Distributed Representation的词向量呢？有很多不同的模型可以用来估计词向量，包括有名的LSA、LDA和神经网络算法。Word2Vec就是使用度比较广的一个神经网络算法实现的词向量计算工具。

s

kmeans聚类

召回率 r = a / (a + c) \* 100%

正确率 p = a / (a + b) \* 100%

F-测度值 F = (2 \* p \* r) /(p + r)

由于在分类结果中，对应每个类别都会有一个召回率和正确率，因此，可以根据每个类别的分类结果评价分类器的整体性能，通常方法有两种：微平均和宏平均。微平均是根据正确率和召回率计算公式直接计算出总得正确率和召回率值。宏平均是指首先计算出每个类别的正确率和召回率，然后对正确率和召回率分别取平均得到总的正确率和召回率。不难看出，宏平均平等对待每一个类别，所以它的值主要受到稀有类别的影响，而微平均平等考虑文档集中的每一个文档，所以它的值受到常见类别的影响比较大。

word2vec向量

一起词频统计，剔除低频词，特征表示，归一化，svm分类

词频需要测试集与训练集一起

Gambling 30324 0.027

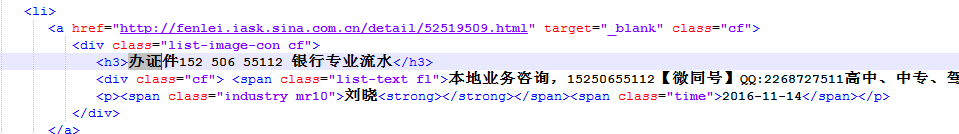
Fake card 59219 0.0538

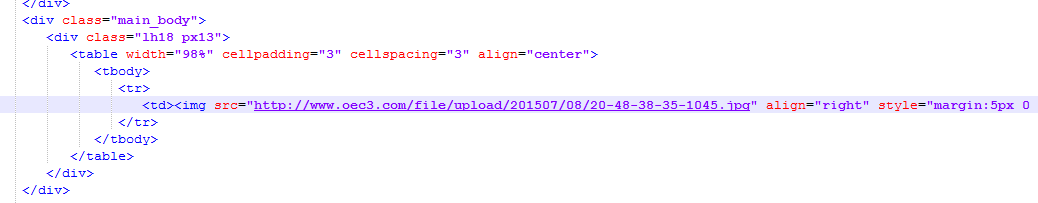
Sexy 23833 0.021

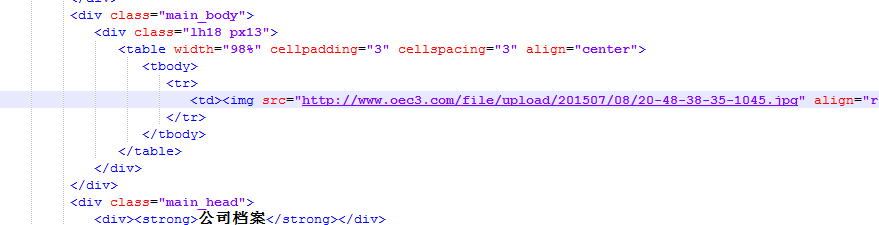
Normal 986623 0.89

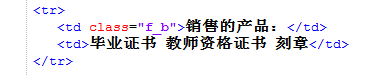
Total : 1099999

Fake card:





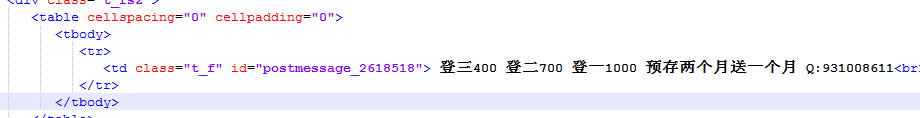


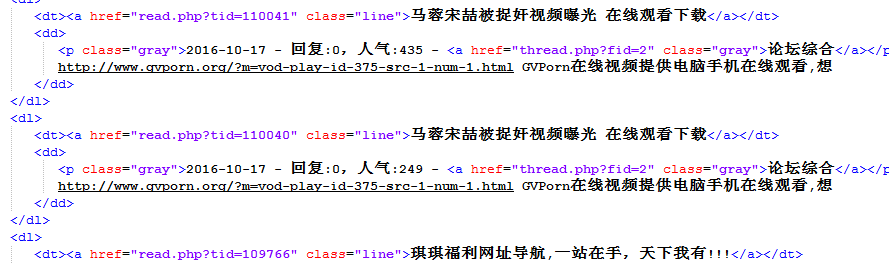


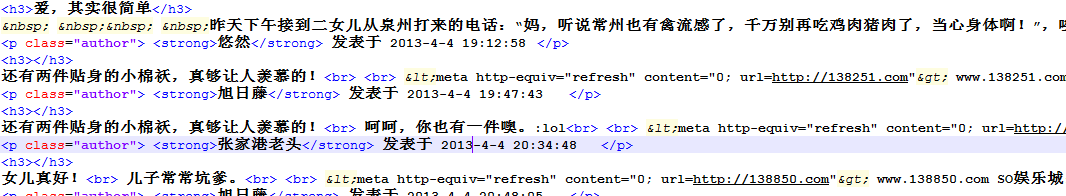
办假证，替考 word2vec

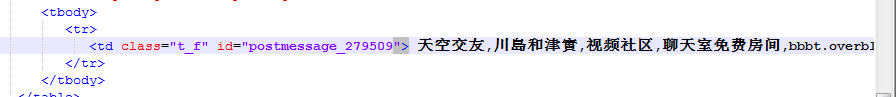
赌和投注

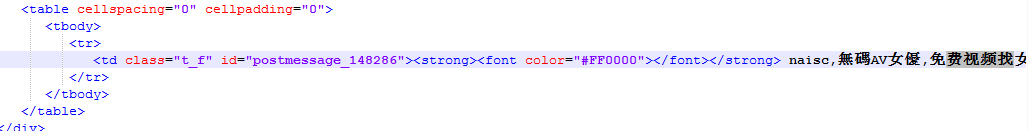






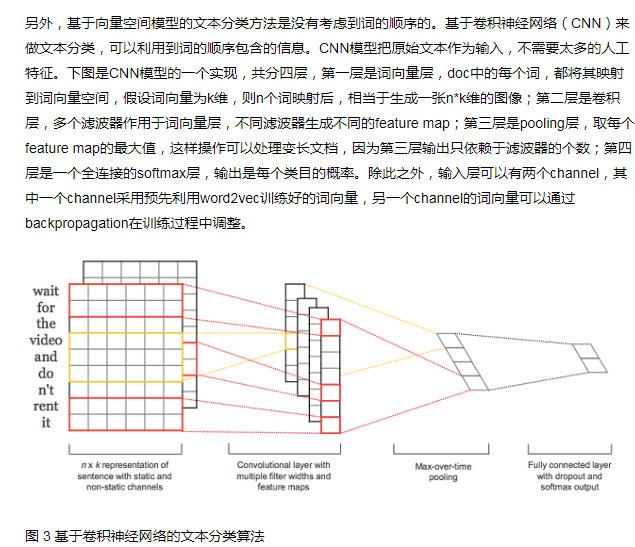


Sexy



如何计算两个文档的相似度二

文本经过分词程序分词后 ,首先去除停用词 ,合并数字和人名等词汇 ;然后统计词频



kmeans聚类

召回率 r = a / (a + c) \* 100%

正确率 p = a / (a + b) \* 100%

F-测度值 F = (2 \* p \* r) /(p + r)

由于在分类结果中，对应每个类别都会有一个召回率和正确率，因此，可以根据每个类别的分类结果评价分类器的整体性能，通常方法有两种：微平均和宏平均。微平均是根据正确率和召回率计算公式直接计算出总得正确率和召回率值。宏平均是指首先计算出每个类别的正确率和召回率，然后对正确率和召回率分别取平均得到总的正确率和召回率。不难看出，宏平均平等对待每一个类别，所以它的值主要受到稀有类别的影响，而微平均平等考虑文档集中的每一个文档，所以它的值受到常见类别的影响比较大。

word2vec向量

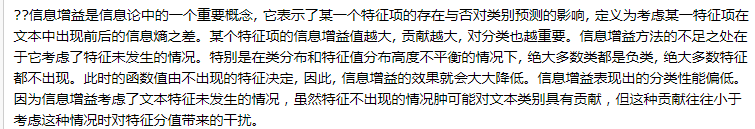
一起词频统计，剔除低频词，特征表示，归一化，svm分类

词频需要测试集与训练集一起

过滤掉一些词

基于向量空间模型的文本自动分类系统的研究与实现

基于归一化词频贝叶斯模型的文本分类方法

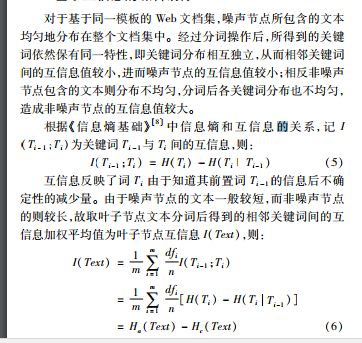


word2vec 聚类，pca降维，LDA,TF-idf

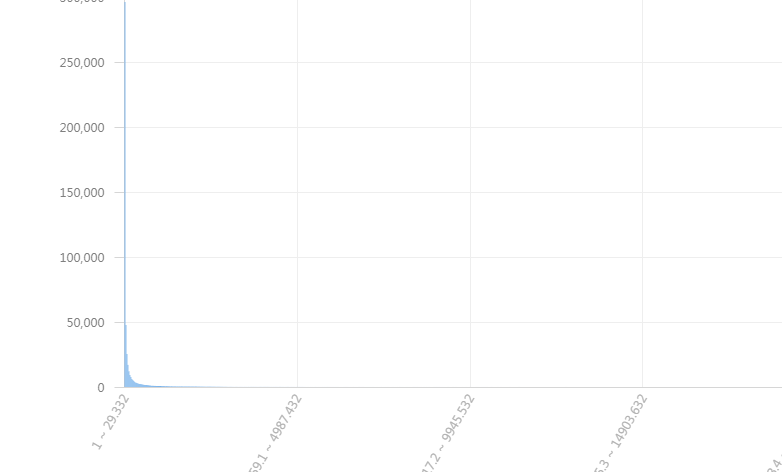
互信息中各词频统计，过滤掉小于500的词频，去除在多个类别同时出现的词，word2vec表示，svm分类

风格多代表这个文字越重要

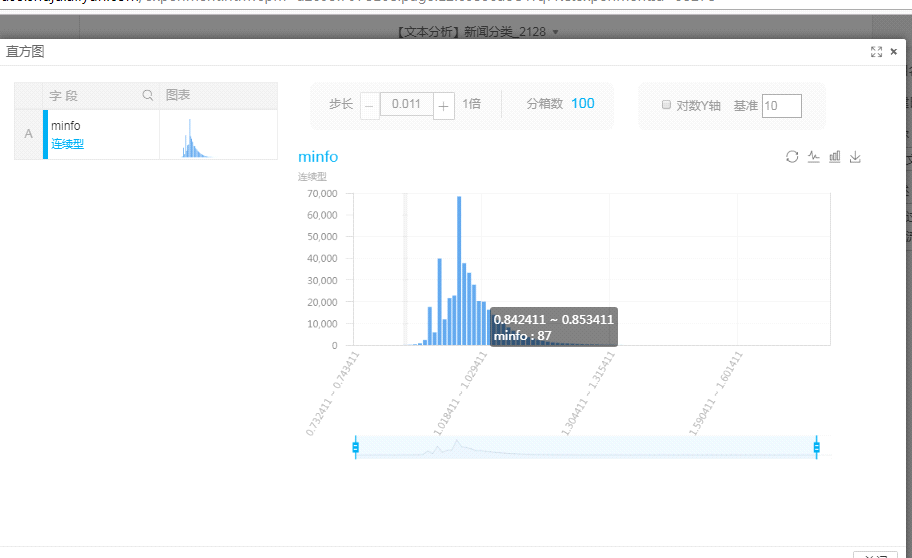
1.word2vec聚类





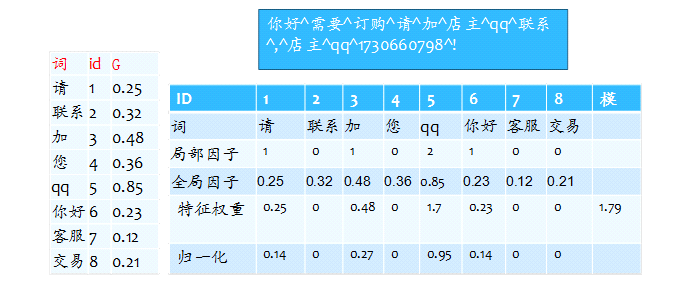


去除normal计算互信息

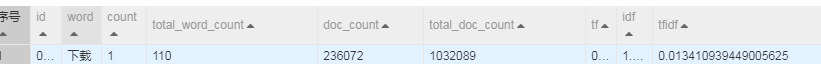


互信息对样本不均匀和低词频信息偏爱

互信息过滤阈值



Tf-idf



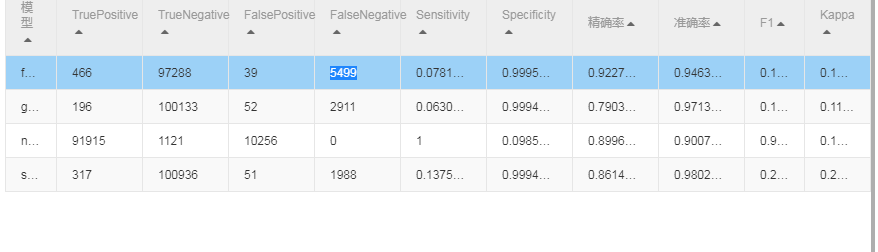
fake card

0.46-0.61 3906

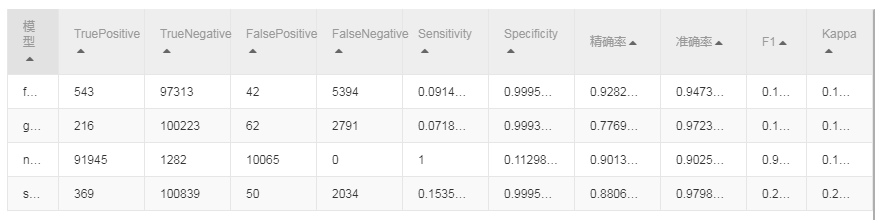
gmabling 0.8-1.2 4000左右

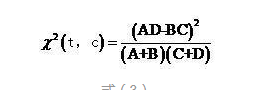
sexy 0.8-1.2 3000左右

全文分析，互信息提取特征 tf-idf



转为词频

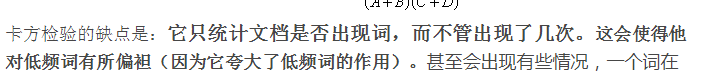


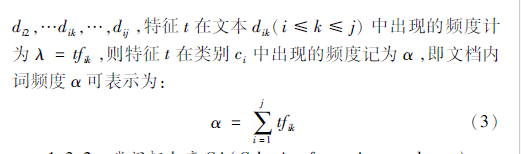


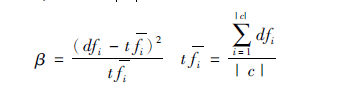
互信息简化公式



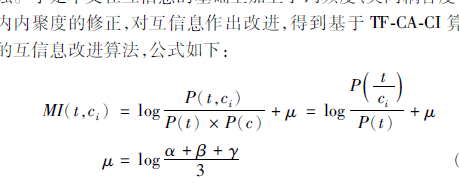


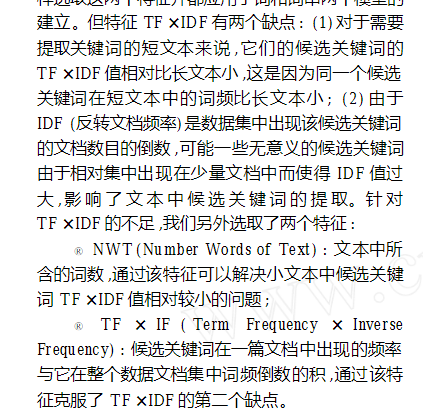












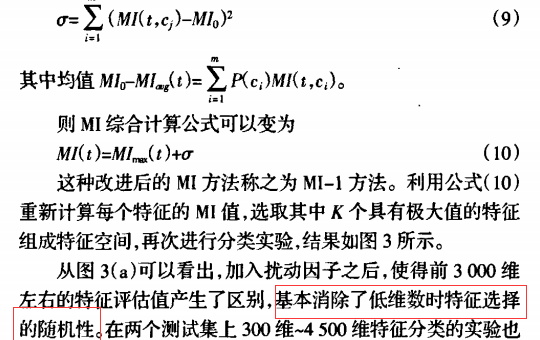
特征抽取的基本思想是利用映射的方法把原始特征项集映射到较低维的空

间中，映射后的特征叫二次特征，是原始特征的某种组合。特征

选择是利用某种评价函数独立地对每个原始特征项进行评估，

然后将它们按评估值地高低排序，从中选取若干个评估值最高

的特征项



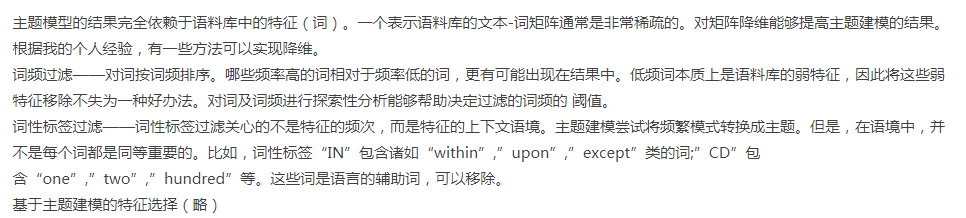
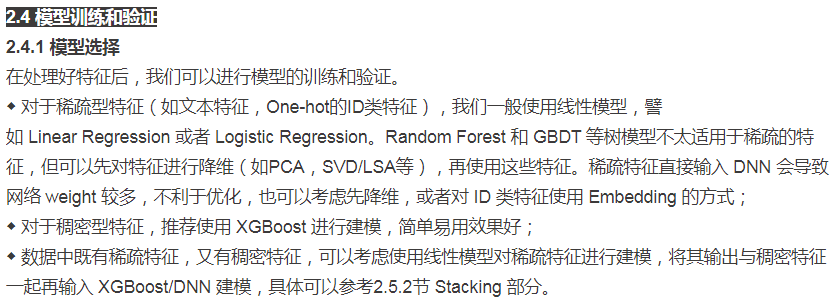
1测试集的tf-idf计算

2.计算互信息类与词的关联，测试集词汇未知

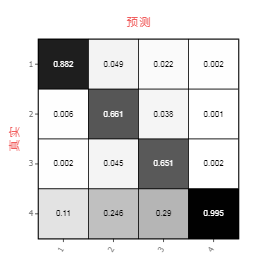
70000normal 4000检测

Lda:

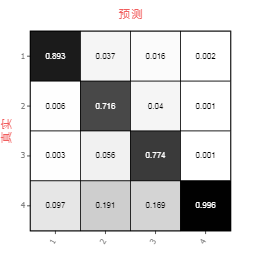
以一定的概率选取上述某个主题，再以一定的概率选取那个主题下的某个单词，



逻辑回归lda 6000

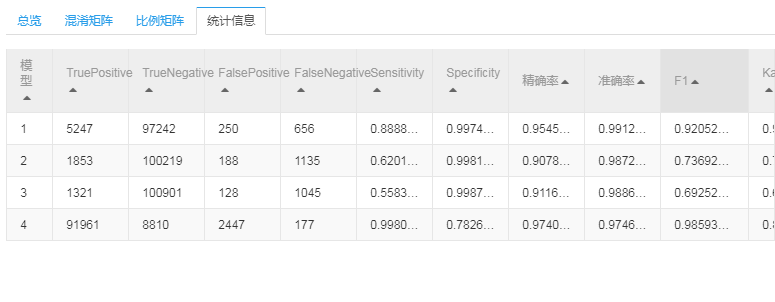


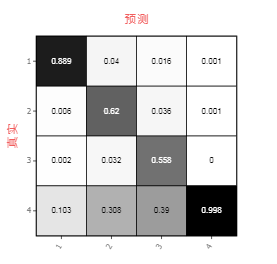
逻辑回归，tf-idf





Lda随机森林





1/全文去除tag 去除risk多个同id

2.获取全文，计算词频，分开训练与测试

3.训练数据》500词频过滤，

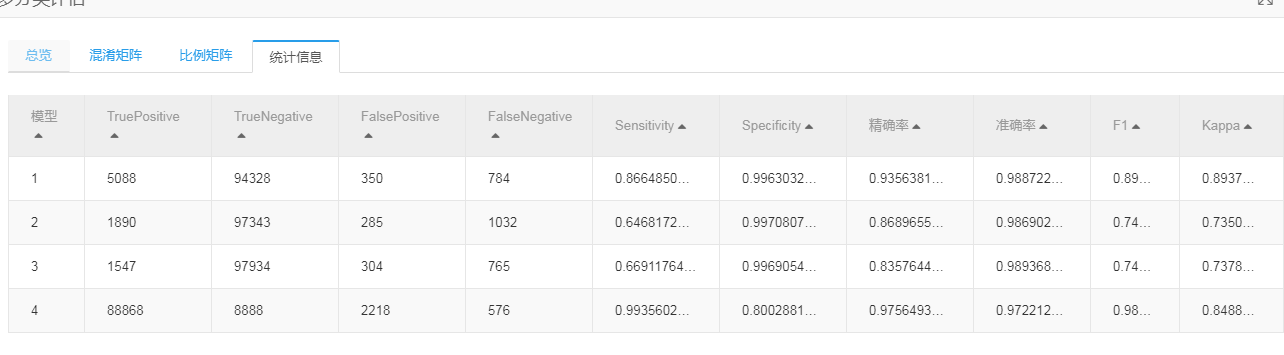
4，去除normal 计算互信息

5，取前1000

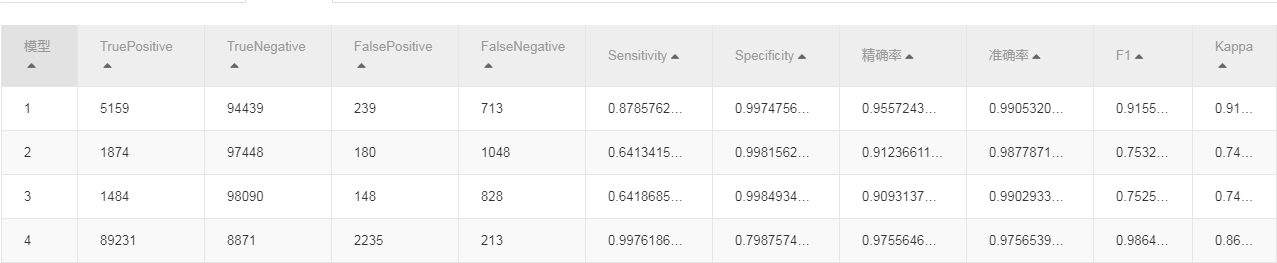
6. lda建模

7逻辑回归

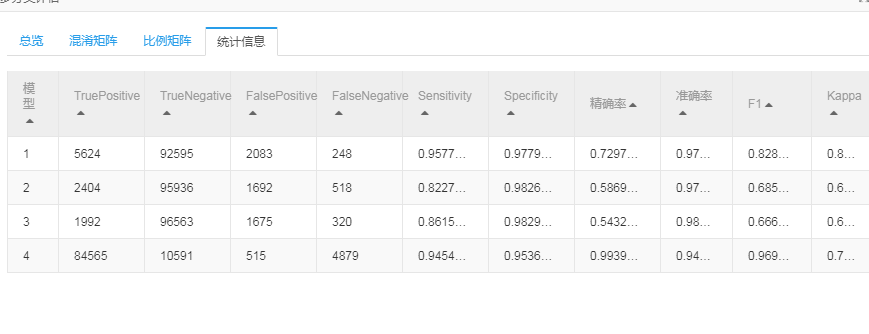
逻辑回归lda 3000



随机森林 lda 3000

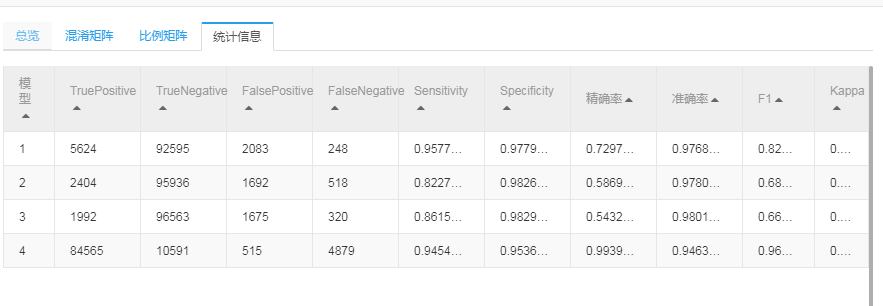


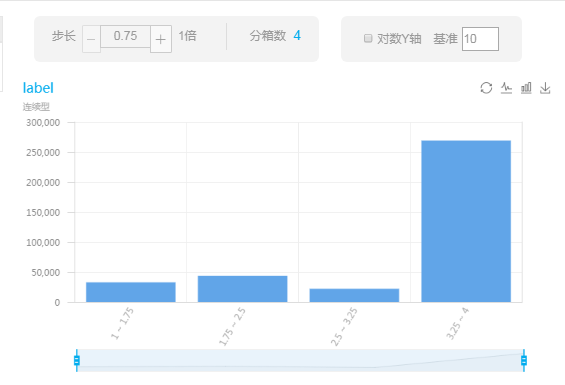
随机森林lda采样





逻辑回归





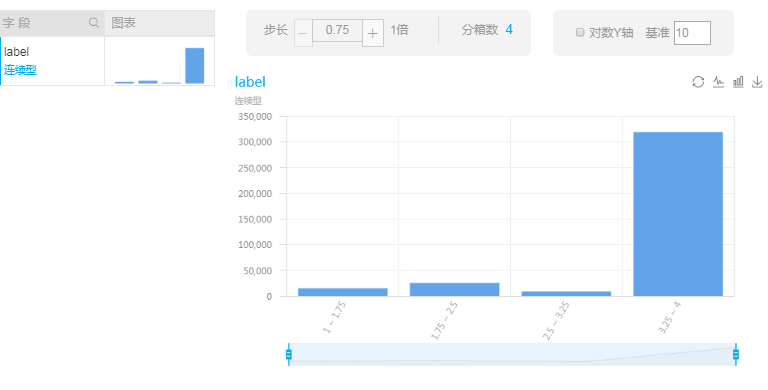
混合



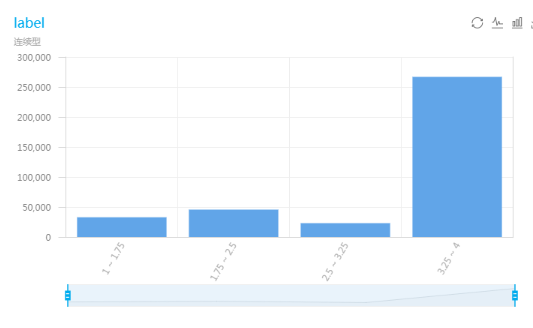
Lranswer 1000,lda,未采样

Lr answer1最后关键词1000，lda,采样

Lr2000 关键词2000，lda



C采样后



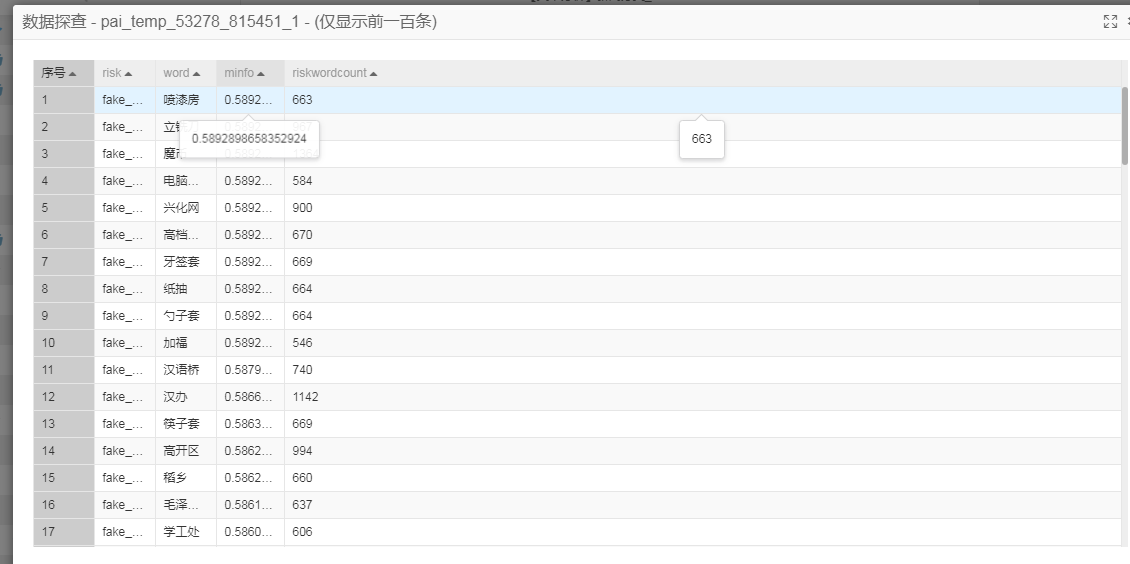
hxbks2ks0643

xxtxxthmhxb0643

TIANCHI\_4142

6C)tRZo2

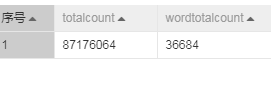
Keywordssn2



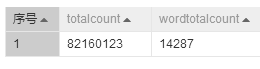


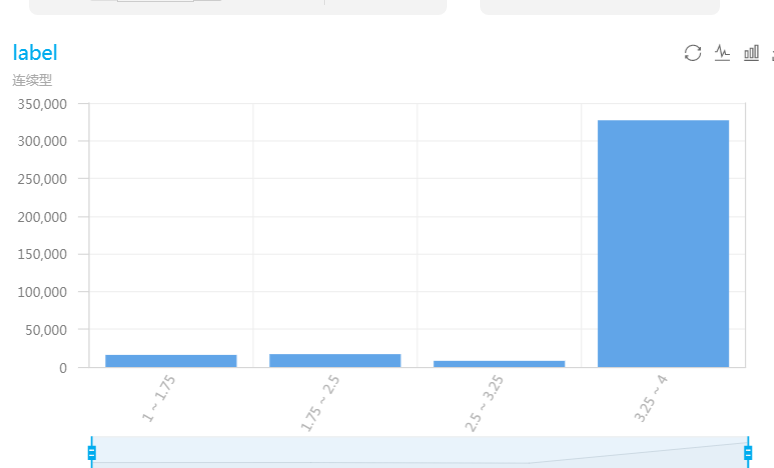


现在



原来





黄第一次提交

16038，17053，8175，327114

过滤100与过滤500，过滤100会有很多噪声数据

互信息加词频会带入噪声数据

Keywordsn2，过滤500，取2000计算互信息，不加词频

Keywordsn1，过滤500，取1000计算互信息，不加词频

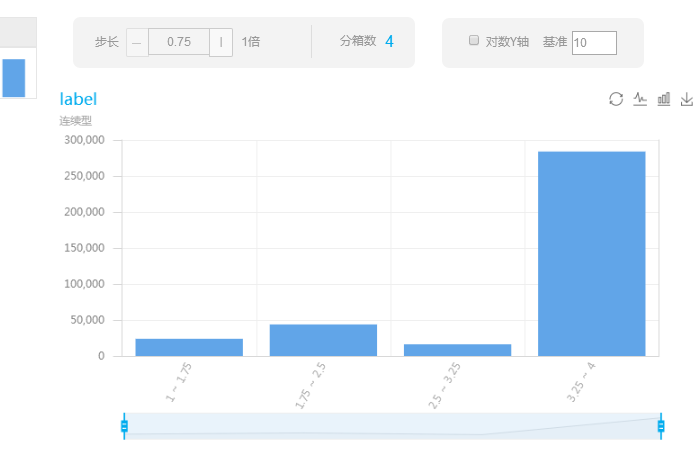
Keywordsf 过滤500，取1000计算互信息，加词频？

W\_keyword过滤500，互信息词频取2000

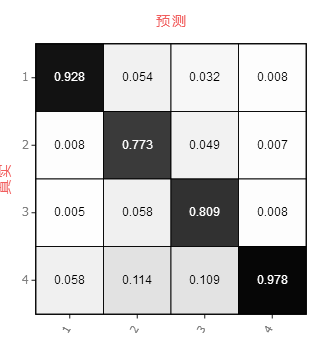
W\_keywords100，过滤100，互信息词频1000

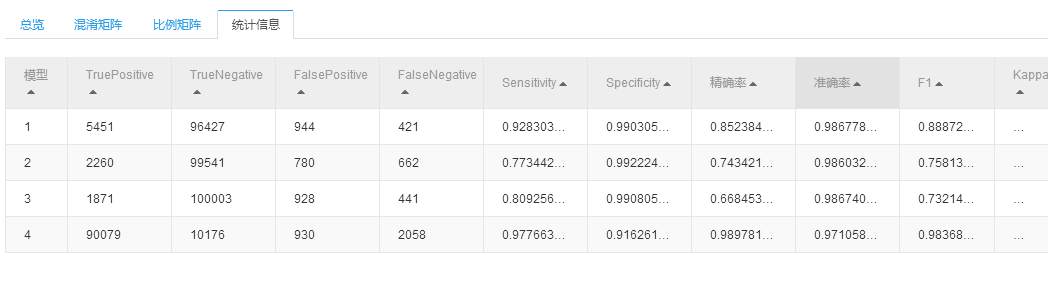
W\_keywords2000,过滤500，互信息2000

采样：l2正则

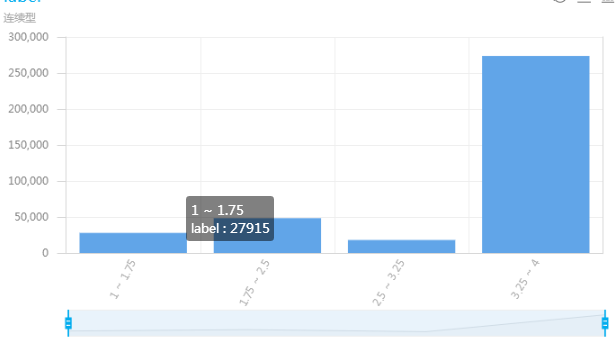


23940 43833 16256 284017

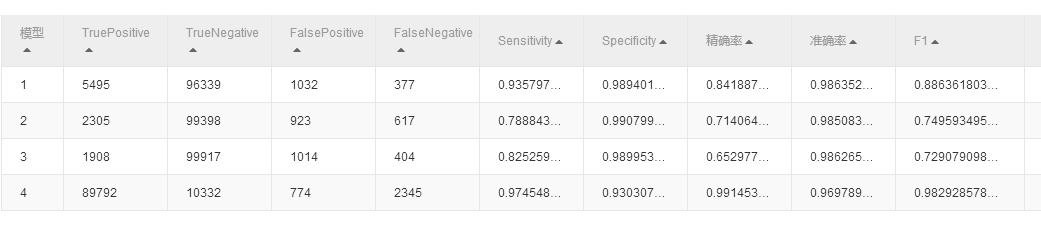




原来



27915 48272 18080 273678



L1正则

25078，48175，17928，276865

最好成绩 0.45

11326 20143 9077 总数328046

次佳成绩 0.42

15037 25365 8834

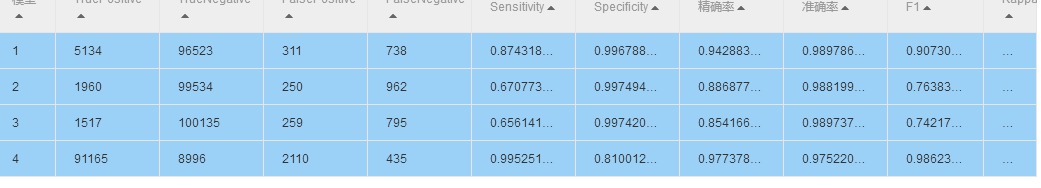
1500成绩 无重复 0.4149

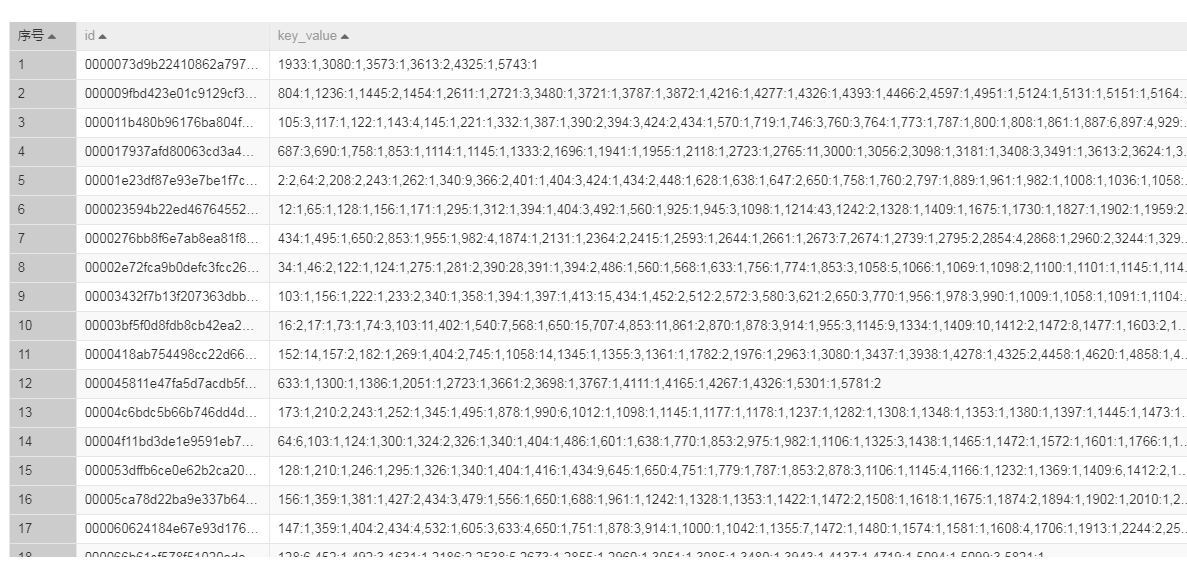
14451 27384 8650

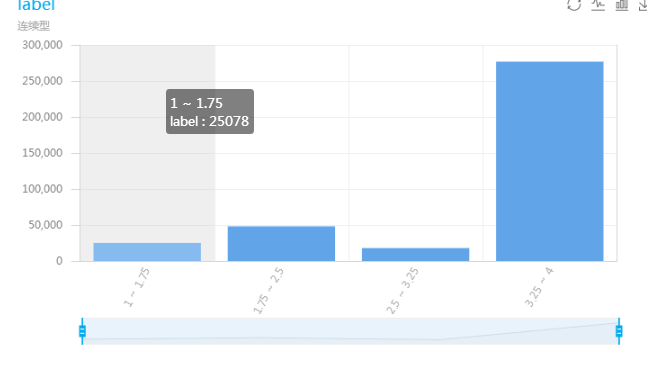
1500 有重复（无重复一样）

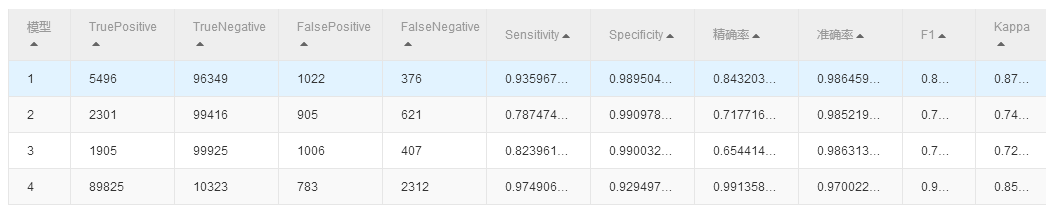
20787 21530 7734

1500增加lda迭代次数 成绩0.3529









去除干扰样本

采样

Lda在wordcount增加时，成绩上升？

逻辑回归以优化总体的精度为目标，不同类别的误分类情况产生的误差是相同的，考虑一个$500:1$的数据集，即使把所有样本都预测为多数类其精度也能达到$500/501$之高，很显然这并不是一个很好的学习效果，因此传统的学习算法在不平衡数据集中具有较大的局限性。

随机过采样则正好相反，即通过多次有放回随机采样从少数类$S\_min$中抽取数据集$E$，采样的数量要大于原有少数类的数量，最终的训练集为$S\_maj+E$。

吉布斯采样：

无论metropolis-hasting算法还是gibbs算法，都需要一个burn in过程，只有在达到平衡状态时候得到的样本才能是平衡状态时候的目标分布的样本，因此，在burn in过程中产生的样本都需要被舍弃。如何判断一个过程是否达到了平衡状态还没有一个成熟的方法来解决，目前常见的方法是看是否状态已经平稳（例如画一个图，如果在较长的过程中，变化已经不大，说明很有可能已经平衡）当然这个方法并不能肯定一个状态是否平衡，你可以举出反例，但是却是实际中没有办法的办法。

**burn-in阶段可能不需要**

当然可以一条链跑到黑，但是一条链跑到黑只能是写学术 paper 的做法，在工程上可能还是要考虑很实际的速度和效率的问题，做 LDA 的时候我们就得考虑每秒钟能处理多少个请求,这时候不得不设置 burn-in。

So one rule I often follow is to start the next run where the last run ended.

Another possible rule is to start at a point, like the mode, known to have reasonably high probability.

<https://www.leiphone.com/news/201706/dTRE5ow9qBVLkZSY.html>

