分析思路：

主要是探讨评分与作者的关系以及，评分与出版社的关系，从中发现它们之间的隐藏关系，

这里是把作者作为因变量和把出版社作为因变量，通过聚类把作者与评分之间的关系，以及出版社和评分之间的隐藏关系找出来

首先，我们先把出版社或者作者单独设为一列，然后用填补的方法，把空缺值补上，接着我们采用分词的方法把中文进行一个分词处理

再去判断是否为中文

```
def is_all_chinese(strs):
    for _char in strs:
        if not '\u4e00' <= _char <= '\u9fa5':
            return False
    return True
```

然后用停用词进行过滤，把这些分好的词保持下来

再去计算每个词语的tf-idf权值，然后去计算它们的权重

```
transformer = TfidfTransformer()

# 第一个fit_transform是计算tf-idf 第二个fit_transform是将文本转为词频矩阵
tfidf = transformer.fit_transform(vectorizer.fit_transform(corpus))
# 获取词袋模型中的所有词语
word = vectorizer.get_feature_names()

# 将tf-idf矩阵抽取出来 元素w[i][j]表示j词在i类文本中的tf-idf权重
weight = tfidf.toarray()
```

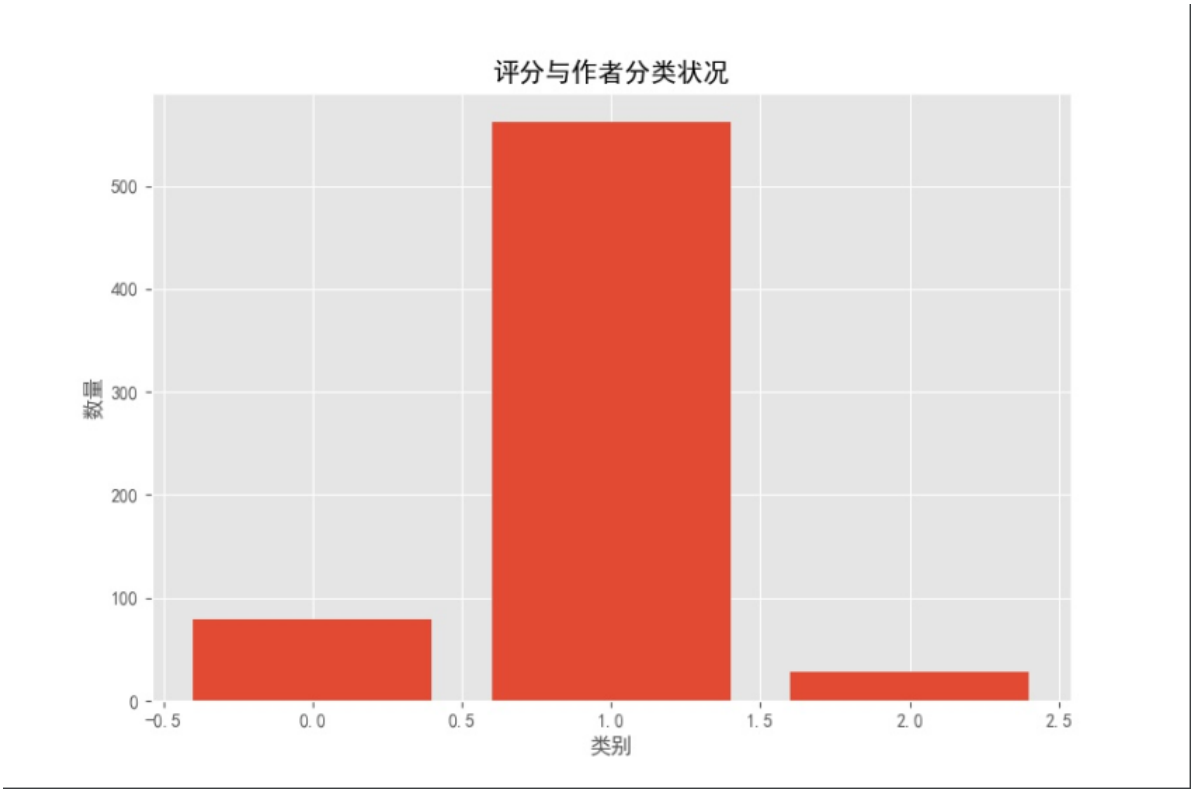
处理好上面的步骤之后，我们开始用k-means，无监督学习的聚类对其进行聚类

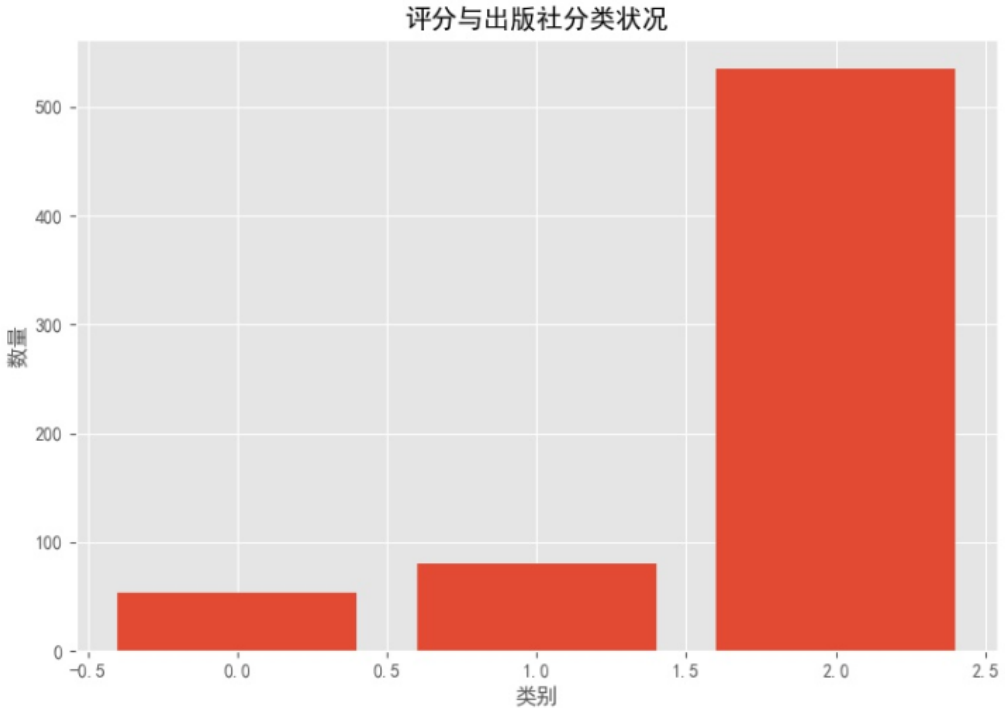
得到我们聚类的值之后，我们进行计算，每个对于分类的数量是多少

```python
predict_y = kmeans.fit_predict(weight)
counts = {}
for p in predict_y:
    counts[p] = counts.get(p, 0) + 1
ls = list(counts.items())
ls.sort(key=lambda x: x[0], reverse=False)
x_data = []
y_data = []
for key, values in ls:
    x_data.append(key)
    y_data.append(values)

plt.figure(figsize=(9, 6))
plt.bar(x_data, y_data)
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.title("评分与作者分类状况")
plt.xlabel("类别")
plt.ylabel("数量")
plt.savefig('评分与作者分类状况.jpg')
plt.show()
```

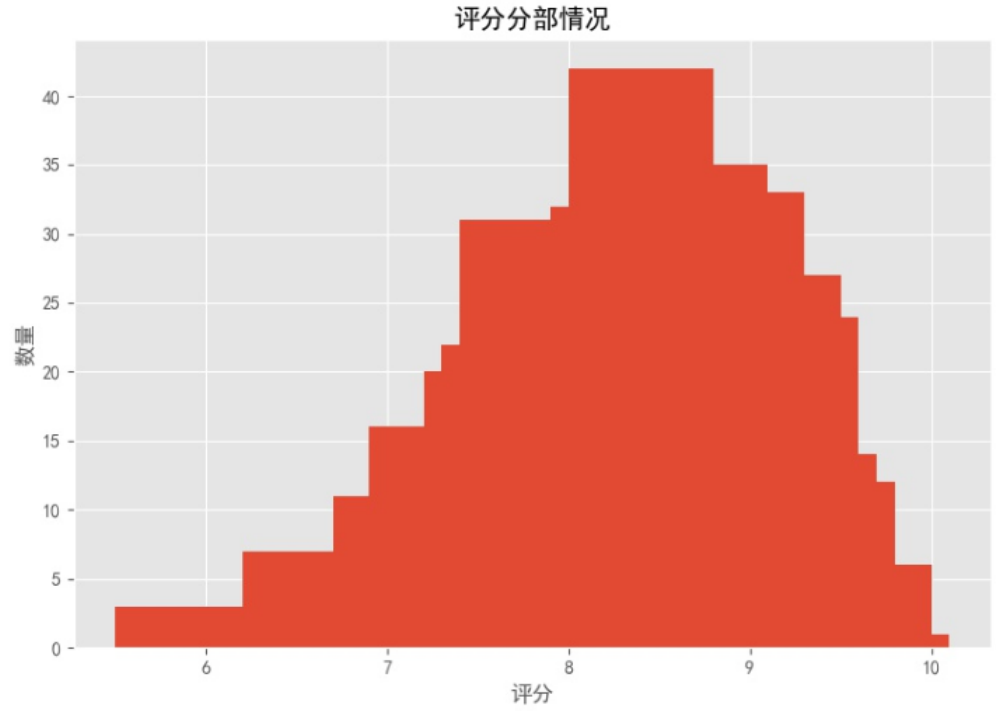
再用图可视化出来





该结果并不是唯一值，但是经过多次运行，我们可以肯定就是每次都会有一类是特别多的，其他的类就较为平均

再把我们的评分进行可视化



评分大多数集中在8以上，说明书籍认可的人数还是较多的