动漫评分的分析和推荐

摘要：相关推荐对于视频网站而言是非常重要的，可以有效地提高用户的使用时间和黏性。而动漫领域作为一个新兴的视频领域分支，拥有着广阔的市场和可观的前景。本项目基于Kaggle上的数据集，获取动漫的相关信息和用户对其的评价。在对整体情况进行可视化分析后，实现两种不同算法的推荐系统，并基于特定样例分析两者的优劣。

数据来源和清洗数据

`myanimelist`是一个重要的动画评分网站，其面向的受众主要是欧美的动画观众。本项目使用了`Kaggle`上的一个基于此网站的数据集，作为分析的数据。其中总计有两个表，包括动漫的信息和用户的评价信息。其中包含**1.2**万的作品信息和**781**万条用户的评价。

其中，动漫信息包含七条信息，包括动画的id，名字，主题类型，形式，集数等信息。评价信息包含用户对动画的评价。具体的情况如下：

```
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   anime_id  12294 non-null  int64
 1   name      12294 non-null  object
 2   genre     12232 non-null  object
 3   type      12269 non-null  object
 4   episodes  12294 non-null  object
 5   rating    12064 non-null  float64
 6   members   12294 non-null  int64
dtypes: float64(1), int64(2), object(4)
memory usage: 672.5+ KB

RangeIndex: 7813737 entries, 0 to 7813736
Data columns (total 3 columns):
 #   Column    Dtype
---  ------    -----
 0   user_id   int64
 1   anime_id  int64
 2   rating    int64
dtypes: int64(3)
memory usage: 178.8 MB
```

对于原始数据，我们进行简单的数据清洗。主要包括以下两个部分：对于数据评分缺失的项目，如评价信息里评分为-1的元组，进行删除；对于动画的名称信息，进行正则表达式的过滤，去除其中的特殊字符。由于合并之后的表很大而缺失值很少，直接删除不会影响其结果。

数据分析和可视化

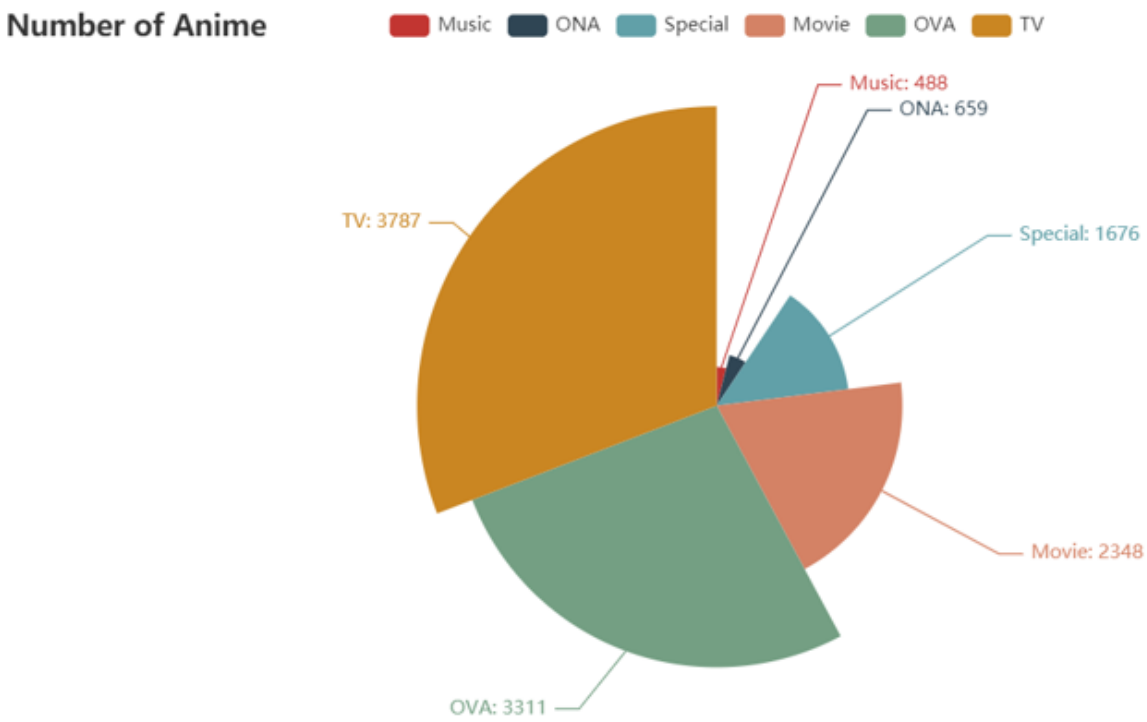
分析作品的类型

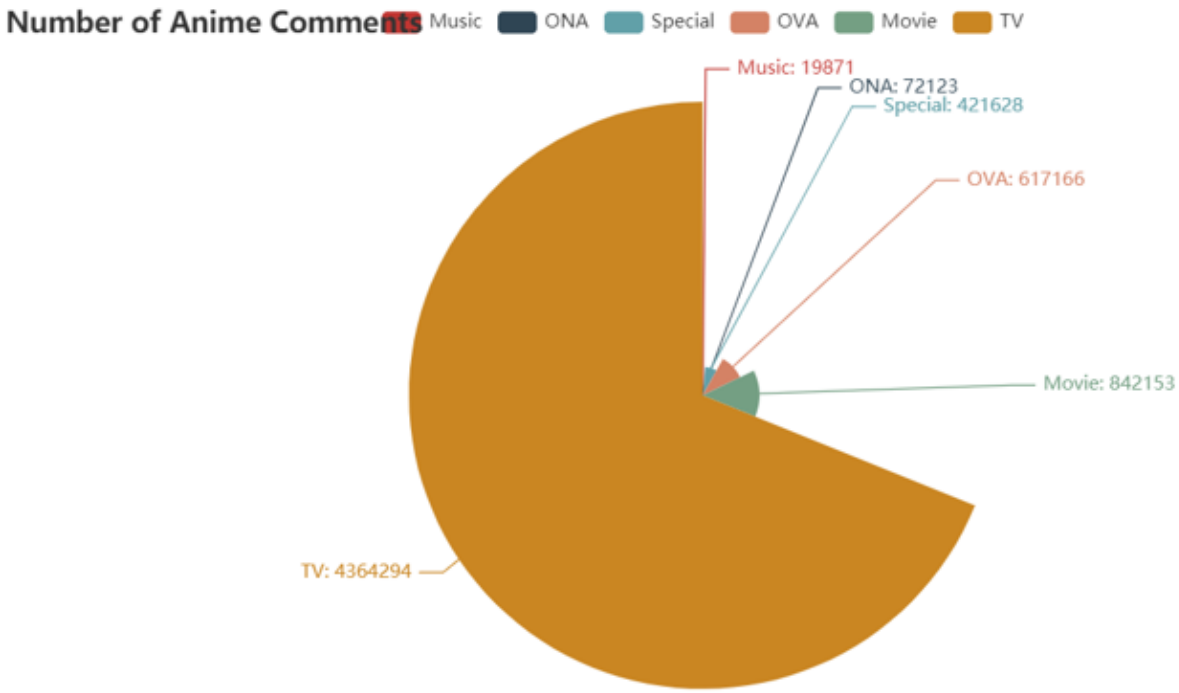
项目先统计了各个类型的作品数量，并绘制了玫瑰图。同时，也绘制了不同类别作品的评分数量的玫瑰图。

玫瑰图又名鸡冠花图、极坐标区域图，是南丁格尔在克里米亚战争期间提交的一份关于士兵死伤的报告时发明的一种图表。它是扇形图的一种，使用圆弧的半径长短表示数据的大小（数量的多少），可以更方便地观察不同类别之间的大小关系。

- 由于半径和面积的关系是平方的关系，南丁格尔玫瑰图会将数据的比例大小夸大，尤其适合对比大小相近的数值。
- 由于圆形有周期的特性，所以玫瑰图也适用于表示一个周期内的时间概念，比如星期、月份。

从玫瑰图上可以看出，**TV**动画和**OVA**的数量是最多的，两者占超过总体的50%。而对TV动画的评价则远远多于其他类别，约为70%。因此，后续将着重分析**TV**动画。





分析作品的标签

在数据集中，每个作品都有若干个标签，显示它们的主题类型。我们采用词云展示不同的标签和所占的比例。

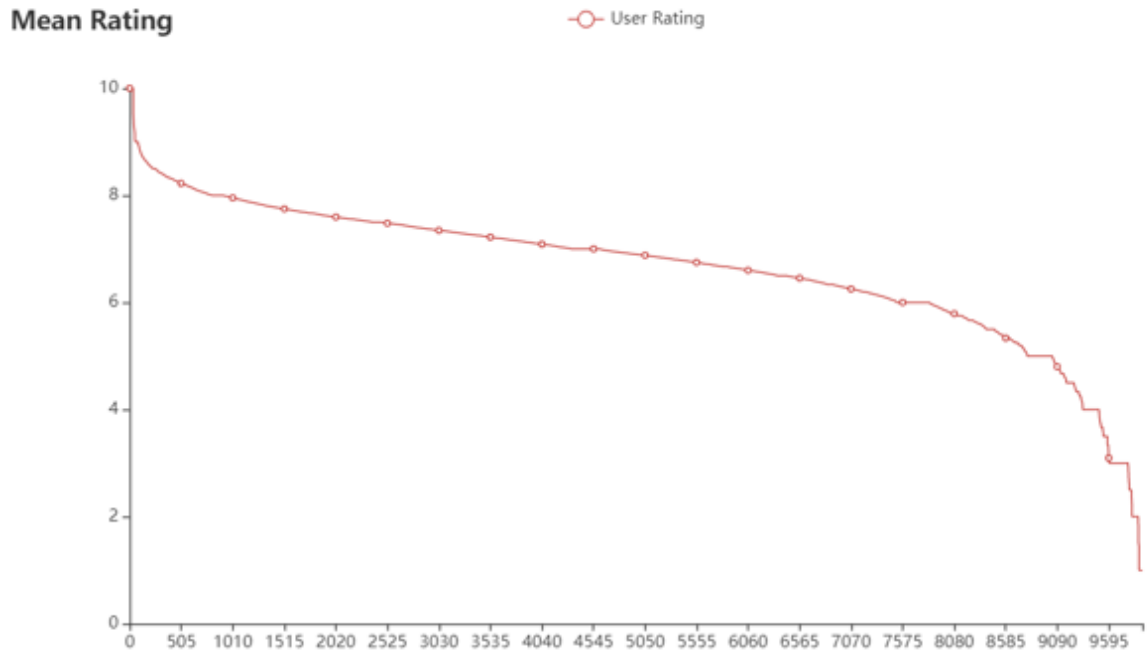
词云是一种词汇统计以后，根据权重生成的词语的排列组合图片，又称文字云，是文本数据的视觉表示。每个词的重要性以字体大小或颜色显示。 词云可以用于快速感知最突出的文字。

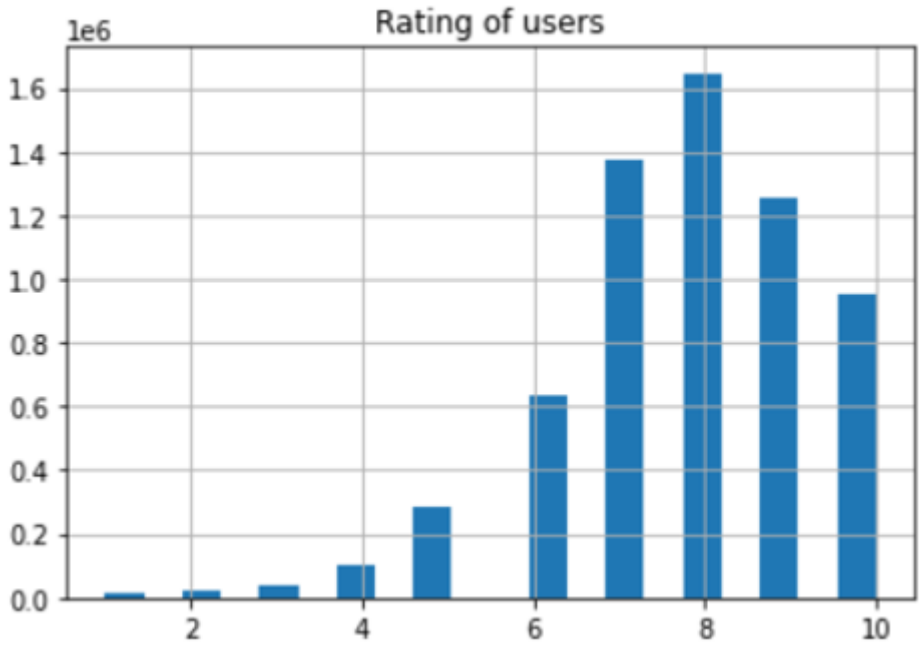
我们分别以标签的数量和评分的平均值作为权重，绘制词云。

从数量上看，`Comedy,Action,Romance`所占数量最多。但从平均得分上看，`Thriller,Police,Josei`的得分最高。从总体上看，平均得分高的标签的数量都不多，也就是说，从统计上看，小众类型的作品的得分会相对高一些。

计算作品的平均得分

然后，我们分析观众对作品评分的大致情况，因为评分同时与用户和作品相关，于是考虑计算作品的平均得分，绘制作品评分的折线图，同时绘制用户评分的直方图。

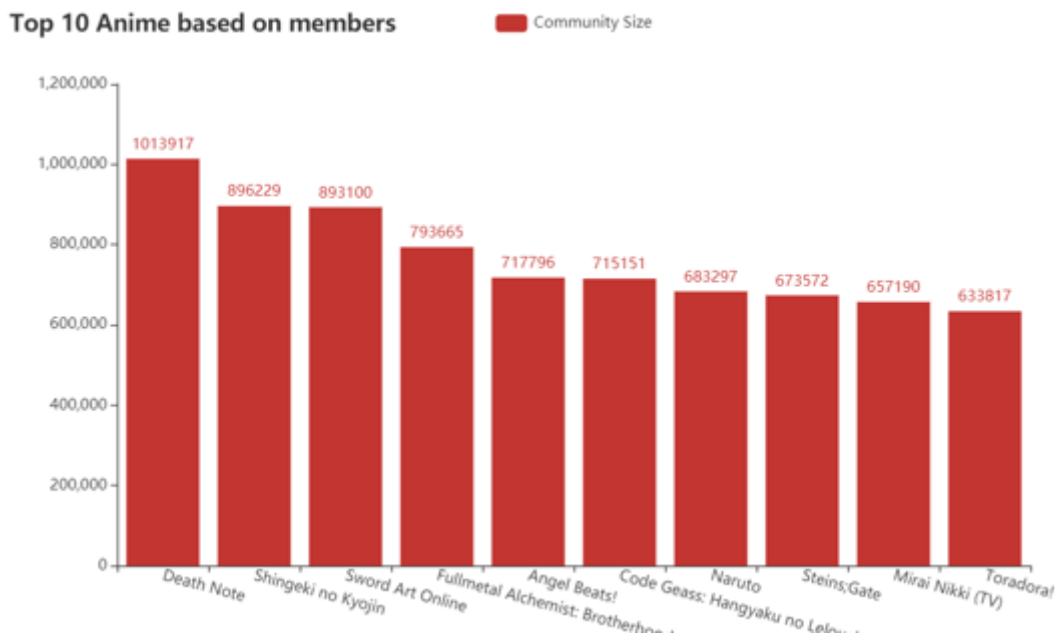




可以看出，绝大部分作品的平均得分都在8分左右，而用户的评分也都集中在8分左右，这两者是相符的。据此，在后续的分析中，可以认为评分在9分及以上的作品是用户明显喜爱的作品。

计算每部作品的评价人数

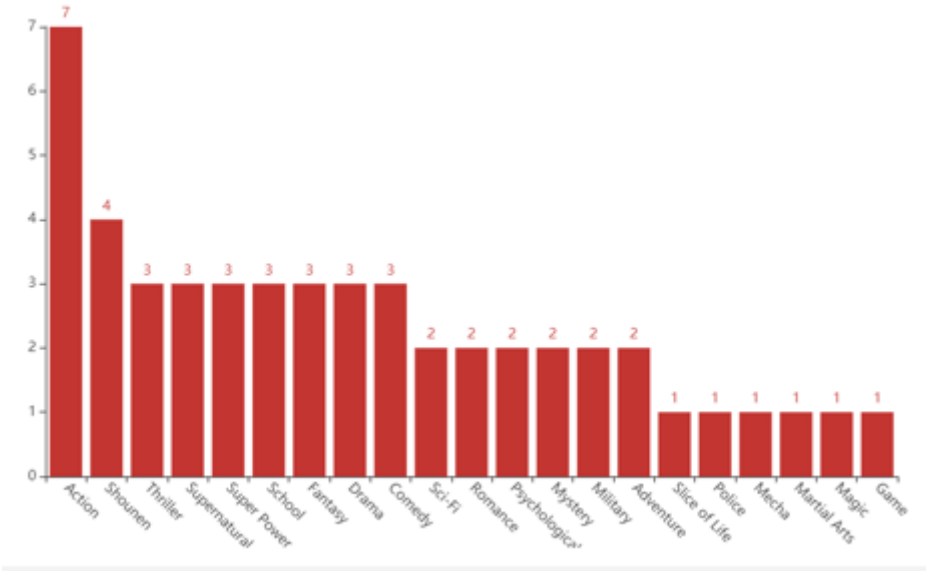
除了每部作品的平均得分，我们也关心作品的人气，也即观看的人数。一个直观的想法是，观看的人数和评分的人数应当是正相关的。因此，在这里用评分的人数进行衡量。绘制评分人数TOP10作品的直方图。



TOP10人气的作品评价人数相差不多，分别是：

1. 死亡笔记
2. 进击的巨人
3. 刀剑神域
4. 钢之炼金术师
5. 天使的心跳
6. 叛逆的鲁鲁修
7. 火影忍者
8. 命运石之门
9. 未来日记
10. 龙与虎

考察TOP10人气作品的标签，绘制直方图。



可以发现，`Action,Shounen(少年)`是出现频率最多的，可见人气最高的动画大多是热血王道类型的。

对于全部作品，按评价人数取对数绘制折线图。

对折线图进行分析，曲线在取对数后呈线性，说明作品的观看人数呈现较为明显的**长尾分布**，超过10万评价的作品只有约500个，只占5%，而低于1000的作品有5000个，占50%。一个可能的原因是，有高热度的作品更容易吸引观众，因此评价人数像滚雪球一样越来越多。这也是符合**马太效应**的一个典型现象。

动漫推荐

在对动漫的整体情况进行分析后，我们希望能根据用户对作品的评分给用户进行推荐。具体来说，推荐采用两种算法。首先，对于观看作品较少的用户，我们对其的了解较少，难以准确判断ta的喜好。这种情况往往被称为冷启动问题，为了解决这一问题，我们考虑利用ta已经看过的作品，采用KNN推荐和ta评分最高的作品相关的作品。其次，对于观看作品较多的用户，KNN算法的表现反而不佳，因为在观看作品数量较多之后，直接使用KNN的招新率不高，也即无法得到新的结果；而预先清理用户看过的结果后，通过KNN得到的作品的相似度都很低。因此，我们直接采用梯度下降矩阵分解的方法求对未知作品的评分。根据评分进行推荐。

清洗数据

要对用户进行推荐，首先要构建用户-作品的评分矩阵。由于评分数据过多，直接对评分矩阵进行处理是有难度的。在前面的分析中，我们发现用户对作品的评分呈明显的长尾分布，评价作品很少的用户占大多数，而这部分用户对于后续的分析贡献不大。因此，在清洗完缺失值后，我们去除评价作品少于200部的用户，以减小矩阵的规模。经过处理，矩阵的规模减少了约$\frac{2}{3}$，可见清洗的效果还是很好的。

使用KNN进行推荐

对于冷启动的情形，采用基于余弦相似度的KNN进行推荐。KNN是通过测量不同特征值之间的距离进行分类。它的思路是：如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别，其中K通常是不大于20的整数。反过来说，对于一个给定的正类样本集合，未知的样本也大概率是和它们同属一个类别。在本项目中，我们取用户看过的作品作为正类，寻找与这部分作品最接近的邻居，并推荐这部分邻居。经测试，推荐的结果基本能达到0.7左右的相似度，峰值甚至能达到0.9以上。

```
Recommendations for Tengai Makyou: Jiraiya Oboro-hen:

1: Gintama: Yorinuki Gintama-san on Theater 2D, with distance of 0.8990722469917743:
2: Gintama: Shinyaku Benizakura-hen, with distance of 0.9208018609443287:
3: Noblesse: Awakening, with distance of 0.9349069235855814:
4: Higashi no Eden Soushuuhen: Air Communication, with distance of 0.9355579226117094:
5: Kyoukai no Kanata Movie: I&#039;ll Be Here - Kako-hen, with distance of 0.9461610295135946:
6: Hajime no Ippo: Boxer no Kobushi, with distance of 0.94729717966649:
7: One Punch Man Specials, with distance of 0.9480061515482447:
8: Tonari no Kaibutsu-kun: Tonari no Gokudou-kun, with distance of 0.9517094130862097:
9: Baby Steps 2nd Season, with distance of 0.9524928555759454:
```

使用矩阵分解进行推荐

对于观看作品较多的观众，使用KNN进行推荐效果不佳，因为推荐的结果基本上都已经看过。因此，考虑使用矩阵分解的方法，使用梯度下降法填充评分矩阵中的未知项。通过对未知项的预测评分给用户进行推荐。具体而言，对于评分矩阵R，算法的目标是找到其分解矩阵P,Q使得

$$\arg\min_{P,Q} ||R - P^T Q||_2^2$$

基于分解矩阵，我们可以得到对用户的缺失评分的预测。通过得到的缺失值，我们可以对用户没有评价过的作品进行排序。

对于这种推荐方式，不容易对推荐结果直接进行评价。因此，考虑基于作品的标签进行评价。首先，根据前面的分析，可以知道评分高于9分的作品是用户所偏爱的，所以选择目标用户偏爱的作品，统计这些作品的标签，与预测作品的标签进行对比分析。通过绘制PR曲线可以发现，这一推荐的结果还是较为不错的。

总结

本项目对动漫的用户评分进行了可视化分析。从类型、标签、评分和人气四个方面对数据进行了展现。得到的结果如下：

1. TV动画占绝大多数；
2. 小众类型的动画相对评分较高；
3. 评分平均在8分左右，可以认为这是用户的基准评分；
4. 最高人气的作品往往是热血王道类型的；
5. 用户的观看作品数呈长尾分布。

随后，基于以上的结论和评分矩阵。我们对用户进行推荐，具体分为冷启动的情形和普通情况下的推荐，分别采用KNN和矩阵分解进行推荐。并分别对推荐结果进行了评价。从总体上看，对推荐结果的评价都能达到预期的目标。

附录

矩阵分解的梯度下降求解代码

```python
def recommend(userID,lr,alpha,d,n_iter,data):
    '''
    userID(int):推荐用户ID
    lr(float):学习率
    alpha(float):权重衰减系数
    d(int):矩阵分解因子(即元素个数)
    n_iter(int):训练轮数
    data(ndarray):用户-电影评分矩阵
    '''
    #获取用户数与电影数
```

```
    m,n = data.shape
    #初始化参数
    x = np.random.uniform(0,1,(m,d))
    w = np.random.uniform(0,1,(d,n))
    #创建评分记录表，无评分记为0，有评分记为1
    record = np.array(data>0,dtype=int)
    print("start training...")
    #梯度下降，更新参数
    for i in range(n_iter):
        p = np.dot(x,w)
        r = np.multiply(record,p-data)
        x_grads = np.dot(r,w.T)
        w_grads = np.dot(x.T,r)
        lr *= 0.9
        alpha *= 0.9
        x = (1-alpha)*x + lr*x_grads
        w = (1-alpha)*w + lr*w_grads
        #if i%10 == 0:
        print(i)
        print(p[userID][-5:])
    print("end training...")
    #预测
    predict = np.dot(x,w)
    #将用户未看过的电影分值从低到高进行排列

    for i in range(n):
        if record[userID][i] == 1 :
            predict[userID][i] = 0

    recommend = np.argsort(predict[userID])
    print(predict[userID][-5:])
    a = recommend[-1]
    b = recommend[-2]
    c = recommend[-3]
    d = recommend[-4]
    e = recommend[-5]
    print('为用户%d推荐的电影为：\n1:%s\n2:%s\n3:%s\n4:%s\n5:%s。'\
          %(user_id[userID],anime_name[a],anime_name[b],
            anime_name[c],anime_name[d],anime_name[e]))
    return [anime_name[i] for i in recommend[-5:]]
```