R&S | 手把手搞推荐[4]：打分预估模型

原创 机智的叉烧 CS的陋室 2019-05-24

点击上方蓝色文字立刻订阅精彩

Cross Me

Ed Sheeran;Chance the Rapper;PnB Rock - Cross Me [Explicit]

【R&S】

本栏目从原来的【RS】改为【R&S】，意为“recommend and search”，即推荐和搜索，结合本人近期的工作方向、最近上的七月在线的课、自己自学、而退出的特别栏目。当然，按照我往期的风格，更加倾向于去讨论一些网上其实讲得不够的东西，非常推荐大家能多看看并且讨论，欢迎大家给出宝贵意见，觉得不错请点击推文最后的好看，感谢各位的支持。

另外，【手把手搞推荐】是我近期开始的连载，结合自己所学，带上代码的手把手和大家分享一些模型和数据处理方式，欢迎关注。

往期回顾：

- R&S | 手把手搞推荐[0]：我的推荐入门小结
- R&S | 手把手搞推荐[1]：数据探索
- R&S | 手把手搞推荐[2]：特征工程指南
- R&S | 手把手搞推荐[3]：数据集存取思路
- RS | 推荐系统整体设计

内容接着前面的讲，如果觉得有点问题请看前面的篇章，后续我也会进行整理回顾。

在前序章节中，已经详细阐述了有关特征工程、数据集存取等方面内容，万事俱备，现在只需要进行模型设计，即可建模的流程，这绝对是觉得最为刺激的重头戏，当然这绝对不是就完事了，后续还有评估、诊断、监控、迭代等内容，后面的事后面再说吧。

模型设计思路

首先再次确认本次建模目标——就是根据给定用户，提供尽可能好的推荐内容，好的标准在于给他推荐的内容（测试集下）评分较高，而此处，作为打分预估，其实就是希望我们对用户A和电影B，能够更为准确地预测打分，所以说白了，就是为了"预测打分"。

此处的打分是1-5一共5个等级的打分，没有半分，与学校课程打分等方面不同，打分的区间小、等级少，相比回归模型，分类模型更为简单直接，而且实际上并非要求要准确预测到分数，其实就是一个准确到整数的问题，因此分类模型更合适，选择通过分类方法处理问题，因此可以处理为一个5分类问题。

在选择分类模型时，需要考虑如下几个方面：

- 第一个版本的模型要求尽可能简单，快速完成功能
- 测试或预测阶段效率尽可能高(毕竟涉及上线，响应时间非常重要)，KNN之类的肯定要抛弃
- 准确性达到基本标注
- 当前数据集内特征大都是离散型特征(已经全部onehot化了，就全都是离散特征了)

所以，此处我选择logistic regression作为此处的模型。补充一下，一般工业界的基线模型，非常喜欢用LR，满足上述特点，非常简单，后续会尝试FM、GDBT等，然后才是深度学习。

开始实现

首先肯定是加载数据了，这块代码上一篇也放过，这里就不赘述细节啦，大写的是自定义的路径。有关hash技巧方面，我还在学习和探索，有关内容可以大家自行了解，也可以期待我的文章，上次谈到的文章在这里：

转 | 数据集存取新方案-认识feature hashing

继续代码！

```
from scipy.sparse import load_npz
def load_dataset(x_path, y_path):
    y_ = []
    y_oh = []
    idx = 0
    with open(y_path, encoding="utf8") as f:
        for line in f:
            ll = line.strip().split(",")
            y_item = max([idx * int(float(ll[idx])) for idx in range(5)]) + 1
            y_oh_item = [int(float(item)) for item in ll[:5]]
            y_.append(y_item)
            y_oh.append(y_oh_item)
            idx = idx + 1
    x_ = load_npz(x_path)
    return x_, y_, y_oh
```

```
x_train, y_train, y_train_oh = load_dataset(GEN_DATA_TRAIN_X_PATH, GEN_DATA_TRAIN_Y_PATH
```

然后就轮到模型的初始化和训练了，这块相比大家非常熟悉，没错，所以用模型就是这么简单，所以才说前面的工作和后面的工作才是重头戏，基线模型本身操作真的少之又少。

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(penalty="l1",n_jobs=-1)
clf.fit(x_train, y_train)
```

后续应有模型评估，此处直接给出可供参考的指标，**后面会有篇章详细讨论评价指标设计**。

```
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import mean_absolute_error, mean_squared_error

def model_rep(y_true, y_pred, average="micro"):
    p = precision_score(y_true, y_pred, average=average)
    r = recall_score(y_true, y_pred, average=average)
    f1score = f1_score(y_true, y_pred, average=average)
    return p, r, f1score

def eu_distances(array1, array2):
    res = 0.0
    for idx in range(len(array1)):
        res = res + (array1[idx] - array2[idx]) ** 2
    return res / len(array1)


def average_distance(y_true_prod, y_pred_prod):
    res = 0.0
    for idx in range(len(y_true_prod)):
        res = res + eu_distances(y_true_prod[idx], y_pred_prod[idx])
    return res / len(y_true_prod)
p, r, f1score = model_rep(y_train, y_pred)      # 分类预测结果
mae = mean_absolute_error(y_train, y_pred)      # mae
mse = mean_squared_error(y_train, y_pred)          # mse
ad = average_distance(y_train_oh, y_prod)     # 平均距离
```

简单说几个这里用到的思路吧：

- 这里首先用的是分类的指标，毕竟我们把它当做分类问题来解决，即查准率、查全率、F1值(我个人不太喜欢叫精确率召回率，有时候含义不明确，查准和查全更直观表示precision和recall的含义)
- mae和mse都是回归问题用到的指标，因为这里的实质是打分预估，所以这里我就考虑把这个指标也放进去了
- 平均距离是算实际打分(onehot)和实际打分(各个打分的概率)查看距离，了解相似度，以确认这个打分与实际接近程度
- AUC是二分类指标，此处使用并不合适，已经考虑用上面的平均距离代替其相关功能

小结

没错，就讲完了，就这么简单，是不是会觉得还没开始就结束了？事实确实如此，上一期也提到了，算法项目远远不止机器学习，机器学习在这里就几行代码，但是前后的数据操作和监控则需要花费很多时间，而且才是能提升结果的，后续慢慢和大家谈吧，敬请期待！

我是叉烧，欢迎关注我！

叉烧，机器学习算法实习生，北京科技大学数理学院统计学研二硕士（保研），本科北京科技大学信息与计算科学、金融工程双学位毕业，硕士期间发表论文6篇，学生一作3篇，1项国家自然科学基金面上项目学生第2参与人，参与国家级及以上学术会议4次，其中，1次优秀论文，国家奖学金。曾任去哪儿网大住宿事业部产品数据，美团点评出行事业部算法工程师。

微信 zgr950123
邮箱 chashaozgr@163.com
知乎 机智的叉烧

微信个人公众号
CS的陋室

喜欢此内容的人还喜欢

属于算法的大数据工具-pyspark：10天吃掉那只pyspark
CS的陋室

18个悬挑脚手架优化做法，脚手架搭设出来后令人耳目一新！
建设施工安全

赖小民案一审宣判，死刑！
人民法院报