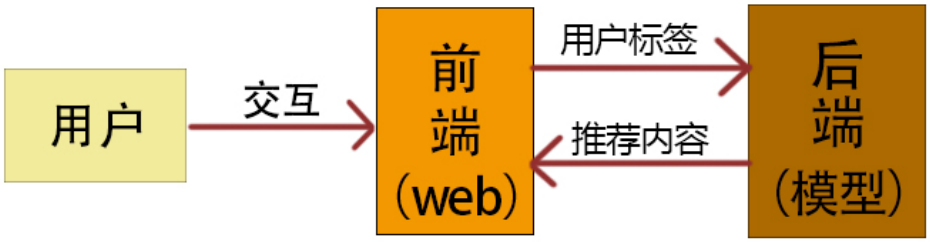
项目总结

1. 系统整体架构



前端：*交互，负责收集用户信息传给模型*

后端：*数据集训练，计算相关性返回推荐结果*

2. 系统配置

2.1 数据集

- Movielens 开源数据集

2.2 执行环境及依赖:

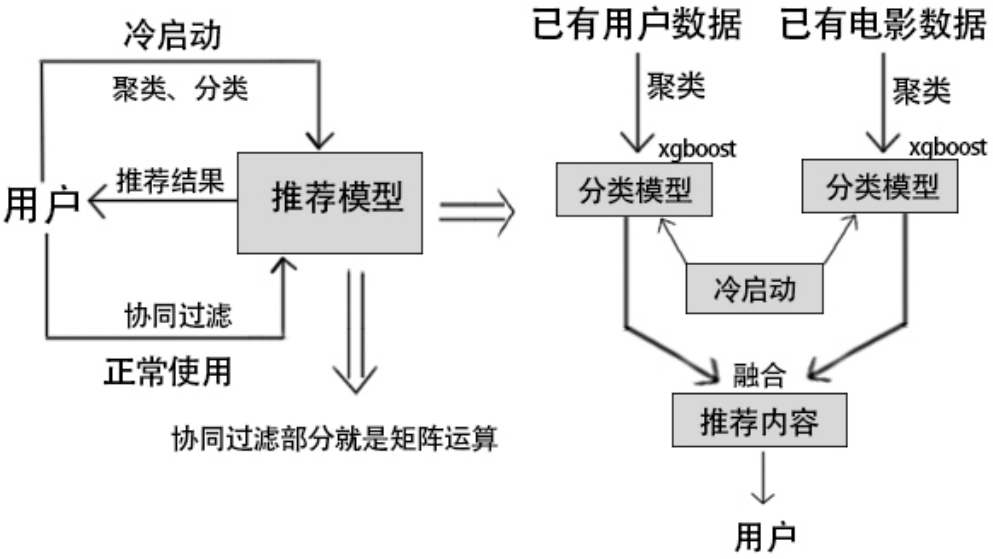
- Ubuntu 18.04
- Python 3.6
- jupyterlab
- pandas
- numpy

2.3 技术路线:

- JS
- Django
- Python

2.4 功能结构

- 前端交互：提供搜索/查看/导航/推荐功能
- 推荐模型主要为两部分：冷启动与动态更新
  - 冷启动：
    用户初次使用时采集用户偏好标签和观影历史信息进行分类算法推荐
  - 动态更新：
    系统采集用户使用过程中的信息，使用协同过滤矩阵，动态更新用户相关性系数进行更新推荐

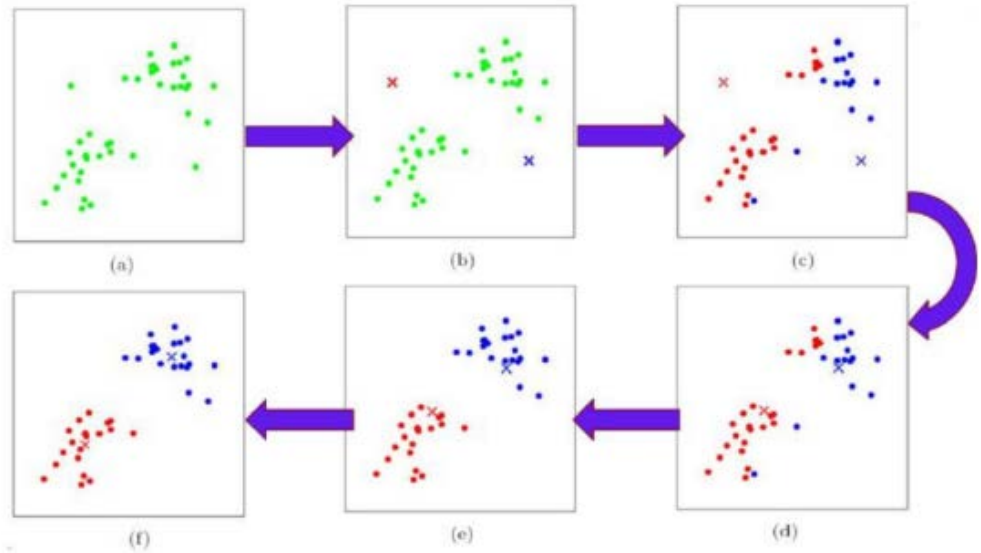


2.5 算法介绍

- 聚类 + 分类 （Cluster & Classification:）
  Make correlation among audience

  1.聚类算法（Cluster）

对已有用户数据，电影数据的数据集，按照数据内部存在的数据特征将数据集划分为多个不同的类别，使类别内的数据比较相似，类别之间的数据相似度比较小，属于无监督学习。聚类算法的重点是计算样本项之间的相似度，有时候也称为样本间的距离。本次系统采用K-Means算法，图解表示如下：



即：

1）原始数据集有N个样本，人为给定两个中心点。

2） 分别计算每个样本到两个中心点之间的距离，可选欧几里得距离，计算公式用 9.1

所提到的公式如下所示 $$d=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

3）把样本分为了两个簇，计算每个簇中样本点的均值为新的中心点。计算公式如下：

$$a_j=\frac{1}{N(c_j)}\sum_{i\in c_j}x_i$$

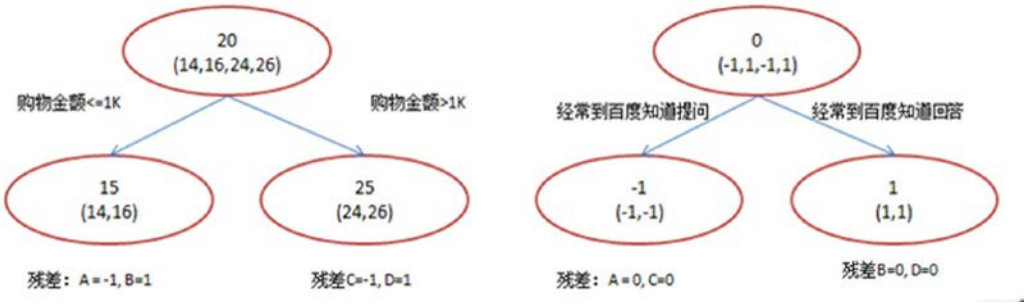
4）重复以上步骤，知道达到前面所说中止条件。

2. xgboost 分类算法（Classification）

对于一个复杂的问题，将多个专家的判断进行适当的综合所得出的判断，要比任何一个专家单独判断好。每一步产生一个弱预测模型(如决策树)，并加权累加到总模型中，可以用于回归和分类问题；如果每一步的弱预测模型生成都是依据损失函数的梯度方向，则称之为梯度提升(Gradient boosting)。

梯度提升算法首先给定一个目标损失函数，它的定义域是所有可行的弱函数集合(基函数)；提升算法通过迭代的选择一个负梯度方向上的基函数来逐渐逼近局部极小值。这种在函数域的梯度提升观点对机器学习的很多领域有深刻影响。

xgboost 是在 GBDT 的基础上对 boosting 算法进行的改进，是一个高级的梯度增强算法（gradient boosting algorithm）内部决策树使用的是回归树，简单回顾 GBDT 如下：



回归树的分裂结点对于平方损失函数，拟合的就是残差；对于一般损失函数（梯度下降），拟合的就是残差的近似值，分裂结点划分时枚举所有特征的值，选取划分点。

最后预测的结果是每棵树的预测结果相加。

Xgboost 算法推导如下：

目标函数：

$$J(f_t) = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + C$$

根据Taylor展开式：$f(x+\Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$，令

$$g_i = \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}, h_i = \frac{\partial^2 L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$$

则：

$$J(f_t) \approx \sum_{i=1}^{n} \left[L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_i^2(x_i) \right] + \Omega(f_t) + C$$

假定某决策树的叶结点数目为T，每个叶结点的权值为$\vec{w} = (w_1, w_2 \dots w_T)$。决策树的学习过程，就是构造如何使用特征得到划分，从而得到这些权值的过程。样本x落在叶结点q中，定义f为：$f_t(x) = w_{q(x)}$。

正则项，决策树的复杂度可考虑叶结点树和叶权值：$\Omega(f_t) = \gamma^{T_t} + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$，其中$T_t$为叶结点数，$w_j$为$j$叶子结点权重。

目标函数计算：

$$J(f_t) \approx \sum_{i=1}^{n} \left[L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_i^2(x_i) \right] + \Omega(f_t) + C$$

$$= \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_i^2(x_i) \right] + \Omega(f_t) + C$$

$$= \sum_{i=1}^{n} \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma^{T_t} + \frac{1}{2}\lambda \sum_{j=1}^{T_t} w_j^2 + C$$

$$= \sum_{j=1}^{T_t} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i) w_j^2 \right] + \gamma^{T_t} + \frac{1}{2}\lambda \sum_{j=1}^{T_t} w_j^2 + C$$

$$= \sum_{j=1}^{T_t} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma^{T_t} + C$$

定义：

$$G_j = \sum_{i \in I_j} g_i, H_j = \sum_{i \in I_j} h_i$$

从而，

$$J(f_t) = \sum_{j=1}^{T_t} \left[G_j w_j + \frac{1}{2}(H_i + \lambda) w_j^2 \right] + \gamma^{T_t} + C$$

对w求偏导，得：

$$\frac{\partial J(f_t)}{\partial w_j} = G_j + (H_j + \lambda) w_j$$

令偏导等于0，得到：

$$w_j = -\frac{G_j}{H_j + \lambda}$$

代回目标函数，得

$$J(f_t) = -\frac{1}{2} \sum_{j=1}^{T_t} \frac{G_j^2}{H_j + \lambda} + \gamma^{T_t}$$

构造决策树的结构：

- 对于某可行划分，计算划分后的$J(f)$;
- 对于所有可行划分，选择$J(f)$降低最小的分割点。

- **协同过滤 + 冷启动**

1. 协同过滤：（Collaborative Filtering recommendation）

协同过滤简单来说是利用某兴趣相投、拥有共同经验之群体的喜好来推荐用户感兴趣的信息，个人通过合作的机制给予信息相当程度的回应（如评分）并记录下来以达到过滤的目的进而帮助别人筛选信息，回应不一定局限于特别感兴趣的，特别不感兴趣信息的纪录也相当重要。与传统的基于内容过滤直接分析内容进行推荐不同，协同过滤分析用户兴趣，在用户群中找到指定用户的相似（兴趣）用户，综合这些相似用户对某一信息的评价，形成系统对该指定用户对此信息的喜好程度预测。

本系统采用以用户为基础（User-based）的协同过滤：

用相似统计的方法得到具有相似爱好或者兴趣的相邻用户，所以称之为以用户为基础（User-based）的协同过滤或基于邻居的协同过滤(Neighbor-based Collaborative Filtering)。 方法步骤：

1.收集用户信息

收集可以代表用户兴趣的信息。一般的网站系统使用评分的方式或是给予评价，这种方式被称为“主动评分”。另外一种是“被动评分”，是根据用户的行为模式由系统代替用户完成评价，不需要用户直接打分或输入评价数据。电子商务网站在被动评分的数据获取上有其优势，用户购买的商品记录是相当有用的数据。

2.最近邻搜索(Nearest neighbor search, NNS)

以用户为基础（User-based）的协同过滤的出发点是与用户兴趣爱好相同的另一组用户，就是计算两个用户的相似度。例如：查找 n 个和 A 有相似兴趣用户，把他们对 M 的评分作为 A 对 M 的评分预测。一般会根据数据的不同选择不同的算法，目前较多使用的相似度算法有 Pearson Correlation Coefficient、Cosine-based Similarity、Adjusted Cosine Similarity。

3.产生推荐结果

有了最近邻集合，就可以对目标用户的兴趣进行预测，产生推荐结果。依据推荐目的的不同进行不同形式的推荐，较常见的推荐结果有 Top-N 推荐和关系推荐。Top-N 推荐是针对个体用户产生，对每个人产生不一样的结果，例如：通过对 A 用户的最近邻用户进行统计，选择出现频率高且在 A 用户的评分项目中不存在的，作为推荐结果。关系推荐是对最近邻用户的记录进行关系规则(association rules)挖掘。

2. 冷启动：（cold boot）

协同过滤推荐基于这样的假设: 为用户找到他真正感兴趣的内容的方法是，首先找与他兴趣相似的用户，然后将这些用户感兴趣的东西推荐给该用户。所以该推荐技术最大的优点是对推荐对象没有特殊的要求，能处理非结构化的复杂对象，如音乐、电影等，并能发现用户潜在的兴趣点。协同过滤推荐算法主要是利用用户对项目的评分数据，通过相似邻居查询，找出与当前用户兴趣最相似的用户群，根据这些用户的兴趣偏好为当前用户提供最可能感兴趣的项目推荐列表。为更进一步地说明协同过滤推荐算法的推荐原理，本文以用户对电影的推荐为例进行阐述。表 1 是用户对电影评分数据的一个简单矩阵的例子，其中每一行代表一个用户，每一列代表一部电影，矩阵中的元素表示用户对所看电影的评分，评分值一般是从 1 到 5 的整数，评分值越大表明用户喜欢该电影。

	Snow white	**Star wars**	**Spider man**	**Supper man**
Alice	5	3	4	1
Bob	5	3	4	1
Chris		3	4	1
Tony	2	1	1	5

表 1 评分矩阵

对表 1 中的数据利用协同过滤推荐算法，系统查找到用户 Alice、Bob 和 Chris 具有相似的兴趣爱好，因为他们对后 3 部电影的评分相同，那么系统会推荐电影 Snow white 给 Chris，因为与其兴趣偏好相似的用户 Alice 和 Bob 对该电影的评分值较高。在表 2 中，对于新用户 Amy，没有评分信息，根据协同过滤推荐算法，无法根据评分信息查找与其兴趣偏好相似的用户，所以系统无法为该用户推荐电影，同样对于新电影 Shrek，因缺乏评分信息系统无法感知它的存在，所以也无法将其推荐出去。这就是协同过滤推荐算法所存在的新用户和新项目问题。

	Snow white	**Star wars**	**Spider man**	**Supper man**	**Shrek**
Alice	5	3	4	1	
Bob	5	3	4	1	
Chris		3	4	1	
Tony	2	1	1	5	
Amy					

表 2 评分矩阵