算法精英大赛指导书——Xgboost预测用户信息

实验背景

用户的人口属性（如性别、年龄、常驻地等）数据一方面可以被用于个性化推荐服务，提升用户体验，另一方面可以用于手机用户群画像分析，帮助厂商了解产品的人群定位，优化产品设计。

实验介绍

1. 年龄是用户人口属性的重要维度，本次比赛任务为根据用户的手机使用行为习惯来预估用户所处的年龄段。每个用户（以唯一ID标识）对应唯一的年龄段。年龄段有6种划分，分别代表不同年龄段，分别为：小于等于18岁， 19-23岁，24-34岁， 35-44岁， 45-54岁，大于等于55岁。参赛同学根据华为提供数据构建预测模型进行年龄段预估，在测试数据集上给出预估结果。
2. xgboost是一种高效的预测分类模型，速度快、支持自行处理缺失值、效果好是该模型的主要特点。
3. 本实验利用xgboost模型预测用户年龄段，分别用1、2、3、4、5、6表示

实验准备

数据说明

评价指标

Accuracy：$Accuracy = \frac{1}{m}\sum_{i=1}^{m} I\left(f\left(x_i\right) = y_i\right)$

测试集形式

用户标识（ID）	预测值（年龄段）	数据形式（数据集中表示）
10001	小于等于18岁	1
10002	19-23岁	2
10003	24-34岁	3
10004	35-44岁	4
10005	45-54岁	5

用户标识（ID）	预测值（年龄段）	数据形式（数据集中表示）
10006	大于等于55岁	6

数据特征说明

age_train.csv	
用户标识（uId）	匿名化处理后的用户唯一标识（ID取值从1000001开始，依次递增）
年龄段（age_group）	年龄范围（取值1； 2； 3； 4； 5；6；）
age_test.csv	
用户标识（uId）	匿名化处理后的用户唯一标识（ID取值从1000001开始，依次递增）
user_basic info.csv	
用户标识（uId）	匿名化处理后的用户唯一标识（ID取值从1000001开始，依次递增）
性别（gender）	男/女（取值空间0,1）
常住地（city）	如深圳市、南京市等（匿名化处理，实际取值c001，c002....）
手机型号（prodName）	如mate10、honor 10等（匿名化处理，实际取值p001、p002.....）
手机ram容量（ramCapacity）	手机ram的大小，以G为单位
ram剩余容量占比（ramLeftRation）	手机剩余的容量占总容量的比例
rom容量（romCapacity）	手机rom的大小，以G为单位
rom剩余容量占比（romLeftRation）	手机剩余rom容量占总rom容量的比例
手机颜色（color）	手机机身的颜色
字体大小（fontSize）	手机设置的字体大小
上网类型（ct）	2G/3G/4G/WIFI
移动运营商（carrier）	移动/联通/电信/其他
手机系统版本（os）	AndroId操作系统的版本号
user_behavior info.csv	
用户标识（uId）	匿名化处理后的用户唯一标识（ID取值从1000001开始，依次递增）
开机次数（bootTimes）	一段时间内(30天)手机的总开机次数
手机A特性使用次数（AFuncTimes）	一段时间内(30天) 手机A特性使用次数

age_train.csv	
...	...
user_app_actived.csv	
用户标识（uId）	匿名化处理后的用户唯一标识（ID取值从1000001开始，依次递增）
应用标识（appId）	匿名化处理后的app唯一标识
user_app_usage.csv	
用户标识（uId）	匿名化处理后的用户唯一标识（ID取值从1000001开始，依次递增）
应用标识（appId）	匿名化处理后的app唯一标识
使用时长（duration）	1天内用户对某app的累计使用时长
打开次数（times）	1天内用户对某app的累计打开次数
使用日期（use_date）	用户对某app的使用日期
app_info.csv	
应用标识（appId）	appId为app应用的唯一标识
应用类型（category）	app所属的应用类型

计算框架

- Xgboost(计算机若有GPU，最好下载GPU版本)

硬件配置

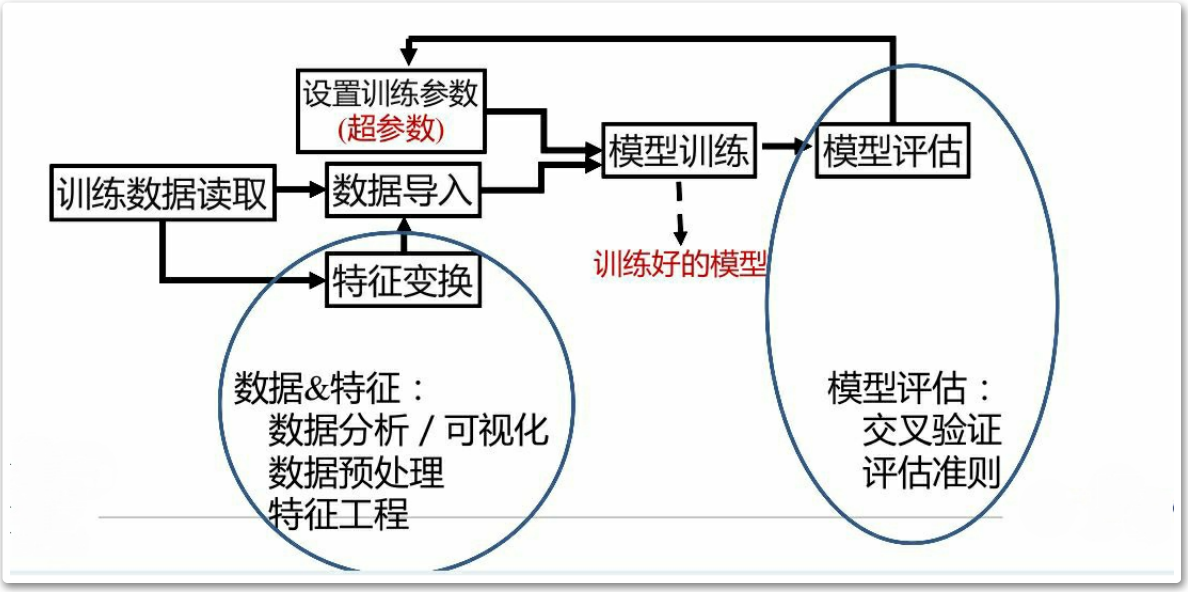
- 处理器：Intel(R) Core(TM) i7-8750H CPU@2.20GHz 2.21 GHz
- RAM：16GB
- 显卡：GeForce GTX 1060
- 显存：6G

依赖第三方库

- Pandas
- Numpy
- Skitlearn
- Xgboost
- Matplotlib
- Dask
- SMOTE

实验内容

处理流程



任务一：特征工程

特征工程及数据分析方法介绍

- **对数据进行探索性的分析的工具包：pandas、matplotlib / seaborn**
- **读取训练数据，取少量样本进行观测，并查看数据规模和数据类型– 标签、特征意义、特征类型等**
- **分析每列特征的分布**
  - 直方图
  - 包括标签列（对分类问题，可看出类别样本是否均衡）
  - 检测奇异点（outliers）

- **分析每两列特征之间的相关性**
  - 特征与特征之间信息是否冗余
  - 特征与标签是否线性相关

- **过采样技术**

  数据非常不平衡的情况下采用过采样的方式扩充数据集：SMOTE

- **再编码**

  XGBoost 模型内部将所有的问题都建模成一个回归预测问题，输入特征只能是数值型。因此对于分类变量需要转化为数值变量，其方法包括：
  - LabelEncoder： 对不连续的数字或者文本进行编号

    - 可用在对字符串型的标签编码（测试结果需进行反变换）
    - 编号默认有序数关系
    - 存储量小
  - 如不希望有序数关系： OneHotEncoder：将类别型整数输入从1维 $\Longrightarrow$ K维的稀疏编码
    - 对XGBoost，OneHotEncoder不是必须，因为XGBoost对特征进行排序从而进行分裂建树；如果用OneHotEncoder得到稀疏编码，XGBoost建树过程中对稀疏特征处理速度块
    - 输入必须是数值型数据（对字符串输入，先调用LabelEncoder变成数字，再用OneHotEncoder ）
    - 存储要求高

- **PCA降维**

  如果数据特征过多，比如几百个特征，可以采样PCA降维的方式缩减特征数量，但是如果计算资源允许的话最好是不要做PCA降维处理，因为毕竟会损失部分信息。

特征工程实施内容

- **特征变换**
  - 常住地：labelencode转化为数值
  - 手机型号：labelencode转化为数值
  - 手机ram容量：数值型暂不处理
  - ram剩余容量占比：数值型暂不处理
  - rom容量：数值型暂不处理
  - rom剩余容量占比：数值型暂不处理
  - 手机颜色：labelencode转化为数值
  - 字体大小：数值型暂不处理
  - 上网类型：onehotencode转化为稀疏特征
  - 移动运营商：onehotencode转化为稀疏特征
  - 手机系统版本（os）：数值型暂不处理
  - 以下特征均为数值暂不处理：
    - 开机次数（bootTimes）
    - 手机A特性使用次数（AFuncTimes）
    - 手机B特性使用次数（BFuncTimes）
    - 手机C特性使用次数（CFuncTimes）
    - 手机D特性使用次数（DFuncTimes）

    - 手机E特性使用次数（EFuncTimes）
    - 手机F特性使用次数（FFuncTimes）
    - 手机G特性使用情况（FFuncSum）

- **其他文件处理**

  app_info.csv、user_app_actived.csv文件关联通过pandas数据分析转换为数十个新的特征变量，特征变量为某种类app安装数量，例如影音娱乐、社交通讯等等

  user_app_usage.csv文件处理过程为:

  1. 从app_info.csv文件提取所有app应用对应的类型总数。
  2. 从user_app_usage.csv文件中提取每个用户对app应用一个月内的使用情况，每个用户使用情况保存至对应csv文件。
  3. 分别从上步得到的用户信息中提取每个用户一个月内对不同类型应用的使用次数times，使用时长duration，使用天数days等情况。一共包括40个类型的应用，以及一个其它类型的应用，并将处理结果保存至csv文件。

代码文件说明

- **文件处理及数据整理**

  代码文件:
    - 特征工程-数据整理-1.py
    - 特征工程-数据整理-2.py
    - 特征工程-大文件处理-1.py
    - 特征工程-大文件处理-2.py

- **数据分析**

  代码文件:
    - 特征工程-数据分析.py
    - 特征工程-组合特征.py

- **过采样扩充数据**

  代码文件:
    - 特征工程-过采样.py

- **PCA降维**

  代码文件:
    - 特征工程-PCA降维.py

任务二：建模调参

Scikit-Learn和Xgboost使用说明

- **Scikit-Learn: Scoring**
  - 用交叉验证（cross_val_score和GridSearchCV）评价模型性能时，用scoring参数定义评价指标。
  - 评价指标是越高越好，因此用一些损失函数当评价指标时，需要再加负号，如neg_log_loss，neg_mean_squared_error
  - 详见sklearn文档：http://scikit-learn.org/stable/modules/model_evaluation.html#log-loss

Scoring	Function	Comment
Classification		
'accuracy'	metrics.accuracy_score	
'average_precision'	metrics.average_precision_score	
'f1'	metrics.f1_score	for binary targets
'f1_micro'	metrics.f1_score	micro-averaged
'f1_macro'	metrics.f1_score	macro-averaged
'f1_weighted'	metrics.f1_score	weighted average
'f1_samples'	metrics.f1_score	by multilabel sample
'neg_log_loss'	metrics.log_loss	requires predict_proba support
'precision' etc.	metrics.precision_score	suffixes apply as with 'f1'
'recall' etc.	metrics.recall_score	suffixes apply as with 'f1'
'roc_auc'	metrics.roc_auc_score	
Clustering		
'adjusted_rand_score'	metrics.adjusted_rand_score	
Regression		
'neg_mean_absolute_error'	metrics.mean_absolute_error	
'neg_mean_squared_error'	metrics.mean_squared_error	
'neg_median_absolute_error'	metrics.median_absolute_error	
'r2'	metrics.r2_score	

- **XGBoost支持的目标函数**
  - Objective： 定义学习任务及相应的学习目标，可选的目标函数如下：
    - “reg:linear” –线性回归。
    - “reg:logistic” –逻辑回归。
    - “binary:logistic” –二分类的逻辑回归问题，输出为概率。
    - “binary:logitraw” –二分类的逻辑回归问题，输出的结果为w T x。
    - “count:poisson” –计数问题的poisson回归，输出结果为poisson分布。
    - “multi:softmax” –让XGBoost采用softmax目标函数处理多分类问题
    - “multi:softprob” –和softmax一样，但是输出的是ndata* nclass的向量，可以将该向量reshape成ndata行nclass列的矩阵。没行数据表示

样本所属于每个类别的概率。

- “rank:pairwise” –set XGBoost to do ranking task by minimizing the pairwise loss

- **XGBoost自定义目标函数**
  - XGBoost在调用obj函数时会传入两个参数：preds和dtrain
    - preds为当前模型完成训练时，所有训练数据的预测值
    - dtrain为训练集，可以通过dtrain.get_label()获取训练样本的label
    - 同时XGBoost规定目标函数需返回当前preds基于训练label的一阶和二阶梯度

```
#user define objective function, given prediction, return gradient
and second order gradient
# this is log likelihood loss
def logregobj(preds, dtrain): #自定义损失函数
    labels = dtrain.get_label()
    preds = 1.0 / (1.0 + np.exp(-preds))
    grad = preds - labels #梯度
    hess = preds * (1.0-preds) #2阶导数
    return grad, hess
#参数：obj= 'logregobj'
```

- **XGBoost支持的评价函数**

’ eval_metric’ The choices are listed below，评估指标:

- “rmse”: root mean square error
- “logloss”: negative log-likelihood
- “error”: Binary classification error rate. It is calculated as #(wrong cases)/#(all cases). For the predictions, the evaluation will regard the instances with prediction value larger than 0.5 as positive instances, and the

  others as negative instances.

- “merror”: Multiclass classification error rate. It is calculated as #(wrong cases)/#(all cases).
- “mlogloss”: Multiclass logloss
- “auc”: Area under the curve for ranking evaluation.
- “ndcg”:Normalized Discounted Cumulative Gain
- “map”:Mean average precision

- **XGBoost参数列表**

参数	说明
max_depth	树的最大深度。树越深通常模型越复杂，更容易过拟合
learning_rate	学习率或收缩因子。学习率和迭代次数 / 弱分类器数目n_estimators相关。 缺省：0.1
n_estimators	弱分类器数目. 缺省:100
slient	参数值为1时，静默模式开启，不输出任何信息
objective	待优化的目标函数，常用值有： binary:logistic 二分类的逻辑回归，返回预测的概率 multi:softmax 使用softmax的多分类器，返回预测的类别(不是概率)。 multi:softprob 和 multi:softmax参数一样，但是返回的是每个数据属于各个类别的概率。支持用户自定义目标函数
nthread	用来进行多线程控制。 如果你希望使用CPU全部的核，那就不用缺省值-1，算法会自动检测它。
booster	选择每次迭代的模型，有两种选择： gbtree：基于树的模型，为缺省值。 gbliner：线性模型
gamma	节点分裂所需的最小损失函数下降值
min_child_weight	叶子结点需要的最小样本权重（hessian）和
max_delta_step	允许的树的最大权重

参数	说明
subsample	构造每棵树的所用样本比例（样本采样比例），同GBM
colsample_bytree	构造每棵树的所用特征比例
colsample_bylevel	树在每层每个分裂的所用特征比例
reg_alpha	L1/L0正则的惩罚系数
reg_lambda	L2正则的惩罚系数
scale_pos_weight	正负样本的平衡
base_score	每个样本的初始估计，全局偏差
random_state	随机种子
seed	随机种子
missing	当数据缺失时的填补值。缺省为np.nan
kwargs	XGBoost Booster的Keyword参数

- **参数类别**
  - 通用参数：这部分参数通常我们不需要调整，默认值就好
  - 学习目标参数：与任务有关，定下来后通常也不需要调整
  - booster参数：弱学习器相关参数，需要仔细调整，会影响模型性能

- **通用参数**
  - booster：弱学习器类型 – 可选gbtree（树模型）或gbliner（线性模型） – 默认为gbtree（树模型为非线性模型，能处理更复杂的任务）
  - silent：是否开启静默模式 – 1：静默模式开启，不输出任何信息 – 默认值为0：输出一些中间信息，以助于我们了解模型的状态
  - ncpu：线程数 – 默认值为-1，表示使用系统所有CPU核

- **学习目标参数**
  - objective: 损失函数 – 支持分类 / 回归 / 排序
  - eval_metric：评价函数

- seed：随机数的种子 – 默认为0 – 设置seed可复现随机数据的结果，也可以用于调整参数

- **booster参数**
  - 弱学习器的参数，尽管有两种booster可供选择，这里只介绍gbtree
  - learning_rate : 收缩步长 vs. n_estimators：树的数目
    - 较小的学习率通常意味着更多弱分学习器
    - 通常建议学习率较小(小于0.1），弱学习器数目n_estimators大
    - 可以设置较小的学习率，然后用交叉验证确定n_estimators
  - 行（subsample）列（colsample_bytree 、colsample_bylevel）下采样比例
    - 默认值均为1，即不进行下采样，使用所有数据
    - 随机下采样通常比用全部数据的确定性过程效果更好，速度更快
    - 建议值：0.3 - 0.8
  - 树的最大深度： max_depth
    - max_depth越大，模型越复杂，会学到更具体更局部的样本
    - 需要使用交叉验证进行调优，默认值为6，建议3-10
  - min_child_weight ： 孩子节点中最小的样本权重和
    - 如果一个叶子节点的样本权重和小于min_child_weight则分裂过程结束

建模调参

步骤

1. **采用缺省参数,此时learning_rate =0.1（较大），观察n_estimators的合适范围**
   - 参数设为1500，earlystop = 50
   - cv函数在n_estimators =1200时停止

```
max_depth=5,
min_child_weight=1,
gamma=0,
subsample=0.3,
colsample_bytree=0.8,
colsample_bylevel=0.7
```

2. **max_depth 和 min_child_weight 参数调整**
   - 这两个参数对结果影响很大
   - 先大范围地粗调参数（步长为2），再小范围微调

- max_depth = range(3,12,2)
- min_child_weight = range(1,10,2)
- 初步得：

  max_depth: 10、min_child_weight: 7

3. **max_depth 和 min_child_weight 参数微调**
   - 在max_depth=10和min_child_weight=7周围微调
   - max_depth = [9,10,11]
   - min_child_weight = [6,7,8]

4. **调整max_depth=9 和 min_child_weight=8后，再次调整n_estimators**
   - 参数设为1500，earlystop = 50
   - cv函数在n_estimators =1400时停止

5. **gamma参数调整**

   缺省值（0）表现还不错，如计算资源允许，可以调整

6. **行列采样参数：subsample和colsample_bytree**
   - 这两个参数可分别对样本和特征进行采样，从而增加模型的鲁棒性
   - 现在[0.3-1.0]之间调整，粗调时步长0.1
   - 微调时步长为0.05

7. **正则参数调整： reg_alpha（L2）和reg_lambda(L0)**
   - reg_alpha = [0.1, 1] # L1, default = 0
   - reg_lambda = [0.5, 1, 2] # L2, default = 1
   - 最终得{'reg_alpha': 1, 'reg_lambda': 0.5)

8. **降低学习率，调整树的数目**
   - 调用xgboost的cv函数

     分别尝试使用0.01、0.04、0.07、0.09

```
xgb = XGBClassifier( learning_rate =0.05,
                     n_estimators=1400,
                     max_depth=9,
                     min_child_weight=8,
                     subsample=0.7,
                     colsample_bytree=0.6,
                     colsample_bylevel=0.7,
                     reg_alpha = 1,
                     reg_lambda = 0.5,
                     objective= 'multi:softprob', seed=3)
```

并行处理和GPU加速

如果数据量比较大，比如几百万几千万行样例、几百个几千个特征，使用单线程cpu或者无GPU计算是非常耗时的，所以可以采用并行处理和利用GPU

- **并行处理的场景**
  1. 交叉验证并行，XGBoost建树不并行
  2. 交叉验证不并行，XGBoost建树并行
  3. 交叉验证并行，XGBoost建树并行

- **使用样例**

```
xgbmodel=XGB(max_depth=max_depth,
                 n_estimators=n_estimators,
                 silent=False,
                 n_jobs=8,#多线程跑模型
                 min_child_weight=min_child_weight,
                 random_state=3,
                 gpu_id=0,#如果有GPU，GPU序号
                 tree_method='gpu_hist',#采用GPU
                 objective='multi:softmax',
                 predictor='cpu_predictor')#防止GPU内存不足，采用CPU预测
```

代码文件说明

以下面三个最重要的参数调参代码为例，其他参数参照这两个文件进行调参

- 建模调参-n_estimators.py
- 建模调参-max_depth-min_child_weight.py

任务三：预测导出数据

示例

通过前面的调参将参数调整好之后，利用调整好的参数即可预测导出数据，代码示例如下：

```
def pre_result(n_estimators,max_depth,min_child_weight,test_data):#只列出这三个参数，其他参数函数内调整
    xgbmodel=XGB(max_depth=max_depth,
                 n_estimators=n_estimators,
                 silent=False,
                 n_jobs=8,#多线程跑模型
                 min_child_weight=min_child_weight,
                 random_state=3,
                 gpu_id=0,#如果有GPU，GPU序号
                 tree_method='gpu_hist',#采用GPU
                 objective='multi:softmax',
                 predictor='cpu_predictor')#防止GPU内存不足，采用CPU预测
    start=time.time()
    xgbmodel.fit(X,Y,eval_metric='mlogloss')#训练模型
    end=time.time()
    print('{0}-{1}-{2}-time:{3}'.format(n_estimators,max_depth,min_child_weight,end-start))#记录模型运行时间
    xtrain_pre=xgbmodel.predict(X)
    x_train_score=accuracy_score(Y,xtrain_pre)
    end=time.time()
    print(end-start,'--训练集')#记录训练集预测运行时间
    print('训练集分数是:{}%'.format(x_train_score*100))
    test_uid=test_data[['uid','gender']]
    test=test_data.drop(['uid'],axis=1)
    test_pre=xgbmodel.predict(test)
    test_uid['label']=test_pre#构建输出结果格式
    end=time.time()
    print(end-start,'--验证集')
    test_uid=test_uid.drop('gender',axis=1)
    test_done=pd.read_csv('test_done.csv')
    test_uid=pd.concat([test_uid,test_done],ignore_index=True)#合并缺少特征数据部分已经预测的结果
    test_uid.to_csv('{0}-{1}-{2}-final-submission.csv'.format(n_estimators,max_depth,min_child_weight),index=False)
    print('{0}-{1}-{2}-submission.csv存储完成!'.format(n_estimators,max_depth,min_child_weight))
    end=time.time()
    print(end-start)
```

数据说明

- X,Y训练集特征举证和标签
- test_data:测试集数据

- test_uid.to_csv('{0}-{1}-{2}-final-submission.csv'.format(n_estimators,max_depth,min_child_weight),index=False)为导出数据接口

代码文件说明

- 建模调参-导出预测数据.py

XGBoost总结

1. **XGBoost是一个用于监督学习的非参数模型** – 目标函数（损失函数、正则项） – 参数（树的每个分支分裂特征及阈值） – 优化：梯度下降
2. **参数优化**

   – 决定模型复杂度的重要参数：learning_rate, n_estimators,max_depth, min_child_weight, gamma, reg_alpha, reg_lamba – 随机采样参数也影响模型的推广性： subsample, colsample_bytree,colsample_bylevel

3. **分布式**

   – AWS – YARN Cluster – …

4. **GPU加速**