

AI Competition 点击反欺诈预测

algorithm

分类: C4.5, Naive Bayes, SVM, KNN, AdaBoost, CART

回归问题} 常见 60%

聚类算法: K-Means, EM
手段 \Rightarrow 降维 中间过程 非最终手段
SVD PCA embedding)

关联分析: Apriori 特定场景: beer & diaper

连接分析: PageRank 网站过程中核心的影响力

LR \Rightarrow baseline

Xgboost
LightGBM

CNN RNN \Rightarrow CV & NLP

Kaggle: 1) Getting Started 2) Playground 3) Research

4) featured 难度较高

阿里云天池 百度AI DataFountain HeyWhale CCF

1) ML

2) NN

3) 启发式算法 \rightarrow 只知道结果

目标评价
loss function

权重参数

AI model process:

EDA

missing stats Label distribution unique (distinct)
is null (sumc)

data preprocess:

fill missing value; standardize data; normalization 归一化

feature engineering:

create new feature, feature transform, encode

build model:

xgboost, lightGBM, Catboost \Rightarrow ML

图像/文本分类: NN (RNN, CNN etc)

通常 lightGBM 更快, 优先使用
 \downarrow
baseline

参数调优:

贝叶斯优化 (主要调树的深度) max_depth 前期采用经验参数

模型融合:

五折交叉验证融合

不同模型之间的融合

xgboost 五个子模型 + 1/5

xgboost & lightGBM 但效果未必那么好
+ SVM NN

点击反欺诈 predict (风控): 二分类 xgboost lightGBM

sid 有用? No

EDA:

info() isnull().sum() 查看每个字段唯一值的个数 unique()
唯一值个数很少说明类别特征

ex: carrier 类别特征

特征变换，对于数值过大的异常值，设为0

LightGBM \rightarrow baseline

在 baseline 上的 improvement:

1) 利用 CSV 特征 2) 利用TimeStamp 提取时间维度 计算时间差

4) 使用五折交叉验证

LightGBM 需要对数据进行归一化处理么？

树模型不需要归一化，连续特征、类别型特征都不用归一化

- 1) 树模型不是基于距离来进行的计算，也没有利用 SGD 等优化算法优化
- 2) LightGBM 的回归树生长过程是利用特征直方图来寻找最优特征以及分簇点
 \Rightarrow 只需要关心取值顺序

使用归一化，各个样本的取值顺序不会发生改变 \Rightarrow 没必要归一化

LightGBM 只需要提前将类别映射到非负整数就可以

官方文档中建议使用从0开始的连续数值进行编码，好处是：

当某个类别特征的个数非常多时，可看成连续特征，或者进行 embedding 编码

如何利用单个模型的多个版本进行融合：

使用五折交叉验证：产生5个子模型，然后进行简单平均得出结果

```
from sklearn.model_selection import StratifiedKFold, KFold
```

StratifiedKFold 采用分层划分（分层随机抽样），即验证集中不同类别占比与原始

样本分布比例一致

stratifiedFold 在划分时，需要传入标签特征

GBM classifier 经验参数

```
clf = lgb.LGBMClassifier(num_leaves=2**5-1, reg_alpha=0.25, reg_lambda=0.25,  
objective='binary', max_depth=5, learning_rate=0.05, min_child_samples=3,  
random_state=2021, n_estimators=1000, subsample=1, colsample_bytree=1)
```

num_leaves = $2^{max_depth} - 1$ # 树的最大叶子数，对称 XGBoost 一般为 2^{max_depth}

reg_alpha L1 正则化系数

reg_lambda L2 正则化系数

max_depth 最大树的深度

n_estimators 树的个数，相当于训练的轮数

subsample 训练样本采样率 (行采样)

colsample_bytree 训练特征采样率 (列采样)

为了防止过拟合，可以抽样本做 Val，如果 Val 的分数下降，就 early stop

XGBvs LightGBM

XGBvs LightGBM 效果相比 LightGBM 可能会好一些

```
xgb=xgb.XGBClassifier(max_depth=6, learning_rate=0.05, n_estimators=2000,  
objective='binary:logistic', tree_method='gpu_hist', subsample=0.7, colsample_bytree=0.7  
min_child_samples=3, eval_metric='auc', reg_lambda=0.5)
```

objective，目标函数，binary：logistic 用于二分类任务

tree_method 使用功能树的树构建方法，hist 代表使用直方图优化的近似贪婪算法

`subsample, colsample_bytree` 是个值得调整的参数，典型的取值为 0.5-0.9
(取 0.效果可能更好)

对 XGBoost 调参：可尝试不同的 `max_depth = [6, 9, 12, 15]`

`learning_rate = 0.025` `n_estimators = [2000, 10000]`

使用 GPU 加速：`tree_method = 'gpu_hist'` `subsample = [0.7, 0.8, 0.9]`

`colsample_bytree = [0.7, 0.8, 0.9]`

Tips: 调参可能会花很多时间，因此运行时间很长，`n_estimators` 比较大，加入各种正则化参数

五折交叉验证：
1) 使用云平台的 jupyter 2) 启动 2-3 个同时运行
3) 将使用模板打印出来 `print(xgb)`
4) 结果文件带上模型的参数

data 归一化使用：1) 和距离相关的 model 中 need use, 比如 LR, SVM, KNN

神经网络建模前进行数据归一化很重要

2) 和距离无关的模型中不需要，比如树模型

神经网络用桶型或塔型

how to avoid 过拟合：设置 `regularizers.l2(0.001)`

`Keras.layers.Dense(200, activation='relu', kernel_regularizer=regularizers.l2(0.001))`

在代码中的 tips: ex: 多版本结果管理

关闭 warnings `import warnings`

`warnings.filterwarnings("ignore")`

二分类NN模型

```
from tensorflow import keras  
from tensorflow.keras.optimizers import Adam  
from tensorflow.keras import regularizers
```

#搭建模型

```
model = keras.Sequential([keras.layers.Dense(200, activation='relu'),  
    input_shape=[len(features.Columns)]), keras.layers.Dense(200, activation='relu'),  
    keras.layers.Dense(200, activation='relu'),  
    keras.layers.Dense(1, activation='sigmoid') #将改用sigmoid])
```

```
model.compile(loss='binary_crossentropy', metrics=['accuracy'], optimizer=adam)
```

#batch_size 根据情况而选择，epochs 可以适当增加一些
model.fit(features, labels, batch_size=10240, epochs=50)

#如果有验证集，可以进行设置

```
model.fit(features, labels, validation_data=(test_feat, test_ans), batch_size=10240,  
    epochs=50)
```