**第二届易观算法大赛——性别年龄预测思路简介**

**团队：江湖交流**

**队员： Fred flyfoxs**

**比赛过程简介**

本大赛从2018年8月29日正式开始，至10月16日结束。在10月15日前，flyfoxs的最高成绩是2.58， Fred的最高成绩是2.569，. 两人于10月16日上午组队成功，经flyfoxs实现简单加权平均后，将成绩提高至 5.559.后经双方合作调优后，最后成绩锁定在5.55832。结束比赛。

**算法特点：**

Flyfoxs主要采用特征优化的方式提高成绩。

Fred的思路主要采用了 多类型神经网络和LightGBM的组合提高成绩。

**Flyfoxs思路介绍**

**一、特征提取算法部分：**

**1、LDA方法使用**

照抄neuronblack同学的算法,使用LDA降维的确效果很不错 <https://github.com/neuronblack/yiguan>。但是该同学仅仅依赖于app的安装与否来进行的LDA判断,我对他从使用角度进行了扩展,比如APP的点击数,APP的使用时长.

**2、word2vec 方法使用**

使用gensim计算每个app的向量, 参数用这个向量训练CNN和LSTM,效果不好,放弃了.但是在后面的APP分类中使用效果不错

**3、KNN/KMeans**

主办方给的数据2/3没有APP分类.

最初我是用Kmeans,忽略主办方给的分类信息,对APP进行分类,发现不管分多少类,用XGB来训练都没什么提高.

后来用KNN来补上APP分类信息的缺失,基于之前的word2vec来计算距离.

**4、TFIDF**

在3个维度计算了TFIDF, 一个是 APP, APP分类##1, APP分类##2, 每个维度分别在2个方向计算TFIDF: app打开的次数, APP使用时间.TFIDF维度依旧比较高,后面会继续对齐进行降维.

**5、SVD**

之前使用了word2vec来对APP做降维处理, 但是word2vec是依赖于app出现的上下文顺序,这个似乎和我们当前的场景不太合适.所以同时选择了用SVD来对上面计算所得的TFIDF做降维.

**二、主要特征介绍**

**1、24小时分片统计APP**

一天24小时,每个APP使用占对应Device使用的百分比, 分别对点击次数和时长计算百分比

**2、按照星期来统计APP**

一周7天,每天的APP使用分布情况

工作日和周末APP使用情况分布及对比

**3、删除低频APP**

删除低频APP后做统计, 这个有点微弱的效果

**三、主要模型介绍**

**1、XGB**

单独对年龄,性别预测. 然后作为DNN模型的输入来参与预测.

不能用年龄和性别做概率运算,这样得到的结果特别差.有人分析过,应该是在这些数据中年龄和性别并不是独立的2个维度.

这个模型做为基模型对性别年龄来预测,效果是比较好的.

**2、LGB**

作为一个树模型的补充,不知道为什么一直调优总是比xgb差一些. 和DNN的结果比较接近

**3、DNN**

单独使用手工提取出来的维度,参与Training也能得到不错的成绩. 但是把树模型Training的结果合并在一起作为输入,分数有比较大的提高.

**4、CNN/LSTM**

都尝试过,效果不太好放弃了

**5、模型融合**

做的比较简单,就是把之前模型中每一个挑出比较好的一个,使用DNN来融合.

**Fred思路介绍**

**一、特征处理**

特征处理主要包括以下几部分：

1. 特征类型

将手机厂商，手机型号，是否安装package，app类型，每日每小时各app点击量，每小时各app日均点击量占比。

1. 缺失值处理

对于类型缺失的package，使用平均日点击量比较接近的package去填充

为了提高精准度，根据数据中的手机型号（估计是http agent里提取的），对手机厂商和手机型号进行了手工调整。把仅有型号信息，没有厂商信息的记录全部手工补全。

1. 提取算法

采用了传统的词袋模型，TfidfVectorizer，Word2Vec方式。

**二、算法优化方向确定**

先后尝试了以下算法：LightGBM，One-Hot + MLP （1层神经网络）、LeNet5 结构的CNN、TextCNN、LSTM，以及CNN+RNN 网络，并简单调优后，发现One-Hot + MLP 表现最优（2.61971）和LightGBM(2.62197)。同时为了保持算法多样化，决定将 2-3层的神经网络和LightGBM作为重点提高方向。

**三、模型借鉴**

**1、stack模型借鉴**

本文详细描写了stack的原理，常用的方法等。决定采用stack方式。

[**http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/**](http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/)

**2、同类比赛借鉴**

**在kaggle上找到类似比赛**

[**https://www.kaggle.com/c/talkingdata-mobile-user-demographics**](https://www.kaggle.com/c/talkingdata-mobile-user-demographics) **，并研究了**

**place3 、place5和 place 11的经验分享，发现place3是用的文本处理的方法，place5的分享比较水，没啥收获。发现place 11的分享和我的判断非常一致，均是使用2-3层的深度网络和boost树结构（xgb）进行处理。重点推荐place11。 因为place11的分享是在github上，所以有些代码直接copy来用。感谢 无私的 ellnzvju。 回头争取联络一下。**

**链接分享下：**

[**https://www.kaggle.com/c/talkingdata-mobile-user-demographics/discussion/23465**](https://www.kaggle.com/c/talkingdata-mobile-user-demographics/discussion/23465)

[**https://www.kaggle.com/c/talkingdata-mobile-user-demographics/discussion/23445**](https://www.kaggle.com/c/talkingdata-mobile-user-demographics/discussion/23445)

[**https://github.com/ellnzvju/Kaggle\_Talking\_Data**](https://github.com/ellnzvju/Kaggle_Talking_Data)

**四、模型设计和调优**

**最终经过反复调优，最终确定以下模型结构**

