课程成绩:
任课教师：郑子彬

中山大學

《数据挖掘项目实验报告》

题目: 基于移动网络通讯行为的风险用户识别

完成时间: 2018.6

学院，专业: 数据科学与计算机学院

计算机科学与技术专业

用户名: Dm15336195

成绩: 0.762272

排名: Rank = 128st / 211

组员&贡献率: 吴立渝

1. 实验平台:

   语言:python – jupyter notebook

   模块:pandas、numpy、lightgbm、sklearn、seaborn、matplotlib

2. 项目目标:

   以模拟的语音通话、短信收发、网站及 App 访问记录等移动网络使用行为为基础，通过数据提取技术和机器学习算法，构建识别风险用户的预测模型，判别用户属于风险用户的可能性。从而为各行业提供风控保障，助力新时代大数字生态的健康有序发展。

3. 项目内容:

   关于本次项目的数据详情:

字段名称	示例	说明
uid	u0001	代表一个手机用户
opp_num	B80A495E52A497684 66B7AB00CA2B5C7	对端通话号码： 已作加密处理，每个号码加密后保持唯一、一致。
opp_head	186	对端号码的前 n 位： 如号码长度>5，则取前 3 位，如： 18612345678->186，01081234567->010 如号码长度<=5，则取前 1 位，如： 10010->1，95555->9，120->1
opp_len	11	对端号码的长度位数，如： 18612345678->11，81234567->8，10010->5，120->3
start_time	01081045	通话发起时间，格式为 DD-HH-MM-SS，如： 01081045 代表第 01 天 08 点 10 分 45 秒
end_time	01081532	通话结束时间，格式同上
call_type	1	通话类型：1-本地，2-省内长途，3-省际长途，4-港澳台长途，5-国际长途。
in_out	0	通话的主被叫类型： 0 为 uid 发起的主叫；1 为 uid 被叫。

字段名称	示例	说明
uid	u0001	代表一个手机号用户
opp_num	B80A495E52A49768 466B7AB00CA2B5C7	对端短信号码： 已作加密脱敏，每个号码加密后保持唯一、一致。
opp_head	186	对端号码的前 n 位： 如号码长度>5，则取前 3 位，如： 18612345678->186，01081234567->010 如号码长度<=5，则取前 1 位，如： 10010->1，95555->9，120->1
opp_len	11	对端号码的长度位数，如： 18612345678->11，81234567->8，10010->5，120->3
start_time	01081045	短信发送时间，格式为 DD-HH-MM-SS，如： 01081045 代表第 01 天 08 点 10 分 45 秒
in_out	0	短信的发送/接收类型： 0 为 uid 发送；1 为 uid 接收。

字段名称	示例	说明
uid	u0001	代表一个手机号用户
wa_name	中国联通手机营业厅	访问的网站或 App 名称（每条记录对应一个网站或 App）
visit_cnt	3	当天访问该网站或 App 的次数
visit_dura	152	当天访问该网站或 App 的总时长，单位：秒
up_flow	683	当天访问该网站或 App 的总上行流量，单位：B
down_flow	2539	当天访问该网站或 App 的总下行流量，单位：B
wa_type	1	网站或 App 区分：0-网站，1-App
date	01	日期，格式为 DD，如 01 代表第 01 天

- 数据预处理:

首先，分别读入本次实验给出的数据，为了提高效率，我们使用 pandas 库中的 concat 函数将训练集和测试集合并同时提取特征。

在合并后，分别观察给出的特征以及合并后是否为 dataframe 对象，方便之后的数据处理。(具体代码会在附件中)

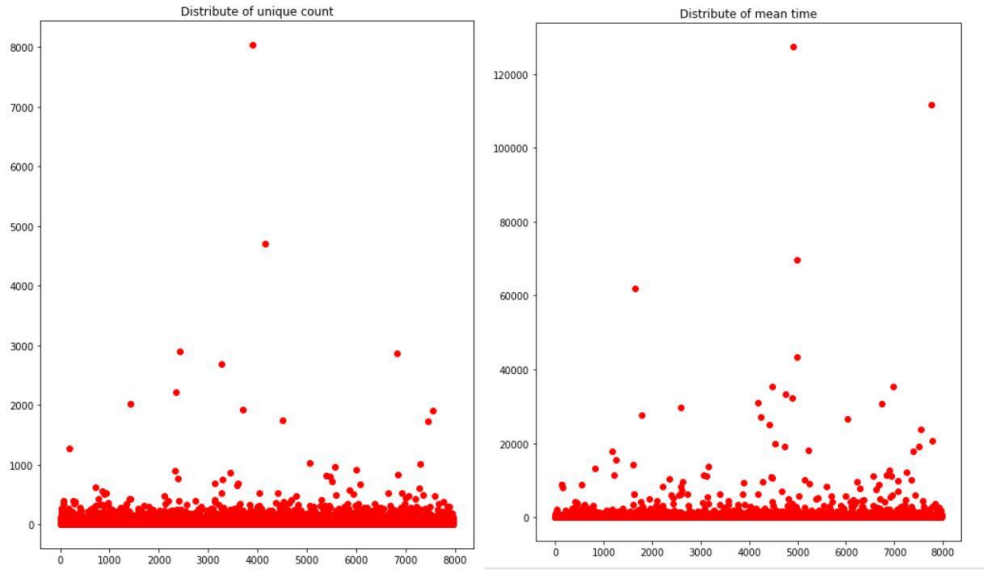
```
In [6]: voice = pd.concat([voice_train, voice_test], axis=0)
        sms = pd.concat([sms_train, sms_test], axis=0)
        wa = pd.concat([wa_train, wa_test], axis=0)

        print(voice.head(3))
        print(type(voice))
        print(sms.head(3))
        print(type(sms))
        print(wa.head(3))
        print(type(wa))
```

```
    uid                             opp_num opp_head  opp_len  start_time  \
0  u0113  38D54642A237A11BB18455FC1E505292      132       11    26115956
1  u0113  38D54642A237A11BB18455FC1E505292      132       11    26115623
2  u0113  38D54642A237A11BB18455FC1E505292      132       11    26174233

   end_time  call_type  in_out
0  26120033          1       1
1  26115707          1       1
2  26174321          1       1
<class 'pandas.core.frame.DataFrame'>
    uid                             opp_num  opp_head  opp_len start_time  \
0  u4003  B378E065731B897E7295926B27CBA0D5       186       11   20174042
1  u4003  B378E065731B897E7295926B27CBA0D5       186       11   20174130
2  u4003  1B15607F3E6D167B44D46046D5993D87       189       11   20015746

   in_out
0       1
1       1
2       0
<class 'pandas.core.frame.DataFrame'>
    uid wa_name  visit_cnt  visit_dura  up_flow  down_flow  wa_type date
0  u0001   155导航        5.0       207.0    313.0      457.0      0.0   14
1  u0001   155导航        7.0       396.0    547.0      659.0      0.0   04
2  u0001   155导航       10.0      3212.0    781.0      941.0      0.0   12
<class 'pandas.core.frame.DataFrame'>
```

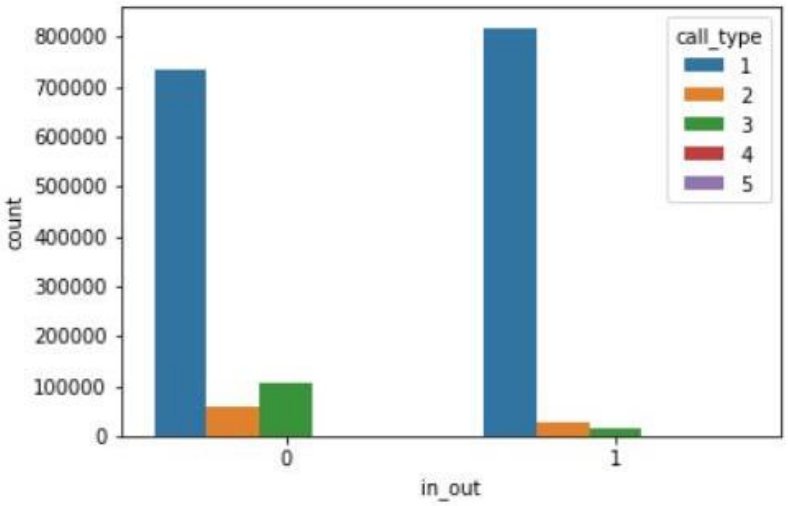
✓ voice:

关于 voice 模块，从中提取 7 个特征，分别为 voice_opp_count、voice_opp_unique_count、voice_opp_head_ unique_count、voice_opp_len、voice_mean_time、voice_call_type、voice_in_out。

其中，我们将 voice_opp_unique_count、voice_mean_time 的分布进行可视化，查看是否有明显的异常点。可以看到有少数的数据分布较突出，所以在模型训练时，参数需要稍作调正避免因异常点造成分类问题，目前 python 中 sklearn 或本次使用的 lightgbm 都有考虑对异常点的包容



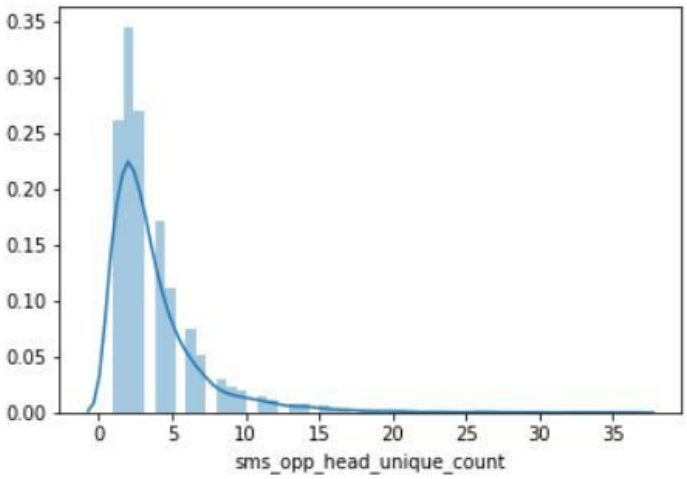
下突为根据 in_out 分类进行 call_type 的统计，可以看出不管是打进或打出，本地电话占大多数。



✓ sms:

关于 sms 模块，从中提取 5 个特征，分别为 sms_opp_num_unique_count、sms_opp_num_count、sms_opp_head_unique_count、sms_opp_len、sms_in_out。

使用 seaborn 中 histplot 直方图，对 sms_opp_head_unique_count 进行统计可视化，可以发现大部分短信的接收端的前缀码集中在 1~5 个号码之间。

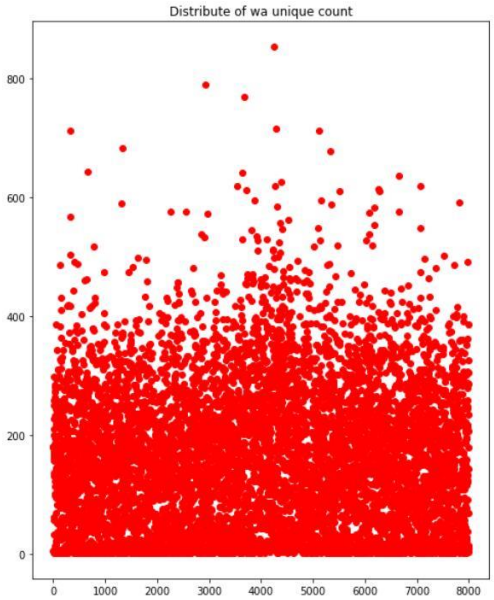


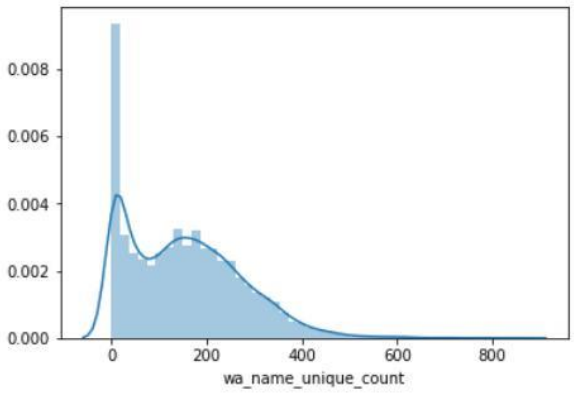
✓ wa:

关于 wa 模块，从中提取 6 个特征，分别为 wa_name_unique_count、wa_name_count、visit_cnt、visit_dura、up_flow、down_flow。

其中在 visit_cnt、visit_dura、up_flow、down_flow 几个特征集中在分别细化并计算他们的标准偏差、最小最大、中位数、平均、总数，这里也使用道聚合函数 agg 的功能，同时执行统计的操作。

关于每个 uid 访问不同网站总数的分布，可以观察到大部分都均匀分布在 0~450 之间的数量，图中有部分较突出的数据点可能为异常点或是风险用户。





- LightGBM 模型训练:

本次使用的分类是基于 lightGBM 构建的模型。LightGBM 是一个梯度 boosting 框架，使用基于学习算法的判定树。它可以说是分布式的，高效的。

✓ 调整参数:

为了得出较准确的模型参数，透过 sklearn 中的 train_test_split 将训练集中的数据再分成训练集跟测试集，再透过反复的交叉验证得出较好的结果并记录参数值。

```
In [3]: from sklearn.cross_validation import train_test_split, KFold
        dy = train['label']
        dx = train.drop(['uid', 'label'], axis = 1)
        X_train, X_test, y_train, y_test = train_test_split(dx, dy, test_size=0.3, random_state= 1)

        lgb_train = lgb.Dataset(X_train,y_train, free_raw_data=False)
        lgb_test = lgb.Dataset(X_test,y_test, free_raw_data=False)
```

首先，给定初始的参数值，具体在交叉验证后再做调整。

```
lgb_params =  {
    'boosting_type': 'gbdt',
    'objective': 'binary',
     'metric': 'auc',
    'is_training_metric': False,
    'min_data_in_leaf': 12,
    'num_leaves': 64,
    'learning_rate': 0.05,
    'feature_fraction': 0.8,
    'bagging_fraction': 0.8,
    'verbosity':-1,
}
```

lightGBM 使用的是 leaf-wise 的算法，因此在调节树的复杂程度时，使用的是 num_leaves 而不是 max_depth。所以我透过 loop 循环语句，分别带入不同数目的 num_leaves 以及 max_depth 进行计算，将得出最好的结果的参数计入我们训练模型的参数中，其中，在模型的损失函数上 metrics 使用的是 auc。代码如下:

```
print('交叉验证')
min_merror = float('Inf')
best_params = {}
print("调参1：提高准确率")
for num_leaves in range(20,200,5):
    for max_depth in range(3,8,1):
        params['num_leaves'] = num_leaves
        params['max_depth'] = max_depth

        cv_results = lgb.cv(
                            lgb_params,
                            lgb_train,
                            seed=2018,
                            nfold=3,
                            metrics=['auc'],
                            early_stopping_rounds=10,
                            verbose_eval=True
                            )

        mean_merror = pd.Series(cv_results['auc-mean']).min()
        boost_rounds = pd.Series(cv_results['auc-mean']).argmin()

        if mean_merror < min_merror:
            min_merror = mean_merror
            lgb_params['num_leaves'] = num_leaves
            lgb_params['max_depth'] = max_depth

#params['num_leaves'] = best_params['num_leaves']
#params['max_depth'] = best_params['max_depth']
lgb_params['num_leaves'] = best_params['num_leaves']
lgb_params['max_depth'] = best_params['max_depth']
```

✓ 模型训练:

分别进行两次训练，第一次在给出初始化参数构件模型并预

测；第二次在调整参数之后再进行训练查看是否有提高准确率。

基本上 lightGBM 给出的训练语句为

```
lgb.cv(lgb_params, dtrain, feval=evalMetric, early_stopping_rounds=10, verbose_eval=5, num_boost_round=10000, nfold=3, metrics=['evalMetric'])

model =lgb.train(lgb_params, dtrain, feval=evalMetric, verbose_eval=5, num_boost_round=300, valid_sets=[dtrain])

pred=model.predict(test.drop(['uid'], axis=1))
```

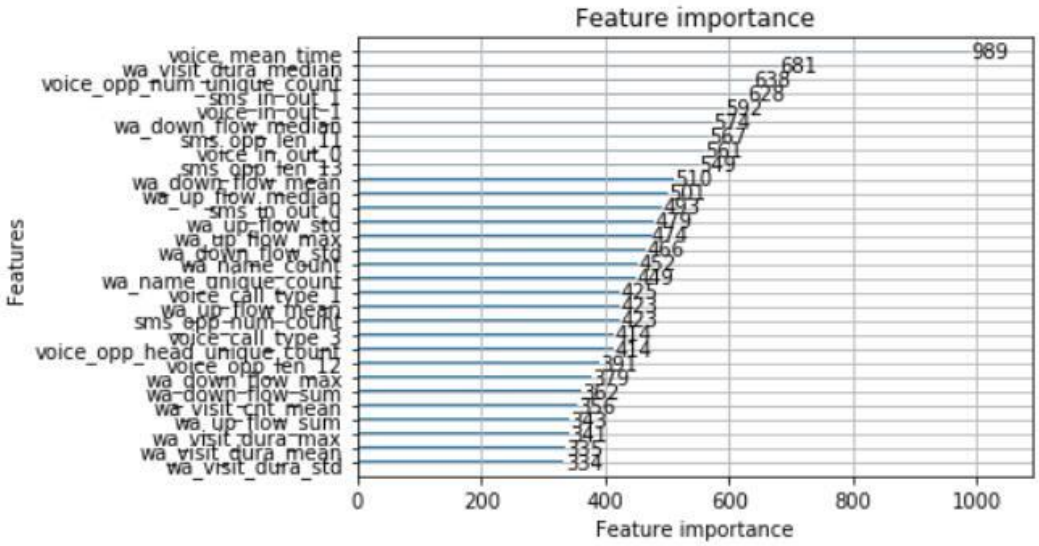
在 feval 参数中使用我们自己定义的函数 evalMetric，在这个函数中计算本次项目的评定标准 score=0.6×auc+0.4×F1，并用这个标准来训练我们的模型，并且下图也给出我们训练每轮迭代的结果。

```
def evalMetric(preds, dtrain):

    label = dtrain.get_label()


    pre = pd.DataFrame({'preds':preds,'label':label})
    pre= pre.sort_values(by='preds',ascending=False)

    auc = metrics.roc_auc_score(pre.label, pre.preds)

    pre.preds=pre.preds.map(lambda x: 1 if x>=0.5 else 0)

    f1 = metrics.f1_score(pre.label, pre.preds)


    res = 0.6*auc +0.4*f1

    return 'res', res, True
```

```
[175]   cv_agg's res: 0.781002 + 0.0116917
[180]   cv_agg's res: 0.780569 + 0.0117981
[185]   cv_agg's res: 0.77877 + 0.0125999
[190]   cv_agg's res: 0.778653 + 0.0132349
[195]   cv_agg's res: 0.777811 + 0.0123092
[200]   cv_agg's res: 0.776879 + 0.012834
[205]   cv_agg's res: 0.779273 + 0.0121625
[210]   cv_agg's res: 0.778251 + 0.0118564
[215]   cv_agg's res: 0.778323 + 0.0130537
[220]   cv_agg's res: 0.778827 + 0.0119821
[225]   cv_agg's res: 0.779687 + 0.0129078
[230]   cv_agg's res: 0.779691 + 0.012443
[235]   cv_agg's res: 0.778554 + 0.0136538
[240]   cv_agg's res: 0.77904 + 0.0137036
[245]   cv_agg's res: 0.777838 + 0.0138044
[250]   cv_agg's res: 0.777557 + 0.0152657
[255]   cv_agg's res: 0.776984 + 0.0152406
[260]   cv_agg's res: 0.777082 + 0.0164029
```

可以根据以下代码来查看我们训练模型时特征的重要性的可视化，方便我们做特征的删减。

```
plt.figure(figsize=(12,15))
lgb.plot_importance(model, max_num_features=30)
plt.title("Feature importance")
plt.show()
```



4. 项目排名、结果:

5. 项目心得与改进处:

本次项目是基于移动网络通讯行为的风险用户识别的实践，透过对实际案例的操作，帮助我更快掌握数据挖掘应用的方法以及增加本身对项目的分析和手作能力。

主要使用 python 中的机器学习模块 sklearn、lightGBM 等等，一些对数据操作的模块 pandas、numpy，以及数据可视化操作 matplotlib、seaborn。

项目主要分成两个部分完成，分别为数据整理、特征提取和分类模型的训练及预测。数据整理使用 dataframe 对象并在该对象上构建特征，这部有参考课件或相关案例等学习一些特征提取做法和想法，在构建训练模型时主要使用基于 lightGBM 的判定树算法，并在该模型上进行参数调整。

虽然最后顺利完成本次案例，但仍碰到些许困难并觉得有多处地方可以改进以及学习。首先，对 lightGBM 并未了解透彻，其中包括在构建该模型时所调用函数中并没有了解所有的参数以及作用，所以部分使用默认参数值，也参考多数现有案例的方案，导致降低了调参过程的效率以及准确性；第二，由于直接调用现有的函数训练，所以对算法实际的工作情况只有粗略的了解。这些部分会在课后自行学习和练习。