

数据挖掘实训第一次周报

陈铭涛
16340024

April 10, 2019

1 甜橙金融杯内容介绍

比赛的内容是根据包含用户操作和用户交易的信息以及用户的黑白标签的训练数据通过训练学习来对包含用户操作和交易数据的测试集中的用户进行用户风险识别。

操作和交易数据的字段信息由比赛介绍页面获取，操作包含了用户操作使用的设备、ip 地址等信息，交易包含了用户的交易资金额、交易平台、对方商户等信息。

比赛的评分标准是使用 ROC 曲线上三种不同的 False Positive 率下的 True Positive 率加权后获得，其计算代码在比赛介绍页面已给出。

2 数据分析与处理

比赛给出的数据包含较多信息，需要进行分析与处理后才能用于模型的训练。已进行的部分分析与处理如下：

1. 数据中给出了交易中的商家信息，因此可以统计出与商家交易的用户中黑用户的比例，作为训练中的特征，在这次的处理中将每个用户交易过的所有商家的黑用户比例的最大值，最小值以及平均值作为训练中使用的特征。

```
def prepare_ratio_data(transaction_df, label, name):
    col = transaction_df.groupby('UID')[name]
    category_sum = dict()
    category_black_count = dict()
    def manage_categories(m):
        uid = m.name
        m = m.unique()
        for c_id in m:
            category_sum[c_id] = category_sum.get(c_id, 0) + 1
            if labels_dict[uid] == 1:
                category_black_count[c_id] = category_black_count.get(c_id, 0) + 1
    col.apply(manage_categories)
```

```

        return {m_id: category_black_count.get(m_id, 0) / m_count for m_id,
                m_count in category_sum.items()}

merchants_ratio = prepare_ratio_data(transaction_df, label, 'merchant'),

```

2. 数据中有交易和操作的日期信息，统计每个日期下的操作与交易数量可获得类似如下的信息：

```

In [110]: (transaction_test.groupby('day').size()[:10],
          transaction_df.groupby('day').size()[:10])

Out[110]: (day
1         2620
2         3092
3         3091
4         2869
5         3473
6         8619
7         6821
8         3637
9         2580
10        2731
dtype: int64, day
1         17767
2         11241
3          3086
4          2826
5          4676
6          3309
7          3437
8          20130
9          10085
10          3304
dtype: int64)

```

不难看出在以 7 天为周期的时间范围内有两天的交易和操作量较大，可以认为这两天为周末时间，因此可以看出在测试集上第一天为周一，在训练集上第一天为周六，由此可以计算每个用户在周末与工作日中进行交易和操作的比率，作为训练的特征。

3. 数据中给出交易和操作的时间，由此可以对用户在每天各个小时时段中操作和交易的比例进行统计作为训练特征。
4. 对数据中的 geo_code 字段使用 geohash 库进行经纬度的解码，将用户交易和操作的经纬度统计数据作为训练特征。

```

geo = transaction_df['geo_code'].apply(lambda x: ((x, x) if
        pd.isna(x) else geohash.decode(x)))
transaction_df['latitude'] = geo.apply(lambda x: x[0])
transaction_df['longitude'] = geo.apply(lambda x: x[1])

```

5. 操作数据中部分设备 (device1) 有着异常高的操作次数，较之平均的 500 次左右的操作这些设备的操作次数达到了数万，在对数据进行处理时把来自这些设备的操作和交易记录排除。
6. 使用 nunique 对操作中用户使用的不同的 os，版本号，设备等数量，以及用户交易中的不同的对方商户，转入转出账号等信息进行处理，作为训练特征。

3 训练模型

所使用的模型为 LightGBM，使用的参数如下：

```

params = {
    'boosting_type': 'gbdt',
    'objective': 'binary',
    'num_leaves': 200,
    'max_depth': -1,
    'learning_rate': 0.1,
    'min_child_samples': 100,
    'n_estimators': 2000,
    'learning_rate': 0.05,
    'scale_pos_weight': float(label[label.Tag==0].shape[0]) /
        label[label.Tag==1].shape[0],
    'boost_from_average': True,
    'min_child_weight': 1e-3,
    'subsample_for_bin': 20000,
    'max_bin': 512,
    'metric': 'auc',
    'reg_alpha': 3,
    'reg_lambda': 5,
    'subsample': 0.9,
    'colsample_bytree': 0.7,
    'subsample_freq': 1,
    'n_jobs': -1,
}

```

在题目给出的评分代码上修改为如下的函数用于传入 sklearn 的 cross_validate 进行交叉验证:

```

def tpr_scorer(estimator, x, y_true):
    y_predict = estimator.predict_proba(x)[:, 1]
    d = pd.DataFrame()
    d['prob'] = list(y_predict)
    d['y'] = list(y_true)
    d = d.sort_values(['prob'], ascending=[0])
    y = d.y

    PosAll = pd.Series(y).value_counts()[1]
    NegAll = pd.Series(y).value_counts()[0]
    pCumsum = d['y'].cumsum()
    nCumsum = np.arange(len(y)) - pCumsum + 1
    pCumsumPer = pCumsum / PosAll
    nCumsumPer = nCumsum / NegAll
    TR1 = pCumsumPer[abs(nCumsumPer-0.001).idxmin()]
    TR2 = pCumsumPer[abs(nCumsumPer-0.005).idxmin()]
    TR3 = pCumsumPer[abs(nCumsumPer-0.01).idxmin()]
    return 0.4 * TR1 + 0.3 * TR2 + 0.3 * TR3

clf = lgb.LGBMClassifier(**params)
kfold = sklearn.model_selection.StratifiedKFold(splits, shuffle=True)



```

```
cv_score = sklearn.model_selection.cross_validate(clf, x, y, cv=kfold,
    scoring={
        'tpr': tpr_scorer,
        'accuracy': 'accuracy',
        'f1': 'f1_micro',
        'roc_auc': 'roc_auc'
    }, return_train_score=True)
```

4 目前结果

```
In [53]: run_cross_validation(train_data.drop('Tag', axis=1).values, train_data.loc[:, 'Tag'].values)

fit_time: [12.17515182 15.04866099 17.74484611 15.62157702 12.12144828]
Average fit_time: 14.542337
score_time: [0.31068516 0.48012209 0.54674673 0.35195208 0.38839984]
Average score_time: 0.415581
test_tpr: [0.8470245 0.83897316 0.86744457 0.83337223 0.84865811]
Average test_tpr: 0.847095
train_tpr: [1. 1. 1. 1. 1.]
Average train_tpr: 1.000000
test_accuracy: [0.97354073 0.97338037 0.97754971 0.97498396 0.97401764]
Average test_accuracy: 0.974694
train_accuracy: [0.99971936 0.99975945 0.99979954 0.99975945 0.99963919]
Average train_accuracy: 0.999735
test_f1: [0.97354073 0.97338037 0.97754971 0.97498396 0.97401764]
Average test_f1: 0.974694
train_f1: [0.99971936 0.99975945 0.99979954 0.99975945 0.99963919]
Average train_f1: 0.999735
test_roc_auc: [0.99117836 0.99114799 0.99387566 0.99026531 0.99015545]
Average test_roc_auc: 0.991325
train_roc_auc: [0.99999995 1. 0.99999997 0.99999998 0.99999997]
Average train_roc_auc: 1.000000
```

排名	排名变化	团队logo	队名	最高得分
 284	-		DM16340024	0.58108

在训练集上进行交叉验证获得的平均分数为 0.847，提交后的分数为 0.581，较之 baseline 的 0.528 的分数有一点提升。

5 TODO

当前所使用的特征基本都为统计特征，但是测试集与训练集的数据的收集时间不同，有些训练集中的数据不能在测试集中得到反映，测试集中的一些商家和 mode 等数据也在训练集中不存在，目前的处理方法是对涉及到这些数据的地方填入训练集对应数据的平均值。但是这样子做的效果并不是很好，在复赛测试集上获得的结果更加差。

接下来可以进行的改进有：

1. 对于 WiFi, MAC 地址, IP 地址等字段进行统计特征处理。
2. 基于时间和操作与交易的地理位置检查用户是否有短时间出现在多个较远的地理位置的情况，这些用户可能使用了代理或其他技术篡改其地址，更为可疑。

3. 对 os, version 等字段统计其在数据中的占比, 作为流行度特征。
4. 根据时间信息统计用户操作和交易的时间间隔作为其操作行为模式。
5. 统计交易商户的交易地址与交易类型, 若变化较为频繁则商家较为可疑。
6. 使用关联图谱等其他方法探索存在的更多特征。