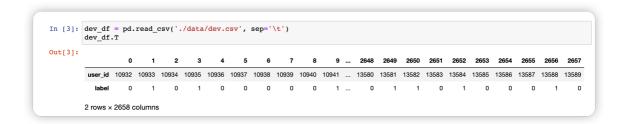
# 毕业设计

某闯关类手游用户流失预测

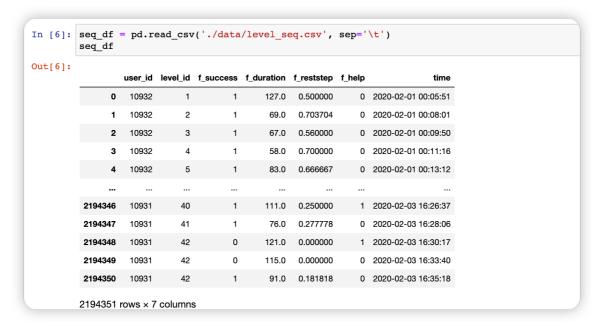
2022年8月22日

### 1. 构建数据集

训练数据一共有8158 个用户,其中流失用户大约占 1/3。同样读取验证集和测试集。



#### 读取用户游戏记录数据



用value\_counts()可以看出各用户的记录次数

```
seq df['user id'].value counts()
4963
         4122
         3893
7884
         2938
12822
11238
         2548
5502
         2369
         . . .
6807
             2
862
            2
397
            2
4860
            2
4283
Name: user id, Length: 13589, dtype: int64
```

将数据按user id分组

```
In [8]: group_user = seq_df.groupby('user_id')
```

提取出与流失率相关的特征,如最高关卡,通关次数,重试次数,总时间,总 共使用提示的次数,总剩余步数,最后一次记录的时间等。

```
In [13]: # 最后一次记录的时间
last_time = group_user['time'].max().apply(lambda x : time.mktime(time.strptime('2020-02-04 23:59:59', "%Y-%m-%d %H:%M:

In [14]: # 最高关卡
    max_level = group_user['f_success'].sum()
    # 重证次数
    sum_success = group_user['f_success'].count() - group_user['f_success'].sum()

In [16]: # 总时间
    sum_duration = group_user['f_duration'].sum()
    # 使用帮助的次数
    sum_help = group_user['f_help'].sum()
    # 总剩余步数
    sum_step = group_user['f_reststep'].sum()

In [17]: # 最后一次玩是否通关
    last_success = group_user.last()['f_success']
```

另外考虑到最近的记录与流失率关系较大、因此也提取出最后一天的特征。

```
In [18]: # 最后一天的记录次数
last_day_play = group_user['time'].apply(lambda x : sum(x > '2020-02-04 00:00:00'))
# 最后一天使用帮助的次数
last_day_help = group_user.apply(lambda x : sum(x['f_help'][x['time'] > '2020-02-04 00:00:00']))
# 最后一天的时间
last_day_duration = group_user.apply(lambda x : sum(x['f_duration'][x['time'] > '2020-02-04 00:00:00']))
```

#### 用这些数据来构建数据集

```
In [21]: def create_dataset(df):
    df['max_level'] = df['user_id'].apply(lambda x: max_level[x])
    df['sum_success'] = df['user_id'].apply(lambda x: sum_success[x])
    df['sum_retry'] = df['user_id'].apply(lambda x: sum_retry[x])
    df['sum_duration'] = df['user_id'].apply(lambda x: sum_duration[x])
    df['sum_help'] = df['user_id'].apply(lambda x: sum_help[x])
    df['last_time'] = df['user_id'].apply(lambda x: last_time[x])
    df['sum_step'] = df['user_id'].apply(lambda x: last_success[x])
    df['last_success'] = df['user_id'].apply(lambda x: max_retry[x])
    df['last_day_play'] = df['user_id'].apply(lambda x: last_day_play[x])
    df['last_day_help'] = df['user_id'].apply(lambda x: last_day_help[x])
    df['last_day_duration'] = df['user_id'].apply(lambda x: last_day_duration[x])
    return df
In [22]: train_df = create_dataset(train_df)
```

### 数据集展示为:

23]:		user_id	label	max_level	sum_success	sum_retry	sum_duration	sum_help	last_time	sum_step	last_success	max_retry	last_day_play	last_day_help
	0	2774	0	134	136	79	25398.0	18	3256.0	40.647103	0	116	31	3
	1	2775	0	116	82	29	18839.0	14	9369.0	28.688567	0	81	35	3
	2	2776	1	123	44	25	6119.0	1	122486.0	12.871456	0	68	0	0
	3	2777	0	164	145	141	40808.0	4	13053.0	35.533978	0	164	49	2
	4	2778	1	122	109	53	32045.0	9	134536.0	48.510957	0	84	0	0
ε	3153	10927	1	207	177	173	34697.0	10	10823.0	58.264718	0	195	54	3
ε	3154	10928	1	48	50	0	4073.0	3	193348.0	23.560141	1	26	0	0
ε	3155	10929	0	122	114	129	28858.0	14	12934.0	28.663942	0	97	39	4
ε	3156	10930	1	39	37	2	6120.0	6	99241.0	14.827311	1	33	0	0
ε	3157	10931	0	170	44	12	6193.0	6	113081.0	15.937665	1	40	0	0

## 2. 建模分析

这里使用了四个模型,分别是XGBoost, Random Forest, SVM, KNN。

1) XGBoost 是基于决策树Boost集成的模型,在训练时,通过十则交叉验

证,计算平均的auc值。

```
XGBoost
In [23]: columns = train_df.columns
    columns = columns.drop(['label', 'user_id'])
In [24]: from xgboost import XGBClassifier

In [25]: from sklearn.model_selection import cross_val_score
    xgb_model = XGBClassifier(objective='binary:logistic', max_depth=2, min_child_weight=140, gamma=0.01, scale_pos_weight
    np.mean(cross_val_score(xgb_model, train_df[columns], train_df['label'], cv=10, scoring='roc_auc'))
Out[25]: 0.8044224409823479
```

### 逐步调整参数,得到最优参数为

```
{
"max_depth": 2,
"min_child_weight": 140,
"gamma": 0.01,
"scale_pos_weight": 1
}
这时交叉验证的AUC值为0.804。
```

2) Random Forest 是基于决策树bagging集成的模型。

```
在训练时调整参数,得到最优参数为
```

```
"max_depth": 10,

"max_features": 0.8,

"max_samples": 0.5,

"min_sample_leaf": 50
```

```
In [34]: from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(max_depth=10, max_features=0.8, max_samples=0.5, min_samples_leaf=50)
np.mean(cross_val_score(rfc, train_df[columns], train_df['label'], cv=10, scoring='roc_auc'))
Out[34]: 0.8051286289227411
```

### 交叉验证的AUC值为0.805

3) SVM模型

```
在训练时调整参数,得到最优参数为
```

```
{
    "kernel": "poly",
    "degree": 2
}
```

交叉验证的AUC值为0.786

### 4) 最后使用基于实例的模型KNN

交叉验证的AUC值为0.797

### 3. 预测

使用加权投票的方法集成上面的四个模型,并用DEV数据集验证,调整各模型的权重,当权重为4:4:1:1时,预测的AUC值最高。

```
加权集成

In [61]: from sklearn.ensemble import VotingClassifier from sklearn.metrics import roc_auc_score

In [62]: dev_df = create_dataset(dev_df) test_df = create_dataset(test_df)

In [63]: voting_clf = VotingClassifier(estimators=[ ('xgb', xgb_model), ('rfc', rfc), ('svc', svc), ('knn', knn) ], voting='soft', weights=[0.4, 0.4, 0.1, 0.1]) voting_clf.fit(train_df[columns], train_df['label']) roc_auc_score(dev_df['label'], voting_clf.predict_proba(dev_df[columns])[:, 1])

Out[63]: 0.8038314476358084
```

使用集成后的模型预测TEST数据集,并将结果输出result.csv

```
In [64]: test_df['proba'] = voting_clf.predict_proba(test_df[columns])[:, 1]
    test_df.to_csv("result.csv", columns = ['user_id', 'proba'], index=False)
```