采用LightGBM方式处理，属于二分类问题：

LightGBM能直接处理类别型特征，但是输入值为**int类型**

特征工程：

1. 将source\_screen\_name、source\_type、genre\_ids、composer、lyricist、bd、isrc中的作为类别型特征，采用类似lable\_Encoder()方法处理，给特征中出现的每一个类别赋予一个数值，作为类别型特征输入lgbm
2. gender采用get\_dummies()方法进行编码，类似One-hot，处理完作为类别型特征输入
3. song\_length 作为数值型特征直接输入
4. msno、song\_id不参与训练与测试
5. LightGBM有对空值的处理机制，所以不需要对空值进行填充
6. LightGBM可以直接处理类别型特征

项目阶段代码：

**import** numpy **as** np  
**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
pd.set\_option(**'display.max\_columns'**,1000)  
pd.set\_option(**'display.width'**,1000)  
pd.set\_option(**'display.max\_colwidth'**,1000)  
  
df\_train=pd.read\_csv(**'train.csv'**,dtype={**'msno'** : **'category'**,  
 **'source\_system\_tab'** : **'category'**,  
 **'source\_screen\_name'** : **'category'**,  
 **'source\_type'** : **'category'**,  
 **'target'** : np.uint8,  
 **'song\_id'** : **'category'**})  
df\_test=pd.read\_csv(**'test.csv'**, dtype={**'msno'** : **'category'**,  
 **'source\_system\_tab'** : **'category'**,  
 **'source\_screen\_name'** : **'category'**,  
 **'source\_type'** : **'category'**,  
 **'song\_id'** : **'category'**})  
  
songs\_extra=pd.read\_csv(**'song\_extra\_info.csv'**)  
members=pd.read\_csv(**'members.csv'**,dtype={**'city'** : **'category'**,  
 **'bd'** : np.uint8,  
 **'gender'** : **'category'**,  
 **'registered\_via'** : **'category'**})  
  
df\_songs=pd.read\_csv(**'songs.csv'**,dtype={**'genre\_ids'**: **'category'**,  
 **'language'** : **'category'**,  
 **'artist\_name'** : **'category'**,  
 **'composer'** : **'category'**,  
 **'lyricist'** : **'category'**,  
 **'song\_id'** : **'category'**})  
  
song\_cols = [**'song\_id'**, **'artist\_name'**, **'genre\_ids'**, **'song\_length'**, **'language'**]  
train = df\_train.merge(df\_songs[song\_cols], on=**'song\_id'**, how=**'left'**)  
test = df\_test.merge(df\_songs[song\_cols], on=**'song\_id'**, how=**'left'**)  
  
  
members[**'registration\_year'**] = members[**'registration\_init\_time'**].apply(**lambda** x: int(str(x)[0:4]))  
members[**'registration\_month'**] = members[**'registration\_init\_time'**].apply(**lambda** x: int(str(x)[4:6]))  
members[**'registration\_date'**] = members[**'registration\_init\_time'**].apply(**lambda** x: int(str(x)[6:8]))  
  
members[**'expiration\_year'**] = members[**'expiration\_date'**].apply(**lambda** x: int(str(x)[0:4]))  
members[**'expiration\_month'**] = members[**'expiration\_date'**].apply(**lambda** x: int(str(x)[4:6]))  
members[**'expiration\_date'**] = members[**'expiration\_date'**].apply(**lambda** x: int(str(x)[6:8]))  
members = members.drop([**'registration\_init\_time'**], axis=1)  
  
**def** isrc\_to\_year(isrc):  
 **if** type(isrc) == str:  
 **if** int(isrc[5:7]) > 17:  
 **return** 1900 + int(isrc[5:7])  
 **else**:  
 **return** 2000 + int(isrc[5:7])  
 **else**:  
 **return** np.nan  
  
songs\_extra[**'song\_year'**] = songs\_extra[**'isrc'**].apply(isrc\_to\_year)  
songs\_extra.drop([**'isrc'**, **'name'**], axis = 1, inplace = **True**)  
  
train = train.merge(members, on=**'msno'**, how=**'left'**)  
test = test.merge(members, on=**'msno'**, how=**'left'**)  
  
train = train.merge(songs\_extra, on = **'song\_id'**, how = **'left'**)  
test = test.merge(songs\_extra, on = **'song\_id'**, how = **'left'**)  
  
*#one-hot编码方式  
#思考：类别的重要性相同，采用类似one-hot编码方式*gender\_train=pd.get\_dummies(train[**'gender'**],drop\_first=**True**)  
gender\_test=pd.get\_dummies(test[**'gender'**],drop\_first=**True**)  
  
*#拼接*train=pd.concat([train,gender\_train],axis=1)  
test=pd.concat([test,gender\_test],axis=1)  
  
*#特征处理后，去掉无用的特征*train.drop([**'gender'**],axis=1,inplace=**True**)  
test.drop([**'gender'**],axis=1,inplace=**True**)  
  
*#将年龄分为一个范围,方便转换为类别型特征*train[**'age\_range'**]=pd.cut(train[**'bd'**],bins=[-45,0,10,18,35,50,80,200])  
test[**'age\_range'**]=pd.cut(test[**'bd'**],bins=[-45,0,10,18,35,50,80,200])  
  
combine=[train,test]  
**for** value **in** combine:  
 value.loc[(value[**'bd'**] > 0) & (value[**'bd'**] <= 10), **'age\_category'**] = 0  
 value.loc[(value[**'bd'**] > 80) & (value[**'bd'**] <= 200), **'age\_category'**] = 1  
 value.loc[(value[**'bd'**] > 50) & (value[**'bd'**] <= 80), **'age\_category'**] = 2  
 value.loc[(value[**'bd'**] > 10) & (value[**'bd'**] <= 18), **'age\_category'**] = 3  
 value.loc[(value[**'bd'**] > 35) & (value[**'bd'**] <= 50), **'age\_category'**] = 4  
 value.loc[(value[**'bd'**] > -45) & (value[**'bd'**] <= 0), **'age\_category'**] = 5  
 value.loc[(value[**'bd'**] > 18) & (value[**'bd'**] <= 35), **'age\_category'**] = 6  
  
  
*#年龄、年龄范围处理完后，删除不用特征*train.drop([**'bd'**,**'age\_range'**],axis=1,inplace=**True**)  
test.drop([**'bd'**,**'age\_range'**],axis=1,inplace=**True**)  
 **del** members, df\_songs  
  
**for** col **in** train.columns:  
 **if** train[col].dtype == object:  
 train[col] = train[col].astype(**'category'**)  
 test[col] = test[col].astype(**'category'**)  
  
X = train.drop([**'target'**], axis=1)  
y = train[**'target'**].values  
  
**from** sklearn.model\_selection **import** train\_test\_split  
*#from sklearn.model\_selection import GridSearchCV*X\_train,X\_val,y\_train,y\_val=train\_test\_split(X,y,test\_size=0.2,random\_state=1)  
  
X\_test = test.drop([**'id'**], axis=1)  
ids = test[**'id'**].values  
  
**del** train, test  
  
**import** lightgbm **as** lgb  
*#d\_train = lgb.Dataset(X, y)  
#watchlist = [d\_train]*lgb\_train=lgb.Dataset(X\_train,y\_train)  
lgb\_val=lgb.Dataset(X\_val,y\_val,reference=lgb\_train)  
  
  
*#Those parameters are almost out of hat, so feel free to play with them. I can tell  
#you, that if you do it right, you will get better results for sure ;)*print(**'Training LGBM model...'**)  
params={  
 **'boosting'**:**'gbdt'**,  
 **'objective'**:**'binary'**,  
 **'metric'**:**'auc'**,  
 **'learning\_rate'**:0.2,  
 **'num\_leaves'**:256,  
 **'max\_depth'**:10,  
 **'num\_rounds'**:200,  
 **'begging\_freq'**:1,  
 **'begging\_seed'**:1,  
 **'max\_bin'**:256,  
 **'n\_jobs'**:-1  
}  
model=lgb.train(params=params,  
 train\_set=lgb\_train,  
 valid\_sets=lgb\_val,  
 early\_stopping\_rounds=5)  
  
**'''  
params = {}  
params['learning\_rate'] = 0.2  
params['application'] = 'binary'  
params['max\_depth'] = 8  
params['num\_leaves'] = 2\*\*8  
params['verbosity'] = 0  
params['metric'] = 'auc'  
params['n\_jobs']=-1  
  
model = lgb.train(params, train\_set=d\_train, num\_boost\_round=50, valid\_sets=watchlist, \  
verbose\_eval=5)'''**print(**'Making predictions and saving them...'**)  
p\_test = model.predict(X\_test)  
  
subm = pd.DataFrame()  
subm[**'id'**] = ids  
subm[**'target'**] = p\_test  
subm.to\_csv(**'submission.csv.gz'**, compression = **'gzip'**, index=**False**, float\_format = **'%.5f'**)  
print(**'Done!'**)

项目需要改进的地方：

**source\_system\_tab source\_screen\_name** **source\_type 这三个特征没有做太多的处理，直接丢失了特征之间的一些相关信息，后面考虑着重处理一下。**

特征工程能够找到算法的极限，超参数调优只能提高模型的极限。

疑问：

想问一下老师，本项目是音乐用户流失量预测，只是在kaggle上得了一个分数，如果是一个系统的话，如何得到用户流失率？是提取出那些标签为1 的用户吗？