Kaggle顶级方案IEEE-Fraud-Xgboost

IEEE-Fraud是一次表格数据的分类预测竞赛，Kaggle顶级大佬**Chris Deotte**向我们展示Xgboost的特征构造、特征筛选、合理有效的本地验证策略、Magic、结构后处理以及快速处理数据，英文原版[XGB Fraud with Magic - [0.9600]]。

特征构造

首先**Chris Deotte**构造了几个特征提取函数：

1. 频率编码，帮助树模型获取feaure-wise信息。

```
# FREQUENCY ENCODE TOGETHER
def encode_FE(df1, df2, cols):
    for col in cols:
        df = pd.concat([df1[col],df2[col]])
        vc = df.value_counts(dropna=True, normalize=True).to_dict()
        vc[-1] = -1
        nm = col+'_FE'
        df1[nm] = df1[col].map(vc)
        df1[nm] = df1[nm].astype('float32')
        df2[nm] = df2[col].map(vc)
        df2[nm] = df2[nm].astype('float32')
        print(nm,', ',end='')
```

2. 标签编码，感觉这里的重点是精准的内存和速度控制，int32和int16....

```
# LABEL ENCODE
def encode_LE(col,train=X_train,test=X_test,verbose=True):
    df_comb = pd.concat([train[col],test[col]],axis=0)
    df_comb,_ = df_comb.factorize(sort=True)
    nm = col
    if df_comb.max()>32000:
        train[nm] = df_comb[:len(train)].astype('int32')
        test[nm] = df_comb[len(train):].astype('int32')
    else:
        train[nm] = df_comb[:len(train)].astype('int16')
        test[nm] = df_comb[len(train):].astype('int16')
    del df_comb; x=gc.collect()
    if verbose: print(nm,', ',end='')
```

3. 联合分组统计

```
# GROUP AGGREGATION MEAN AND STD
# https://www.kaggle.com/kyakovlev/ieee-fe-with-some-eda
def encode_AG(main_columns, uids, aggregations=['mean'], train_df=X_train,
test_df=X_test,
              fillna=True, usena=False):
```

```
    # AGGREGATION OF MAIN WITH UID FOR GIVEN STATISTICS
    for main_column in main_columns:
        for col in uids:
            for agg_type in aggregations:
                new_col_name = main_column+'_'+col+'_'+agg_type
                temp_df = pd.concat([train_df[[col, main_column]],
test_df[[col,main_column]]])
                if usena: temp_df.loc[temp_df[main_column]==-1,main_column] = np.nan
                temp_df = temp_df.groupby([col])
[main_column].agg([agg_type]).reset_index().rename(
                                                        columns={agg_type:
new_col_name})

                temp_df.index = list(temp_df[col])
                temp_df = temp_df[new_col_name].to_dict()

                train_df[new_col_name] = train_df[col].map(temp_df).astype('float32')
                test_df[new_col_name]  = test_df[col].map(temp_df).astype('float32')

                if fillna:
                    train_df[new_col_name].fillna(-1,inplace=True)
                    test_df[new_col_name].fillna(-1,inplace=True)

                print("'"+new_col_name+"'",', ',end='')
```

4. 类别特征组合

```
# COMBINE FEATURES
def encode_CB(col1,col2,df1=X_train,df2=X_test):
    nm = col1+'_'+col2
    df1[nm] = df1[col1].astype(str)+'_'+df1[col2].astype(str)
    df2[nm] = df2[col1].astype(str)+'_'+df2[col2].astype(str)
    encode_LE(nm,verbose=False)
    print(nm,', ',end='')
```

5. 联合nunique统计

```
# GROUP AGGREGATION NUNIQUE
def encode_AG2(main_columns, uids, train_df=X_train, test_df=X_test):
    for main_column in main_columns:
        for col in uids:
            comb = pd.concat([train_df[[col]+[main_column]],test_df[[col]+
[main_column]]],axis=0)
            mp = comb.groupby(col)[main_column].agg(['nunique'])['nunique'].to_dict()
            train_df[col+'_'+main_column+'_ct'] =
train_df[col].map(mp).astype('float32')
            test_df[col+'_'+main_column+'_ct'] =
test_df[col].map(mp).astype('float32')
            print(col+'_'+main_column+'_ct, ',end='')
```

接着，基于分析构造如下特征：

```
# TRANSACTION AMT CENTS
X_train['cents'] = (X_train['TransactionAmt'] -
np.floor(X_train['TransactionAmt'])).astype('float32')
X_test['cents'] = (X_test['TransactionAmt'] -
np.floor(X_test['TransactionAmt'])).astype('float32')
print('cents, ', end='')
# FREQUENCY ENCODE: ADDR1, CARD1, CARD2, CARD3, P_EMAILDOMAIN
encode_FE(X_train,X_test,['addr1','card1','card2','card3','P_emaildomain'])
# COMBINE COLUMNS CARD1+ADDR1, CARD1+ADDR1+P_EMAILDOMAIN
encode_CB('card1','addr1')
encode_CB('card1_addr1','P_emaildomain')
# FREQUENCY ENOCDE
encode_FE(X_train,X_test,['card1_addr1','card1_addr1_P_emaildomain'])
# GROUP AGGREGATE
encode_AG(['TransactionAmt','D9','D11'],
['card1','card1_addr1','card1_addr1_P_emaildomain'],['mean','std'],usena=True)
```

特征筛选

由于问题背景涉及时序，重点需要评估相关特征对未来结构预测的稳定性。**Chris Deotte**采用了一个朴实的方法进行检测，所有特征分别构造一个模型对单特征建模，建模数据采用训练集第一个月份，并在数据集最后一个月份进行验证，期望AUC在训练和验证上均大于0.5（至少有用）。剔除小于0.5的，缺失过大的特征。

```
cols = list( X_train.columns )
cols.remove('TransactionDT')
for c in ['D6','D7','D8','D9','D12','D13','D14']:
    cols.remove(c)

# FAILED TIME CONSISTENCY TEST
for c in ['C3','M5','id_08','id_33']:
    cols.remove(c)
for c in ['card4','id_07','id_14','id_21','id_30','id_32','id_34']:
    cols.remove(c)
for c in ['id_'+str(x) for x in range(22,28)]:
    cols.remove(c)
```

本地验证及预测策略

1. 特征工程验证框架，依然基于时序性能可靠性考虑，建立time-based 的两种本地验证策略：
    - 在时序前75%数据训练，后25%数据验证；
    - 在前4个月训练，跳过一个月，在最后一个月进行验证；
2. 预测策略，基于月份的groupKfold，在12，13，14，15，16，17月份上，fold1 在13-17月进行训练，基于12月验证结果进行early stopping，并对测试数据进行预测，以此类推。最终预测结果为所有fold模型预测的平均。

Magic

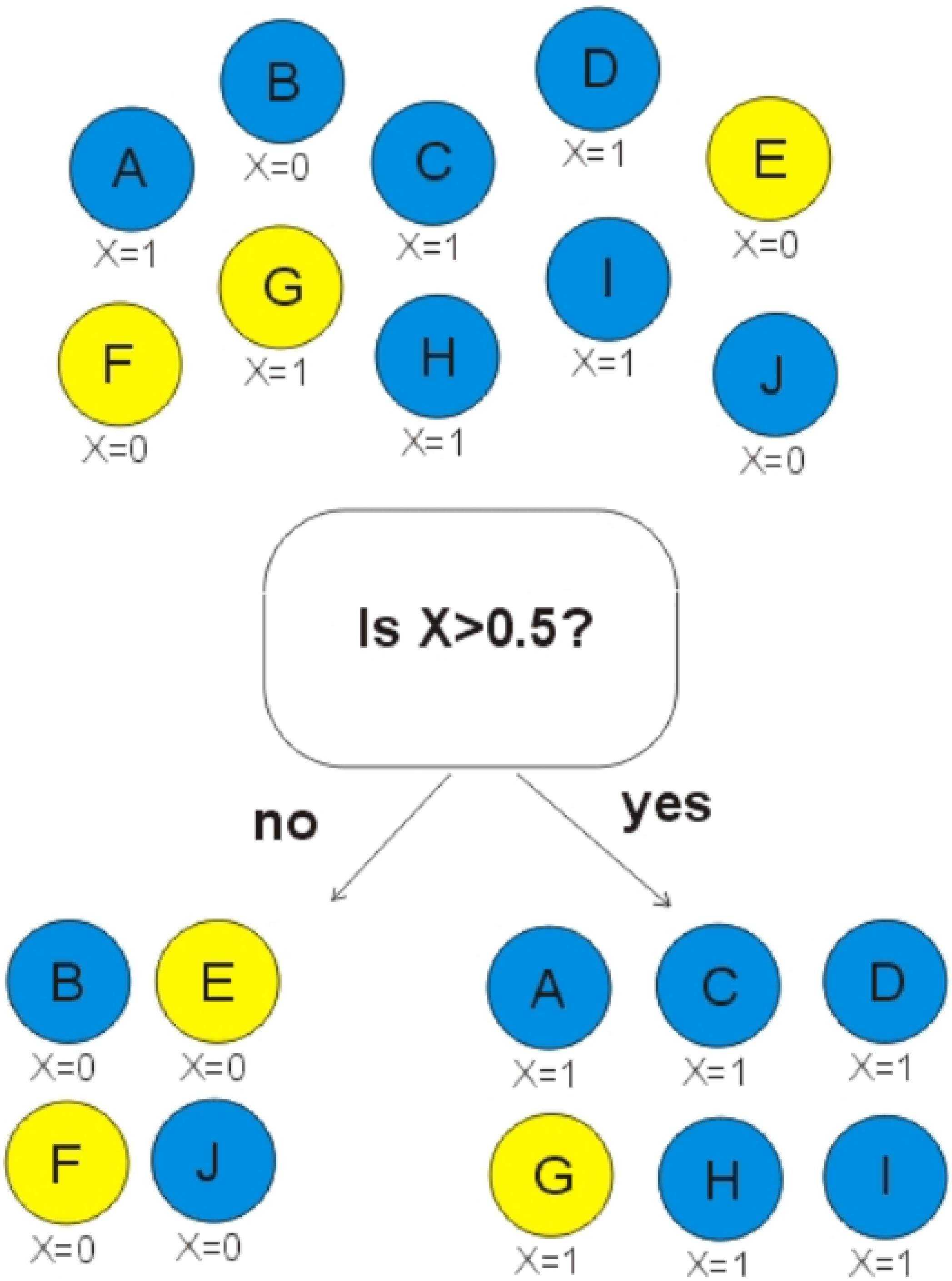
什么魔法？怎么工作的。

1. UID是客户唯一标识；
2. 构造基于UID的组合特征，然后移除UID；

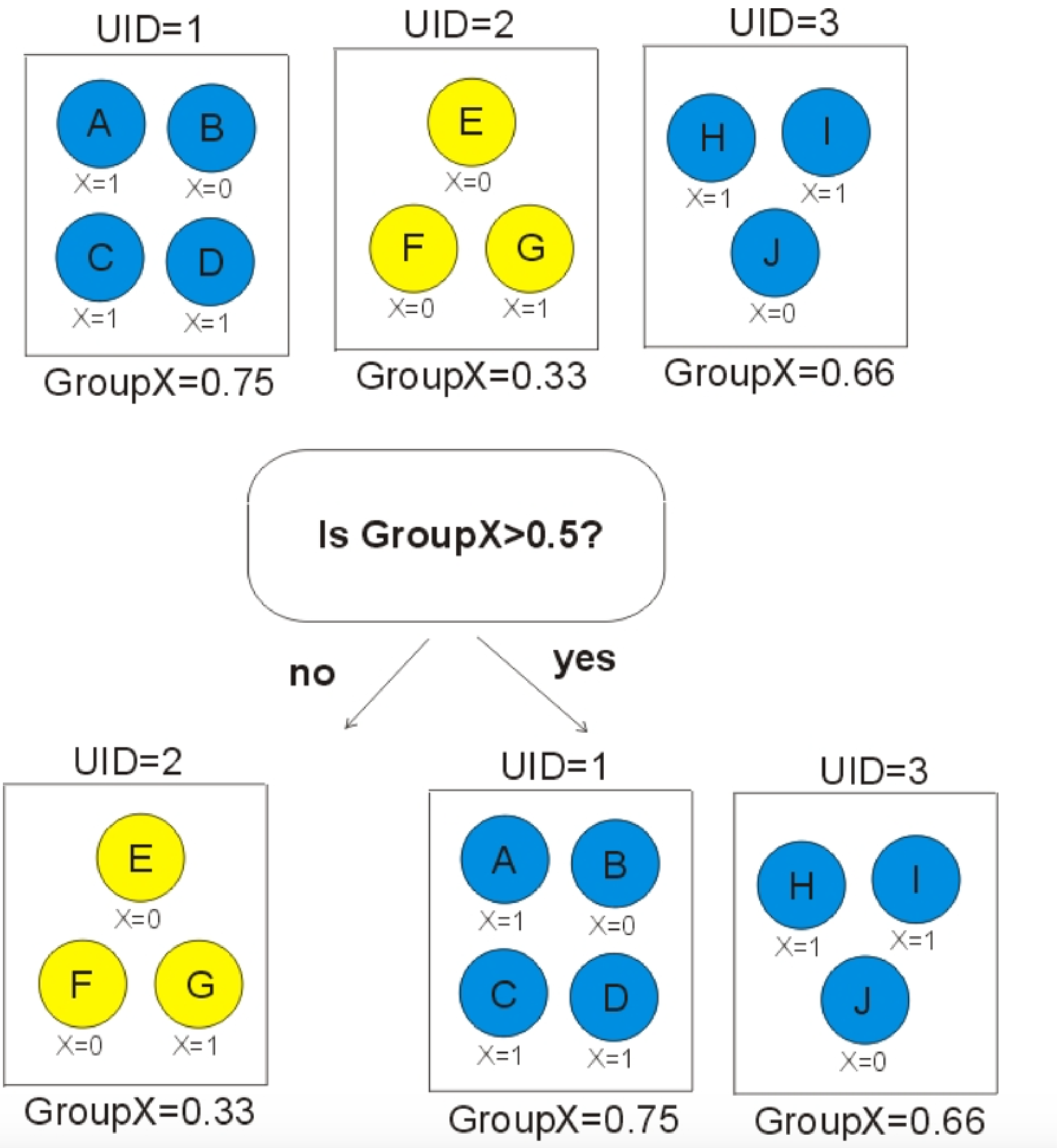
假设有如下10比交易：`A，B，C，D，E，F，G，H，I，J`

TransactionID	UID	FeatureX	GroupX	IsFraud
A	1	1	0.75	1
B	1	0	0.75	1
C	1	1	0.75	1
D	1	1	0.75	1
E	2	0	0.33	0
F	2	0	0.33	0
G	2	1	0.33	0
H	3	1	0.66	1
I	3	1	0.66	1
J	3	0	0.66	1

假如我们只使用特征X，我们可以将70%的交易正确分类（ISFraud），正如下图所展示，树模型在X=0.5作为分裂节点，70%的样本得到了正确预测。



现在我们将UID作为GROUP，计算GROUP内特征X的均值，我们可以100%做出正确划分。注意这个划分并没有直接使用UID，只需要基于UID的对特征X的统计信息；



好了，开始构造Magic，首先我们的UID并不完美，许多UID（客户）包含多个新用户，模型会探测到这个情况并尝试更多的分裂，直到将UID切割到每个唯一用户（信用卡），结合时间信息构造唯一UID。

```
X_train['day'] = X_train.TransactionDT / (24*60*60)
X_train['uid'] = X_train.card1_addr1.astype(str)+'_'+np.floor(X_train.day-
X_train.D1).astype(str)

X_test['day'] = X_test.TransactionDT / (24*60*60)
X_test['uid'] = X_test.card1_addr1.astype(str)+'_'+np.floor(X_test.day-
X_test.D1).astype(str)
```

做基于UID-Group的统计特征-Magic；

```
# FREQUENCY ENCODE UID
encode_FE(X_train,X_test,['uid'])
# AGGREGATE
encode_AG(['TransactionAmt','D4','D9','D10','D15'],['uid'],
['mean','std'],fillna=True,usena=True)
# AGGREGATE
encode_AG(['C'+str(x) for x in range(1,15) if x!=3],['uid'],
['mean'],X_train,X_test,fillna=True,usena=True)
# AGGREGATE
encode_AG(['M'+str(x) for x in range(1,10)],['uid'],['mean'],fillna=True,usena=True)
# AGGREGATE
encode_AG2(['P_emaildomain','dist1','DT_M','id_02','cents'], ['uid'], train_df=X_train,
test_df=X_test)
# AGGREGATE
encode_AG(['C14'],['uid'],['std'],X_train,X_test,fillna=True,usena=True)
# AGGREGATE
encode_AG2(['C13','V314'], ['uid'], train_df=X_train, test_df=X_test)
# AGGREATE
encode_AG2(['V127','V136','V309','V307','V320'], ['uid'], train_df=X_train, test_df=X_test)
# NEW FEATURE
X_train['outsider15'] = (np.abs(X_train.D1-X_train.D15)>3).astype('int8')
X_test['outsider15'] = (np.abs(X_test.D1-X_test.D15)>3).astype('int8')
print('outsider15')
```

Post Process

其团队写了一个脚本创造了一种更加精准的UID构造方法，比Magic UID更加精准，以至于每个新UID下是否欺诈的标签完全一致。因此，后处理逻辑是将所有UID预测结果用其训练集的预测结果均值（UID下）进行替换。应用这个后处理LB得分从0.9602提升到0.9618。

```
X_test['isFraud'] = sample_submission.isFraud.values
X_train['isFraud'] = y_train.values
comb = pd.concat([X_train[['isFraud']],X_test[['isFraud']]],axis=0)

uids = pd.read_csv('/kaggle/input/ieee-submissions-and-
uids/uids_v4_no_multiuid_cleaning..csv',usecols=
['TransactionID','uid']).rename({'uid':'uid2'},axis=1)
comb = comb.merge(uids,on='TransactionID',how='left')
mp = comb.groupby('uid2').isFraud.agg(['mean'])
comb.loc[comb.uid2>0,'isFraud'] = comb.loc[comb.uid2>0].uid2.map(mp['mean'])

uids = pd.read_csv('/kaggle/input/ieee-submissions-and-
uids/uids_v1_no_multiuid_cleaning.csv',usecols=
['TransactionID','uid']).rename({'uid':'uid3'},axis=1)
comb = comb.merge(uids,on='TransactionID',how='left')
mp = comb.groupby('uid3').isFraud.agg(['mean'])
comb.loc[comb.uid3>0,'isFraud'] = comb.loc[comb.uid3>0].uid3.map(mp['mean'])

sample_submission.isFraud = comb.iloc[len(X_train):].isFraud.values
sample_submission.to_csv('sub_xgb_96_PP.csv',index=False)
```

后记

Chris Deotte对结构化数据挖掘、特征、树模型的理解异常深刻，Kaggle顶级方案巡览里应该还会经常出现他的身影，希望这个系列给大家带来更好的思路和理解，想催更就直接点赞关注吧！