In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from functools import partial, reduce
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans,MeanShift, estimate_bandwidth
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import f1_score
```

In [2]:
```
t1 = 'train_accounts.csv'
t2 = 'train_bids.csv'
df1 = pd.read_csv(t1)
df2 = pd.read_csv(t2)
df = pd.merge(df1,df2,on ='bidder_id',how = 'inner')
df['device_n']=df.device.str.extract('(\d+)')
df.head()
```

Out[2]:

	bidder_id	payment_account	address	outcome
0	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.0
1	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.0
2	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.0
3	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.0
4	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.0

In [3]:
```
# quantitative
def statbyfeature(data,features):
    bidders_stat =dict()
    bidder = data['bidder_id'].nunique()
    for feature in features:
        seg = data[['bidder_id',feature]]
        seg_res = seg.groupby('bidder_id')[feature].nunique().reset_index(name = feature+"_nums")
        bidders_stat[feature] = seg_res
    res = bidders_stat[features[0]]
    for feature in features[1:]:
        res = pd.merge(res,bidders_stat[feature],on= 'bidder_id')
    return res

def countByFeature(df,f_count):
    dfs = df
    for f in f_count:
        d1 = df[f].value_counts().reset_index()
        d1.columns = [f,str(f+"_count")]
        dfs = pd.merge(dfs,d1,on = f)
    return dfs
```

In [4]:
```
# Qualitative
def countByFeature_risk(df,f_count):
    dfs = df
    for f in f_count:
        tot = df.groupby(f)['outcome'].sum()
        cnt = df.groupby(f)['outcome'].count()
        tmpt =pd.DataFrame(tot/cnt).reset_index()

        tmpt.columns =[f,str(f+"_risk")]

        dfs = pd.merge(dfs,tmpt,on = f)
    return dfs
```

In [5]:
```
# extract count quantative feature for train data set
feature_count = ['bidder_id','payment_account','address','bid_id','auction','merchandise','device','country'
df_train= countByFeature(df,feature_count)
df_train.head()
```

Out[5]:

	bidder_id	payment_account	address	outcom
0	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.
1	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.
2	0565a780f2c5b46837bbe9739ef15ea92fdu0	474f441d0c8f8416be60291872926cd52br6q	4544d31a6d8a15cdee64e7dafbc63e6drth9a	0.
3	0565a780f2c5b46837bbe9739ef15ea92fdu0	474f441d0c8f8416be60291872926cd52br6q	4544d31a6d8a15cdee64e7dafbc63e6drth9a	0.
4	0565a780f2c5b46837bbe9739ef15ea92fdu0	474f441d0c8f8416be60291872926cd52br6q	4544d31a6d8a15cdee64e7dafbc63e6drth9a	0.

5 rows × 23 columns

In [6]:
```
# extract unique value quantative feature for train data set
features =['device','ip','url','merchandise','auction','country']
dfs = statbyfeature(df,features)
df_train = pd.merge(df_train, dfs,on = 'bidder_id')
df_train.head()
```

Out[6]:

	bidder_id	payment_account	address	outcome
0	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.0
1	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.0
2	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.0
3	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.0
4	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	0.0

5 rows × 29 columns

In [7]:
```
#extract qualtative feature for train data set
risk_feature =['bidder_id','payment_account','address','bid_id','auction','merchandise','device','country',
df_train_f = countByFeature_risk(df_train,risk_feature)
df_train_f.head()
```

Out[7]:

	bidder_id	payment_account	address	outcor
0	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	(
1	ea5948061b2059bcbd900d3ae7e86e428ae8g	cee2b2bb0c91cac4cf7c3d0a7f6f0dddt2lku	a3d2de7675556553a5f08e4c88d2c228n7xkp	(
2	c2d1b8c1b7a9c8010f5b215e8aadb1d7gahpp	a47ec76145b2abb39b940289f696386b1ccvn	65e92ffdba45917e11cc034e53cd8f5b189h5	(
3	c2d1b8c1b7a9c8010f5b215e8aadb1d7gahpp	a47ec76145b2abb39b940289f696386b1ccvn	65e92ffdba45917e11cc034e53cd8f5b189h5	(
4	1af4685cda2979a608c4b6ad83473774qgicn	e35a74340ac6d90fcb4fe9b84c6eaa2135wqx	ceda5914ce08b87ec934e614452442903j17m	

5 rows × 39 columns

In [8]:
```
# get the numerical training dataset
col_drop= ['device','ip','url','merchandise','auction','country','address','time','payment_account','bid_id'
df_train= df_train_f.drop(col_drop, axis=1)
df_train.head()
```

Out[8]:

	outcome	device_n	bidder_id_count	payment_account_count	address_count	bid_id_count	auction_count	merchandise_count	de
0	0.0	53	272	272	272	1	32	220307	
1	0.0	150	272	272	272	1	223	220307	
2	0.0	53	226	226	226	1	223	220307	
3	0.0	8	226	226	226	1	263	220307	
4	1.0	53	6002	6002	6002	1	223	220307	

5 rows × 28 columns

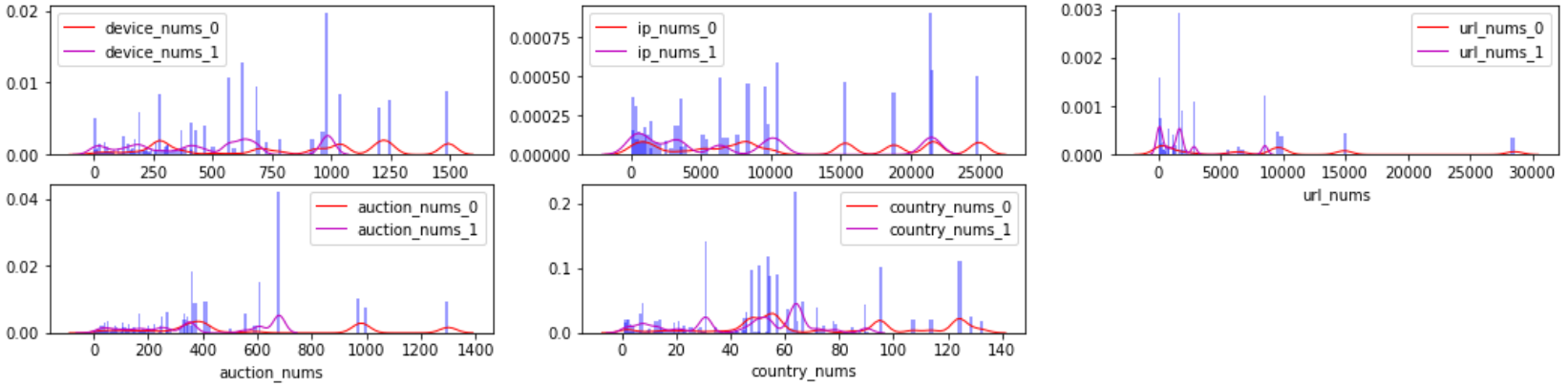
Exploration data Analysis

Histogram for train dataset

In [100]:
```
features =['device_nums','ip_nums','url_nums','auction_nums','country_nums']
index =0
plt.figure(figsize=(18,4))
data_0 = df_train[df_train.outcome ==0]
data_1 = df_train[df_train.outcome ==1]

for fea in features:
    index +=1
    plt.subplot(2,3,index)
    plt.xlabel(fea)
    sns.distplot(data_0[fea],kde =True,bins =100,color ='b',kde_kws={'color':"r","lw":1,"label":fea+"_0"})
    sns.distplot(data_1[fea],kde =True,bins =100,color ='b',kde_kws={'color':"m","lw":1,"label":fea+"_1"})


plt.show()
```



Bar chart for Feature space

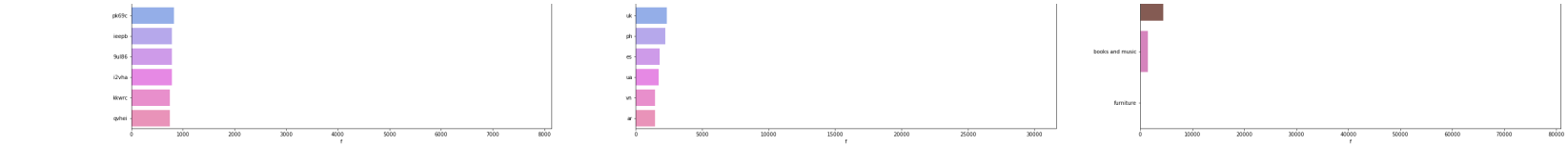
In [26]:

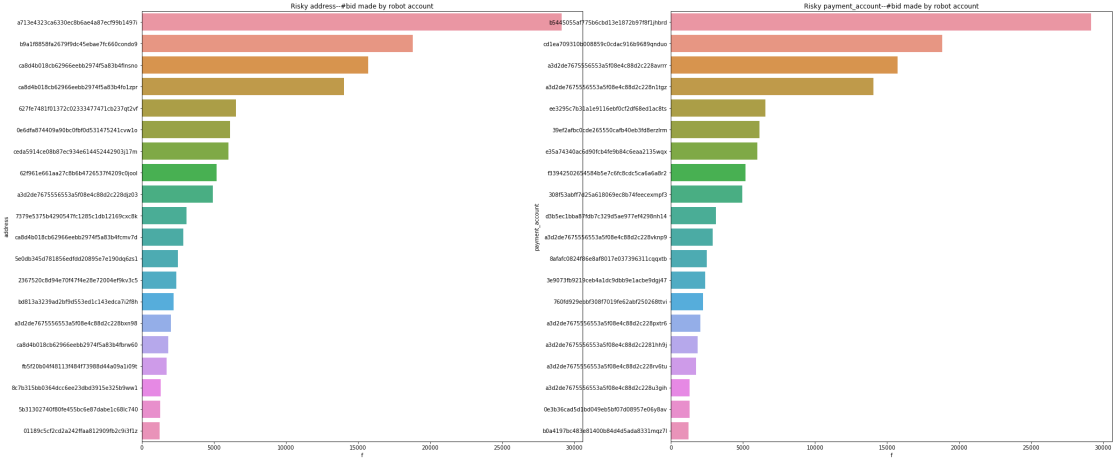
```
features =['device','ip','url','auction','country','merchandise','address','payment_account']
index =0
plt.figure(figsize=(50,50))
# data_0 = df_train[df_train.outcome ==0]
# data_1 = df_train[df_train.outcome ==1]

for fea in features:
    index +=1
    data = df.groupby(fea)['outcome'].sum().sort_values(ascending=False).reset_index(name = 'f')[:20]
    plt.subplot(3,3,index)
    plt.title("Risky "+str(fea)+"--#bid made by robot account")
    plt.xlabel(fea)
    sns.barplot(y=fea,x ='f',data =data)



plt.show()
```


pk69c
ieepb
9ul86
i2vha
kkwrc
qvhei
0
1000
2000
3000
4000
5000
6000
7000
8000
f
uk
ph
es
ua
vn
ar
0
5000
10000
15000
20000
25000
30000
f
books and music
furniture
0
10000
20000
30000
40000
50000
60000
70000
80000
f


Risky address--#bid made by robot account
a713e4323ca6330ec8b6ae4a87ecf99b1497i
b9a1f8858fa2679ffdc45ebae7fc660condn9
ca8d4b018cb62966eebb2974f5a83b4fnsno
ca8d4b018cb62966eebb2974f5a83b4fs1rpr
627fe7481f01372c02333477471cb237qf2vf
0e6dfa074409a90bc0fbf0d531475241cvw1o
ceda5914ce08b87ec934e61445244290 3j17m
62f961e661aa27cfb6b4726537f4209c0ppd
a3d2de7675556553a5f08e4c88d2c228dqj03
7379e5375b4290547fc1285c1db12169cxc8k
ca8d4b018cb62966eebb2974f5a83b4fcmv7d
5e0db345d781856edfdd20895e7e190dq6zs1
2367520cbd94e70f47f4e28e72004ef9kv3c5
bd813a3239ad2bf9d553ed1c143edca7i2f8h
a3d2de7675556553a5f08e4c88d2c228bxn98
ca8d4b018cb62966eebb2974f5a83b4fbrw60
fb5f20b04f4811 3f484f7398dd44a09a1i09t
8c7b315bb0364dcc6ee23dbd3915e325b9wwa1
5b313027405f80fe455bc6e87dabe1c68c740
01189c5cf2cd2a242ffaa812909fb2c93f1z
0
5000
10000
15000
20000
25000
30000
f
Risky payment_account--#bid made by robot account
b54855af775b6cbd13e1872b978flj1hbrid
cd1ea709310b008859c0cdac916b968f9qndus
a3d2de7675556553a5f08e4c88d2c228avrrr
a3d2de7675556553a5f08e4c88d2c228n1tgz
ee3295c7b31a1e9116ebf0cf2df68ed1ac8ts
39ef2afbc02de265550cafb40eb3fd8erzlrm
e35a74340ac5d901fcb4fe9b84c6eaa2135wqx
f3394250260454b4b5e7c6fc8cdc5ca6a6afb2
308f53abbff7d25a618069ec8b74feecexmpf3
d0b5ec1bbabf7fdb7c329d5ae977ef4298nh14
a3d2de7675556553a5f08e4c88d2c228vknp9
8afafc0824f96e8af8017e037396311cqqxtb
3e9073fb9213ceb4a1dc9dbb9e1acbe9dgj47
760fd929ecbf308f7019fe62abf250268ttvi
a3d2de7675556553a5f08e4c88d2c228pxtr6
a3d2de7675556553a5f08e4c88d2c2281hh9j
a3d2de7675556553a5f08e4c88d2c228rv6tu
a3d2de7675556553a5f08e4c88d2c228u3gih
0e3b36cad5d1bd049eb5bf07d08957e06y8av
bba4197bc483e81400b84d445a6a8331mqz7t
0
5000
10000
15000
20000
25000
30000
f

In [108]:
```
df.groupby('payment_account')['outcome'].sum().sort_values(ascending=False).reset_index(name="f")
```

Out[108]:

	payment_account	f
0	b5445055af775b6cbd13e1872b97f8f1jhbrd	29146.0
1	cd1ea709310b008859c0cdac916b9689qnduo	18826.0
2	a3d2de7675556553a5f08e4c88d2c228avrrr	15727.0
3	a3d2de7675556553a5f08e4c88d2c228n1tgz	14038.0
4	ee3295c7b31a1e9116ebf0cf2df68ed1ac8ts	6540.0
...	...	...
391	a3d2de7675556553a5f08e4c88d2c228nxmdi	0.0
392	a3d2de7675556553a5f08e4c88d2c228o5xqr	0.0
393	a3d2de7675556553a5f08e4c88d2c228o5y1h	0.0
394	a3d2de7675556553a5f08e4c88d2c228oggrt	0.0
395	0298ffee623bea598e46dab6e40983a2sr9js	0.0

396 rows × 2 columns

Model

In [9]:
```
col = df_train.columns[1:]
x = df_train[col]
y = df_train['outcome']
## feature scale
x_train,x_test ,y_train, y_test = train_test_split(StandardScaler().fit_transform(x),y,test_size =.33,random

df_train.head()
```

Out[9]:

	outcome	device_n	bidder_id_count	payment_account_count	address_count	bid_id_count	auction_count	merchandise_count	de
0	0.0	53	272	272	272	1	32	220307	
1	0.0	150	272	272	272	1	223	220307	
2	0.0	53	226	226	226	1	223	220307	
3	0.0	8	226	226	226	1	263	220307	
4	1.0	53	6002	6002	6002	1	223	220307	

5 rows × 28 columns

In [10]:
```
## Prediction for test_account(real)
t3 = 'test.csv'
t4 = 'test_bids.csv'
df3 = pd.read_csv(t3)
df4 = pd.read_csv(t4)
df_t = pd.merge(df3,df4,on ='bidder_id')
df_t['device_n']=df_t.device.str.extract('(\d+)')
df_t.drop('Unnamed: 0',axis=1,inplace=True)
df_t.head()
```

Out[10]:

	bidder_id	payment_account	address	outcome	
0	fb92fc925c5df50f4da7874a7542cf2210z1f	a3d2de7675556553a5f08e4c88d2c2285fa8i	a3d2de7675556553a5f08e4c88d2c228g8zew	1	2
1	fb92fc925c5df50f4da7874a7542cf2210z1f	a3d2de7675556553a5f08e4c88d2c2285fa8i	a3d2de7675556553a5f08e4c88d2c228g8zew	1	2
2	fb92fc925c5df50f4da7874a7542cf2210z1f	a3d2de7675556553a5f08e4c88d2c2285fa8i	a3d2de7675556553a5f08e4c88d2c228g8zew	1	2
3	fb92fc925c5df50f4da7874a7542cf2210z1f	a3d2de7675556553a5f08e4c88d2c2285fa8i	a3d2de7675556553a5f08e4c88d2c228g8zew	1	1
4	fb92fc925c5df50f4da7874a7542cf2210z1f	a3d2de7675556553a5f08e4c88d2c2285fa8i	a3d2de7675556553a5f08e4c88d2c228g8zew	1	1

In [11]:
```
df_test= countByFeature(df_t,feature_count)
df_test.head()
```

Out[11]:

	bidder_id	payment_account	address	outcome
0	fb92fc925c5df50f4da7874a7542cf2210z1f	a3d2de7675556553a5f08e4c88d2c2285fa8i	a3d2de7675556553a5f08e4c88d2c228g8zew	1
1	fae553f133602fba6e9e6051dfb27fefkruax	a3d2de7675556553a5f08e4c88d2c228xsfa5	a3d2de7675556553a5f08e4c88d2c2288cnnj	1
2	a4e83190edb97fefdc4a6cbfd00f41fbm7c4o	a3d2de7675556553a5f08e4c88d2c228qteoy	3a7e6a32b24aeab0688e91a41f3188e2mhc65	0
3	a4e83190edb97fefdc4a6cbfd00f41fbm7c4o	a3d2de7675556553a5f08e4c88d2c228qteoy	3a7e6a32b24aeab0688e91a41f3188e2mhc65	0
4	a4e83190edb97fefdc4a6cbfd00f41fbm7c4o	a3d2de7675556553a5f08e4c88d2c228qteoy	3a7e6a32b24aeab0688e91a41f3188e2mhc65	0

5 rows × 23 columns

In [12]:
```
dfs_t = statbyfeature(df_t,features)
df_test = pd.merge(df_test,dfs_t,on = 'bidder_id')
risk_feature =['bidder_id','payment_account','address','bid_id','auction','merchandise','device','country',
df_test_f = countByFeature_risk(df_test,risk_feature)
df_test_f.head()
df_test_f.drop(col_drop, axis=1, inplace=True)
```

In [13]:
```
test_col = df_test_f.columns[1:]

test_col
```

Out[13]:
```
Index(['device_n', 'bidder_id_count', 'payment_account_count', 'address_count',
       'bid_id_count', 'auction_count', 'merchandise_count', 'device_count',
       'country_count', 'ip_count', 'url_count', 'device_nums', 'ip_nums',
       'url_nums', 'merchandise_nums', 'auction_nums', 'country_nums',
       'bidder_id_risk', 'payment_account_risk', 'address_risk', 'bid_id_risk',
       'auction_risk', 'merchandise_risk', 'device_risk', 'country_risk',
       'ip_risk', 'url_risk'],
      dtype='object')
```

In [14]:
```
x_t = df_test_f[test_col]
x_test_account = StandardScaler().fit_transform(x_t)
y_test_account = df_test_f['outcome']
df_test_f.head()
```

Out[14]:

	outcome	device_n	bidder_id_count	payment_account_count	address_count	bid_id_count	auction_count	merchandise_count	de
0	1	219	25	25	25	1	22	12753	
1	1	720	180	180	180	1	1	12753	
2	0	219	3573	3573	3573	1	19	12753	
3	0	212	3573	3573	3573	1	1	12753	
4	0	178	3573	3573	3573	1	17	12753	

5 rows × 28 columns

Logistic Regression

In [15]:
```
def lg_reg(x_train,x_test ,y_train, y_test,x_test_account,y_test_account):
    log_reg =  LogisticRegression()
    #models = [xgboost]
    models = [log_reg]

    for model in models:
        print("*********Logistic Regression *****")
        model.fit(x_train,y_train)
        fpr,tpr,  _ = metrics.roc_curve(y_test,model.predict_proba(x_test)[:,1])
        auc = metrics.auc(fpr,tpr)
        y_pred_train = model.predict(x_test)
        y_pred_test = model.predict(x_test_account)
        f1_train = f1_score(y_test, y_pred_train,average='micro')
        f1_test = f1_score(y_test_account, y_pred_test,average='macro')
        print('auc score:{}',auc)
        print("train score:{}",model.score(x_train,y_train))
        print("test score_train",model.score(x_test,y_test))
        print("Actually test score",model.score(x_test_account,y_test_account))
        print("F1 Train-test score_train",f1_train)
        print("F1 Actually test score",f1_test)
```

Logistic regression performance

In [16]:
```
lg_reg(x_train,x_test ,y_train, y_test,x_test_account,y_test_account)
```

```
*********Logistic Regression *****
auc score:{} 1.0
train score:{} 1.0
test score_train 1.0
Actually test score 1.0
F1 Train-test score_train 1.0
F1 Actually test score 1.0
```

In [17]:
```
def xgboost_model(x_train,x_test ,y_train, y_test,x_test_account,y_test_account):
    eval_set = [(x_test, y_test)]
    xgboost = XGBClassifier(eval_metric=["error", "logloss"], eval_set=eval_set, verbose=True, objective='bi
                            n_estimators=100,
                             )
    #log_reg =  LogisticRegression()
    #models = [xgboost,log_reg]
    models = [xgboost]

    for model in models:
        print("*********Xgboost**********")
        model.fit(x_train,y_train)
        fpr,tpr,  _ = metrics.roc_curve(y_test,model.predict_proba(x_test)[:,1])
        auc = metrics.auc(fpr,tpr)
        y_pred_train = model.predict(x_test)
        y_pred_test = model.predict(x_test_account)
        f1_train = f1_score(y_test, y_pred_train,average='micro')
        f1_test = f1_score(y_test_account, y_pred_test,average='micro')
        print('auc score:{}',auc)
        print("train score:{}",model.score(x_train,y_train))
        print("test score_train",model.score(x_test,y_test))
        print("Actually test score",model.score(x_test_account,y_test_account))
        print("F1 Train-test score_train",f1_train)
        print("F1 Actually test score",f1_test)
```

In [18]:
```
xgboost_model(x_train,x_test ,y_train, y_test,x_test_account,y_test_account)
```

```
*********Xgboost**********
auc score:{} 1.0
train score:{} 1.0
test score_train 1.0
Actually test score 1.0
F1 Train-test score_train 1.0
F1 Actually test score 1.0
```

In [ ]:
```

```