In [1]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import average_precision_score, roc_curve, auc,recall_sco
re, precision_score
from sklearn.model_selection import train_test_split
import datetime, time
from datetime import datetime
from time import mktime
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
import matplotlib.pyplot as plt
from xgboost import XGBClassifier
```

In [2]:
```
#num = LabelEncoder()
train = pd.read_csv("D:\Academics\PGDBA\Competitions\Predixion/train.csv")
test = pd.read_csv("D:\Academics\PGDBA\Competitions\Predixion/test.csv")
item_data = pd.read_csv("D:\Academics\PGDBA\Competitions\Predixion/item_data.c
sv")
view_data = pd.read_csv("D:\Academics\PGDBA\Competitions\Predixion/view_log.cs
v")
```

In [3]:
```
train.shape, test.shape
```

Out[3]:
```
((197093, 7), (40516, 7))
```

In [4]:
```
train.head()
```

Out[4]:

	impression_id	time_stamp	cust_id	app_code	os_version	lte_flag	click
0	c4ca4238a0b923820dcc509a6f75849b	11/15/2029 0:00	I7862	E22	obsolete	0	
1	a87ff679a2f3e71d9181a67b7542122c	11/15/2029 0:00	E238	D71	trending	0	
2	eccbc87e4b5ce2fe28308fd9f2a7baf3	11/15/2029 0:00	F8442	B27	trending	0	
3	c81e728d9d4c2f636f067f89cc14862c	11/15/2029 0:00	I9464	B29	medium	0	
4	45c48cce2e2d7fbdea1afc51c7c6ad26	11/15/2029 0:01	G3410	E67	trending	1	

In [5]:
```
train['click_flag'].value_counts()/train.shape[0]
```

Out[5]:
```
0    0.954494
1    0.045506
Name: click_flag, dtype: float64
```

In [6]:
```
item_data.head()
```

Out[6]:

	item_id	item_price	product_type
0	C6880	2301.0	D040
1	F4939	1756.5	G822
2	E0383	412.5	B619
3	I777	1177.5	F264
4	B13705	633.5	B0239

In [7]:
```
item_data.product_type.nunique()
```

Out[7]:
```
7959
```

In [8]:
```
print(len(np.intersect1d(test.cust_id,view_data.cust_id)))
print(len(np.intersect1d(test.cust_id,train.cust_id)))
```

```
18383
11911
```

In [9]:
```
print(view_data.shape)
view_data.head()
```

```
(1235944, 6)
```

Out[9]:

	Unnamed: 0	server_time	device_type	session_id	cust_id	item_id
0	1	11/15/2029 0:00	android	B65430	B570	C7894
1	2	11/15/2029 0:00	android	B4305	E2867	D5447
2	3	11/15/2029 0:00	android	H9862	H1850	F2937
3	4	11/15/2029 0:00	android	E38850	H9140	C342
4	5	11/15/2029 0:00	android	G19497	D5051	H2284

In [10]:
```
view_data = view_data.drop(['Unnamed: 0'], axis = 1)
```

In [40]:
```
view_data = (view_data.groupby(view_data.columns.tolist()).size().reset_index
().rename(columns={0:'records'}))
view_item = pd.merge(view_data, item_data, on='item_id')
```

In [12]:
```
view_item.head()
```

Out[12]:

	server_time	device_type	session_id	cust_id	item_id	records	item_price	product_type
0	11/15/2029 0:00	android	B004608	G7493	F5196	1	124.5	E832
1	11/15/2029 18:04	android	B80263	E4112	F5196	1	124.5	E832
2	11/15/2029 18:19	android	B56	D2704	F5196	1	124.5	E832
3	11/16/2029 18:00	android	D86759	C2072	F5196	1	124.5	E832
4	11/16/2029 2:54	android	J23969	I731	F5196	1	124.5	E832

In [ ]:
```
#del tr_data, ts_data
```

In [13]:
```
view_item.columns
```

Out[13]:
```
Index(['server_time', 'device_type', 'session_id', 'cust_id', 'item_id',
       'records', 'item_price', 'product_type'],
      dtype='object')
```

In [14]:
```
temp_data_1 = pd.DataFrame(view_item.groupby(['cust_id'])['item_id'].nunique()
.reset_index())
temp_data_1.columns = ['cust_id', 'unique_items_viewed']

temp_data_2 = pd.DataFrame(view_item.groupby(['cust_id'])['session_id'].nuniqu
e().reset_index())
temp_data_2.columns = ['cust_id', 'unique_sessions']

temp_data_3 = pd.DataFrame(view_item.groupby(['cust_id'])['product_type'].nuni
que().reset_index())
temp_data_3.columns = ['cust_id', 'unique_product_type']

temp_data_4 = pd.DataFrame(view_item.groupby(['cust_id'])['item_price'].mean()
.reset_index())
temp_data_4.columns = ['cust_id', 'avg_item_price']

temp_data_5 = pd.DataFrame(view_item.groupby(['cust_id'])['records'].sum().res
et_index())
temp_data_5.columns = ['cust_id', 'total_records']
```

In [15]:
```
user_df = ((temp_data_1.merge(temp_data_2, on=['cust_id'], how='left')).merge(
temp_data_3, on=['cust_id'], how='left')).merge(temp_data_4, on=['cust_id'], h
ow='left')
user_df = user_df.merge(temp_data_5, on=['cust_id'], how='left')
```

In [41]:
```
view_item['server_time'] = pd.to_datetime(view_item['server_time'])
```

In [42]:
```
view_item.head()
```

Out[42]:

	server_time	device_type	session_id	cust_id	item_id	records	item_price	product_type
0	2029-11-15 00:00:00	android	B004608	G7493	F5196	1	124.5	E832
1	2029-11-15 18:04:00	android	B80263	E4112	F5196	1	124.5	E832
2	2029-11-15 18:19:00	android	B56	D2704	F5196	1	124.5	E832
3	2029-11-16 18:00:00	android	D86759	C2072	F5196	1	124.5	E832
4	2029-11-16 02:54:00	android	J23969	I731	F5196	1	124.5	E832

In [43]:
```
view_item["log_Year"] = view_item["server_time"].dt.year

view_item["log_Month"] = view_item["server_time"].dt.month

view_item["log_Day"] = view_item["server_time"].dt.day

view_item["log_WeekDay"] = view_item["server_time"].dt.weekday

view_item["log_time"] = view_item["server_time"].dt.time

view_item[['log_h','log_m','log_s']] = view_item['log_time'].astype(str).str.s
plit(':', expand=True).astype(int)
```

In [19]:
```
days_active = view_item.reset_index().groupby(['cust_id'])['server_time'].agg(
lambda x: (x.max() - x.min()).days if (x.max() - x.min()).days !=0 else 1)
unique_days_active = view_item.reset_index().groupby(['cust_id'])['server_tim
e'].agg(lambda x: len(np.unique(x.dt.dayofyear)))
user_time_features = days_active.reset_index().merge(unique_days_active.reset_
index(),on='cust_id',how = 'left')
user_time_features.columns = ['cust_id','log_days_active','log_unique_days_act
ive']
```

In [ ]:
```
#view_item = view_item.merge(user_time_features, on=['cust_id'], how='left')
```

In [ ]:
```
#view_item.head()
```

In [20]:
```
view_item.log_Month.unique()
```

Out[20]:
```
array([11, 12], dtype=int64)
```

In [21]:
```
log_Month_df = pd.pivot_table(view_item, values="session_id", index="cust_id", columns="log_Month", aggfunc="count", fill_value=0).reset_index()
print(log_Month_df.columns)

log_Month_df.columns = ["cust_id"] + ["log_Month_"+str(i) for i in range(11,13)]
```

```
Index(['cust_id', 11, 12], dtype='object', name='log_Month')
```

In [22]:
```
log_WeekDay_df = pd.pivot_table(view_item, values="session_id", index="cust_id", columns="log_WeekDay", aggfunc="count", fill_value=0).reset_index()
print(log_WeekDay_df.columns)

log_WeekDay_df.columns = ["cust_id"] + ["log_WeekDay_"+str(i) for i in range(0,7)]
```

```
Index(['cust_id', 0, 1, 2, 3, 4, 5, 6], dtype='object', name='log_WeekDay')
```

In [ ]:
```
#view_item = (view_item.merge(log_Month_df, on=['cust_id'], how='left')).merge(log_WeekDay_df, on=['cust_id'], how='left')
```

In [ ]:
```
#view_item.head()
```

In [23]:
```
bins = [0,7,15,22,31]

group_names = [1, 2, 3, 4]
view_item['week_month'] = pd.cut(view_item['log_Day'], bins, labels=group_names)
view_item.head()
```

Out[23]:

	server_time	device_type	session_id	cust_id	item_id	records	item_price	product_type	log
0	2029-11-15 00:00:00	android	B004608	G7493	F5196	1	124.5	E832	
1	2029-11-15 18:04:00	android	B80263	E4112	F5196	1	124.5	E832	
2	2029-11-15 18:19:00	android	B56	D2704	F5196	1	124.5	E832	
3	2029-11-16 18:00:00	android	D86759	C2072	F5196	1	124.5	E832	
4	2029-11-16 02:54:00	android	J23969	I731	F5196	1	124.5	E832	

In [24]:
```
view_item.week_month.unique()
```

Out[24]:
```
[2, 3, 4, 1]
Categories (4, int64): [1 < 2 < 3 < 4]
```

In [25]:
```
view_item['week_month'] = pd.to_numeric(view_item['week_month'])
```

In [26]:
```
log_week_month_df = pd.pivot_table(view_item, index="cust_id", columns="week_month", values="session_id", aggfunc="count", fill_value=0).reset_index()
log_week_month_df.columns = ["cust_id"] + ["log_week_month_"+str(i) for i in range(1,5)]
```

In [27]:
```
temp_data_6 = pd.DataFrame(view_data.groupby(['cust_id'])['device_type'].nunique().reset_index())
temp_data_6.columns = ['cust_id', 'unique_devices']
user_df = user_df.merge(temp_data_6, on=['cust_id'], how='left')
```

In [ ]:
```
#temp_data_10 = pd.pivot_table(view_data, index = 'user_id', values = ['records'], columns = ['server_time_min'], aggfunc = 'count', fill_value = 0)
```

In [ ]:
```
# list_col = list(temp_data_10.columns.values)
# list_col
# list_col_copy = []
# for i in list_col:
#     str1 = str(i[0])
#     str2 = str(i[1])
#     list_col_copy.append('views_in_server_time_min_'+str2)

# d = {ord(x):"_" for x in ":-() &"}
# new_list = [x.translate(d) for x in list_col_copy]

# temp_data_10.columns = new_list
# temp_data_10 = temp_data_10.reset_index()
```

In [ ]:
```
#temp_data_11 = pd.pivot_table(view_data, index = 'user_id', values = ['records'], columns = ['server_time_hr'], aggfunc = 'count', fill_value = 0)
```

In [ ]:
```
#list_col = list(temp_data_11.columns.values)
#list_col
#list_col_copy = []
#for i in list_col:
#    str1 = str(i[0])
#    str2 = str(i[1])
#    list_col_copy.append('views_in_server_time_hr_'+str2)

#d = {ord(x):"_" for x in ":-() &"}
#new_list = [x.translate(d) for x in list_col_copy]

#temp_data_11.columns = new_list
#temp_data_11 = temp_data_11.reset_index()
```

In [ ]:
```
#user_df = ((user_df.merge(temp_data_9, on=['user_id'], how='left'))).merge(temp_data_11, on=['user_id'], how='left')
```

In [31]:
```
user_df.tail()
```

Out[31]:

	cust_id	unique_items_viewed	unique_sessions	unique_product_type	avg_item_price	tot
76527	J995	2	1	2	958.500000	
76528	J996	2	3	2	1874.666667	
76529	J997	14	6	14	6247.312500	
76530	J998	8	4	8	8461.000000	
76531	J999	2	1	2	1319.750000	

In [44]:
```
view_item.sort_values(['cust_id',"server_time"],ascending=True,inplace=True)
user_df['cumcount_1']=view_item.groupby("cust_id")["session_id"].cumcount() +
1

view_item.sort_values(['cust_id','item_id',"server_time"],ascending=True,inpla
ce=True)
user_df['cumcount_2']=view_item.groupby(["cust_id",'item_id'])["session_id"].c
umcount() + 1

view_item.sort_values(['cust_id','session_id','item_id',"server_time"],ascendi
ng=True,inplace=True)
user_df['cumcount_3']=view_item.groupby(["cust_id",'session_id','item_id'])["s
ession_id"].cumcount() + 1
```

In [45]:
```
view_item['device_type']=view_item['device_type'].astype('category')
view_item['session_id']=view_item['session_id'].astype('category')
view_item['item_id']=view_item['item_id'].astype('category')

server_time = view_item.server_time
view_item.drop(['server_time'],axis=1,inplace=True)
```

In [46]:
```
view_item.head()
```

Out[46]:

	device_type	session_id	cust_id	item_id	records	item_price	product_type	log_Year
632212	android	B44465	A	B26865	1	2368.0	J344	2029
40360	android	B44465	A	G0603	1	2614.0	C823	2029
566626	android	I6373	A	B16073	1	2090.5	B24	2029
616876	android	J21046	A	B2439	1	9920.0	J015	2029
632000	android	J68970	A	B26865	1	2368.0	J344	2029

In [48]:
```
view_item[view_item['cust_id'] == 'J995']
```

Out[48]:

	device_type	session_id	cust_id	item_id	records	item_price	product_type	log_Year
376675	android	F97857	J995	B24676	1	480.0	J362	2029
376680	android	F97857	J995	B24676	1	480.0	J362	2029
376685	android	F97857	J995	B24676	1	480.0	J362	2029
468431	android	F97857	J995	D9095	1	1437.0	I579	2029
468437	android	F97857	J995	D9095	1	1437.0	I579	2029
468439	android	F97857	J995	D9095	1	1437.0	I579	2029

In [51]:
```
cat_agg=['count','nunique']
num_agg=['min','mean','max','sum']
agg_col={
    'device_type':cat_agg, 'session_id':cat_agg, 'item_id':cat_agg,'item_pric
e':num_agg,
       'product_type':['count','nunique']
}

for k in view_item.columns:
    if k.startswith('cumcount'):
        agg_col[k]=num_agg
```

In [56]:
```
view_item1=view_item.groupby('cust_id').agg(agg_col)
```

In [57]:
```
view_item1.head()
```

Out[57]:

	device_type		session_id		item_id		item_price			
	count	nunique	count	nunique	count	nunique	min	mean	max	sum
cust_id										
A	9	1	9	4	9	6	576.0	2466.888889	9920.0	22202.(
B0	3	1	3	3	3	3	1112.0	10076.166667	27840.0	30228.5
B00	13	1	13	2	13	11	316.5	7148.269231	63680.0	92927.5
B000	4	1	4	2	4	2	905.5	1239.125000	2240.0	4956.5
B0001	8	1	8	1	8	6	208.0	12068.750000	72000.0	96550.(

In [58]:
```
view_item1.columns=['view_' + '_'.join(col).strip() for col in view_item1.colu
mns.values]
view_item1.reset_index(inplace=True)
view_item1.head()
```

Out[58]:

	cust_id	view_device_type_count	view_device_type_nunique	view_session_id_count	view_sess
0	A	9	1	9	
1	B0	3	1	3	
2	B00	13	1	13	
3	B000	4	1	4	
4	B0001	8	1	8	

In [ ]:
```
```

Using all test and train data

In [91]:
```
train['flag'] = 'Train'
test['flag'] = 'Test'
test['click_flag'] = None

full_df = train.append(test, ignore_index = True)
```

In [92]:
```
full_df.head()
```

Out[92]:

	impression_id	time_stamp	cust_id	app_code	os_version	lte_flag	click
0	c4ca4238a0b923820dcc509a6f75849b	11/15/2029 0:00	I7862	E22	obsolete	0	
1	a87ff679a2f3e71d9181a67b7542122c	11/15/2029 0:00	E238	D71	trending	0	
2	eccbc87e4b5ce2fe28308fd9f2a7baf3	11/15/2029 0:00	F8442	B27	trending	0	
3	c81e728d9d4c2f636f067f89cc14862c	11/15/2029 0:00	I9464	B29	medium	0	
4	45c48cce2e2d7fbdea1afc51c7c6ad26	11/15/2029 0:01	G3410	E67	trending	1	

In [63]:
```
#del train,test
```

In [93]:
```
full_df.os_version.unique()
```

Out[93]:
```
array(['obsolete', 'trending', 'medium'], dtype=object)
```

In [94]:
```
full_df.app_code.nunique()
```

Out[94]:
```
490
```

In [95]:
```
full_df['time_stamp']=pd.to_datetime(full_df['time_stamp'])
```

In [96]:
```
full_df['lte_flag']=full_df['lte_flag'].astype('category')
full_df['app_code']=full_df['app_code'].astype('category')

timestamp = full_df.time_stamp
full_df.drop(['time_stamp'],axis=1,inplace=True)

full_df.head()
```

Out[96]:

	impression_id	cust_id	app_code	os_version	lte_flag	click_flag	flag
0	c4ca4238a0b923820dcc509a6f75849b	I7862	E22	obsolete	0	0	Train
1	a87ff679a2f3e71d9181a67b7542122c	E238	D71	trending	0	0	Train
2	eccbc87e4b5ce2fe28308fd9f2a7baf3	F8442	B27	trending	0	0	Train
3	c81e728d9d4c2f636f067f89cc14862c	I9464	B29	medium	0	0	Train
4	45c48cce2e2d7fbdea1afc51c7c6ad26	G3410	E67	trending	1	1	Train

In [97]:
```
full_df['ad_timestamp'] = timestamp
```

In [98]:
```
full_df["ad_Year"] = full_df["ad_timestamp"].dt.year

full_df["ad_Month"] = full_df["ad_timestamp"].dt.month

full_df["ad_Day"] = full_df["ad_timestamp"].dt.day

full_df["ad_WeekDay"] = full_df["ad_timestamp"].dt.weekday

full_df["ad_time"] = full_df["ad_timestamp"].dt.time

full_df[['ad_h','ad_m','ad_s']] = full_df['ad_time'].astype(str).str.split(':'
, expand=True).astype(int)
```

In [99]:
```
ad_days_active = full_df.reset_index().groupby(['cust_id'])['ad_timestamp'].ag
g(lambda x: (x.max() - x.min()).days if (x.max() - x.min()).days !=0 else 1)
ad_unique_days_active = full_df.reset_index().groupby(['cust_id'])['ad_timesta
mp'].agg(lambda x: len(np.unique(x.dt.dayofyear)))
ad_user_time_features = ad_days_active.reset_index().merge(ad_unique_days_acti
ve.reset_index(),on='cust_id',how = 'left')
ad_user_time_features.columns = ['cust_id','ad_days_active','ad_unique_days_ac
tive']
```

In [100]:
```
ad_Month_df = pd.pivot_table(full_df, values="impression_id", index="cust_id", columns="ad_Month", aggfunc="count", fill_value=0).reset_index()
print(ad_Month_df.columns)

ad_Month_df.columns = ["cust_id"] + ["ad_Month_"+str(i) for i in range(11,13)]
```

```
Index(['cust_id', 11, 12], dtype='object', name='ad_Month')
```

In [101]:
```
ad_WeekDay_df = pd.pivot_table(full_df, values="impression_id", index="cust_id", columns="ad_WeekDay", aggfunc="count", fill_value=0).reset_index()
print(ad_WeekDay_df.columns)

ad_WeekDay_df.columns = ["cust_id"] + ["ad_WeekDay_"+str(i) for i in range(0,7)]
```

```
Index(['cust_id', 0, 1, 2, 3, 4, 5, 6], dtype='object', name='ad_WeekDay')
```

In [102]:
```
full_df['ad_week_month'] = pd.cut(full_df['ad_Day'], bins, labels=group_names)
full_df.head()
```

Out[102]:

	impression_id	cust_id	app_code	os_version	lte_flag	click_flag	flag
0	c4ca4238a0b923820dcc509a6f75849b	I7862	E22	obsolete	0	0	Train
1	a87ff679a2f3e71d9181a67b7542122c	E238	D71	trending	0	0	Train
2	eccbc87e4b5ce2fe28308fd9f2a7baf3	F8442	B27	trending	0	0	Train
3	c81e728d9d4c2f636f067f89cc14862c	I9464	B29	medium	0	0	Train
4	45c48cce2e2d7fbdea1afc51c7c6ad26	G3410	E67	trending	1	1	Train

In [103]:
```
full_df['ad_week_month'] = pd.to_numeric(full_df['ad_week_month'])
```

In [104]:
```
ad_week_month_df = pd.pivot_table(full_df, index="cust_id", columns="ad_week_month", values="impression_id", aggfunc="count", fill_value=0).reset_index()
ad_week_month_df.columns = ["cust_id"] + ["ad_week_month_"+str(i) for i in range(1,5)]
```

In [ ]:
```

```

Merging all user features to full_df

In [105]:
```
full_df = pd.merge(full_df, user_df, on= 'cust_id', how='left')
full_df = full_df.merge(user_time_features, on=['cust_id'], how='left')
full_df = full_df.merge(log_Month_df, on=['cust_id'], how='left')
full_df = full_df.merge(log_WeekDay_df, on=['cust_id'], how='left')
full_df = full_df.merge(log_week_month_df, on=['cust_id'], how='left')
full_df = full_df.merge(view_item1, on=['cust_id'], how='left')
```

Merging all ad features to full_df

In [106]:
```
full_df = full_df.merge(ad_user_time_features, on=['cust_id'], how='left')
full_df = full_df.merge(ad_Month_df, on=['cust_id'], how='left')
full_df = full_df.merge(ad_WeekDay_df, on=['cust_id'], how='left')
full_df = full_df.merge(ad_week_month_df, on=['cust_id'], how='left')
```

In [107]:
```
full_df.columns
```

Out[107]:
```
Index(['impression_id', 'cust_id', 'app_code', 'os_version', 'lte_flag',
       'click_flag', 'flag', 'ad_timestamp', 'ad_Year', 'ad_Month', 'ad_Day',
       'ad_WeekDay', 'ad_time', 'ad_h', 'ad_m', 'ad_s', 'ad_week_month',
       'unique_items_viewed', 'unique_sessions', 'unique_product_type',
       'avg_item_price', 'total_records', 'unique_devices', 'cumcount_1',
       'cumcount_2', 'cumcount_3', 'log_days_active', 'log_unique_days_activ
e',
       'log_Month_11', 'log_Month_12', 'log_WeekDay_0', 'log_WeekDay_1',
       'log_WeekDay_2', 'log_WeekDay_3', 'log_WeekDay_4', 'log_WeekDay_5',
       'log_WeekDay_6', 'log_week_month_1', 'log_week_month_2',
       'log_week_month_3', 'log_week_month_4', 'view_device_type_count',
       'view_device_type_nunique', 'view_session_id_count',
       'view_session_id_nunique', 'view_item_id_count', 'view_item_id_nuniqu
e',
       'view_item_price_min', 'view_item_price_mean', 'view_item_price_max',
       'view_item_price_sum', 'view_product_type_count',
       'view_product_type_nunique', 'ad_days_active', 'ad_unique_days_activ
e',
       'ad_Month_11', 'ad_Month_12', 'ad_WeekDay_0', 'ad_WeekDay_1',
       'ad_WeekDay_2', 'ad_WeekDay_3', 'ad_WeekDay_4', 'ad_WeekDay_5',
       'ad_WeekDay_6', 'ad_week_month_1', 'ad_week_month_2', 'ad_week_month_
3',
       'ad_week_month_4'],
      dtype='object')
```

In [108]:
```
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 237609 entries, 0 to 237608
Data columns (total 68 columns):
impression_id                237609 non-null object
cust_id                      237609 non-null object
app_code                     237609 non-null category
os_version                   237609 non-null object
lte_flag                     237609 non-null category
click_flag                   197093 non-null object
flag                         237609 non-null object
ad_timestamp                 237609 non-null datetime64[ns]
ad_Year                      237609 non-null int64
ad_Month                     237609 non-null int64
ad_Day                       237609 non-null int64
ad_WeekDay                   237609 non-null int64
ad_time                      237609 non-null object
ad_h                         237609 non-null int32
ad_m                         237609 non-null int32
ad_s                         237609 non-null int32
ad_week_month                237609 non-null int64
unique_items_viewed          215290 non-null float64
unique_sessions              215290 non-null float64
unique_product_type          215290 non-null float64
avg_item_price               215290 non-null float64
total_records                215290 non-null float64
unique_devices               215290 non-null float64
cumcount_1                   215290 non-null float64
cumcount_2                   215290 non-null float64
cumcount_3                   215290 non-null float64
log_days_active              215290 non-null float64
log_unique_days_active       215290 non-null float64
log_Month_11                 215290 non-null float64
log_Month_12                 215290 non-null float64
log_WeekDay_0                215290 non-null float64
log_WeekDay_1                215290 non-null float64
log_WeekDay_2                215290 non-null float64
log_WeekDay_3                215290 non-null float64
log_WeekDay_4                215290 non-null float64
log_WeekDay_5                215290 non-null float64
log_WeekDay_6                215290 non-null float64
log_week_month_1             215290 non-null float64
log_week_month_2             215290 non-null float64
log_week_month_3             215290 non-null float64
log_week_month_4             215290 non-null float64
view_device_type_count       215290 non-null float64
view_device_type_nunique     215290 non-null float64
view_session_id_count        215290 non-null float64
view_session_id_nunique      215290 non-null float64
view_item_id_count           215290 non-null float64
view_item_id_nunique         215290 non-null float64
view_item_price_min          215290 non-null float64
view_item_price_mean         215290 non-null float64
view_item_price_max          215290 non-null float64
view_item_price_sum          215290 non-null float64
view_product_type_count      215290 non-null float64
view_product_type_nunique    215290 non-null float64
ad_days_active               237609 non-null int64
```

```
ad_unique_days_active        237609 non-null int64
ad_Month_11                  237609 non-null int64
ad_Month_12                  237609 non-null int64
ad_WeekDay_0                 237609 non-null int64
ad_WeekDay_1                 237609 non-null int64
ad_WeekDay_2                 237609 non-null int64
ad_WeekDay_3                 237609 non-null int64
ad_WeekDay_4                 237609 non-null int64
ad_WeekDay_5                 237609 non-null int64
ad_WeekDay_6                 237609 non-null int64
ad_week_month_1              237609 non-null int64
ad_week_month_2              237609 non-null int64
ad_week_month_3              237609 non-null int64
ad_week_month_4              237609 non-null int64
dtypes: category(2), datetime64[ns](1), float64(36), int32(3), int64(20), obj
ect(6)
memory usage: 119.4+ MB
```

In [109]:
```
full_df['os_version']= [1 if x == 'obsolete' else x for x in full_df["os_versi
on"]]
full_df['os_version']= [2 if x == 'medium' else x for x in full_df["os_versio
n"]]
full_df['os_version']= [3 if x == 'trending' else x for x in full_df["os_versi
on"]]
full_df['os_version']= pd.to_numeric(full_df['os_version'])
```

In [110]:
```
full_df['lte_flag']= pd.to_numeric(full_df['lte_flag'])
```

Training Model

In [146]:
```
df_train=full_df[full_df['click_flag'].isnull()==False].copy()
df_test=full_df[full_df['click_flag'].isnull()==True].copy()

df_train['click_flag']= pd.to_numeric(df_train['click_flag'])
```

In [147]:
```
print(df_test.shape, df_train.shape, full_df.shape)
```

```
(40516, 68) (197093, 68) (237609, 68)
```

In [148]:
```
full_df.isnull().sum()
```

Out[148]:

impression_id	0
cust_id	0
app_code	0
os_version	0
lte_flag	0
click_flag	40516
flag	0
ad_timestamp	0
ad_Year	0
ad_Month	0
ad_Day	0
ad_WeekDay	0
ad_time	0
ad_h	0
ad_m	0
ad_s	0
ad_week_month	0
unique_items_viewed	22319
unique_sessions	22319
unique_product_type	22319
avg_item_price	22319
total_records	22319
unique_devices	22319
cumcount_1	22319
cumcount_2	22319
cumcount_3	22319
log_days_active	22319
log_unique_days_active	22319
log_Month_11	22319
log_Month_12	22319
	...
log_week_month_2	22319
log_week_month_3	22319
log_week_month_4	22319
view_device_type_count	22319
view_device_type_nunique	22319
view_session_id_count	22319
view_session_id_nunique	22319
view_item_id_count	22319
view_item_id_nunique	22319
view_item_price_min	22319
view_item_price_mean	22319
view_item_price_max	22319
view_item_price_sum	22319
view_product_type_count	22319
view_product_type_nunique	22319
ad_days_active	0
ad_unique_days_active	0
ad_Month_11	0
ad_Month_12	0
ad_WeekDay_0	0
ad_WeekDay_1	0
ad_WeekDay_2	0
ad_WeekDay_3	0
ad_WeekDay_4	0
ad_WeekDay_5	0
ad_WeekDay_6	0

```
ad_week_month_1                 0
ad_week_month_2                 0
ad_week_month_3                 0
ad_week_month_4                 0
Length: 68, dtype: int64
```

Random Forest

In [160]:
```
from catboost import CatBoostClassifier,Pool, cv
from lightgbm import LGBMClassifier
from sklearn.model_selection import StratifiedKFold,train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,roc_auc_score
from sklearn.metrics import f1_score, classification_report
```

In [150]:
```
X,y = df_train.drop(['impression_id', 'cust_id', 'app_code', 'flag', 'ad_time'
, 'ad_timestamp', 'click_flag'], axis = 1), df_train['click_flag']
Xtest=df_test.drop(['impression_id', 'cust_id', 'app_code', 'flag', 'ad_time',
'ad_timestamp', 'click_flag'],axis=1)
print(X.shape,Xtest.shape)

for i in X.columns:
    X[i].fillna(0, inplace = True)
    Xtest[i].fillna(0, inplace = True)

Xtrain,X_val,ytrain,y_val = train_test_split(X,y,test_size=0.20,random_state =
1996,stratify=y)
print(Xtrain.shape, X_val.shape)
```

```
(197093, 61) (40516, 61)
(157674, 61) (39419, 61)
```

In [151]:
```
X.isnull().sum()
```

Out[151]:
```
os_version                  0
lte_flag                    0
ad_Year                     0
ad_Month                    0
ad_Day                      0
ad_WeekDay                  0
ad_h                        0
ad_m                        0
ad_s                        0
ad_week_month               0
unique_items_viewed         0
unique_sessions             0
unique_product_type         0
avg_item_price              0
total_records               0
unique_devices              0
cumcount_1                  0
cumcount_2                  0
cumcount_3                  0
log_days_active             0
log_unique_days_active      0
log_Month_11                0
log_Month_12                0
log_WeekDay_0               0
log_WeekDay_1               0
log_WeekDay_2               0
log_WeekDay_3               0
log_WeekDay_4               0
log_WeekDay_5               0
log_WeekDay_6               0
                           ..
log_week_month_2            0
log_week_month_3            0
log_week_month_4            0
view_device_type_count      0
view_device_type_nunique    0
view_session_id_count       0
view_session_id_nunique     0
view_item_id_count          0
view_item_id_nunique        0
view_item_price_min         0
view_item_price_mean        0
view_item_price_max         0
view_item_price_sum         0
view_product_type_count     0
view_product_type_nunique   0
ad_days_active              0
ad_unique_days_active       0
ad_Month_11                 0
ad_Month_12                 0
ad_WeekDay_0                0
ad_WeekDay_1                0
ad_WeekDay_2                0
ad_WeekDay_3                0
ad_WeekDay_4                0
ad_WeekDay_5                0
ad_WeekDay_6                0
```

```
ad_week_month_1              0
ad_week_month_2              0
ad_week_month_3              0
ad_week_month_4              0
Length: 61, dtype: int64
```

In [152]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
clf = RandomForestClassifier(n_estimators = 350, random_state = 7000)
clf.fit(Xtrain, ytrain)
```

Out[152]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=No
ne,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=350,
                       n_jobs=None, oob_score=False, random_state=7000,
                       verbose=0, warm_start=False)
```

In [154]:
```
sel = SelectFromModel(clf)
sel.fit(Xtrain, ytrain)
selected_feat= X_train.columns[(sel.get_support())]
```

In [155]:
```
preds_rf = clf.predict(X_val)
print(confusion_matrix(y_val.astype('int32'), preds_rf))
print(accuracy_score(y_val.astype('int32'), preds_rf))
```

```
[[37143   482]
 [ 1692   102]]
0.9448489307186889
```

LightGBM

In [176]:
```
err=[]
y_pred_tot=[]
y_out = []
from sklearn.model_selection import KFold,StratifiedKFold
fold=StratifiedKFold(n_splits=10,shuffle=True,random_state=1994)
i=1
#print(X.shape, y.shape)
y = y.astype('int32')
y_test = y_test.astype('int32')
#from sklearn.utils.multiclass import type_of_target
#print(type_of_target(y))
#a,b = fold.split(X,y)
for train_index, test_index in fold.split(X,y):
    Xtrain, X_test = X.iloc[train_index], X.iloc[test_index]
    ytrain, y_test = y[train_index], y[test_index]
    m=LGBMClassifier(n_estimators=1000,random_state=1994,learning_rate=0.08,co
lsample_bytree=0.2,objective='binary',scale_pos_weight=1)
    m.fit(Xtrain,ytrain,eval_set=[(X_test, y_test)],eval_metric='f1', early_st
opping_rounds=200,verbose=200)
    preds=m.predict_proba(X_test)[:,-1]
    #print("err: ",roc_auc_score(y_test,preds))
    preds_class = m.predict(X_test)
    print("accuracy: ", accuracy_score(y_test, preds_class))
    err.append(roc_auc_score(y_test,preds))
    p = m.predict_proba(Xtest)[:,-1]
    i=i+1
    y_pred_tot.append(p)
    out = m.predict(Xtest)
    y_out.append(out)
```

```
Training until validation scores don't improve for 200 rounds
[200]   valid_0's binary_logloss: 0.172999
[400]   valid_0's binary_logloss: 0.171755
[600]   valid_0's binary_logloss: 0.171559
Early stopping, best iteration is:
[554]   valid_0's binary_logloss: 0.171475
accuracy:  0.954439370877727
Training until validation scores don't improve for 200 rounds
[200]   valid_0's binary_logloss: 0.172962
[400]   valid_0's binary_logloss: 0.171732
[600]   valid_0's binary_logloss: 0.170946
[800]   valid_0's binary_logloss: 0.171001
[1000]  valid_0's binary_logloss: 0.171262
Did not meet early stopping. Best iteration is:
[814]   valid_0's binary_logloss: 0.170845
accuracy:  0.954337899543379
Training until validation scores don't improve for 200 rounds
[200]   valid_0's binary_logloss: 0.174034
[400]   valid_0's binary_logloss: 0.172916
[600]   valid_0's binary_logloss: 0.172463
[800]   valid_0's binary_logloss: 0.172476
Early stopping, best iteration is:
[613]   valid_0's binary_logloss: 0.172339
accuracy:  0.9539320142059868
Training until validation scores don't improve for 200 rounds
[200]   valid_0's binary_logloss: 0.172322
[400]   valid_0's binary_logloss: 0.170826
[600]   valid_0's binary_logloss: 0.170328
[800]   valid_0's binary_logloss: 0.170678
Early stopping, best iteration is:
[667]   valid_0's binary_logloss: 0.170103
accuracy:  0.9546930492135972
Training until validation scores don't improve for 200 rounds
[200]   valid_0's binary_logloss: 0.174258
[400]   valid_0's binary_logloss: 0.173144
[600]   valid_0's binary_logloss: 0.172918
Early stopping, best iteration is:
[574]   valid_0's binary_logloss: 0.172843
accuracy:  0.9544370592115278
Training until validation scores don't improve for 200 rounds
[200]   valid_0's binary_logloss: 0.174598
[400]   valid_0's binary_logloss: 0.173036
[600]   valid_0's binary_logloss: 0.173319
Early stopping, best iteration is:
[499]   valid_0's binary_logloss: 0.172839
accuracy:  0.9545892739357654
Training until validation scores don't improve for 200 rounds
[200]   valid_0's binary_logloss: 0.172742
[400]   valid_0's binary_logloss: 0.171743
[600]   valid_0's binary_logloss: 0.171708
[800]   valid_0's binary_logloss: 0.17173
Early stopping, best iteration is:
[617]   valid_0's binary_logloss: 0.171482
accuracy:  0.9544877974529403
Training until validation scores don't improve for 200 rounds
[200]   valid_0's binary_logloss: 0.172254
[400]   valid_0's binary_logloss: 0.171013
```

```
[600]   valid_0's binary_logloss: 0.170931
[800]   valid_0's binary_logloss: 0.170603
Early stopping, best iteration is:
[672]   valid_0's binary_logloss: 0.170535
accuracy:  0.9544877974529403
Training until validation scores don't improve for 200 rounds
[200]   valid_0's binary_logloss: 0.174411
[400]   valid_0's binary_logloss: 0.172907
[600]   valid_0's binary_logloss: 0.172542
[800]   valid_0's binary_logloss: 0.172547
[1000]  valid_0's binary_logloss: 0.17252
Did not meet early stopping. Best iteration is:
[889]   valid_0's binary_logloss: 0.172148
accuracy:  0.9540311532802274
Training until validation scores don't improve for 200 rounds
[200]   valid_0's binary_logloss: 0.173961
[400]   valid_0's binary_logloss: 0.172701
[600]   valid_0's binary_logloss: 0.172628
Early stopping, best iteration is:
[535]   valid_0's binary_logloss: 0.172502
accuracy:  0.9543840064948245
```

In [174]:
```
np.random.uniform(0.15,0.3,1)
```

Out[174]:
```
array([0.25884092])
```

In [184]:
```
test['click_flag']=np.mean(y_out,0)
test.click_flag = test.click_flag.astype('int32')
submit = test.drop(['flag'], axis=1)
submit.head()
```

Out[184]:

	impression_id	time_stamp	cust_id	app_code	os_version	lte_flag	click
0	de59f5885f41cf82f8f12d1c20d0471f	12/7/2029 0:00	F3450	D8	medium	1	
1	96970a8395afcfbbad29b200755b5c61	12/7/2029 0:00	J1192	C07	trending	0	
2	8a1d38954e079d4222f54fa81db0caf2	12/7/2029 0:00	H4151	B90	obsolete	0	
3	519a0e84d5a19de6555f71deb3f21f97	12/7/2029 0:00	D7356	B90	trending	0	
4	14136e7ff83fa5ea23d336409ee5a347	12/7/2029 0:00	F7418	D18	trending	1	

In [186]:
```
submit.to_csv("D:\Academics\PGDBA\Competitions\Predixion/submit.csv", index = False)
```

In [ ]:
```

```