In [10]:

```
import gc
import time
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
print(os.listdir('D:/Kaggle/instacart'))
```

```
['.ipynb_checkpoints', 'aisles.csv', 'code.ipynb', 'departments.csv', 'inst
arcart_notes.docx', 'notes.docx', 'orders.csv',
'order_products__prior.csv', 'order_products__train.csv', 'products.csv', '
sample_submission.csv', '~$starcart_notes.docx']
```

In [12]:

```
def load_data(path_data):

    priors = pd.read_csv(path_data + 'order_products__prior.csv',
                         dtype={
                                'order_id': np.int32,
                                'product_id': np.uint16,
                                'add_to_cart_order': np.int16,
                                'reordered': np.int8})
    train = pd.read_csv(path_data + 'order_products__train.csv',
                        dtype={
                                'order_id': np.int32,
                                'product_id': np.uint16,
                                'add_to_cart_order': np.int16,
                                'reordered': np.int8})

    orders = pd.read_csv(path_data + 'orders.csv',
                             dtype={
                                    'order_id': np.int32,
                                    'user_id': np.int64,
                                    'eval_set': 'category',
                                    'order_number': np.int16,
                                    'order_dow': np.int8,
                                    'order_hour_of_day': np.int8,
                                    'days_since_prior_order': np.float32})

    products = pd.read_csv(path_data + 'products.csv')
    aisles = pd.read_csv(path_data + "aisles.csv")
    departments = pd.read_csv(path_data + "departments.csv")
    sample_submission = pd.read_csv(path_data + "sample_submission.csv")

    return priors, train, orders, products, aisles, departments, sample_subm
ission

class tick_tock:
    def __init__(self, process_name, verbose=1):
        self.process_name = process_name
        self.verbose = verbose
    def __enter__(self):
        if self.verbose:
            print(self.process_name + " begin ......")
            self.begin_time = time.time()
    def __exit__(self, type, value, traceback):
```

```
    def __exit__(self, type, value, traceback):
        if self.verbose:
            end_time = time.time()
            print(self.process_name + " end ......")
            print('time lapsing {0} s \n'.format(end_time - self.begin_time)
)

def ka_add_groupby_features_1_vs_n(df, group_columns_list, agg_dict, only_n
ew_feature=True):

    with tick_tock("add stats features"):
        try:
            if type(group_columns_list) == list:
                pass
            else:
                raise TypeError(k + "should be a list")
        except TypeError as e:
            print(e)
            raise

        df_new = df.copy()
        grouped = df_new.groupby(group_columns_list)

        the_stats = grouped.agg(agg_dict)
        the_stats.columns = the_stats.columns.droplevel(0)
        the_stats.reset_index(inplace=True)
        if only_new_feature:
            df_new = the_stats
        else:
            df_new = pd.merge(left=df_new, right=the_stats, on=group_columns
_list, how='left')

    return df_new

def ka_add_groupby_features_n_vs_1(df, group_columns_list,
target_columns_list, methods_list, keep_only_stats=True, verbose=1):

    with tick_tock("add stats features", verbose):
        dicts = {"group_columns_list": group_columns_list ,
"target_columns_list": target_columns_list, "methods_list" :methods_list}

        for k, v in dicts.items():
            try:
                if type(v) == list:
                    pass
                else:
                    raise TypeError(k + "should be a list")
            except TypeError as e:
                print(e)
                raise

        grouped_name = ''.join(group_columns_list)
        target_name = ''.join(target_columns_list)
        combine_name = [[grouped_name] + [method_name] + [target_name] for m
ethod_name in methods_list]

        df_new = df.copy()
        grouped = df_new.groupby(group_columns_list)

        the_stats = grouped[target_name].agg(methods_list).reset_index()
        the_stats.columns = [grouped_name] + \
```

```
                                ['_%s_%s_by_%s' % (grouped_name, method_name, ta
rget_name) \
                                 for (grouped_name, method_name, target_name) in
combine_name]
        if keep_only_stats:
            return the_stats
        else:
            df_new = pd.merge(left=df_new, right=the_stats, on=group_columns
_list, how='left')
        return df_new
```

In [14]:

```
path_data = 'D:/Kaggle/instacart/'
priors, train, orders, products, aisles, departments, sample_submission =
load_data(path_data)


priors_orders_detail = orders.merge(right=priors, how='inner',
on='order_id')


priors_orders_detail.loc[:,'_user_buy_product_times'] =
priors_orders_detail.groupby(['user_id', 'product_id']).cumcount() + 1

agg_dict = {'user_id':{'_prod_tot_cnts':'count'},
            'reordered':{'_prod_reorder_tot_cnts':'sum'},
            '_user_buy_product_times': {'_prod_buy_first_time_total_cnt':la
mbda x: sum(x==1),
                                        '_prod_buy_second_time_total_cnt':l
bda x: sum(x==2)}}
prd = ka_add_groupby_features_1_vs_n(priors_orders_detail, ['product_id'],
agg_dict)

prd['_prod_reorder_prob'] = prd._prod_buy_second_time_total_cnt / prd._prod
_buy_first_time_total_cnt
prd['_prod_reorder_ratio'] = prd._prod_reorder_tot_cnts /
prd._prod_tot_cnts
prd['_prod_reorder_times'] = 1 + prd._prod_reorder_tot_cnts /
prd._prod_buy_first_time_total_cnt
prd.head()
```

```
add stats features begin ......
```

```
D:\Anaconda3\lib\site-packages\pandas\core\groupby.py:4036: FutureWarning:
using a dict with renaming is deprecated and will be removed in a future ve
rsion
  return super(DataFrameGroupBy, self).aggregate(arg, *args, **kwargs)
```

```
add stats features end ......
time lapsing 130.34890985488892 s
```

Out[14]:

	product_id	_prod_tot_cnts	_prod_reorder_tot_cnts	_prod_buy_first_time_total_cnt	_prod
0	1	1852	1136.0	716	276

	2 product_id	90 _prod_tot_cnts	12.0 _prod_reorder_tot_cnts	78 _prod_buy_first_time_total_cnt	8 _prod
2	3	277	203.0	74	36
3	4	329	147.0	182	64
4	5	15	9.0	6	4

In [ ]:

In [15]:

```
agg_dict_2 = {'order_number':{'_user_total_orders':'max'},
              'days_since_prior_order':{'_user_sum_days_since_prior_order':
sum',
                                        '_user_mean_days_since_prior_order'
'mean'}}
users = ka_add_groupby_features_1_vs_n(orders[orders.eval_set == 'prior'],
['user_id'], agg_dict_2)
agg_dict_3 = {'reordered':
              {'_user_reorder_ratio':
               lambda x: sum(priors_orders_detail.ix[x.index,'reordered']==1
)/
                         sum(priors_orders_detail.ix[x.index,'order_number']
> 1)},
              'product_id':{'_user_total_products':'count',
                            '_user_distinct_products': lambda x: x.nunique()
}}
us = ka_add_groupby_features_1_vs_n(priors_orders_detail, ['user_id'], agg_
dict_3)
users = users.merge(us, how='inner')
users['_user_average_basket'] = users._user_total_products / users._user_to
tal_orders
us = orders[orders.eval_set != "prior"][['user_id', 'order_id', 'eval_set',
'days_since_prior_order']]
us.rename(index=str, columns={'days_since_prior_order':
'time_since_last_order'}, inplace=True)

users = users.merge(us, how='inner')
users.head()
```

```
add stats features begin ......
```

```
D:\Anaconda3\lib\site-packages\pandas\core\groupby.py:4036: FutureWarning:
using a dict with renaming is deprecated and will be removed in a future ve
rsion
  return super(DataFrameGroupBy, self).aggregate(arg, *args, **kwargs)
```

```
add stats features end ......
time lapsing 0.46395301818847656 s

add stats features begin ......
```

```
D:\Anaconda3\lib\site-packages\ipykernel\__main__.py:15:
DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing
```

```
See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate_ix
```

```
add stats features end ......
time lapsing 1108.6702029705048 s
```

Out[15]:

	user_id	_user_total_orders	_user_sum_days_since_prior_order	_user_mean_days_since_
0	1	10	176.0	19.555555
1	2	14	198.0	15.230769
2	3	12	133.0	12.090909
3	4	5	55.0	13.750000
4	5	4	40.0	13.333333

In [16]:

```
agg_dict_4 = {'order_number':{'_up_order_count': 'count',
                              '_up_first_order_number': 'min',
                              '_up_last_order_number':'max'},
              'add_to_cart_order':{'_up_average_cart_position': 'mean'}}

data = ka_add_groupby_features_1_vs_n(df=priors_orders_detail,
                                                        group_columns_list=['
er_id', 'product_id'],
                                                        agg_dict=agg_dict_4)

data = data.merge(prd, how='inner', on='product_id').merge(users,
how='inner', on='user_id')

data['_up_order_rate'] = data._up_order_count / data._user_total_orders
data['_up_order_since_last_order'] = data._user_total_orders - data._up_las
t_order_number
data['_up_order_rate_since_first_order'] = data._up_order_count / (data._us
er_total_orders - data._up_first_order_number + 1)

train = train.merge(right=orders[['order_id', 'user_id']], how='left', on='
order_id')
data = data.merge(train[['user_id', 'product_id', 'reordered']], on=['user_
id', 'product_id'], how='left')

del priors_orders_detail, orders
gc.collect()
data.head()
```

```
add stats features begin ......
```

```
D:\Anaconda3\lib\site-packages\pandas\core\groupby.py:4036: FutureWarning:
using a dict with renaming is deprecated and will be removed in a future ve
rsion
  return super(DataFrameGroupBy, self).aggregate(arg, *args, **kwargs)
```

```
add stats features end ......
time lapsing 75.1117730140686 s
```

Out[16]:

	user_id	product_id	_up_order_count	_up_first_order_number	_up_last_order_number	_u
0	1	196	10	1	10	1.4
1	1	10258	9	2	10	3.3
2	1	10326	1	5	5	5.0
3	1	12427	10	1	10	3.3
4	1	13032	3	2	10	6.3

5 rows × 27 columns

In [22]:

```
import xgboost
from sklearn.cross_validation import train_test_split

train = data.loc[data.eval_set == "train",:]
train.drop(['eval_set', 'user_id', 'product_id', 'order_id'], axis=1, inpla
ce=True)
train.loc[:, 'reordered'] = train.reordered.fillna(0)

X_test = data.loc[data.eval_set == "test",:]

X_train, X_val, y_train, y_val = train_test_split(train.drop('reordered', a
xis=1), train.reordered,
                                                  test_size=0.9,
random_state=125)
d_train = xgboost.DMatrix(X_train, y_train)
xgb_params = {
    "objective"         : "reg:logistic"
    ,"eval_metric"      : "logloss"
    ,"eta"              : 0.1
    ,"max_depth"        : 6
    ,"min_child_weight" :10
    ,"gamma"            :0.70
    ,"subsample"        :0.76
    ,"colsample_bytree" :0.95
    ,"alpha"            :2e-05
    ,"lambda"           :8
}

watchlist= [(d_train, "train")]
bst = xgboost.train(params=xgb_params, dtrain=d_train, num_boost_round=100,
evals=watchlist, verbose_eval=10)
xgboost.plot_importance(bst)

d_test = xgboost.DMatrix(X_test.drop(['eval_set', 'user_id', 'order_id', 'r
eordered', 'product_id'], axis=1))
X_test.loc[:,'reordered'] = (bst.predict(d_test) > 0.25).astype(int)
X_test.loc[:, 'product_id'] = X_test.product_id.astype(str)
submit = ka_add_groupby_features_n_vs_1(X_test[X_test.reordered == 1],
                                        group_columns_list=['order_i
],
```

```
                                                    target_columns_list= ['produ
_id'],
                                                    methods_list=[lambda x: ' '.
in(set(x))], keep_only_stats=True)
submit.columns = sample_submission.columns.tolist()
submit_final = sample_submission[['order_id']].merge(submit, how='left').fi
llna('None')
submit_final.to_csv("python_test1.csv", index=False)
```

```
D:\Anaconda3\lib\site-packages\ipykernel\__main__.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy
D:\Anaconda3\lib\site-packages\pandas\core\indexing.py:517:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```

```
[0] train-logloss:0.625879
[10] train-logloss:0.335623
[20] train-logloss:0.268368
[30] train-logloss:0.251049
[40] train-logloss:0.246337
[50] train-logloss:0.244756
[60] train-logloss:0.244051
[70] train-logloss:0.243595
[80] train-logloss:0.243202
[90] train-logloss:0.242851
add stats features begin ......
add stats features end ......
time lapsing 3.057374954223633 s
```