Age-Gender Code Notes

1. **age\_gender\_daily\_etl.py**

This code extracts basic bundle-age/gender statistics summarization tables and ctr data. Extraction is on daily basis, and it extracts from the last date on record to 7 days before current date by default.

* 1. **populate\_ctr\_history**

Get sum of n\_clicks, n\_ctc and n\_impression from science\_core for different uid, date, country, pub\_type, category, adomain, creative\_type, banner\_size combinations into ctr\_dataset table day by day.

After looping through the dates, create a new table that aggregates over dates and replace the original ctr\_dataset table.

* 1. **populate\_raw\_uid\_bundle\_gender/age**

Extract unique uid, bundle, pub\_id, sp\_user\_gender, os, dt, cntry from science\_core into dq\_raw\_uid\_bundle\_gender, where gender is not null/age is between 5 and 85, day by day.

* 1. **populate\_bundle\_qlty\_for\_age**

Extract age bracket (11 brackets) from science\_core for difference bundle, pub\_id, os and uid (will be referred to as BPOU later). Count number of records in each of the bracket for different BPOU. Then aggregate over uid, also compute number of user in each BPO combination and number of users that have difference age bracket. Extract and insert into dq\_bundle\_qlt\_for\_age day by day.

* 1. **populate\_bundle\_qlty\_for\_gender**

Similar extraction process as **c**, extract gender information for different PBOU. Then aggregate over uid and counts the number of users and users that have different gender for each PBO combination.

1. **copy\_science\_userstore.py**
   1. **create\_ust\_kv\_sql**

Create external table based on hdfs files located on /user/xad/up\_aggr/featurestats/201\*/\*\*/\*\*/allmonthsdata/basic8/. Then aggregate over country, uid, feature, feature\_val to get the monthly sum for frequency and freq\_hr, and delete data from the external table.

* 1. **create\_ust\_no\_sprs\_vect**

Basically, just import the hdfs files located at

/user/xad/up\_aggr/featurestats/201\*/\*\*/\*\*/basic8/userstore/ into hive as table ust\_YYMMDD

1. **age\_gender\_modeling\_with\_facebook.sql**

Not sure if this file can be executed directly. It is safer to run step by step for now.

Modifications required before running any queries in this file.

a. Replace previous date for ag\_config.MODEL\_N\_UST\_DT with the new date. For example, if the old date is yymmdd and new date is YYMMDD, then the replacement includes replace “yyyy\_mm\_dd” with “YYYY\_MM\_DD”; replace “yymmdd” with “YYMMDD”

b. find and change table bndli\_ga\_profile\_yymmdd/ ga\_profile\_ffm\_yymmdd tables to most updated version.

**Step 1-4:** load facebook data from hdfs into hive and create table fb\_bundle\_demo\_distrib\_entropy, which stores gender/age ratio and percentage for each bundle.

**Step 5:** inner join with ust\_kv\_yymmdd to find all the bundle that is in current user store and facebook profile table. And store all the bundle and its corresponding uid information into table fb\_user\_demo\_vectors\_171231.

**Step 6:** compute weighted entropy for entropies in table fb\_user\_demo\_vectors\_171231. The new table is group by country, dt, uid. Number of bundles for each uid is also computed. The weight is essentially the sum of entropy across bundles over the total entropy for a uid. New table is fb\_user\_demo\_vector\_yymmdd.

**Step 7.1&7.2:** for each uid, find the age/gender bracket that has the highest weighted frequency. Results are stored in fb\_preds\_by\_bundle\_yymmdd.

**Step 8:** gather data generated from step 7.1 and 7.2. give predictions on age bracket/gender for uid whose weighted entropy exceeds certain threshold.

1. **age\_gender\_bi\_weekly.py**

Route 1(-s y –e y): run all the step 1.\*.\* then exit. Doing this will only general new age-gender stats.

Route 2 (-s y –e n –m n) run 1.1.1 to 3.2, but do not run the last merge step.

Route 3 (-s y –e n –m y) run everything and merge with the previous profile.

**S1.1.1 gen\_bundle\_quality\_summ\_gender\_tbl**

Based on table from **1.c** dq\_bundle\_qlt\_for\_age, select data from “start date” to “end date”, where start date and end date are specified in configuration file. Aggregate over date, to get user count for each of BPO combination. Compute the ratio of each bracket and if there is one bracket that has percentage higher than 85%, then we say it is a bad bundle. Result is stored in table dq\_bundle\_qlt\_summ\_age\_...

**S1.1.2 gen\_bundle\_quality\_summ\_gender\_tbl**

Similar as S1.1.1, compute the ratio of male and female, then check if there is any bundle that has gender ratio higher than 98%.

**S1.2.1 gen\_uid\_age\_counts\_by\_bundle**

Get country, uid, age information from dq\_raw\_uid\_bundle\_age where the bundle is not bad for age according to S1.1.1. Also count the number of bundles, number of days and number of rows for each country, uid, age combination. Results are stored in table uid\_age\_counts\_by\_bundle.

**S1.2.0.0 gen\_by\_bundle\_FB\_like\_age\_distribution**

Generate Facebook-like data based on dq\_raw\_uid\_bundle\_age.

**S1.2.0.1 gen\_by\_bundle\_KLDs**

Compute the KL divergence between facebook data and our data. KL divergence is computed with 6 brackets distribution and 3 brackets distribution.

**S1.2.1 gen\_uid\_age\_counts\_by\_bundle\_use\_kld**

Similar as the first S1.2.1, but this time we consider KL divergence. Apart from bundle that has high percentage from one bracket, we also remove the bundles that has KL-divergence higher than certain thresholds. Results are stored in uid\_age\_counts\_by\_bundle\_kld.

**S1.2.2 gen\_uid\_gender\_counts\_by\_pub**

Similar as S1.2.1 which extract country, uid, gender, number of bundles, number of days and number of rows from dq\_raw\_uid\_... with bad bundles removed.

**S1.3 create\_uid\_age\_gender\_stats**

For each country, uid, get the mean, min, max age. Results are stored in table uid\_age\_stat\_...

**S1.4 gen\_age\_gender\_repeat\_uids**

Get the uids that appears in more than one day in both dq\_raw\_uid\_bundle\_age/gender and save the result in table dq\_age\_gender\_repeat\_uids. Seems that currently this table is not in use.

**P2.1.1 gen\_pre\_prfl\_ffm\_age\_brkt\_stg**

Why join itself? What is the counterpart of cs in method?

Based on table uid\_age\_stat\_... get country, uid, age bracket (4 brackets) such that standard deviation for age <3. Store result as pre\_profile\_ffm\_\_uid\_age\_brkt\_...\_stg

**P2.2 gen\_pre\_prfl\_ffm\_age\_binary**

Based on stage table from P2.1.1 and class\_proba\_ffm\_...(prediction results), extract the binary age. First use stage table to create sc binary age table. Then based on the prediction result, insert rows that we are about >90% sure and does not already exist in the table.

**P2.3 gen\_pre\_prfl\_ffm\_age\_brkt\_preds**

Insert into the bracket stage table from P2.1.1 with prediction values that we are relatively sure of and do not already exist on the table. Then for the last bracket (21-), use the binary age result.

**P2.4 gen\_pre\_prfl\_gender**

First generate gender profile stage table based on uid\_gender\_stat from S1.3. Then insert with prediction values that we are >90% sure of.

**P3.1 compile\_uids**

Generate country and uids for ga\_profile.

**P3.2 gen\_ga\_profile**

Generate ga profile based on stage tables built on P2.2, P2.3, P2.4. By default, it creates a new profile as ga\_profile\_ffm\_YYMMDD.