Capstone Project Daniel Fang

Machine Learning NanoDegree 07/30/2016

Demographic feature prediction for cell phone users

I. Definition

Project Overview

This project is derived from the open Kaggle competition: TalkingData Mobile User Demographics. Its purpose is to build a model predicting users' demographic characteristics based on their app usage, geolocation, and mobile device properties. Doing so will help developers and brand advertiser to know their users better and customize the user experience.

Datasets

This project is about predicting the gender and age of mobile users given their activities and phone information. The dataset is provided by TalkingData, a leading mobile data platform, and can be downloaded here: https://www.kaggle.com/c/talkingdata-mobile-user-demographics/data. The relationship between different data is shown in fig.1.

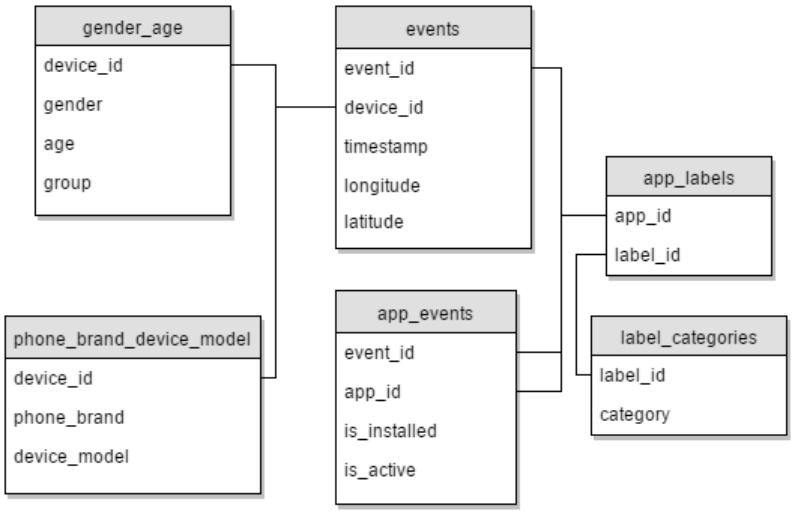

Figure 1. The data schema

Detailed file description:

- **gender_age_train.csv, gender_age_test.csv** - the training and test set

    - group: this is the target variable you are going to predict
- **events.csv, app_events.csv** - when a user uses TalkingData SDK, the event gets logged in this data. Each event has an event id, location (lat/long), and the event corresponds to a list of apps in app_events.
    - timestamp: when the user is using an app with TalkingData SDK
- **app_labels.csv** - apps and their labels, the label_id's can be used to join with label_categories
- **label_categories.csv** - apps' labels and their categories in text
- **phone_brand_device_model.csv** - device ids, brand, and models

There are some problem with the above dataset: After the initial data exploration, it was found that many devices in the gender_age_test.csv don't have any record in events.csv, which means the only information left to predict such devices is just phone brand and phone models. So if a brand or a model is very popular with different groups of people, like Samsung Note, or Huawei Mate7, the best guess given such limited information is simply averaging possibilities of different groups, which is not a machine learning problem.

So I changed the dataset by requesting that we only predict devices with valid an event record. It also has good side-effects: the dataset becomes much smaller, so we can expedite the iteration of feature engineering and model training. Also, we need to set up our own test dataset that can evaluate our model accurately and immediately.

Metrics

The performance of a model is evaluated by multi-class logarithmic loss. Each device has been labeled with one true class. For each device, our learner must produce a set of predicted probabilities (one for each class). The formula is then,

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \log p_{ij}$$

II. Analysis

Data exploration

As I mentioned in Section I, the dataset used in this project has been truncated to make it more learnable. After the processing, the total number of distinct device id is 23309. We also need to put 20% away as test dataset, so the number device id used for training is about 18500.

Table 1 lists the first 10 row of the training data. The column features that can be submitted to training includes 'device_id', 'phone_brand, and 'device_model'. The target column is 'group', which has 12 classifications based on users' gender and age.

Table 1. Gender_age and phone_brand_device_model dataset

device_id	gender	age	group	phone_brand	device_model
-5827952925479472594	M	30	M29-31	小米	MI One Plus
773248989809697210	M	20	M22-	vivo	X5Pro
-5913071468598874323	M	40	M39+	三星	S7566
375031242916141301	M	32	M32-38	索尼	Xperia Z
-1025284795963440768	M	56	M39+	三星	Galaxy Note 2
6666299284550885153	M	24	M23-26	华为	荣耀 7
7102818890309647918	M	34	M32-38	华为	荣耀 6 Plus
-5926490704406813688	M	35	M32-38	小米	红米
8490622669015199443	M	33	M32-38	华为	荣耀 6
4072804797843084687	F	31	F29-32	小米	MI 2S

There are also events data registered for each 'device_id' that can be very useful for our model training. Table 2 lists the first 3 rows of events data. For each record, it has a 'timestamp' and location information ('longitude' and 'latitude'). Furthermore, for some events, we are also offered with detailed App data, as listed in table 3. It represents which app ('app_id') is installed and active during an event for a given device. Note that there are many events that has no record in apps. Also, if an event is corresponding to any apps, it usually has multiple app records. This will become more obvious after we process the data.

Table 2. Events data part 1

event_id	device_id	timestamp	longitude	latitude
1	29182687948017175	5/1/2016 0:55	121.38	31.24
2	-6401643145415154744	5/1/2016 0:54	103.65	30.97
3	-4833982096941402721	5/1/2016 0:08	106.6	29.7

Table 3. Events data part 2

Event_id	app_id	is_installed	is_active
2	5927333115845830913	1	1
2	-5720078949152207372	1	0
2	-1633887856876571208	1	0

There are tons of apps in the market. To make the context more complete, each app is labeled with one or multiple categories, as shown in table 4.

Table 4. App label data

app_id	label_id	category
7324884708820027918	251	Finance
-4494216993218550286	251	Finance
6058196446775239644	406	unknown
6058196446775239644	407	DS_P2P net loan
8694625920731541625	406	unknown

Data processing and visualization

After the basic data description in the above section, it's time dive into and get our hands dirty. First, let's check the age and gender distribution of training data. As shown in fig.1, male users takes up 65.5% of all data, and the female takes the rest. Because of the dominance in number of male users, the top three categories are M32-39, M39+, and M23-26.

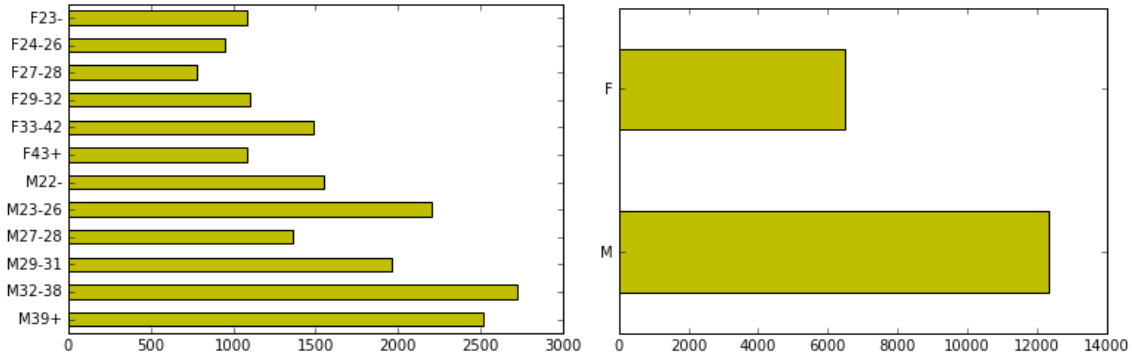

Figure 2. Age and gender distribution of training data

Fig.3 presents the popularity of different phone brands and models. There are in total of 88 brands and 880 models in the dataset. The top 3 brands are Xiaomi(小米), Huawei(华为) and Samsung(三星) respectively. Top 2 models are all manufactured by Xiaomi, and the third one is Galaxy Note 3 manufactured by Samsung.

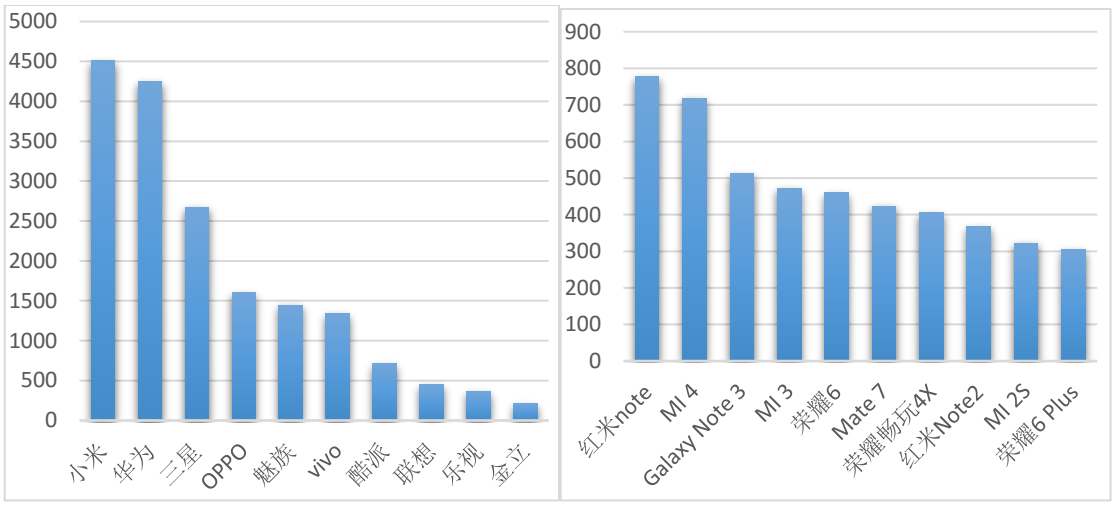

Figure 3. Top 10 Phone brands and device models

Different brands may have different product strategy. The number of models offered by the top 10 brands is plotted in fig.4. From that, we may notice that Samsung and Huawei have much more models than Xiaomi, but still Xiaomi is the most popular brand in terms of number of devices. Actually, most models have less than 5 device in the dataset. For these models, we probably cannot learn valid information from them because of the limited number of data.

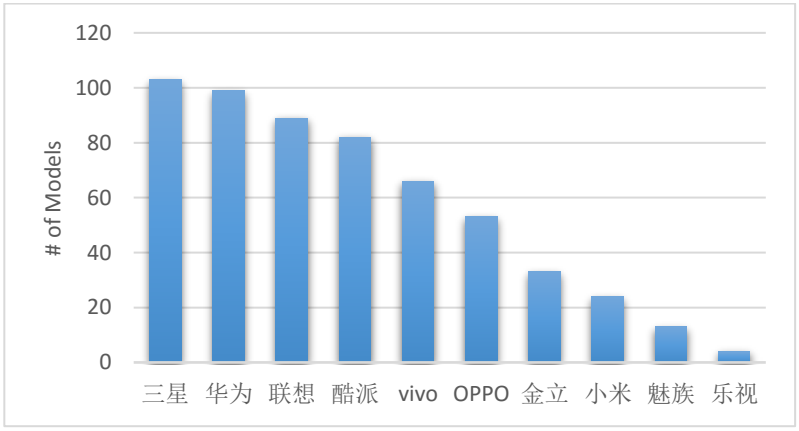

Figure 4 Number of models manufactured by top 10 brands

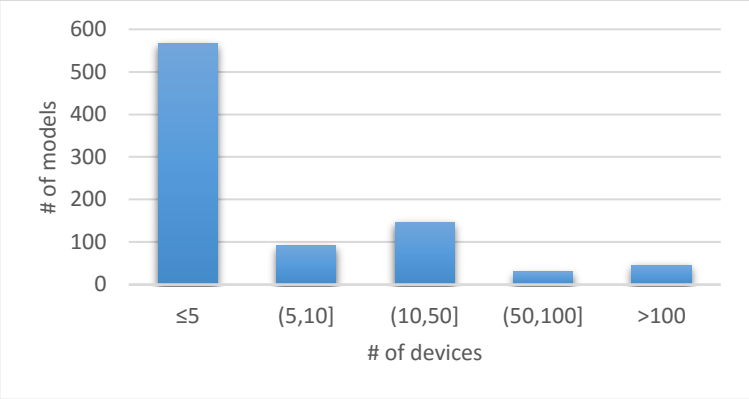

Figure 5. Distribution of models of different number of devices

Benchmark

Before we deploy any machine learning stuff, we need a benchmark to compare across performances. The final submission should give probabilities of 12 classes for each 'device_id' in test dataset. Therefore, the benchmark is set up by pure statistical inferring: using the frequency of classes in the training dataset as the predicting probabilities.

Table 5. Benchmark submission

device_id	F23-	F24-26	F27-28	F29-32	F33-42	F43+
7102818890309647918	0.06819	0.05623	0.04201	0.06198	0.07431	0.056620
	M22-	M23-26	M27-28	M29-31	M32-38	M39+
	0.09969	0.12847	0.07389	0.09720	0.12651	0.114861

The resulted log-loss of the benchmark submission is 2.4235.

Events data preprocessing

In the previous section, table 2-4 shows the related information about events. Tloo facilitate the model training and feature engineering, we need to merge them into one table. As shown in fig.3, for each 'event_id', the installed and active 'app_id' are merged into a list separately. And the corresponding 'label_id' for each 'app_id' is also aggregated. I also applied TFIDF model to create two new features: 'active_tfidf' and 'install_tfidf', which reflects the importance of a label in the whole corpus.

event_id	device_id	timestamp	longitude	latitude	active_label
1	29182687948017175	2016-05-01 00:55:25	121.38	31.24	None
2	-6401643145415154744	2016-05-01 00:54:12	103.65	30.97	549 710 548 704 172 405 730 747 749 776 782 785 ...
3	-4833982096941402721	2016-05-01 00:08:05	106.6	29.7	None

active_app_id	active_tfidf	installed_label	installed_app_id
None	None	None	None
[5927333115845830913, -653184325010919369, ...	{'747': 4.9236469360864525, ...	721 302 303 704 548 251 263 405 730 756 757 775 ...	[-5720078949152207372, -1633887856876571208, ...
None	None	None	None

install_tfidf	phone_brand	device_model
None	小米	红米note
{'704': 0.08500655644858628, ...	三星	Galaxy Grand Prime
None	魅族	MX4 Pro

[3 rows x 13 columns]

Figure 6. Merged events dataset

Until now, the ‘timestamp’ has not been properly processed. In this project, the time data combined with active/installed app data should be useful in finding patterns of certain devices. Therefore, I transformed ‘timestamp’ into 2 additional features: ‘weekday’ and ‘period’. The ‘weekday’ denotes the day of the week, from Monday to Sunday. The ‘period’ refers to which time slot the ‘timestamp’ falls in: {‘midnight’: 00:00 – 06:00, ‘morning’: 06:00-12:00, ‘afternoon’: 12:00-18:00, ‘evening’: 18:00-00:00}. Finally, figure 7 shows what the whole training dataset looks like.

device_id	gender	age	group	phone_brand	device_model	device	event_id	timestamp
29182687948017175	M	46	M39+	小米	红米note	小米红米 note	1	20160501T005525
-4833982096941402721	M	47	M39+	魅族	MX4 Pro	魅族MX4 Pro	3	20160501T000805
1476664663289716375	M	19	M22-	华为	Mate 7	华为Mate 7	6	20160501T002721

longitude	latitude	active_label	active_app_id	active_tfidf
121.38	31.24		None	None
106.6	29.7		None	None
0.0	0.0	549 721 302 303 548 704 183 713 704 548 549 710 ...	[-7377004479023402858, 6284164581582112235, ...	{'709': 3.52229, '303': 1.35591, '704': 1.18421, ...

installed_label	installed_app_id	install_tfidf	weekday	period
	None	None	Sunday	midnight
	None	None	Sunday	midnight
1012 548 549 130 549 713 721 302 303 548 704 211 ...	[6924337203824723352, -2102196853266493861, ...	{'151': 2.17236, '214': 1.67982, '211': 1.93535, ...	Sunday	midnight

[3 rows x 19 columns]

Figure 7. the complete training dataset

III. Experiments

Method – Gradient boosting decision tree

There are both numerical and categorical features in the dataset. This can be perfectly handled by decision trees. There are two popular tree-based techniques: random forest and gradient boosting decision trees (GBDT). Random forest uses many fully grown decision trees. The basic idea is to resample the data over and over and for each sample train a new tree. So different trees overfit the data in different ways, and through bagging, the overfitting effect are averaged out. GBDT also uses a lot of trees, but in a boosting way. That means for each iteration, GBDT trained a shallow tree based on the previous ones to decrease the overall loss function and then add the new tree to the ensemble. Compared to random forest, GBDT generally converges faster, and is robust against overfitting. Thus, GBDT is applied in this project.

Implementation

Before we deploy GBDT to the dataset, a validation data need to be created. A 5-folded validation schema is used in the project. When creating validation set, we can not just randomly slice out some records from training dataset as validation dataset, because there are multiple records (different events_id) corresponding to one unique device_id in the data. A complete segregation of 'device_id' need to be enforced. Otherwise, the validation score is meaningless for model evaluation. Then GBDT is applied on the training dataset with the build-in GBDT function.

Cross validation set and model training codes

```
import time
train = gl.cross_validation.shuffle(ga_train_events, random_seed=time.time())
folds = gl.cross_validation.KFold(ga_train_events, num_folds=5)
fold_train, fold_val = folds[0]
target = "group"
```

```
features = ['longitude', 'latitude', 'active_tfidf', 'install_tfidf','phone_brand',
'device_model']
model = gl.classifier.boosted_trees_classifier.create(fold_train,target=target,
features=features, validation_set=fold_val, max_iterations = 10)
```

First of all, the individual features are tested on their performance. For each test, the GBDT run 50 iterations, and record the log-loss of validation dataset. The training accuracy or training log-loss is always increasing monotonically, while we want to observe when the overfitting come into being.

The results are presented in figure 8, which provides us many information about features. Strong overfitting is observed with the location features and phone brand features. As we discussed, GBDT is very robust to overfitting, then what is the reason for such strong overfitting in very early stage of iteration? The answer is in the dataset. There are multiple events records corresponding to one unique device_id, which share almost the same features of longitude, latitude, phone brand and device model. This makes the learner very easy to get overfitted, since all it needs to do is to simply predict group base on the exact match of its location or phone brand.

For the timestamp feature (weekday and period), small overfitting was observed at the end of the iteration with its best log-loss score around 2.36. Meanwhile, the app feature seems to be the most informative based on its performance: it didn't show large overfitting, and the new tfidf feature bring the log-loss score down to 2.31.

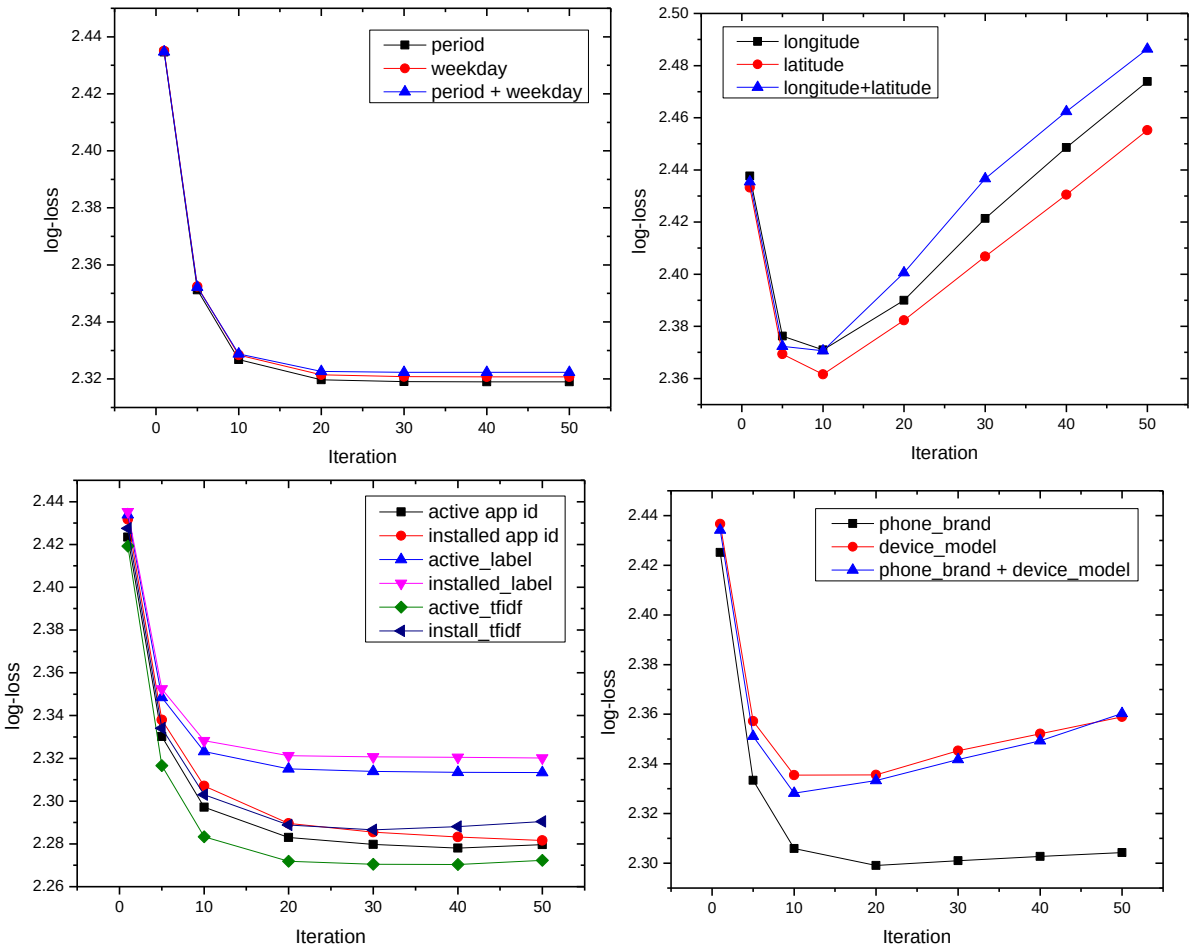

Figure 8. Log-loss score on validation data for models using different features

Results

Based on the above results, the most informative feature is app_label and app_id. But it is really unwise to throw away features of location and phone brand because of the overfitting problem. In the final model, all features are included, but regularization techniques are applied to prevent overfitting while decreasing log-loss further.

`Final model training codes:`

```
target = "group"
features = ['longitude', 'latitude', 'active_tfidf', 'install_tfidf', 'phone_brand',
'device_model', 'weekday','period']
model = gl.classifier.boosted_trees_classifier.create(fold_train,  target=target,
features=features, validation_set=fold_val, max_iterations = 200, max_depth=6,
step_size=0.1,column_subsample = 0.3)
```

In the above codes, the column_subsample refers the subsample ratio of columns in each iteration of tree construction. Setting this to a value to a small value help prevent overfitting. Also, step_size denotes

shrinkage of the prediction of each weak learner to make the boosting process more conservative. The smaller the step size, the more conservative the algorithm will be.

Figure 9 shows the resulted log-loss score of a validation dataset. The sweet point is around 100th iteration, and have a log-loss of 2.275. The log-loss score of other validation set are listed in table 6.

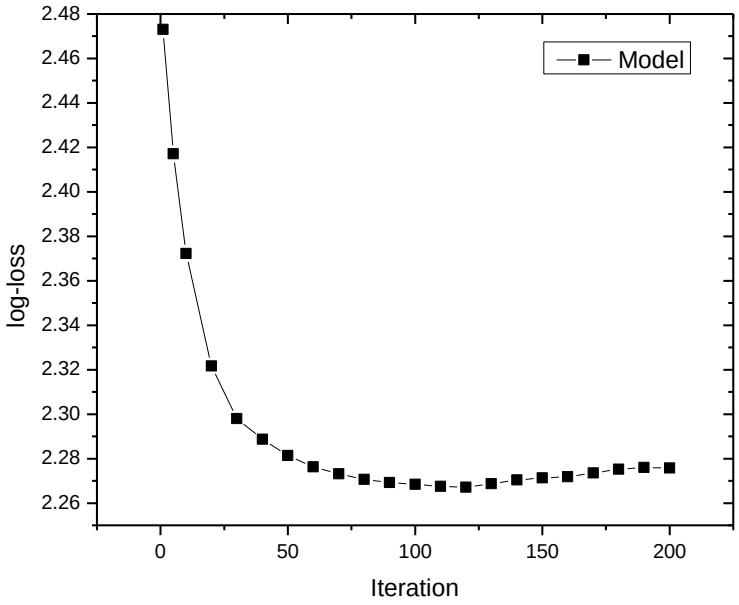

Figure 9. log-loss curve of validation data in the final model

The score on test set obtained by applying the final model is 2.254. Compared to the benchmark, the log-loss score is improved by 7.0%.

Table 6 5-fold validation set score

Validation set	Log-loss
Fold 1	2.275107
Fold 2	2.255033
Fold 3	2.271240
Fold 4	2.266012
Fold 5	2.250840

Conclusions and Reflections

In this project, the demographic feature of cell phone users is predicted based on different features. The importance of different features are investigated by feeding them individually into GBDT algorithms. The final score on test set is 2.254, compared to the benchmark of 2.423. Because there are some overfitting problem for some feature in the dataset, the improvement is not very ideal. Possible improvement can be

made by reorganizing the dataset, and training different learners, or using emsemble/stacking method to fully tap the potential of the data.