from sklearn.cross\_validation import [train\_test\_split](http://scikit-learn.org/dev/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split)

* Random split into test and train set
* Ideally use stratification and train until error on test set is not decreasing. Run training on each fold. Ensemble e.g. 5 folds. Then it’s a 80 train 20 test split.

from sklearn.preprocessing import [LabelEncoder](http://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.LabelEncoder.html#sklearn.preprocessing.LabelEncoder)

turns categorical variables like text into numerical label. Uesful for xgboost that doesn’t handle text categoricals

Le.fit(array of labels)  
le.transform(array of categoricals)  
le.inverse\_transform(array of numericals) -> returns original categories

from sklearn.metrics import [log\_loss](http://scikit-learn.org/dev/modules/generated/sklearn.metrics.log_loss.html#sklearn.metrics.log_loss)

logistic loss or cross-entropy loss: -log P(yt|yp) = -(yt log(yp) + (1 - yt) log(1 - yp))

**log\_loss**(y\_true, y\_pred) returns float

number of events, instead of binary: replace np.ones(d.shape[0])

with:

d['size']

add continuous features like number of apps, but scale to not mess up regularization.

* Features:
* proportion of events where an app appears on each device
* Frequency of app usage

FROM   
<https://www.kaggle.com/c/talkingdata-mobile-user-demographics/forums/t/22410/best-single-model?page=8>

number of layers didn't seem to change my results too much. The levers that had the most impact for me were **number of epochs** (which could rapidly overfit), the **samples per epoch** (which produced some interesting changes**), neurons per layer**, and finally the **choice of random seed**.

 0,0 latitude and longitude coordinate (~60%) (0,0), (1,0), (0,1)... are errors in the data caused by several reasons. You may take them as NaN.  
~60% (63% from memory) of the gender\_age train and test rows don't have events, i.e., don't have a matching device\_id in event. This is part of the reason there are so many 0,0 lat/long

This means that 60% is only being trained on features in phone\_brand\_device\_model.

I have learnt that I can't label encode category, phone brand, and device model since it means I am ranking them (still a noob here).

This is not 100% correct. You can label encode categories as long as you use tree-based methods. For regression based models you are partly right.

|  |
| --- |
| Our approach so far:  1) use brands and models only to predict all samples  2) use predictions from 1 for stacking  3) use all features to predict all samples  4) use predictions from 3 for stacking  5) weight average "no event" samples from 2) and 4)  The final prediction is "has\_event" samples from 4) and "no event" samples from 5). I guess we will try your ideas since we are kind of out of ideas. |
| #39 | Posted 39 days ago  [rcarson's image](https://www.kaggle.com/jiweiliu)  [rcarson](https://www.kaggle.com/jiweiliu)    ANOVA - <https://en.wikipedia.org/wiki/Analysis_of_variance>  @Manel, we did something very similar and stacking is better than weighted average. One stupid mistake I made is that two single models' predictions for test data have different row order so it messed up when submitting. Otherwise the cv of stacking is quite close to lb. Another weird thing is that when I used sklearn randomforest for stacking, the lb is way worse than cv. - rcarson |

[**Creating ensembles from submission files**](http://mlwave.com/kaggle-ensembling-guide/)ensemble uncorrelated models,  
average models

@sugianto, I one-hot encoded everything and ended up with 21035 columns in a sparse matrix. Full train matrix only has about 1 million nonzero values, so it does not take up a lot of memory. I also found that I can usesklearn.feature\_selection.VarianceThreshold to drop about a half of these features without losing much performance.

I weight averaged my best three models(2.250,2.251,2.252) and got 2.248.

|  |
| --- |
| I've a nnet with CV:2.25265 lb:2.24818 |
| #67 | Posted 36 days ago  [Michael Jahrer](https://www.kaggle.com/mjahrer)  I have finished my stack. Now I need more ideas :(.  My 5-CV scores:   * Score with events: 1.928310868 * Score without events: 2.38589622132 * Score: 2.24300845924      |  |  | | --- | --- | | 1. xgboost with gblinear 2. Keras with no hidden layers and softmax output 3. OneVsMany wrapper over Logistic regression [**http://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html**](http://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html)   All of them are slightly different versions of multiclass logistic regression => I would expect ensemble being the best. | | | #75 | Posted 33 days ago  [Vladimir Iglovikov](https://www.kaggle.com/iglovikov) | | | 2 | sklearn [**LogisticRegression**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression) has built-in multiclass handling, so the wrapper is unnecessary, I think. The default multiclass option is "one vs rest" but it also supports "multinomial" mode which is what I used. It gives better results in this competition because it optimizes the metric we're evaluated on =).   |  |  |  | | --- | --- | --- | | **Arcady27 wrote:**  **Chenglong Chen wrote:**  **Michael Jahrer wrote:**  I've a nnet with CV:2.25265 lb:2.24818  Finally got a nnet with 5-fold CV of 2.255217 (std 0.007015) and LB 2.24206 :)  Wow, a single nnet for both with- and without events data? I found out that nnets perform well for subset with events but even an ensemble of 5 nnets resulted only in ~1.97 on samples with events and ~2.2487 overall on LB.  good point. Currently my blend stack is build from 7 models. All nnets. All build on all data. No data split. I should try to split between have/dontHave events data as many people suggest. Best single nnet is now CV2.25038 / LB2.24405. Basically the improvement comes from updating my feature gen code to capture all data fields. It is very interesting that xgboost/gbtree does not work well out of the box with same dataset (at least my params does not work, got stuck at 2.29xx somewhere). It worked so well for all kind of competitions previously.   |  | | --- | | My current nnet (bagged 10 times, same params, different seeds) is on all of the data and gets a 2.23266 on the LB. Might be possible to get below 2.23 by splitting the data. | | #89 | Posted 31 days ago  [Branden Murray](https://www.kaggle.com/brandenkmurray) | | |  | | | It seems as though people are getting good results with neural  nets on all the data without splitting. But if people are interested in splitting  I had good results using this to split the data:  train.loc[train['device\_id'].isin(set(train['device\_id']).intersection(events['device\_id'].unique()))]  I update my 5-cv scores:   * Score with events: 1.90988677946 * Score without events: 2.38774726373 * Score: 2.23852829357 * LB: 2.23138   Best single models for users with devices:   * nnet: 1.94721 (2.25201 for all devices) * xgb: 1.98828 (2.27095 for all devices)   I need to improve the part for users without events but I make overfit when I try to stack (for the moment I continue doing weighted average of three models)  @wilam It is difficult to give any information without revealing too much. But almost all I did is in the forum or public scripts, excepting the way that I make the stack ---which is nothing involved.  I use more than two models to stack (4 different at this moment)... | |

My current nnet (bagged 10 times, same params, different seeds) is on all of the data and gets a 2.23266 on the LB. Might be possible to get below 2.23 by splitting the data.

Have you (or anyone else) tried to evaluate how much gain you get from the bagging effect on a single model ? (drawing full dataset with replacements). Eg. error vs. number of bags. It this similar to train 10x 5CV (+avg at the end) with different random seed in weight init ?

I shouldn't have said bagging here. I didn't do any sampling with replacement (though I've done it a few other times), I just ran 10 times for each fold, averaged those predictions (and kept them for stacking), then ran 10 times on the full dataset, made test predictions, and averaged them. Either way, haven't ever tried evaluating the gain.

|  |
| --- |
| How far can one go with XGBoost (without using NN)?  2.25577 is my best with XGB |
| #132 | Posted 9 days ago  [Permalink](https://www.kaggle.com/c/talkingdata-mobile-user-demographics/forums/t/22410/best-single-model/132396#post132396) | [Quote](https://www.kaggle.com/forums/messages/quote?forumMessageId=132396) | [Flag](https://www.kaggle.com/forums/messages/flag?forumMessageId=132396)  [Branden Murray](https://www.kaggle.com/brandenkmurray) |

It depends on the data, competition and everything.. try it on your own which one scores better on leaderboard. In an early stage I made a submission with a model that scored 2.24905 by training on all data, then I run 5CV and predict the test by average the 5 models from each fold and got 2.24818. Since this was better I use 5CV with no retraining for building up the ensemble.

2.

FROM  
<https://www.kaggle.com/c/talkingdata-mobile-user-demographics/forums/t/22817/help-w-visualizing-features-validity-of-proposed-feature/130974#post130974>

I thought that it would be worthwhile to separate weekday from weekend activity and then compare the distribution of each user's app events throughout the weekday AND to add in data about how much of each particular user's events fall within this time period on *other* weekdays.

A hypothetical example of one user's app\_events could be the following:

user of "Device\_ID A" has 24 app events over multiple days w/this distribution: weekdays: 21 events weekends: 3 events

within weekdays, the app events are distributed as follows: between 1AM - 3AM 12 events on 5 different weekdays between 6AM - 8AM 3 events on the same day between noon - 2PM 5 events on 2 different weekdays between 2PM - 4PM 1 event

My hypothesis is that those calls between 1-3AM on multiple days (for example) may be a predictor of a user's age & gender group.

3.

FROM

<https://www.kaggle.com/dvasyukova/talkingdata-mobile-user-demographics/brand-and-model-based-benchmarks/comments>

Looks like 6 devices with duplicate rows have different values for brand and model. in train: 1

in test: 5

Drop them.

Weighting some rare device models somehow – look up

4.

FROM

<https://www.kaggle.com/laurae2/talkingdata-mobile-user-demographics/a-linear-model-on-apps-and-labels>

5.

FROM

<https://www.kaggle.com/c/talkingdata-mobile-user-demographics/forums/t/23286/you-were-only-supposed-to-blow-the-doors-off?page=2>

Similarly, I got ~2.37 5-fold CV score only using the normalized order of the observations. It did not carry over to the test set though (LB 2.42), though maybe there's a better way to exploit it. (row number)/(total rows)

Since the train and test set are different sizes the order needs to be normalized, otherwise the extra rows in the test set won't fall within the range of the train set.

You should be able to beat the Random Forest benchmark using only the ID with a smart processing of it. Seems a bit non sense; but that's what I looked on the train set during the first days. You can get a 5-fold CV to 2.30-2.37 if I remember the first days I worked on that (depending on the random folds - it varies a lot) just with that, way better than random predictions. I didn't even bother submitting it on the public LB because it was way too strange (I never predicted on the test set anyway). Make sure you don't work straight on the 12-class but separate age+gender (I never managed to have a model "learn" from IDs to predict the 12 classes directly - instead I stacked age+gender using out-of-fold predictions).

note: my ID features were around 200 or so. You can look at the ID and make features, and try to make a model learn from it. If you don't get under ~9 RMSE for age, then you need to make better features as it is the close to the threshold of random predictions.

Extract the numbers and the signs from the ID (ignore NA warnings) using this in R, then work on your features from the variable char\_list (first index is the sign: if it is 1 it is negative, if it is NA then it is positive):

data <- as.data.frame(fread("gender\_age\_train.csv", header = TRUE, sep = ",", colClasses = c("character", "character", "numeric", "character")))  
  
chars <- nchar(data$device\_id)  
char\_list <- data.frame(matrix(nrow = nrow(data), ncol = 20))  
  
temp\_chars <- sprintf("%20s", data$device\_id)  
  
char\_list[grep("-", temp\_chars), 1] <- 1  
  
for (i in 2:20) {  
  char\_list[, i] <- as.numeric(substr(temp\_chars, i, i))  
}

If there are other IDs to exploit, then it might be interesting to look at them. I didn't test those other than in the gender age train set.

Lessons learnt for future competitions:

Separate data-read-in, data munging, models, ensembling, csv-output

* Train-set need to be in a form for ready consumption by multiple algorithms.
* Enable automatic saving of model building stats
* Save every model and its submission for re-use – together with its stats.
* Enable param tuning – must an on/off wrapper for each algorithm