Predict whether a mobile ad will be clicked

In online advertising, click-through rate (CTR) is a very important metric for evaluating ad performance. As a result, click prediction systems are essential and widely used for sponsored search and real-time bidding.



For this competition, we have provided 11 days worth of Avazu data to build and test prediction models. Can you find a strategy that beats standard classification algorithms? The winning models from this competition will be released under an open-source license.

**Started:** 10:15 pm, Friday 24 October 2014 UTC   
**Ends:** 11:59 pm, Monday 26 January 2015 UTC (94 total days)   
**Points:** this competition awards standard [ranking points](https://www.kaggle.com/wiki/UserRankingAndTierSystem)   
**Tiers:** this competition counts towards [tiers](https://www.kaggle.com/wiki/UserRankingAndTierSystem)

# Evaluation

Submissions are evaluated using the [Logarithmic Loss](https://www.kaggle.com/wiki/LogarithmicLoss) (smaller is better).

## Submission Format

The submissions should contain the predicted probability of click for each ad impression in the test set using the following format:

id,click  
60000000,0.384  
63895816,0.5919  
759281658,0.1934  
895936184,0.9572  
...

Logarithmic Loss (LogLoss)

The logarithm of the likelihood function for a Bernoulli random distribution.

In plain English, this error metric is typically used where you have to predict that something is true or false with a probability (likelihood) ranging from definitely true (1) to equally true (0.5) to definitely false(0).

The use of log on the error provides extreme punishments for being both confident and wrong. In the worst possible case, a single prediction that something is definitely true (1) when it is actually false will add infinite to your error score and make every other entry pointless. In Kaggle competitions, predictions are bounded away from the extremes by a small value in order to prevent this.

LogLoss=−1n ∑ i=1 n [y i log(y ^  i )+(1−y i )log(1−y ^  i )]

Matlab code:

epss=0.001; %arbitrary value, may be model tuning parameter

y\_pred=min(max(y\_pred,epss),1-epss);

LogLoss=-mean(y.\*log(y\_pred)+(1-y).\*log(1-y\_pred));

language: ini

Python code:

import scipy as sp

def llfun(act, pred):

epsilon = 1e-15

pred = sp.maximum(epsilon, pred)

pred = sp.minimum(1-epsilon, pred)

ll = sum(act\*sp.log(pred) + sp.subtract(1,act)\*sp.log(sp.subtract(1,pred)))

ll = ll \* -1.0/len(act)

return ll

language: python

R code:

llfun <- function(actual, prediction) {

epsilon <- .000000000000001

yhat <- pmin(pmax(prediction, epsilon), 1-epsilon)

logloss <- -mean(actual\*log(yhat)

+ (1-actual)\*log(1 - yhat))

return(logloss)

}

language: matlab

Data fields

* id: ad identifier
* click: 0/1 for non-click/click
* hour: format is YYMMDDHH, so 14091123 means 23:00 on Sept. 11, 2014. Timezone is UTC.
* C1, C17-C24: anonymized categorical variables
* C2-C16: named variables
  + banner\_pos
  + site\_id
  + site\_domain
  + site\_category
  + app\_id
  + app\_domain
  + app\_category
  + device\_id
  + device\_ip
  + device\_os
  + device\_make
  + device\_model
  + device\_type
  + device\_conn\_type
  + device\_geo\_country

# New dataset released 11/18/2014

Data fields

* id: ad identifier
* click: 0/1 for non-click/click
* hour: format is YYMMDDHH, so 14091123 means 23:00 on Sept. 11, 2014 UTC.
* C1 -- anonymized categorical variable
* banner\_pos
* site\_id
* site\_domain
* site\_category
* app\_id
* app\_domain
* app\_category
* device\_id
* device\_ip
* device\_model
* device\_type
* device\_conn\_type
* C14-C21 -- anonymized categorical variables

# Notes

Training data (6.1 GB csv file)

Start date 10/21/2014 12:00 am (‘14102100’) UTC

End date 10/30/2014 11:00 pm (‘14103023’) UTC

40,428,967 observations, 24 variables

6,865,066 clicked (16.98 %)

Uneven number of samples per a day, per an hour

240 unique hour segments in the data (10 full days)

Average number of samples per hour = 169,018.3

Standard deviation of samples per hour = 74,309.3

Minimum samples per an hour = 14,876

Maximum samples per an hour = 447,783

I need to pull samples from each hour segment



Percentage of observations that clicked grouped by hour of the day (entire dataset)

Okay, hour isn’t going to be a great predictor

Looks like dataset was sampled to produce the same proportions of clicked for every hour



Correlation values for each variable (predictor) to the click variable (response)

100k Sample dataset

Train.csv – 99,949 observations

Val.csv – 50,049 observations

16.9 % clicked for both

500k Sample dataset (4 hours to create)

Train.csv – 500,048 observations

Val.csv – 99,949 observations

16.9 % clicked for both

240 unique values for hour category for both (every hour segment was sampled from)

Probability density functions for the hour for the entire dataset, the training dataset, and the validation dataset are identical!

# Feature engineering