Solution for Avito Context Ad Clicks
https://www.kaggle.com/c/avito-context-ad-clicks
3rd place, Lucas Silva and Dmitry Efimov

1 Approach summary

The purpose of this challenge was to build a model which is able to predict if individual users click a given context ad. Avito has provided eight comprehensive relational datasets to explore. Our solution contains the following steps:

1. Cross validation splitting
2. Basic feature engineering
3. FM [4], [5], [6] and FTRL [3] feature engineering
4. XGBOOST models [1] and ensembling.

2 Cross validation splitting

For each UserID the train dataset has been sorted by SearchDate. We have splitted the train and test datasets in four parts:

- **test**: searches from the test dataset
- **val**: last three searches of UserID during the period of the test dataset (date of search > minimum date of search from the test dataset)
- **tr**: three searches before the last three searches of UserID during the period of the test dataset
- **hist**: all other searches of UserID.

3 Basic feature engineering

The following list of features has been generated:

- **SearchRussian**: 1 - if SearchQuery is in Russian, 0 - otherwise
- **SearchAdCount**: number of ads for the SearchID
- **SearchAdT1Count**, **SearchObjectType1Count**: number of ads with ObjectType = 1 for the SearchID

- **SearchAdT2Count**: number of ads with ObjectType = 2 for the SearchID
- **SearchAdT3Count**, **SearchObjectType3Count**: number of ads with ObjectType = 3 for the SearchID
- **SearchOrdUsrAsc**: order number of SearchID sorted by SearchDate for each UserID
- **SearchOrdUsrDesc**: order number of SearchID sorted by SearchDate in descending order for each UserID
- **UserPrevPhoneRequest**: 1 - if UserID requested the phone number, 0 - otherwise
- **UserPrevVisitReq**: number of visits for each SearchID
- **UserPrevVisitReqUni**: number of unique AdID for each SearchID
- **SearchParamsSZ**: size of SearchParams
- **SearchQuerySZ**: size of SearchQuery
- **CountIPUser**: number of unique UserID for each IPID
- **CountUserSearchLocation**: number of unique LocationID for each UserID
- **CountUserSearchCategory**: number of unique CategoryID for each UserID
- **CountUserSearch**: number of unique SearchID for each UserID
- **CountUserAd**: number of unique AdID for each UserID
- **CountUserAdT1**: number of unique AdID with ObjectType=1 for each UserID
- **CountUserAdT3**: number of unique AdID with ObjectType=3 for each UserID
- **CountUserAdDupT1**: number of duplicated AdID with ObjectType=1 for each UserID

- **CountUserAdDupT3**: number of duplicated AdID with ObjectType=3 for each UserID
- **UserQryTotalTime**: time from the first SearchID to the current SearchID for each UserID
- **UserPrevQryDate**, **SearchIDPreviousAge**: time from the previous SearchID to the current SearchID for each UserID
- **UserPrevPrevQryDate**: time from the one before previous SearchID to the current SearchID for each UserID
- **UserPrevPrevPrevQryDate**: time from the one before previous before previous SearchID to the current SearchID for each UserID
- **CountAdSearch**: number of unique SearchID by each AdID
- **RatioAdPos1**: ratio of ad shows on the first position to the total number of ad shows for each AdID
- **CountAdUsers**: number of unique UserID by each AdID
- **CountAdSearchLoc**: number of unique LocationID by each AdID
- **CountAdSearchCat**: number of unique CategoryID by each AdID
- **RatioSearchRuss**: ratio of ad shows with SearchQuery in Russian to the total number of ad shows for each AdID
- **AdTitleSZ**: size of AdTitle
- **AdPosition1Count**: number of times on the first position for each AdID
- **AdPosition7Count**: number of times on the seventh position for each AdID
- **UserSearchUniqueCount**: number of unique SearchID for each UserID
- **LocationUserUniqueCount**: number of unique UserID for each LocationID

- **CategoryUserUniqueCount**: number of unique UserID for each CategoryID
- **UserAdCount**: number of ad shows for each AdID and UserID
- **AdCategoryPriceDeviation**: (Ad price - Median of prices by ad category) / Median of prices by ad category
- **UserAdViewTotalCount**: number of visits by each UserID
- **UserAdViewUniqueCount**: number of unique visited AdID by each UserID
- **UserAdCategoryPriceMean**: average price of visited AdID by each UserID and AdCategory
- **UserAdCategoryPriceMedian**: median price of visited AdID by each UserID and AdCategory
- **UserAdCategoryPriceMin**: min price of visited AdID by each UserID and AdCategory
- **UserAdCategoryPriceMax**: max price of visited AdID by each UserID and AdCategory
- **UserAdViewTotalCount2**: number of phone requests by each UserID
- **UserAdViewUniqueCount2**: number of unique AdID with phone requested by each UserID
- **UserAdCategoryPriceMean2**: average price of phone requested AdID by each UserID and AdCategory
- **UserAdCategoryPriceMedian2**: median price of phone requested AdID by each UserID and AdCategory
- **UserAdCategoryPriceMin2**: min price of phone requested AdID by each UserID and AdCategory
- **UserAdCategoryPriceMax2**: max price of phone requested AdID by each UserID and AdCategory

- **SearchAdCommonWordCount**: number of common words in SearchQuery and AdTitle

The following notations explain the rest set of features:

- features with suffix **Bin**: discretized features
- features with prefix **Prob** or suffix **likeli**: likelihood features calculated on
  1. **hist** part to get values for the **tr** part
  2. **hist** and **tr** parts to get values for the **val** part
  3. **hist**, **tr** and **val** parts to get values for the **test** part

4 FM and FTRL feature engineering

The several Factorization Machine (FM) [4], [5], [6] and Follow the Proximal Regularized Leader (FTRL) [3] have been trained to generate additional set of features. Each of the following models has been trained on

1. **hist** part to get prediction for the **tr** part
2. **hist** and **tr** parts to get prediction for the **val** part
3. **hist**, **tr** and **val** parts to get prediction for the **test** part

The list of the models are

1. **fm.01**:

   parameters: k = 16, t = 20, r = 0.02, l = 0.00001, s = 6

   feature set: AdCatID AdHistCTRBin AdID AdParams AdPriceBin AdTitleSZBin Position SearchAdCount SearchAdT1Count SearchAdT2Count SearchAdT3Count SearchCatID SearchLocID SearchParamsSZBin SearchQuerySZBin SearchRussian UserID UserIPID UserPrevQryDateBin UserQryTotalTimeBin.

2. **fm.02**:

   parameters: k = 12, t = 5, r = 0.015, l = 0.00001, s = 6

   feature set: AdCatID AdHistCTRBin AdParams AdPriceBin AdTitleSZBin Position SearchAdT1Count.

3. **fm.03**:

   parameters: k = 12, t = 3, r = 0.008, l = 0.00001, s = 12

   feature set: SearchAdCount SearchAdT1Count SearchAdT2Count SearchAdT3Count SearchCatID SearchLocID SearchParamsSZBin SearchQuerySZBin SearchRussian Position.

4. **fm.04**:

   parameters: k = 12, t = 5, r = 0.004, l = 0.00001, s = 12

   feature set: UserAgentFamilyID UserAgentID UserAgentOSID UserDeviceID UserPrevPhoneRequest UserPrevPrevPrevQryDateBin UserPrevPrevQryDateBin UserPrevQryDateBin UserPrevVisitReq UserPrevVisitReqUni UserQryTotalTimeBin Position.

5. **fm.05**:

   parameters: k = 12, t = 5, r = 0.004, l = 0.00001, s = 12

   feature set: CountAdSearchBin CountAdSearchCatBin CountAdSearchLocBin CountAdUsersBin CountIPUserBin CountUserAdBin CountUserAdDupT1Bin CountUserAdDupT3Bin CountUserAdT1Bin CountUserAdT3Bin CountUserSearchBin CountUserSearchCategoryBin CountUserSearchLocationBin RatioAdPos1Bin RatioSearchRussBin Position.

6. **ftrl.04**:

   parameters: α = 0.07, β = 1.0, l1 = 0.01, l2 = 1.0, dropout = 0, epoch = 1

   feature set: AdCatID AdHistCTRBin AdID AdParams AdPriceBin AdTitleSZBin Position SearchAdCount SearchAdT1Count SearchAdT2Count SearchAdT3Count SearchCatID SearchLocID SearchParamsSZBin SearchQuerySZBin SearchRussian UserAgentFamilyID UserAgentID UserAgentOSID UserDeviceID UserID UserIPID UserPrevPhoneRequest UserPrevPrevPrevQryDateBin UserPrevPrevQryDateBin UserPrevQryDateBin UserPrevVisitReq UserPrevVisitReqUni two-way interactions: AdUs UsSearch AdSearch AdPos UsPos PosSearch

7. **ftrl.05**:

   parameters: $\alpha = 0.008$, $\beta = 0.1$, l1 = 0.1, l2 = 0.15, dropout = 0, epoch = 2

   feature set: AdCatID AdHistCTRBin AdID AdParams AdPriceBin AdTitleSZBin CountAdSearchBin CountAdSearchCatBin CountAdSearchLocBin CountAdUsersBin CountIPUserBin CountUserAdBin CountUserAdDupT1Bin CountUserAdDupT3Bin CountUserAdT1Bin CountUserAdT3Bin CountUserSearchBin CountUserSearchCategoryBin CountUserSearchLocationBin Position RatioAdPos1Bin RatioSearchRussBin SearchAdCount SearchAdT1Count SearchAdT2Count SearchAdT3Count SearchCatID SearchLocID SearchOrdUsrAsc SearchOrdUsrDesc SearchParamsSZBin SearchQuerySZBin SearchRussian UserAgentFamilyID UserAgentID UserAgentOSID UserDeviceID UserID UserIPID UserLogged UserPrevPhoneRequest UserPrevPrevPrevQryDateBin UserPrevPrevQryDateBin UserPrevQryDateBin UserPrevVisitReq UserPrevVisitReqUni UserQryTotalTimeBin

   two-way interactions: AdUs UsSearch AdSearch AdPos UsPos PosSearch

8. **ftrl.06**:

   parameters: $\alpha = 0.07$, $\beta = 1.0$, l1 = 0.01, l2 = 1.0, dropout = 0, epoch = 2

   feature set: AdCatID AdHistCTRBin AdID AdParams AdPriceBin AdTitleSZBin CountAdSearchBin CountAdSearchCatBin CountAdSearchLocBin CountAdUsersBin CountIPUserBin CountUserAdBin CountUserAdDupT1Bin CountUserAdDupT3Bin CountUserAdT1Bin CountUserAdT3Bin CountUserSearchBin CountUserSearchCategoryBin CountUserSearchLocationBin Position RatioAdPos1Bin RatioSearchRussBin SearchAdCount SearchAdT1Count SearchAdT2Count SearchAdT3Count SearchCatID SearchLocID SearchOrdUsrAsc SearchOrdUsrDesc SearchParamsSZBin SearchQuerySZBin SearchRussian UserAgentFamilyID UserAgentID UserAgentOSID UserDeviceID UserID UserIPID UserLogged UserPrevPhoneRequest UserPrevPrevPrevQryDateBin UserPrevPrevQryDateBin UserPrevQry-

DateBin UserPrevVisitReq UserPrevVisitReqUni UserQryTotalTimeBin

two-way interactions: AdIDSearchCatID AdIDUserID AdCatIDSearchCatID AdIDSearchLocID SearchCatIDUserID AdCatIDUserID SearchLocIDUserID AdIDPos AdCatIDPos SearchCatIDPos SearchLocIDPos UserIDPos SearchRussianPos SearchAdT1AdID SearchAdT1AdCatID SearchAdT1Pos AdIDUserAgentOSID AdIDUserAgentFamilyID AdCatIDAdPriceBin AdPriceBinUserID

5 XGBOOST models and ensembling

We have trained 4 different XGBOOST models [1] with different sets of features and different sets of parameters. All XGBOOST models have been trained on **tr** and **val** parts only.

1. **l1.xgb.03**:

   parameters: objective = “binary:logistic”, eval_metric = “logloss”, eta = 0.2, max_depth = 10, gamma = 0.8, colsample_bytree = 0.7, colsample_bylevel = 0.8, nrounds = 75, epoch = 20

   feature set: AdCatID, AdHistCTR, AdID, AdParams, AdPrice, AdTitleSZ, CountAdSearch, CountAdSearchCat, CountAdSearchLoc, CountAdUsers, CountIPUser, CountUserAd, CountUserAdDupT1, CountUserAdDupT3, CountUserAdT1, CountUserAdT3, CountUserSearch, CountUserSearchCategory, CountUserSearchLocation, Position, RatioAdPos1, RatioSearchRuss, SearchAdCount, SearchAdT1Count, SearchAdT2Count, SearchAdT3Count, SearchCatID, SearchDate, SearchLocID, SearchOrdUsrAsc, SearchOrdUsrDesc, SearchParamsSZ, SearchQuerySZ, SearchRussian, UserAgentFamilyID, UserAgentID, UserAgentOSID, UserDeviceID, UserID, UserIPID, UserLogged, UserPrevPhoneRequest, UserPrevPrevPrevQryDate, UserPrevPrevQryDate, UserPrevQryDate, UserPrevVisitReq, UserPrevVisitReqUni, UserQryTotalTime, ProbAdID, ProbAdCatID, ProbAdParams, ProbUserID, ProbUserIPID, ProbUserAgentID, ProbUserAgentOSID, ProbUserDeviceID, ProbUserAgentFamilyID, ProbSearchLocID, ProbSearchCatID, ProbAdCatIDUserAgentFamilyID, ProbAdIDUserAgentFamilyID, ProbAdCatIDUserAgentOSID, ProbAdIDUserAgentOSID, ProbAdCatIDUserID, ProbAdIDUserID, ProbAdCatIDUserIPID, ProbAdIDUserIPID, ProbAdCatIDSearchCatID, ProbAdIDSearchCatID, ProbAdCatIDSearchLocID, ProbAdIDSearchLocID, ProbSearchCatIDUserAgentFamilyID, ProbSearchLocIDUserAgentFamilyID, ProbSearchCatIDUserAgentOSID, ProbSearchLocIDUserAgentOSID, ProbSearchCatIDUserID, ProbSearchLocIDUserID, ProbSearchCatIDUserIPID, ProbSearchLocIDUserIPID, ProbSearchLocIDSearchCatID

2. **l1.xgb.05**:

   parameters: objective = “binary:logistic”, eval_metric = “logloss”, eta = 0.18, max_depth = 10, gamma = 0.8, colsample_bytree = 0.7, colsample_bylevel = 0.8, min_child_weight = 5, nrounds = 75, epoch = 10

   feature set: AdCatID, AdHistCTR, AdID, AdParams, AdPrice, AdTitleSZ, CountAdSearch, CountAdSearchCat, CountAdSearchLoc, CountAdUsers, CountIPUser, CountUserAd, CountUserAdDupT1, CountUserAdDupT3, CountUserAdT1, CountUserAdT3, CountUserSearch, CountUserSearchCategory, CountUserSearchLocation, Position, RatioAdPos1, RatioSearchRuss, SearchAdCount, SearchAdT1Count, SearchAdT2Count, SearchAdT3Count, SearchCatID, SearchDate, SearchLocID, SearchOrdUsrAsc, SearchOrdUsrDesc, SearchParamsSZ, SearchQuerySZ, SearchRussian, UserAgentFamilyID, UserAgentID, UserAgentOSID, UserDeviceID, UserID, UserIPID, UserLogged, UserPrevPhoneRequest, UserPrevPrevPrevQryDate, UserPrevPrevQryDate, UserPrevQryDate, UserPrevVisitReq, UserPrevVisitReqUni, UserQryTotalTime, ProbAdID, ProbAdCatID, ProbAdParams, ProbUserID, ProbUserIPID, ProbUserAgentID, ProbUserAgentOSID, ProbUserDeviceID, ProbUserAgentFamilyID, ProbSearchLocID, ProbSearchCatID, ProbAdCatIDUserAgentFamilyID, ProbAdIDUserAgentFamilyID, ProbAdCatIDUserAgentOSID, ProbAdIDUserAgentOSID, ProbAdCatIDUserID, ProbAdIDUserID, ProbAdCatIDUserIPID, ProbAdIDUserIPID, ProbAdCatIDSearchCatID, ProbAdIDSearchCatID, ProbAdCatIDSearchLocID, ProbAdIDSearchLocID, ProbSearchCatIDUserAgentFamilyID, ProbSearchLocIDUserAgentFamilyID, ProbSearchCatIDUserAgentOSID, ProbSearchLocIDUserAgentOSID, ProbSearchCatIDUserID, ProbSearchLocIDUserID, ProbSearchCatIDUserIPID, ProbSearchLocIDUserIPID, SearchDayYear, Search-

Position2Count, SearchPosition6Count, SearchPosition7Count, AdPosition1Count, AdPosition7Count, SearchParamsCount, LocationUserUniqueCount, CategoryUserUniqueCount, SearchIDPreviousAge, AdParamsSize, AdParamsCount, UserAdCount, AdCategoryPriceDeviation, UserAdViewTotalCount, UserAdViewUniqueCount, UserAdCategoryPriceMean, UserAdCategoryPriceMedian, UserAdCategoryPriceMin, UserAdCategoryPriceMax, UserAdViewTotalCount2, UserAdViewUniqueCount2, UserAdCategoryPriceMean2, UserAdCategoryPriceMedian2, UserAdCategoryPriceMin2, UserAdCategoryPriceMax2

3. **l2.xgb.02**:

   parameters: objective = "binary:logistic", eval_metric = "logloss", eta = 0.18, max_depth = 10, gamma = 0.8, colsample_bytree = 0.7, colsample_bylevel = 0.8, nrounds = 75, epoch = 20

   feature set: AdCatID, AdHistCTR, AdID, AdParams, AdPrice, AdTitleSZ, CountAdSearch, CountAdSearchCat, CountAdSearchLoc, CountAdUsers, CountIPUser, CountUserAd, CountUserAdDupT1, CountUserAdDupT3, CountUserAdT1, CountUserAdT3, CountUserSearch, CountUserSearchCategory, CountUserSearchLocation, Position, RatioAdPos1, RatioSearchRuss, SearchAdCount, SearchAdT1Count, SearchAdT2Count, SearchAdT3Count, SearchCatID, SearchDate, SearchLocID, SearchOrdUsrAsc, SearchOrdUsrDesc, SearchParamsSZ, SearchQuerySZ, SearchRussian, UserAgentFamilyID, UserAgentID, UserAgentOSID, UserDeviceID, UserID, UserIPID, UserLogged, UserPrevPhoneRequest, UserPrevPrevPrevQryDate, UserPrevPrevQryDate, UserPrevQryDate, UserPrevVisitReq, UserPrevVisitReqUni, UserQryTotalTime, ProbAdID, ProbAdCatID, ProbAdParams, ProbUserID, ProbUserIPID, ProbUserAgentID, ProbUserAgentOSID, ProbUserDeviceID, ProbUserAgentFamilyID, ProbSearchLocID, ProbSearchCatID, ProbAdCatIDUserAgentFamilyID, ProbAdIDUserAgentFamilyID, ProbAdCatIDUserAgentOSID, ProbAdIDUserAgentOSID, ProbAdCatIDUserID, ProbAdIDUserID, ProbAdCatIDUserIPID, ProbAdIDUserIPID, ProbAdCatIDSearchCatID, ProbAdIDSearchCatID, ProbAdCatIDSearchLocID, ProbAdIDSearchLocID, ProbSearchCatIDUserAgentFamilyID, ProbSearchLocIDUserAgentFamilyID, ProbSearchCatIDUserAgentOSID, ProbSearchLocIDUserAgentOSID, ProbSearchCatIDUserID, ProbSearchLocIDUserID, ProbSearchCatIDUserIPID, ProbSearchLocIDUserIPID, SearchDayYear, Search-

Position2Count, SearchPosition6Count, SearchPosition7Count, AdPosition1Count, AdPosition7Count, SearchParamsCount, LocationUserUniqueCount, CategoryUserUniqueCount, SearchIDPreviousAge, AdParamsSize, AdParamsCount, UserAdCount, AdCategoryPriceDeviation, UserAdViewTotalCount, UserAdViewUniqueCount, UserAdCategoryPriceMean, UserAdCategoryPriceMedian, UserAdCategoryPriceMin, UserAdCategoryPriceMax, UserAdViewTotalCount2, UserAdViewUniqueCount2, UserAdCategoryPriceMean2, UserAdCategoryPriceMedian2, UserAdCategoryPriceMin2, UserAdCategoryPriceMax2, FM and FTRL features

4. **xgb.dtry**:

   parameters: objective = “binary:logistic”, eval_metric = “logloss”, eta = 0.2, max_depth = 10, gamma = 0.8, colsample_bytree = 0.7, colsample_bylevel = 0.8, min_child_weight = 4, nrounds = 75, epoch = 15

   feature set: Position, HistCTR, AdIDlikeli, UserIDlikeli, SearchLocationIDlikeli, SearchCategoryIDlikeli, AdCategoryIDlikeli, SearchObjectType3Count, SearchObjectType1Count, SearchPosition2Count, SearchPosition6Count, SearchPosition7Count, AdPosition1Count, AdPosition7Count, IsUserLoggedOn, SearchQuerySize, SearchRussian, SearchParamsSize, SearchParamsCount, UserSearchUniqueCount, LocationUserUniqueCount, CategoryUserUniqueCount, SearchIDPreviousAge, Price, AdParamsSize, AdParamsCount, AdTitleSize, UserAdCount, AdCategoryPriceDeviation, UserAgentIDlikeli, UserAgentOSIDlikeli, UserDeviceIDlikeli, UserAgentFamilyIDlikeli, UserAdViewTotalCount, UserAdViewUniqueCount, UserAdCategoryPriceMean, UserAdCategoryPriceMedian, UserAdCategoryPriceMin, UserAdCategoryPriceMax, UserAdViewTotalCount2, UserAdViewUniqueCount2, UserAdCategoryPriceMean2, UserAdCategoryPriceMedian2, UserAdCategoryPriceMin2, UserAdCategoryPriceMax2, UserIDAdIDlikeli, UserIDSearchLocationIDlikeli, UserIDSearchCategoryIDlikeli, UserIDAdCategoryIDlikeli, UserIDIPIDlikeli, UserIDUserAgentOSIDlikeli, UserIDUserAgentFamilyIDlikeli, AdIDSearchLocationIDlikeli, AdIDSearchCategoryIDlikeli, AdIDAdCategoryIDlikeli, AdIDIPIDlikeli, AdIDUserAgentOSIDlikeli, AdIDUserAgentFamilyIDlikeli, SearchLocationIDSearchCategoryIDlikeli, SearchLocationIDAdCategoryIDlikeli, SearchLocationIDIPIDlikeli, SearchLocationIDUserAgentOSIDlikeli, SearchLocationIDUserAgentFamilyIDlikeli, SearchCategoryIDAdCategoryIDlikeli, SearchCategoryIDIPIDlikeli, SearchCategoryIDUserAgentOSIDlikeli, SearchCategoryIDUserAgentFamilyIDlikeli, AdCategoryIDIPIDlikeli, AdCategoryIDUserAgentOSIDlikeli, AdCategoryIDUserAgentFamilyIDlikeli, IPIDUserAgentOSIDlikeli, IPIDUserAgentFamilyIDlikeli, UserAgentOSIDUserAgentFamilyIDlikeli, SearchAdCommonWordCount

The final ensemble was a geometric linear combination:

$$1.1 \cdot \mathbf{l1.xgb.03}^{0.216} \cdot \mathbf{l1.xgb.05}^{0.1} \cdot \mathbf{l2.xgb.02}^{0.54} \cdot \mathbf{xgb.dtry}^{0.144}$$

The private and public scores are summarized in the following table:

Model name	**Public score**	**Private score**
l1.xgb.03	0.04086	0.04111
l1.xgb.05	0.04076	0.04104
l2.xgb.02	0.04043	0.04069
xgb.dtry	0.04096	0.04122
final ens	0.04021	0.04046

6 File list and training

Task	File name	Description
Combine all files	main.R	**Output:** csv file with predictions in the data\submission folder
Build datasets	data.build.R data.build.tree.R data.build.dtry.R data.combine.R	**Output:** different data tables in the folder data\output-r
Models for feature engineering	train.l1.fm.01.R train.l1.fm.02.R train.l1.fm.03.R train.l1.fm.04.R train.l1.fm.05.R train.l1.ftrl.04.R train.l1.ftrl.05.R train.l1.ftrl.06.R	**Output:** additional set of features
XGBOOST models	train.l1.xgb.03.R train.l1.xgb.05.R train.l2.xgb.02.R train.xgb.dtry.R	**Output:** predictions in the folder data\output-py
Final ensembling	train.zens.R	**Output:** predictions in the folder data\submission
Helper functions	_fn.base.R _utils.R	

To calculate the final ensembling:

1. Put all data files (extracted) into the “data\input” folder.
2. Open R session with folder “avito-context-click-r” set as working dir.
3. Run the R script main.R in the folder “avito-context-click-r”.
4. The predictions will be saved in the data\submission folder.

Pre-requisites:

All files consider the folder “avito-context-click-r”’ as the working dir. Using RStudio and loading the project inside will do it.

The model can be run on Linux Ubuntu 12.04 or Mac OS.

Dependencies:

Python: pandas 0.12.0, numpy 1.8.0, scikit-learn 0.16.1 [2], xgboost, argparse, warnings, os, gzip, csv, collections, diatomite, sys, math, random, pickle, inspect, gc, pylearn2, timeit, ml_metrics, ast, itertools, six, getopt.

R: doSNOW, foreach, cvTools, data.table, compiler, ffbase, SOAR, SparseM, Matrix, matrixStats, Rcpp, xgboost, infotheo, tm, parallel, rlecuyer.

All the listed versions are the used ones. It will probably work with newer versions, but it was not tested. The R version used was 3.1.0, Python version used 2.7.3.

References

[1] https://github.com/dmlc/xgboost

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay *Scikit-learn: Machine Learning in Python.* 1991, Journal of Machine Learning Research, 12, pp. 2825-2830.

[3] H.Brendan McMahan et al. ”Ad click prediction: a view from the trenches.” *In KDD*, Chicago, Illinois, USA, August 2013.

[4] Michael Jahrer et al. ”Ensemble of collaborative filtering and feature engineered models for click through rate prediction.” *In KDD Cup*, 2012

[5] Steffen Rendle. ”Social network and click-through prediction with factorization machines.” *In KDD Cup*, 2012

[6] Wei-Sheng Chin et al. ”A learning-rate schedule for stochastic gradient methods to matrix factorization.” *In PAKDD*, 2015.