News Article Recommender

Feature Engineering

ITEM_ID: As there are more than 20,000 unique item ids, the percentage of ads clicked to the total ads visited are considered.

UUID: As the unique user ids are more , the percentage of ads clicked to the total ads visited are considered.

SITE_ID:The site id i.e browser which the user is using is considered.

AD_UNIT_ID: The ad_unit_id which are present in the impression attributes is considered

URL: The url in which the ad is present is considered as feature

Count: From the UVH data present in impression attributes, the no of ads clicked by the user till then is taken into account.

User to User : Based on the similars user who visited the same item ids, the chance of user visiting other item ids are calculated and taken as a feature.

Approach

The main idea of my solution is stacking. Stacking helps you to combine different machine learning methods predictions as "metafeatures".

To obtain metafeature for train, you split your data into K folds, training K models on K-1 parts while making prediction for 1 part that was left aside for each K-1 group. To obtain metafeature for test, we can make a

single prediction based on all train data. After that you train metaclassifier on features & metafeatures.

To stack the data I used a python library called vecstack (although could be done without it).

My final solution is a two level stacking.

- In the first level **ExtraTreeClassifier** (default), **RandomForest**(default),**XGB** native model with tuned parameters) and a 2 hidden network **Neural network** models are chosen as meta features (since they gave the most different accurate predictions)
- In the second level the raw features and the 4 four metafeatures are trained on the same **XGB,Neural network** and a Randomised **ExtraTreeClassifer** and these three models predictions are bagged using the weight as **0.9* [XGBOOST^0.7) *NN^0.3] + 0.1 * [ET]** (the weights are tuned)