Strategy:

Our solution heavily depends on negative down-sampling [1, 2], which means we use all positive examples (i.e., is_attributed == 1) and down-sampled negative examples on model training. We down-sampled negative examples such that their size becomes equal to the number of positive ones. It discards about 99.8% of negative examples, but we didn't see much performance deterioration when we tested with our initial features. Moreover, we could get better performance when creating a submission by bagging five predictors trained on five sampled datasets created from different random seeds. This technique allowed us to use hundreds of features while keeping LGB training time less than 30 minutes.

Features:

First, we started from features from Kernels. On Thanks pranav84 and other Kagglers for sharing their awesome insights! We did feature engineerings using all the data examples instead of the down-sampled ones.

- five raw categorical features (ip, os, app, channel, device)
- time categorical features (day, hour)
- some count features

Then, we created a bunch of features in a brute-force way. For each combination of five raw categorical features (ip, os, app, channel, and device), we created the following click series-based feature sets (i.e., each feature set consists of 31 (=(2^5) - 1) features):

- click count within next one/six hours
- forward/backward click time delta
- average attributed ratio of past clicks

We didn't do feature selection. We just added all of them to our model. At that point, our LGB model's score was 0.9808.

Next, we tried categorical feature embedding by using LDA/NMF/LSA. Here is the pseudo code