# In-class Kaggle Competition

Team: change
Members: Ping-Hsun Lee (pl2775) and Shi Jie Koh (sk4812)

## The method you used to produce the final submission

### Feature Engineering

- We converted the original "hour" column into two separate columns - "hour" which takes values 0 to 23 and represents the hour of day, and "day of week" which takes on two values only. We noted that the month and year is irrelevant here since the whole training and testing data only contained data over two days.
- We removed rare features, defined as those categories of a feature that appear only once. We group them together as one "Unknown" category.
- Following the original winning team's proposal, we defined a new predictor "user" based on characteristics of the device of an observation.
- We introduced four new columns which are counts of a certain level of a feature appearing - "device_ip_count", "device_id_count", "hour_count" and "hourly_user_count".
- Lastly, we created a "click_history" predictor, which tracks the click history made by the same user.

### Data Preprocessing

- We performed the same feature engineering on the test set.
- To evaluate hyperparameters and to choose models, we created a validation set using the training set data. Due to the time nature of the dataset, we split the dataset based on time as a random split will not capture this order. Observations after 4pm will be grouped into the validation set.
- To make the validation set better resemble the unseen test set, we modified the "click_history" feature. Any clicks made by the user in the time corresponding to the validation set will not be recorded as the same user's click history since we will not know the target in the unseen test set.
- We also update the counts for the training set to reflect only information that is present there.
- We used hashing to convert features of string data type into integers so that they can be processed by our boosted tree model - xgboost and lightgbm.
- We also prepared a separate dataset for catboost retaining all categorical features and changing the relevant predictors' data type to categorical.

### Modeling

- We decided early on two approaches - deep learning and boosted trees. The deep learning approach is explained more in the other methods we tried.
- For boosting, we decided to use xgboost for its excellent performance on Kaggle competitions, lightgbm due to its massive computational advantage (especially with such

a large dataset), and catboost which will handle the preprocessing of our categorical features very well.

Bagging Ensemble

- To improve the performance of just a single model, we bag the above models. We obtain the optimal hyperparameters using BayesSearchCV by tuning it on the training set and repeat this process for a collection of boosted trees.
- We tried several different approaches to aggregate the bagged models - using simple average and weighted averages obtained using logistic regression and neural networks.

Other methods you tried (which may or may not work well)

- We experimented with other forms of encoding the categorical predictors, such as K-fold target encoding and sklearn's FeatureHasher. Both did not give good results compared to Python's inbuilt hash() function.
- We spent significant time on deep learning models. We tried to build a MLP from scratch using keras and experimenting with different architectures of neural networks by varying the number of hidden layers, the number of neurons per layer, the dropout rate and the activation function. We also attempted to build an embedding layer for our categorical features which proved difficult to complete with our limited knowledge.
- We tried existing neural networks libraries (deepctr and deeptables) that were well suited to predict clickthrough rate problems. We experimented with DeepFM and DCN but we could not successfully define a suitable architecture that produces good results due to our lack of understanding.

Resources consulted:

- https://arxiv.org/pdf/2009.05794
- https://www.csie.ntu.edu.tw/~r01922136/slides/kaggle-avazu.pdf
- https://www.alpha-quantum.com/blog/ctr-prediction/ctr-prediction-using-hashing-trick-logistic-regression-sgd-and-only-simple-python/
- https://towardsdatascience.com/mobile-ads-click-through-rate-ctr-prediction-44fdac40c6ff