Kaggle: CTR prediction.

Постановка задачи.

Есть датасет кликов пользователей на рекламу, названия колонок-признаков скрыты. Нужно предсказать вероятность клика пользователя на представленную рекламу.

Описание датасета.

В тренировочном датасете представлен трафик Criteo за 7 дней. В каждой строке представлена реклама и пользователь. В первом столбце указано, произошел клик (1) или нет (0). В остальных столбцах указаны признаки без указания их семантического значения, некоторые данные могут быть пропущены. Всего есть 13 целочисленных признаков и 26 категориальных. Строки упорядочены в хронологическом порядке.

В тестовом датасете представлены данные по рекламам на следующий день после временных рамок тренировочного датасета. Колонка с меткой (событие клика) была удалена.

Решение.

В первую очередь было опробовано решение с логистической регрессией на первых двух признаках one-hot encoding, без регуляризации (regParam = 0, elasticNetParam = 0). Logreg, one-hot, 1+2 fts: *0.69811*.

Затем я просмотрела соотношение данных в датасете и заметила, что реклам с отсутствием клика в данных примерно в три раза больше, чем с кликом. И хотя в большинстве случаев в реальности клика, действительно, не случается, но я подумала, что это может давать слишком сильный баес модели и давать меньший вес данным с кликом. Поэтому я попыталась немного уменьшить разрыв в данных, просемплировав тренировочный датасет, взяв из него 70% изначальных нулей и все единицы.
Logreg, one-hot, balanced, 1+2 fts: *0.69867*.

Дальше я обратила внимание на сами признаки (категориальные). Их 26 штук, пока же использовались только два первых. Попытки запустить на трех первых признаках привели к тому, что модель очень долго обучалась, и я просто не дожидалась конца обучения. Поэтому первой мыслью было, модель получается слишком сложной, и мне не хватает мощности машины. Отсюда появилась идея выбрать “лучшие” признаки. Для этого модель была обучена 26 раз, всегда с одним категориальным признаком + всеми целочисленными. Результаты получились следующие:

k0: 0.685
k1: 0.7025
k2: что-то плохое
k3: что-то плохое
k4: 0.685
k5: 0.686
k6: 0.72
k7: 0.685
k8: 0.687
k9: что-то плохое
k10: 0.718
k11: что-то плохое
k12: 0.716
k13: 0.697
k14: 0.727
k15: что-то плохое
k16: 0.698
k17: 0.716
k19: 0.686
k20: что-то плохое
k21: 0.685
k22: 0.695
k23: что-то плохое
k24: 0.687
k25: что-то плохое
k18: 0.687

(Что-то плохое означает, что модель очень долго не завершала процесс обучения, так что эти признаки не использовались далее при тренировке)

По полученной информации были выбраны два и пять потенциально “лучших” признаков, на которых обучили модели.
Logreg, one-hot, balanced, 6 + 14 fts: *0.74846*
Logreg, one-hot, balanced, 6+ 10 + 12 + 14 + 17 fts: *0.74866*

Подбор гиперпараметров показал, что regParam = 0, elasticNetParam = 0 дают лучшие результаты, так что с этой стороны улучшений не было.

Далее модель Logistic Regression была заменена на XGBoost. Все дальнейшие результаты были получены после подбора гиперпараметров. Тестирование признаков на данной модели дало следующий результат:

k0: 0.724523
k1: 0.73465
k2: что-то плохое
k3: что-то плохое
k4: 0.72465
k5: 0.72606
k6: 0.73192
k7: 0.72452
k8: 0.72491
k9: что-то плохое
k10: 0.73157
k11: что-то плохое
k12: 0.73369
k13: 0.73430
k14: 0.73452
k15: что-то плохое
k16: 0.72749
k17: 0.73454
k19: 0.72738
k20: что-то плохое
k21: 0.72497
k22: 0.73015
k23: что-то плохое
k24: 0.72743
k25: что-то плохое
k18: 0.72664

XGB, one-hot, balanced, 1+2 fts: *0.72771.*
XGB, one-hot, balanced, 1 + 14 fts: *0.73611*

И последним шагом была замена one-hot encoding на mean-target encoding со сглаживанием. Как и ожидалось, это дало неплохой прирост.
XGB, mean-target, balanced, 14 + 17 fts: *0.75234*
XGB, mean-target, balanced, 6 + 10 + 12 + 14 + 17 fts: *0.76429*

Неиспользованные идеи.

- FFM и глубокое обучение: я нашла некоторое количество статей с описанием применения данных алгоритмов конкретно к задаче CTR, но реализовывать не решилась.
- Была попытка сделать что-то вроде эмбеддинга. Вектор строится только из целочисленных фич, для новых записей клик проверяется чем-то вроде knn, метрика расстояния --- cosine similarity. Но ничего хорошего не вышло уже на train/validate, так что засылать на стала.