CTR prediction

План

Изначально планировал получить три решения:

1. xgb на всем датасете с hyperopt
2. Модель из DeepCTR
3. Модель из xlearn

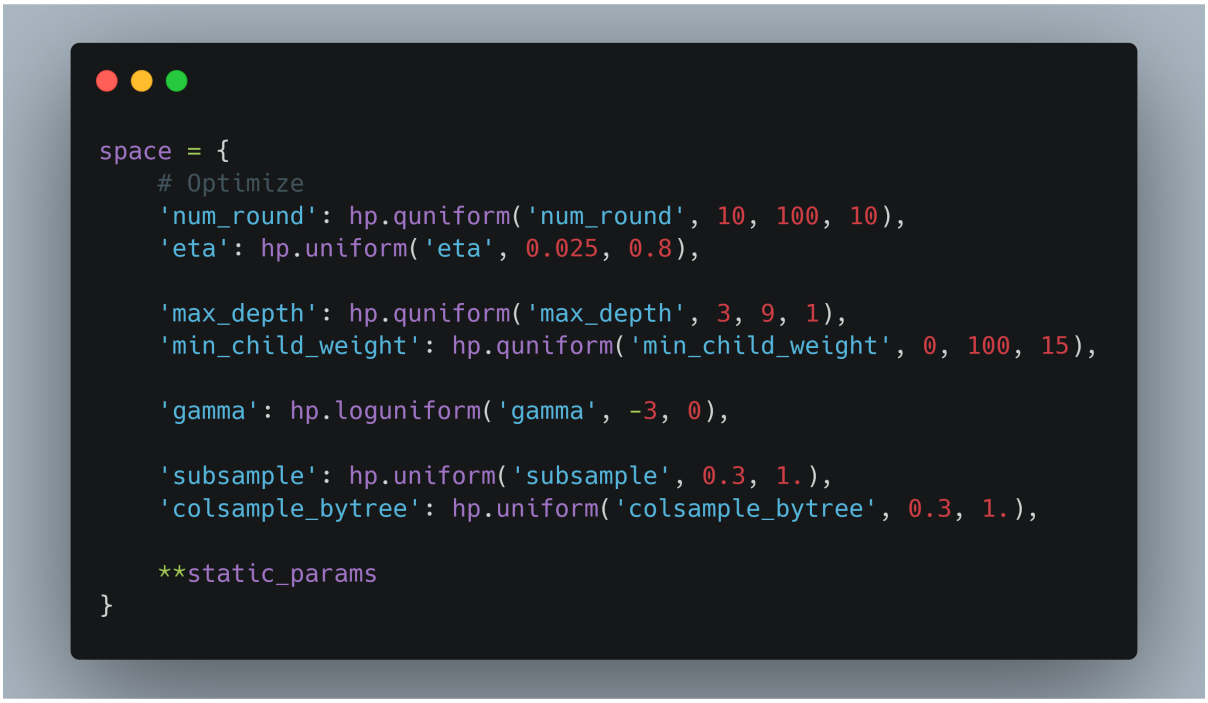
---

1. xgb

Ключевые моменты:

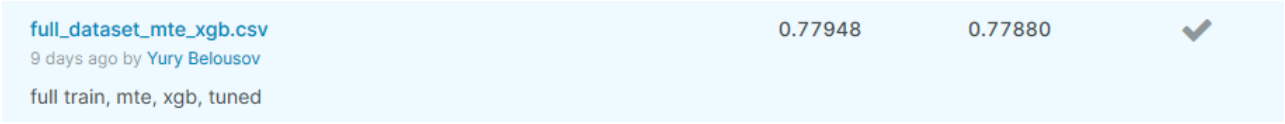
- пробовал делать one hot encoding для всех категориальных фичей, и это даже считалось, но одна итерация обучения бустинга занимала около 30 минут, что очень долго для подбора параметров
- поэтому решил использовать другой подход для категориальных фичей: а именно catboost encoder из categorical encoding (правда этот пакет не умеет работать с pyspark, поэтому пришлось вначале сдеать все в pandas Dataframe, а затем перевести в spark — что могло бы являться проблемой, если данных было бы сильно больше)
- Для подбора параметров:
    - не просто `hp.choice`, так как зачастую есть порядок в гиперпараметрах (`num_round`, `eta`, ...), а `hp.quniform`, `hp.uniform` (hyperopt parameter expresion)
    - решил подбирать все параметры *сразу*, а не поочередно

    - Общее пространство поиска параметров:



```
space = {
    # Optimize
    'num_round': hp.quniform('num_round', 10, 100, 10),
    'eta': hp.uniform('eta', 0.025, 0.8),

    'max_depth': hp.quniform('max_depth', 3, 9, 1),
    'min_child_weight': hp.quniform('min_child_weight', 0, 100, 15),

    'gamma': hp.loguniform('gamma', -3, 0),

    'subsample': hp.uniform('subsample', 0.3, 1.),
    'colsample_bytree': hp.uniform('colsample_bytree', 0.3, 1.),

    **static_params
}
```

    - выставил 30 попыток, что, возможно, маловато
    - поиск занял 1 час 20 минут
    - лучшие параметры нашлись при последней (*sic!*) попытке
- Подбор параметро дал `1.4` прироста ate по roc-auc относительно модели со стандартными параметрами
- На kaggle с этим решением получил следующий результат:



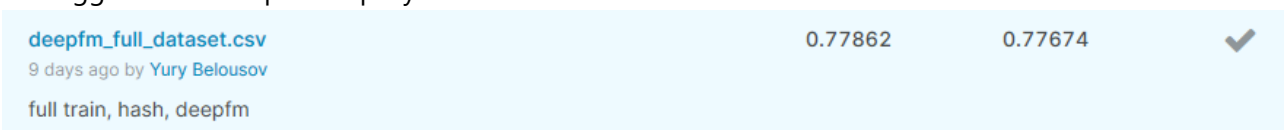
full_dataset_mte_xgb.csv 0.77948 0.77880
9 days ago by Yury Belousov
full train, mte, xgb, tuned

- (возможно), если бы отправил нетюненую модель, то все равно стал бы вторым
- Обучение на всем датасете дало `0.045` прироста по roc-auc в сравнении с обучением на случайной половине с подбором параметров (разница с моделью на половине без подбора - `0.056`), что достаточно много, и в очередной раз показывает, что обучение на подвыборке не самый лучший подход.
- код доступен в 01.ctr_xgb.ipynb

---

2. DeepCTR

- Понравилась библиотека, достаточно легко с ней работать.
- Использовал самую стандартную модель DeepFM, легко завести и работать (единственное что tensorflow почему-то не может строить граф с переменными, которые содержат нижнее подчеркивание, поэтому нужно обязательно переименовать колонки: `train.rename(columns=lambda x: x.replace('_', ''))`).
- и нужно обязательно dense фичи скастить к `np.float32`, а sparse к `string`
- Для категориальных фичей использовал DeepCTR feature hashing
- На kaggle вполне хороший результат:



deepfm_full_dataset.csv 0.77862 0.77674
9 days ago by Yury Belousov
full train, hash, deepfm

- И это при том, что я не особо экспериментировал и побирал параметры
- код доступен в 02.1.ctr_deepctr_deepfm.ipynb
- После окончания контеста попробовал другую модель: FGCNN, и она тоже показала себя неплохо (правда обучалась в 4 раза дольше):

- код доступен в 02.2.ctr_deepctr_fgcnn.ipynb

---

3. xlearn

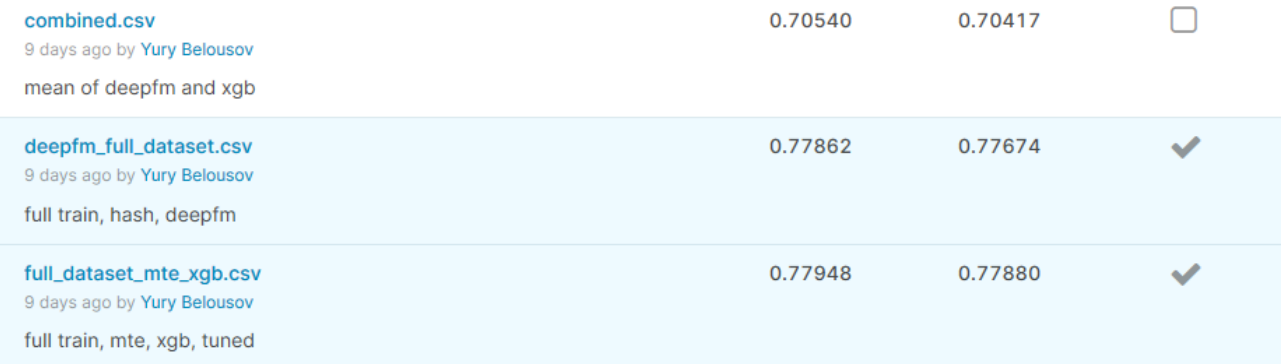
- Захотел использовать алгоритм FFM, который использовался в выигрышном решении оригинального соревнования. Но оригинальная реализация (LIBFFM) достаточно древняя, поэтому решил использовать более новую и быструю (по заявлению авторов) реализацию xlearn
- Однако этот интерфейс я тоже не до конца понял, а именно что за `libffm` формат (конкретно как составляется field).
- Поэтому использовал просто factorization machine (FM), так как нашел понятный FM пример с использованием scikit learn api.
- Но результат совсем не очень:

- код доступен в 03.ctr_xl_fm.ipynb

---

4. Ансамбль (самая интересная часть)

- Как известно, зачастую в соревнованиях побеждают комбинации различных алгоритмов (а формально — теорема Кондорсе о присяжных).
- Поэтому я захотел объединить результаты первого и второго подхода.
- Быстро написав и отправив код, я очень удивился:



- Так как результат объединения **сильно** хуже каждого из подхода по отдельности.
- Это была последняя попытка в последний день, поэтому я ничего не мог уже исправить

- Лишь на следующий день, когда я посмотрел на результаты первых двух подходов:

deepfm_full_dataset.csv

```
id,proba
566935904713,0.4102064
566935904715,0.3677692
566935904727,0.40551883
566935904737,0.5109164
566935904741,0.38027614
566935904745,0.08481385
566935904752,0.32745838
566935904759,0.012269236
566935904767,0.5162818
566935904768,0.077426665
566935904778,0.7996467
566935904785,0.06712084
566935904789,0.7352005
566935904804,0.25856566
566935904810,0.4860264
566935904817,0.10084507
566935904819,0.28937745
566935904830,0.4274971
566935904847,0.27776772
566935904848,0.24683109
566935904849,0.04271594
566935904853,0.24116129
566935904867,0.13399896
```

full_dataset_mte_xgb.csv

```
id,proba
566936179499,0.15740602
584116096827,0.044796906
575525985092,0.102862604
575525778555,0.5752256
584115725080,0.3822638
566936098987,0.40038332
575526158169,0.34860474
584115768000,0.3369564
575525703619,0.6717163
575525995952,0.117264174
566935961087,0.13080785
575525855671,0.4796365
584115565705,0.08831968
584115652121,0.55467355
584115683934,0.08740573
575525681971,0.112382025
584115674185,0.30997187
592705576855,0.4577617
575526053659,0.052949328
592705495073,0.4116129
584115934718,0.14455374
584116016277,0.31964454
566936160190,0.18716021
```

  я заметил, что в csv от pyspark `id` идут группами по 5 (возможно это количество партиций).
- а вот как у меня было написано объединение:

```
probs = []

submission_filenames = ['deepfm_full_dataset.csv', 'full_dataset_mte_xgb.csv']

for submission_filename in submission_filenames:
    submission = pd.read_csv(submission_filename)
    probs.append(submission['proba'])

proba = np.array(probs).mean(axis=0)
```

- то есть у меня никак не учитывался порядок `id` 😭😭😭
- Для исправления достаточно было добавить `sort_values` или, ещё лучше, написать через `merge`.
- В итоге, исправив, я получил следующий результат:

Submission and Description	Private Score	Public Score	Use for Final Score
combined.csv 8 days ago by Yury Belousov deemfm & xgb correct combining	0.78771	0.78639	☐

  Что почти на `0.01` больше чем у первого места 😦
- *неправильный* код доступен в 04.CombineSolutions.ipynb