# Общие замечания.

Классы несбалансированные кликов около 3%. Проблемой является обучение со сбалансированной ошибкой. Классификатор не заточенный под несбалансированные выборки, будет часто ошибаться при классификации класса малого размера(клики). В случае с кликами, это приводит к тому, что при малой ошибке классификатора, за счет хорошей классификации 0, CTR может уходит к нулю. И если продуктово важен CTR, то мы можем стремится к сбалансированной ошибке при небольшой точности классификатора. Если же нам очень важна точность клика, то мы можем хотеть получить высокую точности в классе 1. Простым решением является намеренное, смещение выборки и использование весов в функции штрафа. Более продвинутым, использование специальных методов imbalanced classification, которые во время обучения, могут гибко подстраивать параметры метода обучения под ошибку, например, добирая специальных примеров малого класса во время обучения. Я решил не использовать продвинутые методы, т.к. не знаю точных требований к ошибкам в классах и не уверен в информативности факторов. Вдруг, есть киллер фича, лучше сначала провести разведку.

Также я решил игнорировать трех идиотов с kaggke и их Field-aware Factorization Machine. Который требует отдельного разбирательства. Представить, что мне ничего не известно и начать с нуля.

# Описание работы скриптов

1. Скрипт criteo\_freq.sh выбирает топ 30 частотных признаков-категорий и сохраняет их с частотами в файл freq30\_categorial.txt. Далее это файл будет использоваться скриптом *criteoCTR.py* для преобразования признаков-категорий в бинарные признаки.
2. Запуск скрипта *criteoCTR.py* *--fc* приводит к построению двух файлов *features\_flat\_category.txt* и *features\_flat\_category\_c1.txt* , где признаки-категории ( признаки с индексом i > 13 в исходных данных) преобразуются в бинарные признаки, со значением, соответствующем наличию категории в строке. Первый файл *features\_flat\_category.txt* полученпреобразованием всех строк в *day\_0\_small\_data.txt*; второй – только строк, с кликами из большого файла *day\_0\_big\_data.txt*. После процедуры количество факторов стало равным **482.**
3. Запуск скрипта *criteoCTR.py* *--fstat* считает статистики для частотных категорий по *day\_0\_small\_data.tx.* Результат сохраняется в *fcdump.txt.*
4. Запуск скрипта *criteoCTR.py* *--fs* производит выбор наиболее информативных факторов. Используется статистика, посчитанная на предыдущем шаге. Создаются фалы с конечным набором факторов *features.txt* – производный от *day\_0\_small\_data.txt* и *features\_c1.txt* – производный от кликов в *day\_0\_big\_data.txt.* Эти наборы данных будем использовать при классификации.
5. Запуск скрипта с параметром *criteoCTR.py* *-с* производит обучение, строит классификатор, сохраняет результат.
6. Запуск скрипта *criteoCTR.py* *--fp* строит частотные паттерны отдельно для класса 0 и 1. Была попытка найти киллер фичу. Что-то находиться, но в эту тему нужно углубляться.

# Результаты

Обучил две модели SVM, GBDT, поверх них обучил линейную модель.

GradientBoostingClassifier

gbclogit\_cqf.pkl

Classifacation

c1Train=411026

c0Train=399426

c1/c0 Train=1.029042

size train, test = 810452 103465

c0 Proportion=35

c0 weight =2

0.303909534625

size train, test = 810452 103465

1test=2891

1train=411026

Test:

nErr1= 0.076790

nErr0= 0.088908

sklearn.svm import SVC

Classifacation

c1Train=8133

c0Train=7915

c1/c0 Train=1.027543

size train, test = 16048 2139

c0 Proportion=37

c0 weight=2

0.857410004675

size train, test = 16048 2139

1test=54

1train=8133

nErr1= 0.000000

nErr0= 0.019070

305.0 0.0

1test=542

1train=81943

nErr1= 0.357934

nErr0= 0.011376

# Замечания

Пробовал делать количественные факторы из категорийных. Для этого я искал частотные паттерны (области в выборке, описанные категориями), где классы хорошо разделяются, или присутствует только преимущественно один класс(высока условная вероятность одного из классов). Такие области малы, и порожденный ими фактор почти всегда равен 0. Но если обучить модель “неплохо” предсказывающую эту область, вероятность предсказания для всей выборки можно использовать как полный количественный фактор.