Ресурсы

1 вирутальная машина в Яндес.Облаке: 4 ядра, 16GB оперативной памяти.

Что пробовал и не упало

1. Логистическая регрессия
2. XGBoost
3. XGBoost + наивный MeanTarget encoding (берём среднее значение)
4. XGBoost + честный MeanTarget encoding (делаем "правильно": усредняем значения предыдущих targets)
5. Логистическая регрессия поверх XGBoost

В этом разделе XGBoost запускался на численных признаках + 2х категориальных признаках.

Для XGBoost + наивный MeanTarget подобрал вручную значение параметра alpha регуляризации (из Addittive https://en.wikipedia.org/wiki/Additive_smoothing). Вручную: перепробовал 5-6 и остановился на самом простом.

Аналогичную штуку проделал XGBoost+ честный MeanTarget.

Результаты

	public leaderboard ROC AUC	private leaderboard ROC AUC
Логистическая регрессия	0.69861	0.69601
XGBoost	0.72557	0.72374
XGBoost+наивный MT	**0.73240**	**0.73032**
XGBoost+наивный MT (3 кат. признака)	0.72768	0.72567
XGBoost+честный MT	0.73142	0.72959

Логистическая регрессия поверх листьев XGBoost (лист каждого дерева - категориальный признак) работала также или хуже логистической регрессии на при валидации. Поэтому не включена в таблицу.

Про "XGBoost+наивный MT (3 кат. признака)" в следующей части.

Не заработало

1. XGBoost с большим колличеством категориальных признаков
2. XGBoost с параметрами из hyperopt
3. XGBoost+MeanTarget с параметрами из hyperopt

В этом разделе не смог сделать посылки. Ибо их сделать, если XGBoost упал или java развалась по дороге???

Пробовал выдать spark 12 GB, вместо значения по умолчанию, что не помогло.

Постоянное падение XGBoost при множественных попытках отойти от решения задачи, данного notebook, жутко бесило. В какой-то момент я даже был в гневе: пытался добавить лишь один категориальный признак, XGBoost или spark продолжал падать после его добавления.
Бесило ещё потому, что я потралил на это много времени, оно не зароботало, а я мог бы потратить это время на реализацию логистической регрессии поверх XGBoost.

Как я пытался добавлять категориальные признаки?

Посчитал число различных значений первых 10 категориальных признаков. Выбрал 4 признака с самым маленьким числом вхождений. Попрабовал каждый из них добавить в XGBoost+наивный MeanTarget encoding. Каждый раз было всего 3 категориальных признака+все численные. Процедура обучения вычисления subsion.csv упала на каждом варианте. По памяти.

Замечания: До добавления или на сэмплированной обучающей выборке оно не падало. Уменьшив глубину значения max_depth до 5 я всё-таки смог добавить 1 признак и обучиться на тренировочной выборке, но это ухудшило результат модели.

Как я пытался использовать ответ hyperopt?

Взял параметры XGBoost из 5й домашки. Параметры max_depth=10, num_round=100 значильно увеличили длительность обучения и размер модели. При обучении на треннировочной выборке XGBoost заврешался. После этого Java падала, разражало. Я повторил 3 раза, в 2 из которых не запускал лишние ячейки, чтобы не забивать память. Java продолжила падать сразу после завершения XGBoost.

Если нужно, могу повторить эксперименты и записать screencast падения spark+XGBoost.

Возможно, надо было не жадничать, и выделить больше GB оперативной памяти в Яндекс.Облаке, что было бы дороже.