Как мы обучали бустинги на ГПУ и обломались

GPU для сеток

GPU для бустингов

Почему бустинги и на GPU?

- Основные продовые модели: риски, лиды, антифрод
- Бустинги отлично справляются с задачами классификации
- Нет проблем катить в прод:

  - модели легкие, обычно до 5 mb

  - модели интерпретируемые

- GPU дает возможность быстро экспериментировать:

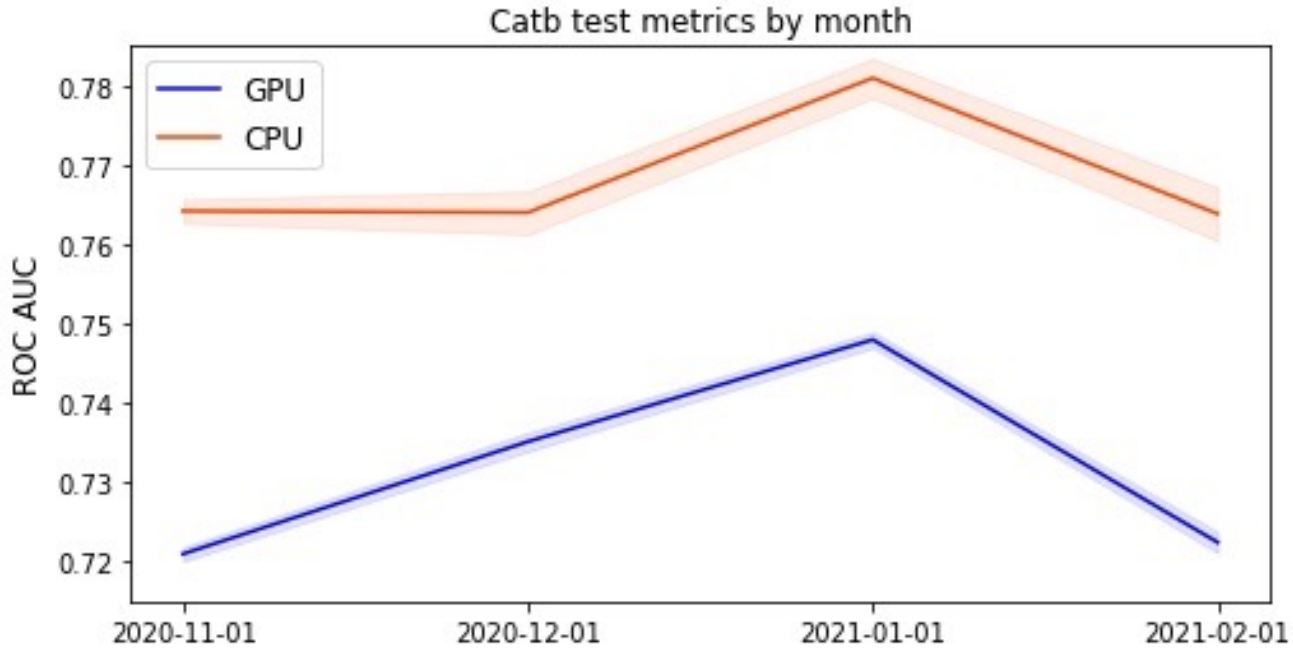
  - подбирать параметры алгоритма

  - подбирать лучший трейн сет

  - и главное: проводить итеративный отбор признаков (3500 –> 100-200)

Почему обломались?

- Обучая catboost на CPU и GPU выявили большую разницу в качестве: модели очень разные, при одинаковых параметрах
- Есть issue годовалой давности по теме – все еще открытый
- Насколько результат случаен? Что насчет xgboost и lightgbm?



CPU vs GPU : замеряемые статистики

- ROC AUC на отложенном тесте
- стабильность распределения предиктов
- стабильность feature importance
- отличия в дефолтных параметрах

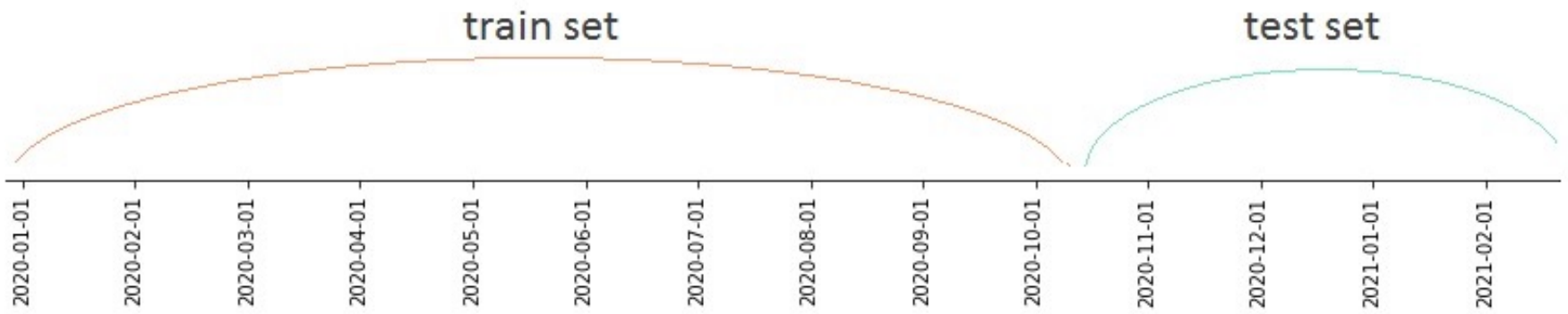
- размер моделей
- время обучения
- время инференса

XGBoost

- время перекладывания данных на GPU

О датасете

- Абоненты МТС, имеющие кредитные продукты банков
- Таргет – просрочка по кредитным платежам
- 500 К строк в трейне, 200К строк в тесте
- Валидация – отложенная по времени
- 292 вещественных признака
- 11 категориальных признаков (есть высококардинальные)



О дизайне эксперимента

- 3 библиотеки – catboost, xgboost, lightgbm
- 3 сета параметров разной сложности для каждой библиотеки
- 10 моделей на CPU и 10 на GPU для каждого сета параметров
- Рандомная генерация seed

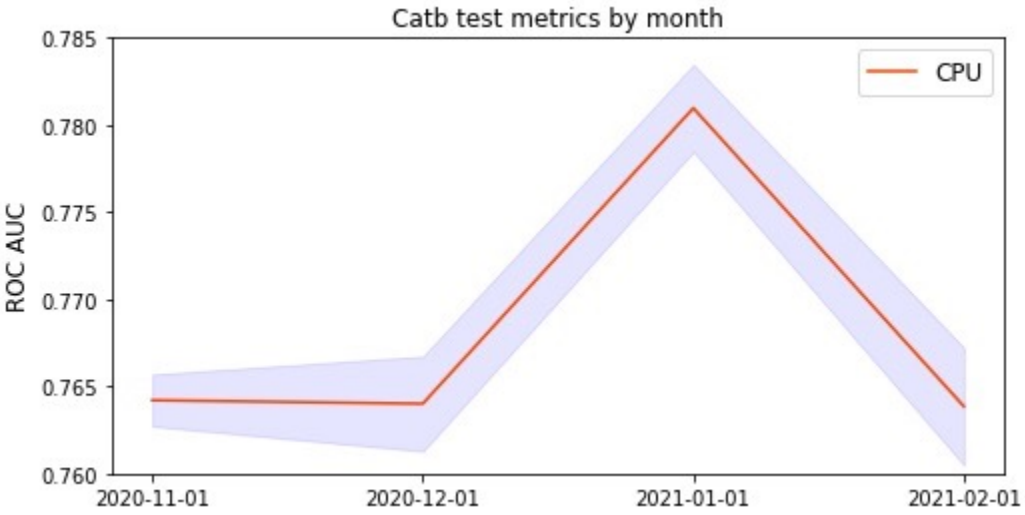
Параметры:

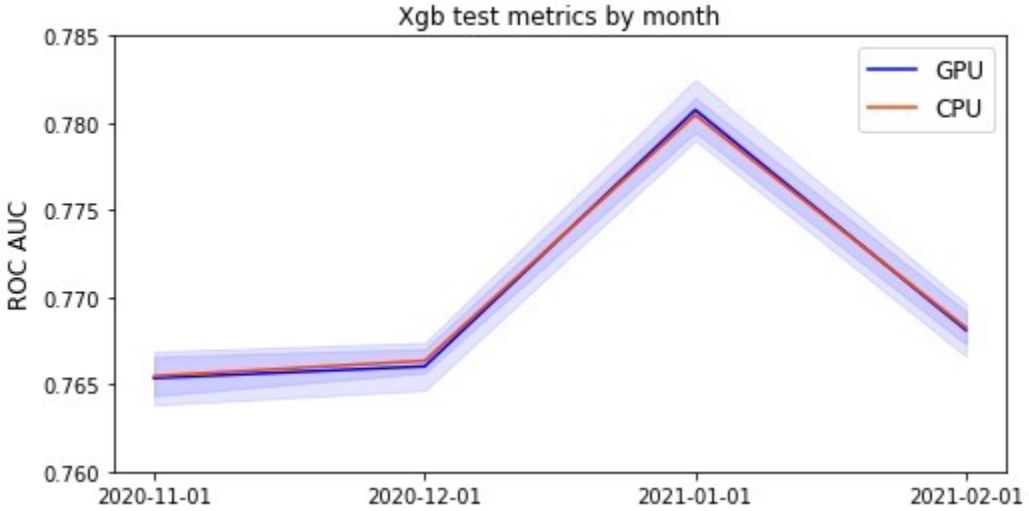
	hard	**medium**	**light**
n_estimators	1000	700	800
max_depth	6	5	4
subsample	0.8	0.8	0.8
learning_rate*	0.02	0.04	0.05

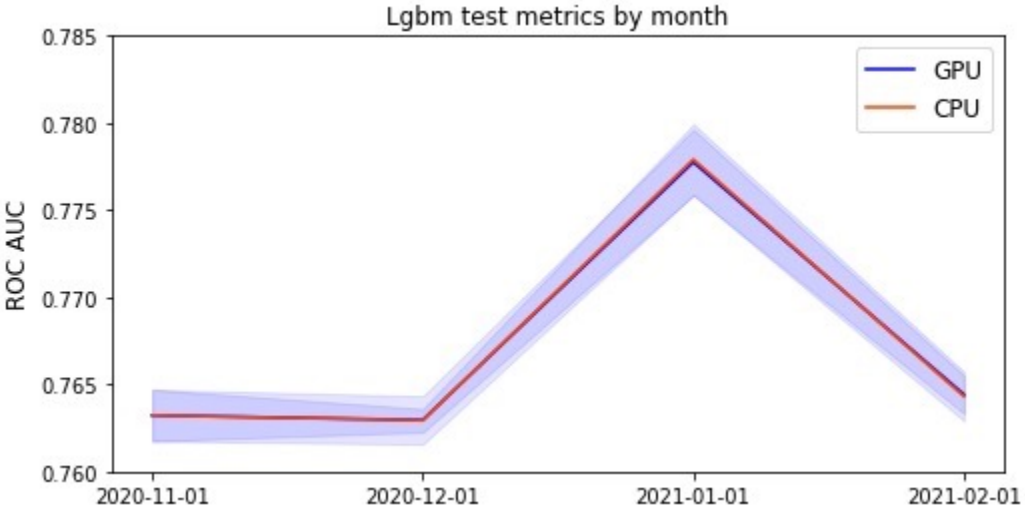
* learning_rate для xgboost и lightgbm, для catboost - дефолтный

Метрика ROC AUC

- Тестовые метрики xgboost и lightgbm практически идентичны
- Дисперсия метрик catboost CPU моделей в 7 раз выше чем у xgboost, в 3 раза - чем у lightgbm

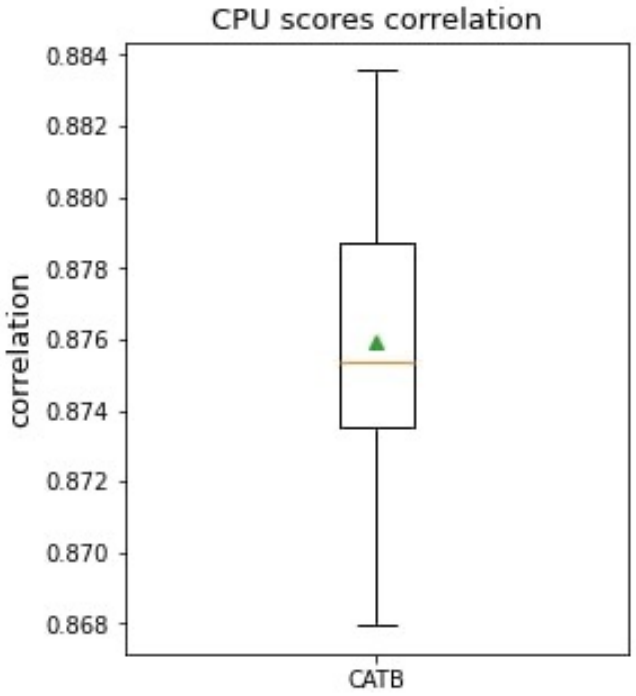


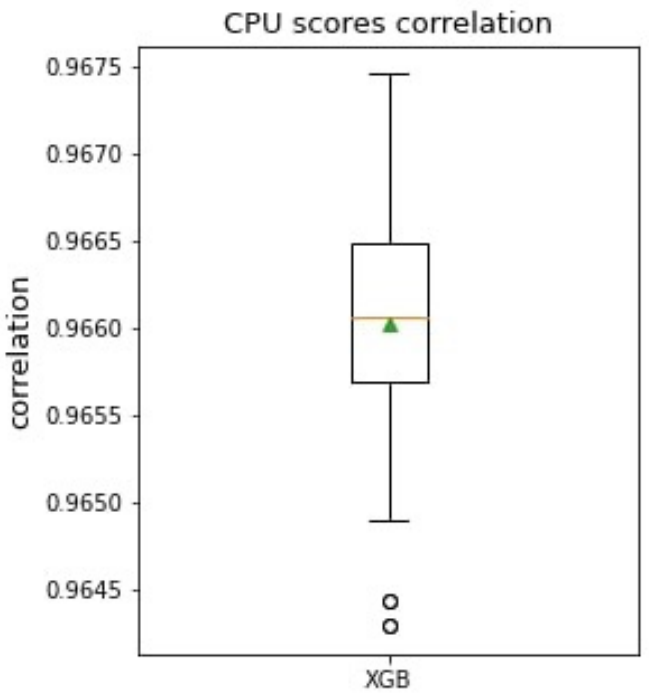


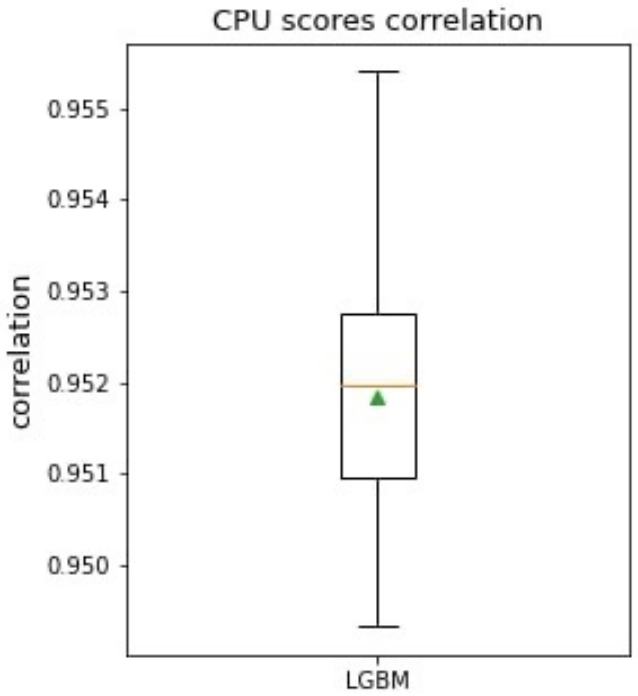


Корреляция предиктов CPU моделей

- Предикты CPU моделей catboost довольно сильно зависят от seed (min corr: 86.8)
- Самые стабильные предикты выдает xgboost



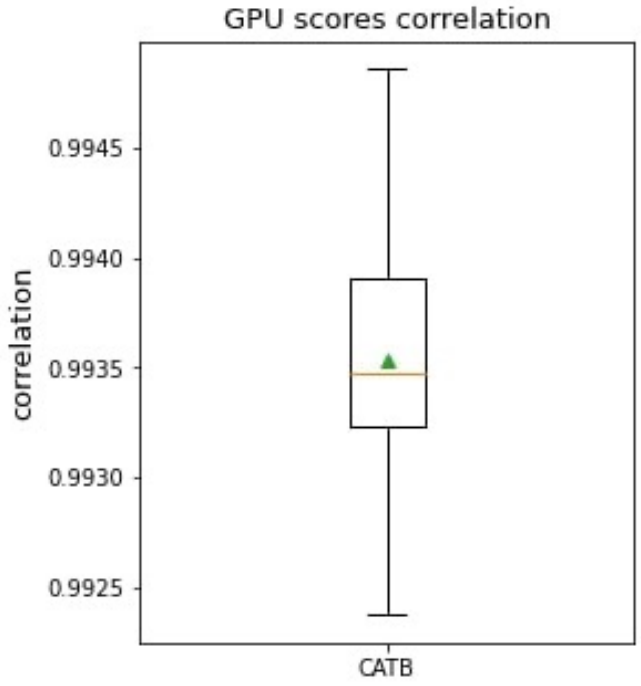


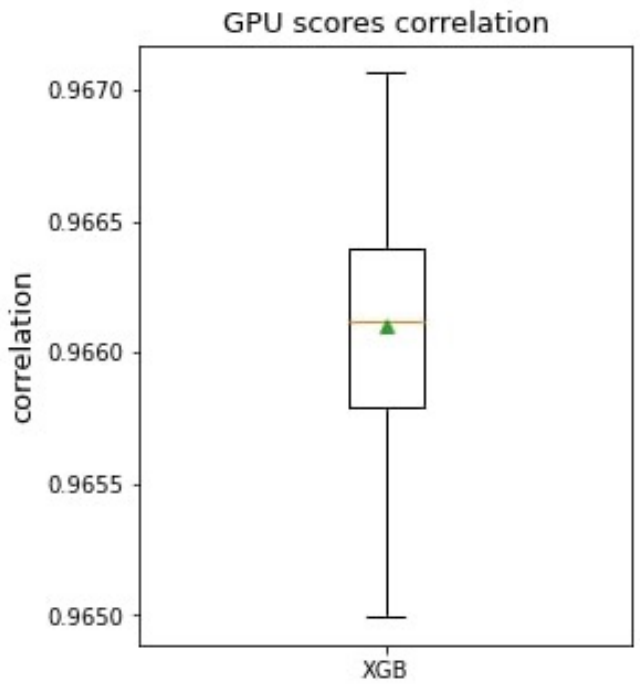


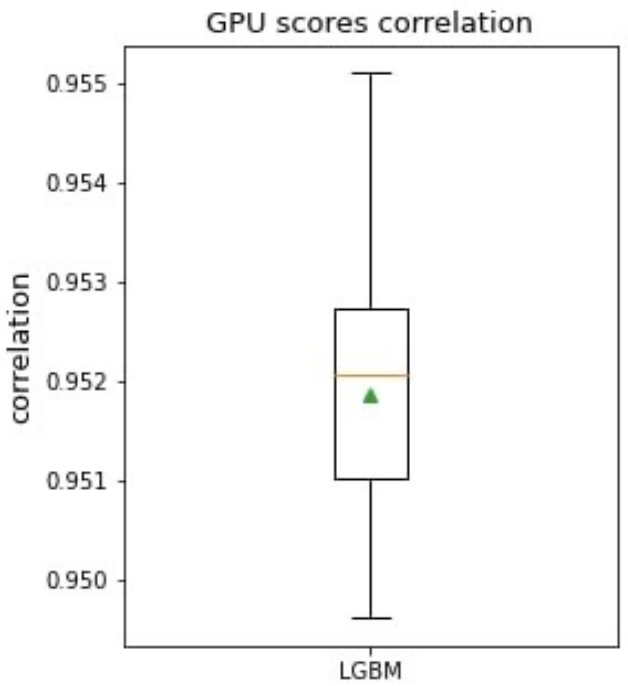
Вывод: лучше указывать seed (у catboost и lightgbm он генерируется на лету)

Корреляция предиктов GPU моделей

- Все корреляции > 0.94
- Самые стабильные предикты теперь у catboost

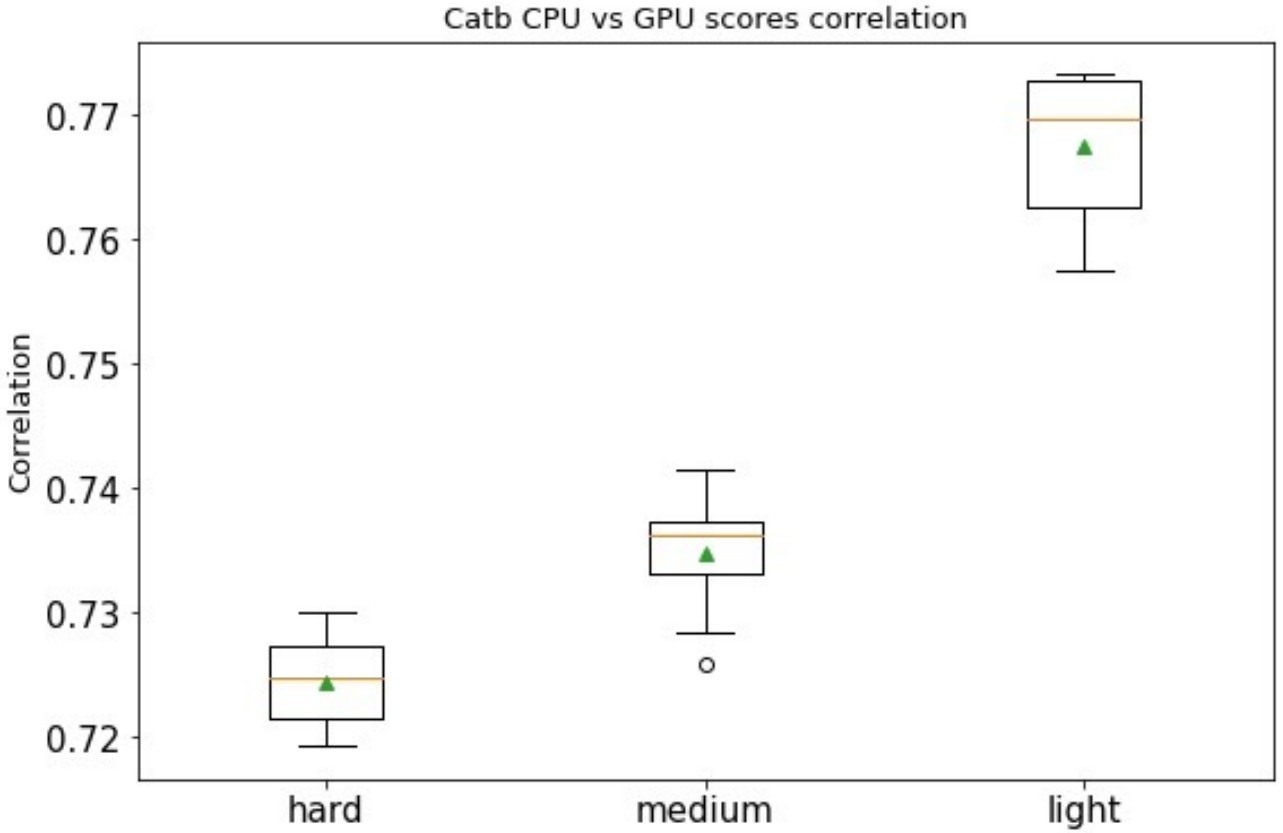






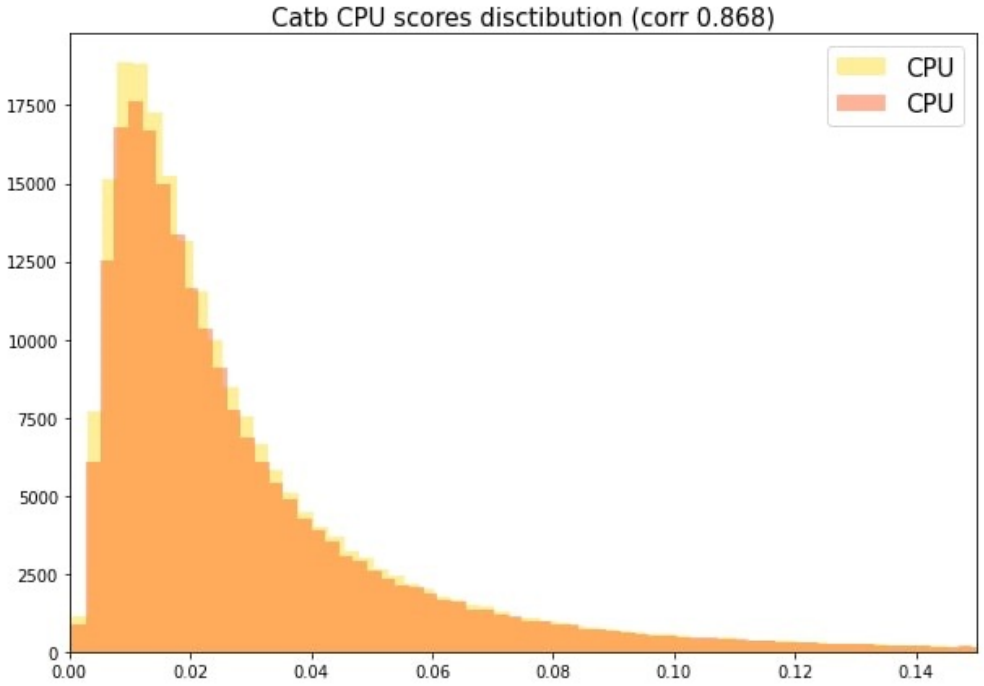
Корреляция предиктов CPU vs GPU

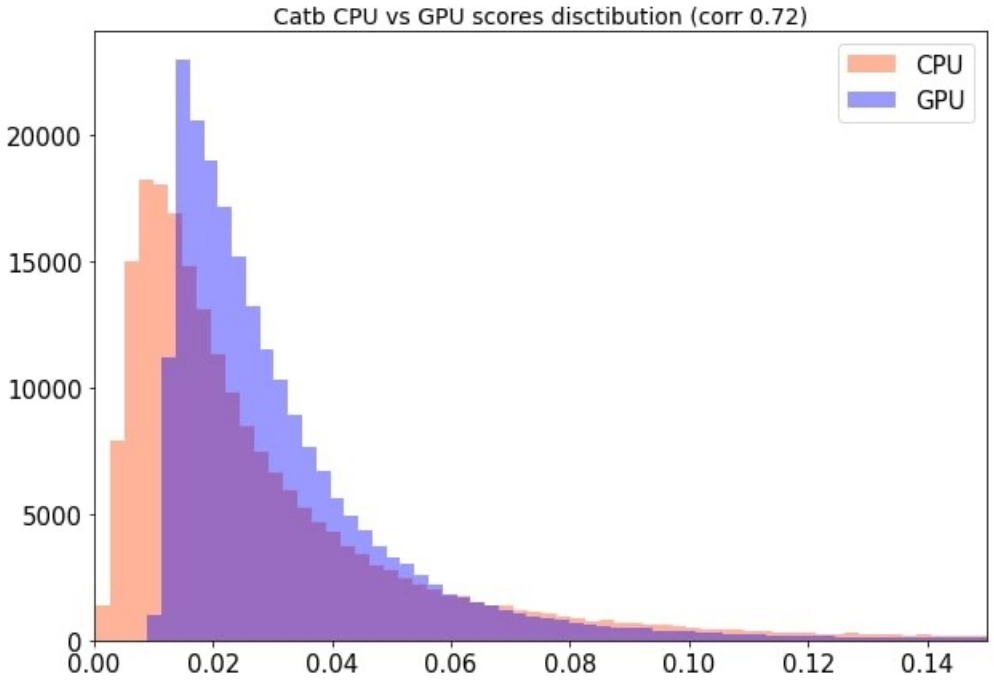
- Корреляция предиктов xgboost (min 0.96) и lightgbm (min 0.97) остается на высоком уровне
- Корреляция предиктов catboost сильно меньше (min 0.72)
- Корреляция catboost зависит от сложности параметров: проще параметры – выше корреляция



Распределение предиктов

- Распределения предиктов catboost CPU vs GPU заметно отличаются





- Некоторые пары моделей catboost CPU и пары моделей catboost GPU проходят тест Колмогорова-Смирнова на равенство распределений. Пары CPU-GPU – никогда его не проходят
- Пары xgboost моделей чаще проходят тест, чем нет (84% тестов в нашем эксперименте)
- Любые пары lightgbm моделей почти всегда проходят тест (99% тестов)

Порядок признаков: корреляция

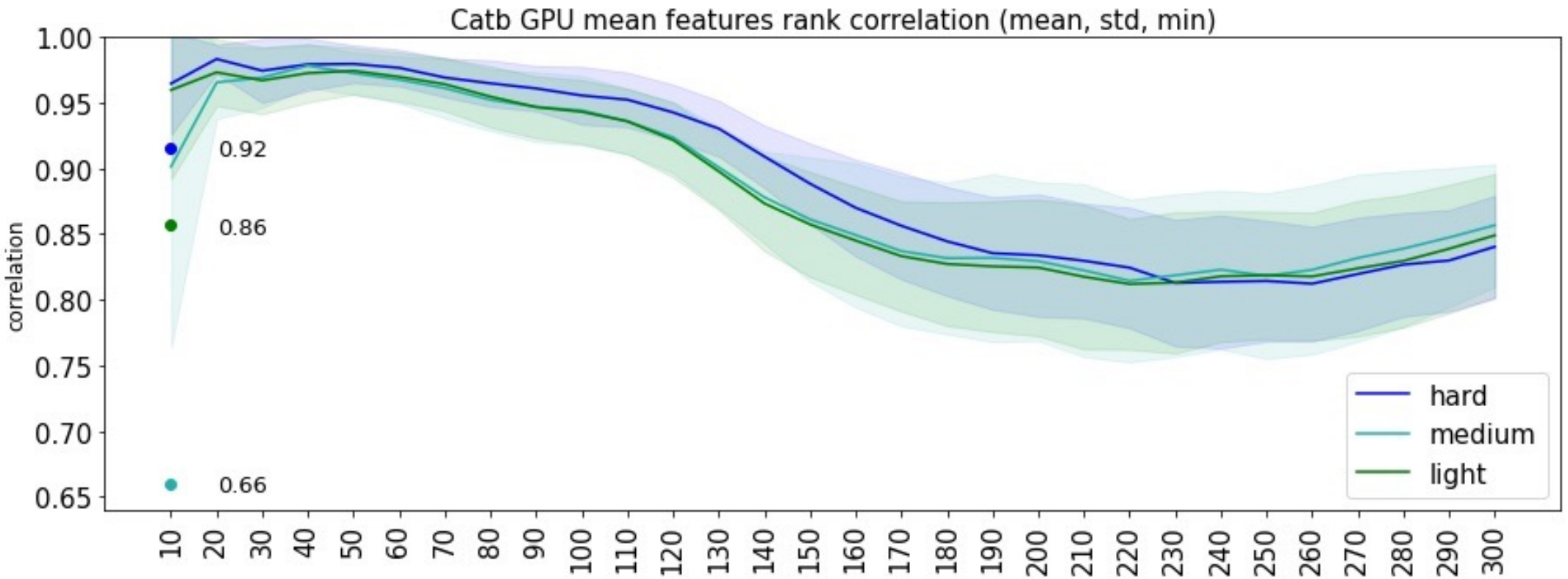
Хотим: понять насколько отличается порядок признаков:

- внутри пула моделей обученных на CPU
- внутри пула моделей обученных на GPU
- между моделями CPU и GPU

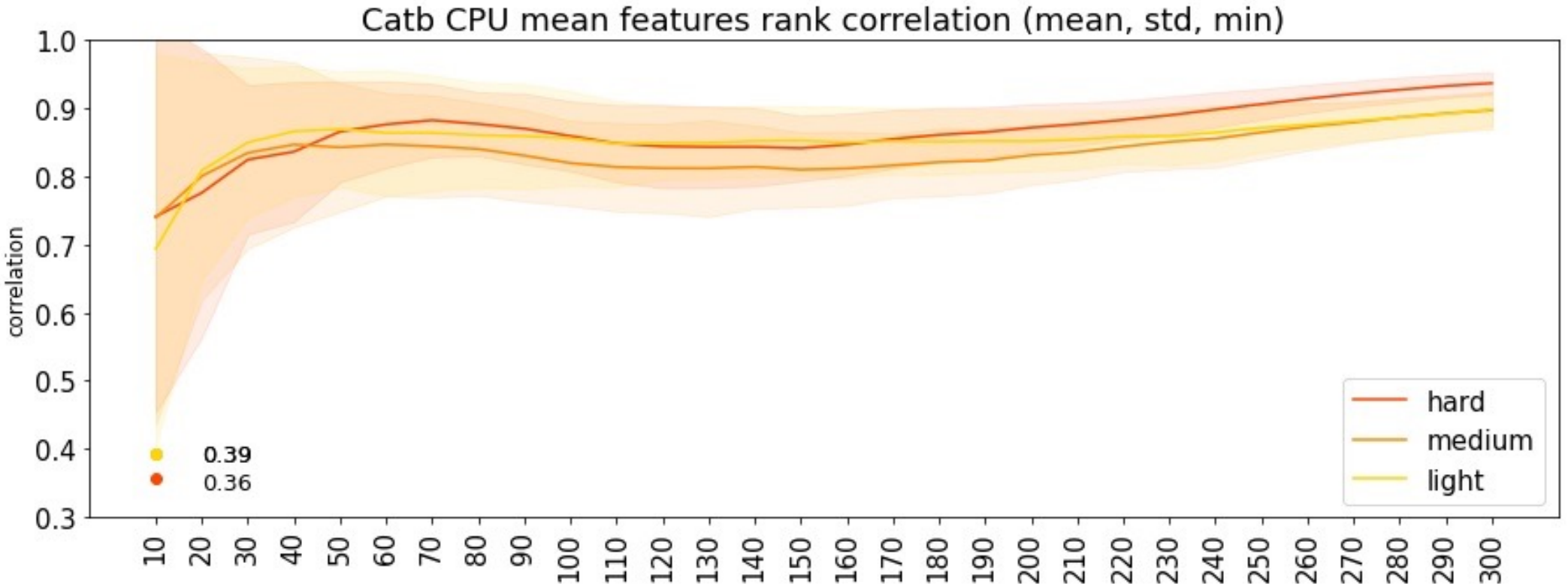
Идея:

- возьмем df с feature_importance для каждой модели
- будем шагать по нему, прибавляя по 10 фичей
- на каждом шаге для каждой пары моделей посчитаем ранговую корреляцию Спирмена
- усредним ответы для каждого шага (топ 10 фичей, топ 20 фичей и т.д.)
- построим график с доверительным интервалом, посмотрим на среднее и минимумы

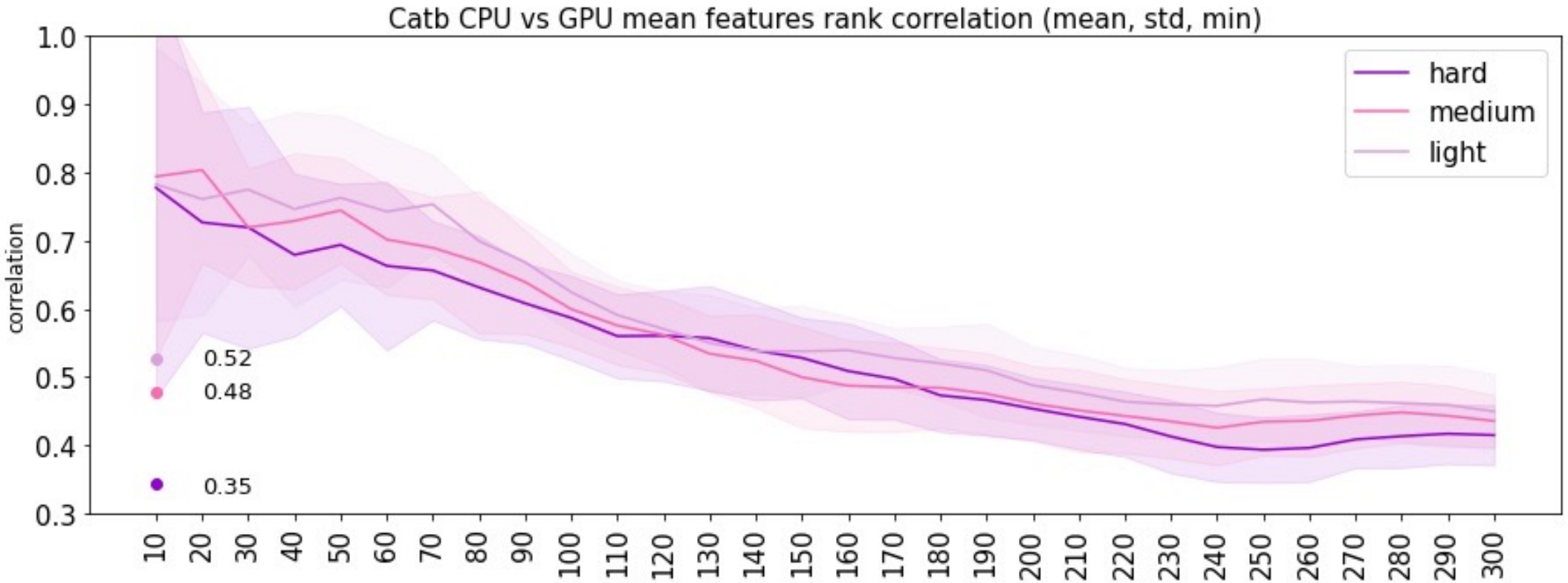
Порядок признаков: catboost GPU



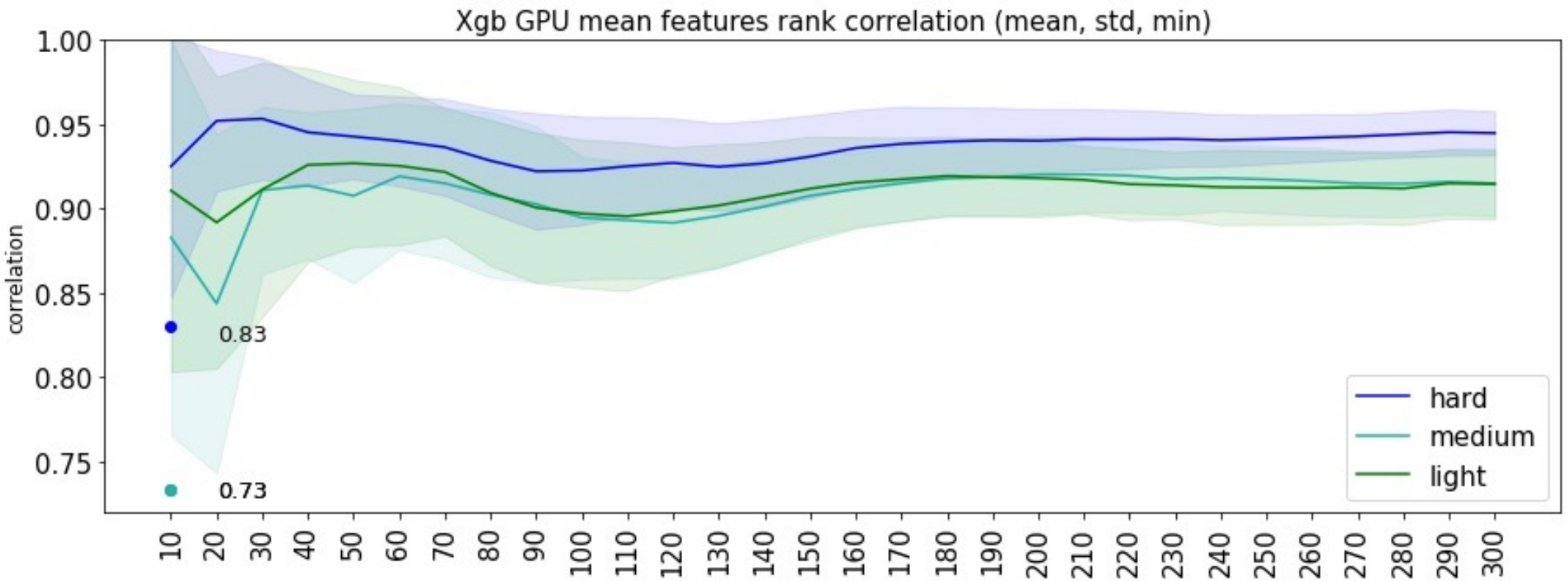
Порядок признаков: catboost CPU



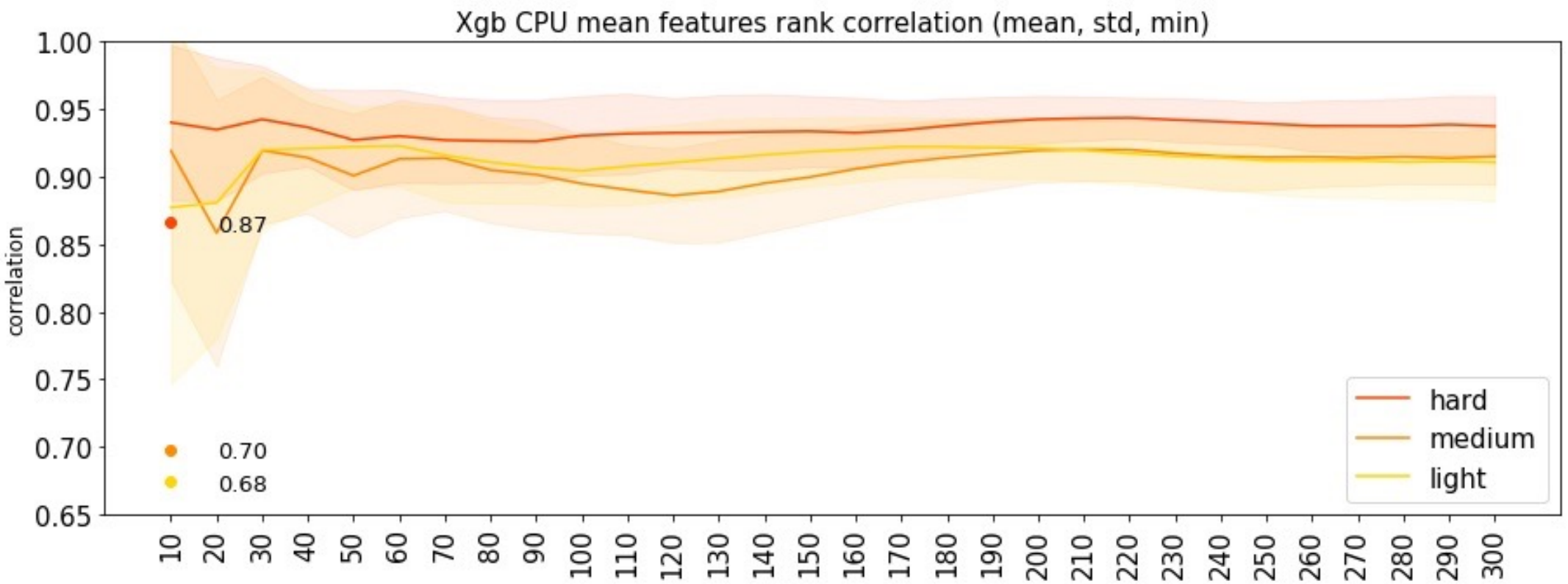
Порядок признаков: catboost CPU vs GPU



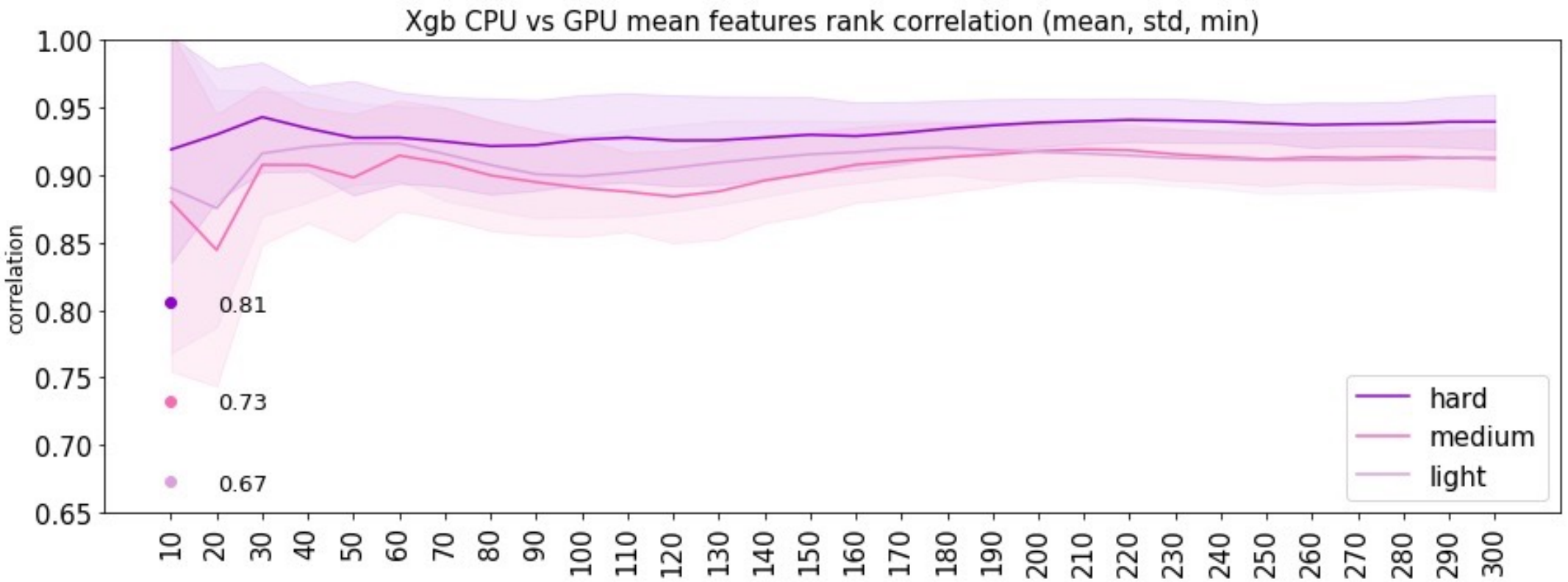
Порядок признаков: xgboost GPU



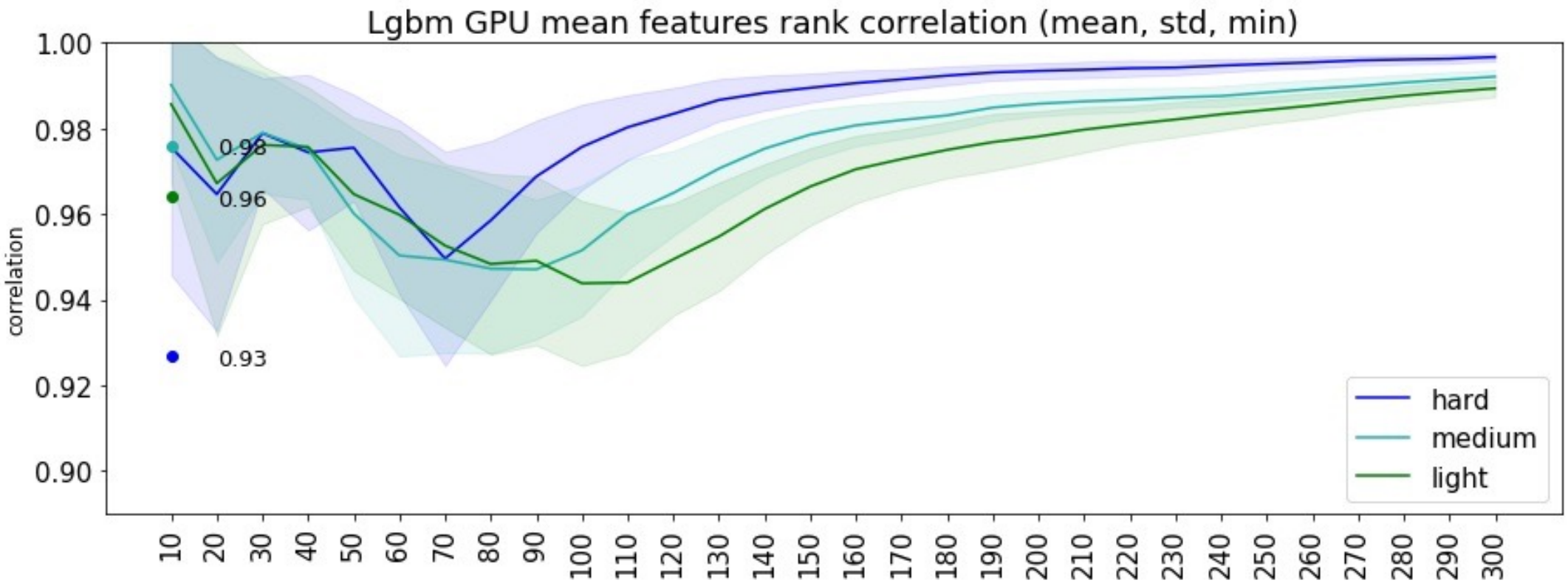
Порядок признаков: xgboost CPU



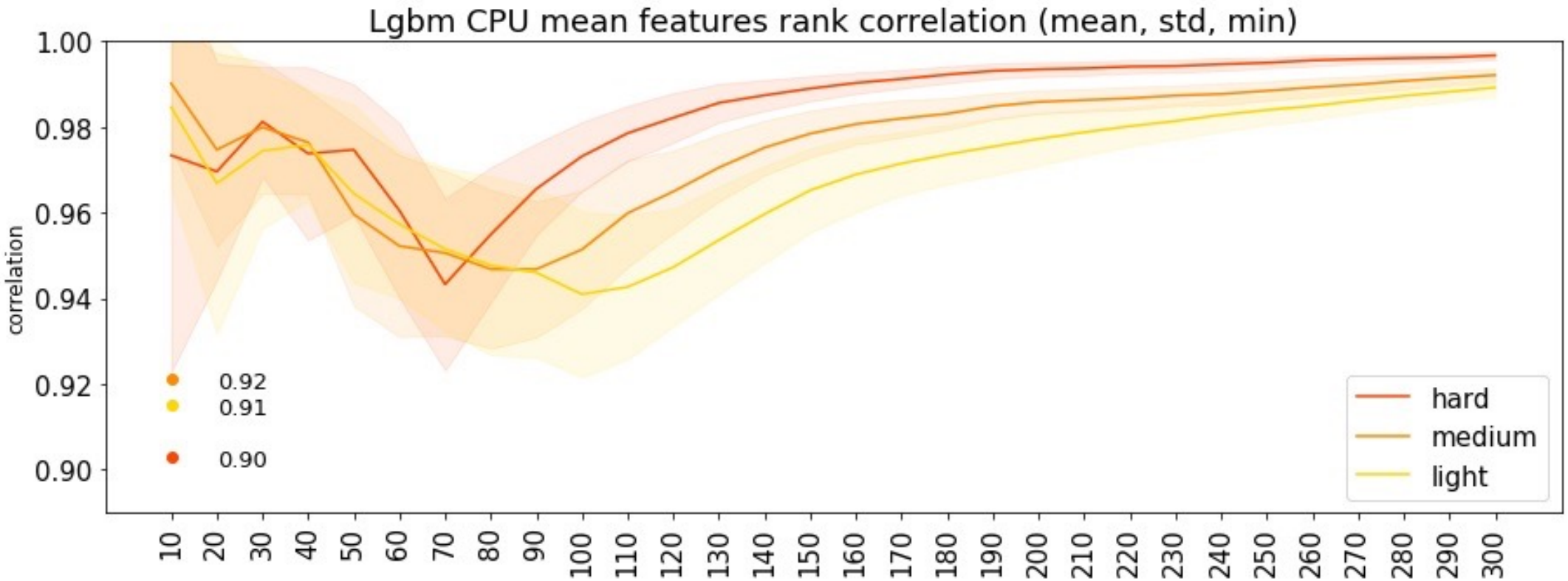
Порядок признаков: xgboost CPU vs GPU



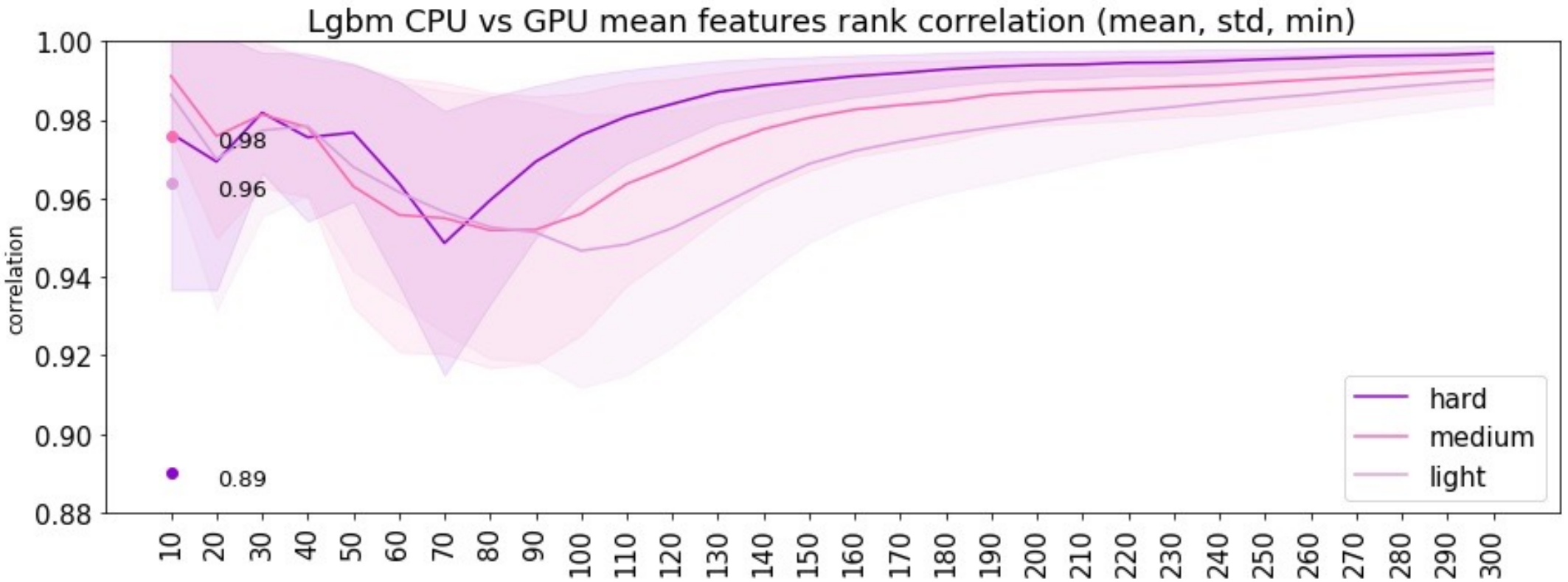
Порядок признаков: lightgbm GPU



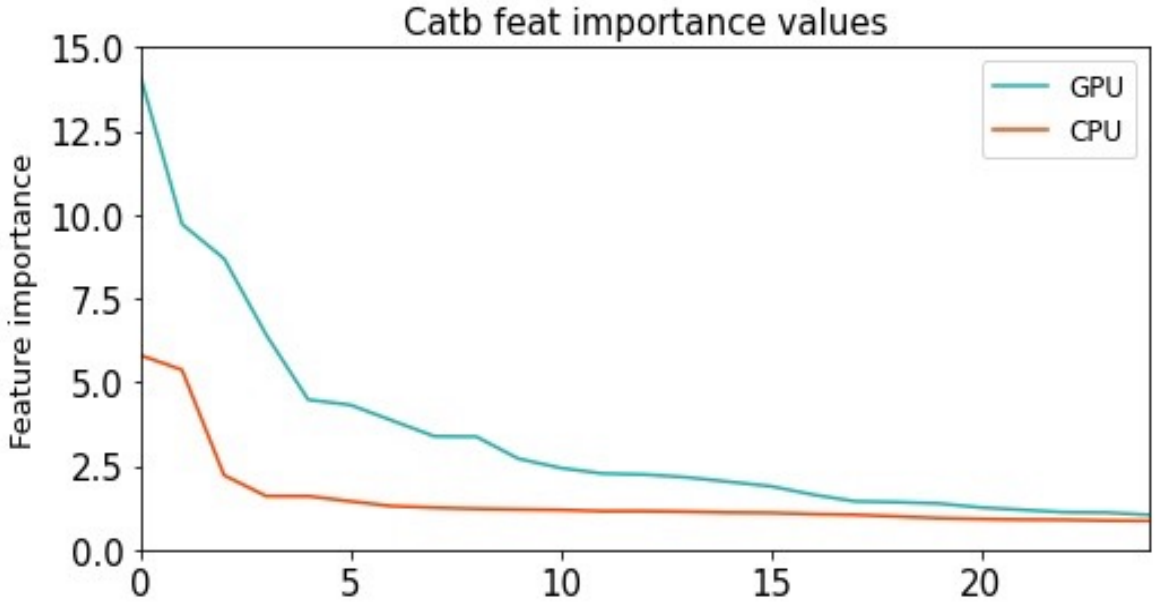
Порядок признаков: lightgbm CPU

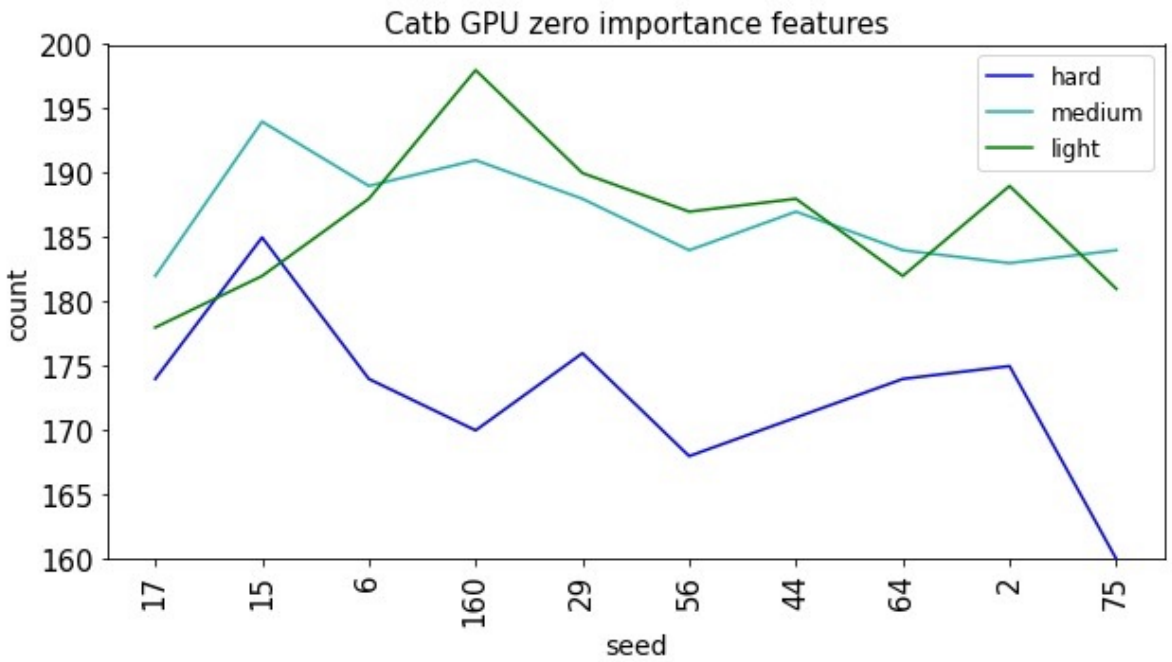


Порядок признаков: lightgbm CPU vs GPU



Важность признаков: зануление





- GPU catboost модели зануляют больше половины признаков
- Все остальные модели – CPU catboost, xgboost и lightgbm – не больше 8

Отличия в дефолтных параметрах (CPU vs GPU)

- У lgbm параметры полностью совпадают
- У xgb отличается только метод поиска лучшего сплита (exact vs hist)

Отличия catboost моделей:

Other parameters	CPU	GPU
border_count	256	128
bootstrap_type	MVS	Bayesian
bagging_temperature	-	1

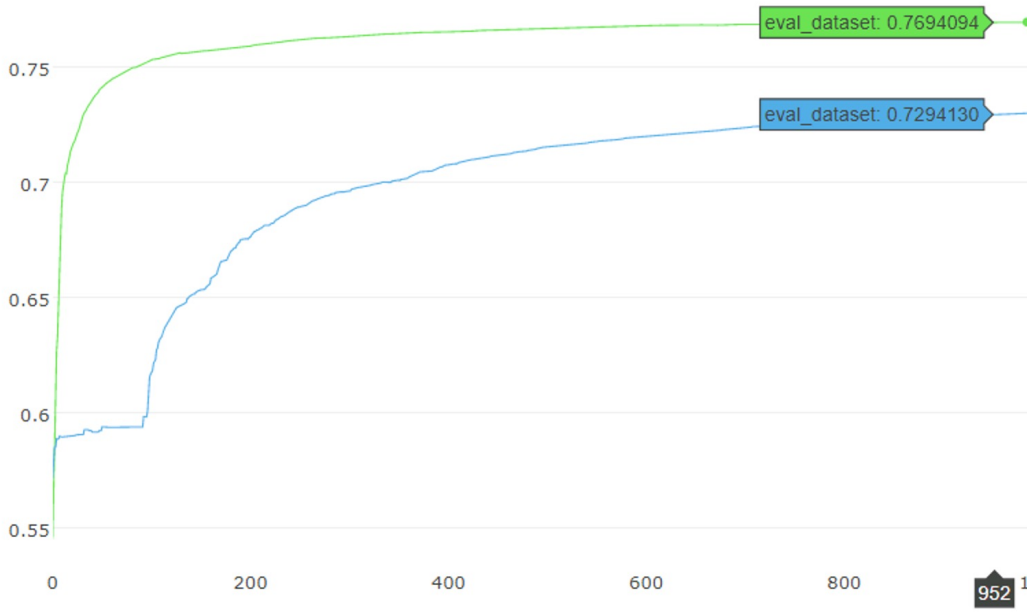
LR parameter	CPU	GPU
LR params hard	0.147	0.024
LR params medium	0.204	0.033
LR params light	0.180	0.029

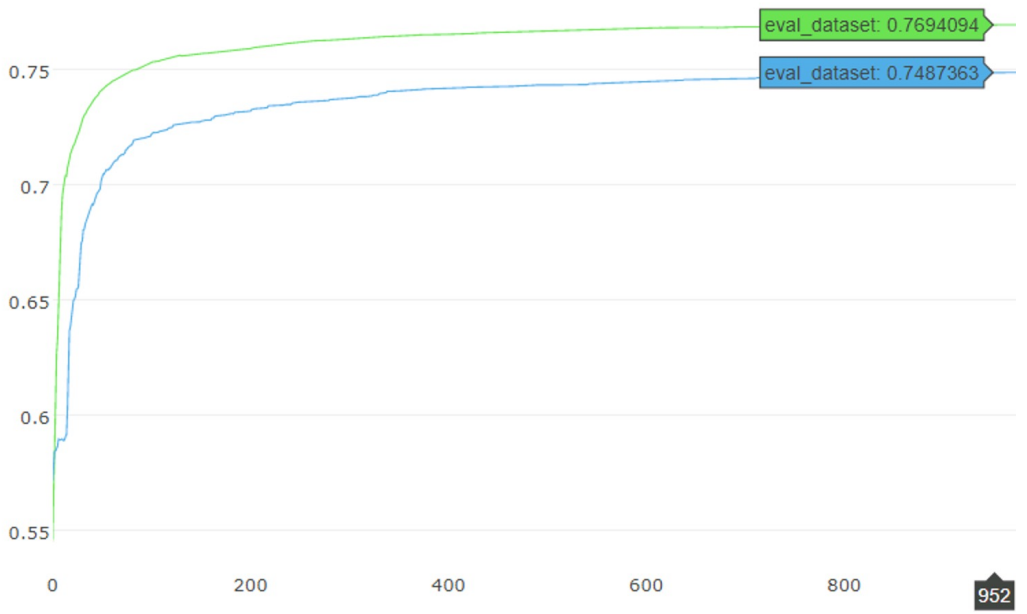
Изменение гранулярности вещественных признаков, типа bootstrap-а, параметров семплирования значимо не повлияло на метрику качества GPU

GPU **learning_rate** всегда примерно в 6 раз меньше

GPU модели недообучаются?

- Увеличение итераций не помогает – и без того маленький learning_rate автоматически уменьшается
- Фиксация learning_rate на уровне CPU дает прирост, сокращая разрыв с до 2 AUC



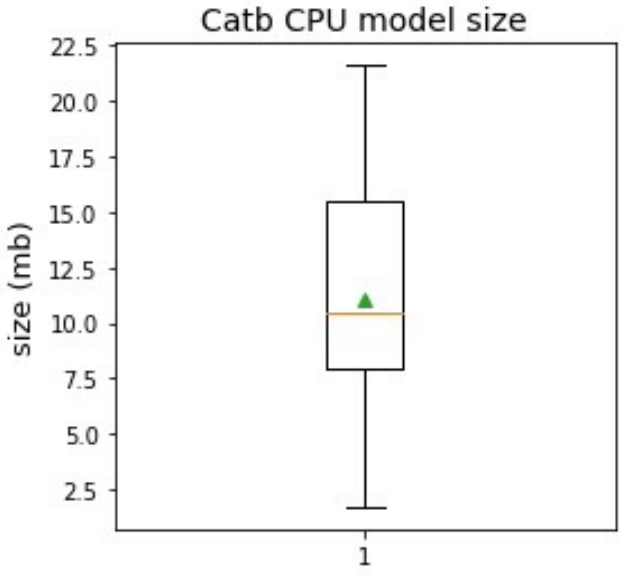


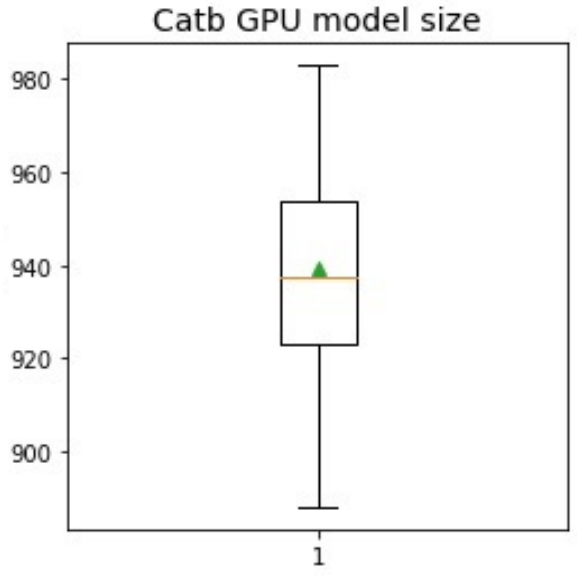
Таким образом, catboost не всегда хорошо подбирает learning_rate, обучаясь на GPU – модели могут получиться сильно недообученными

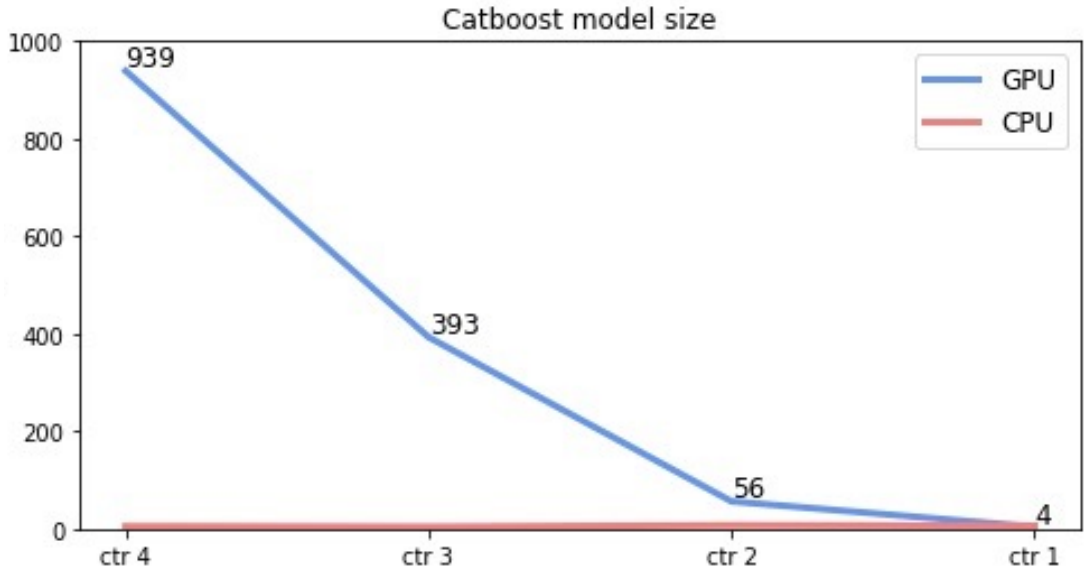
Проблема размера моделей

Обратили внимание на слишком большой размер catboost GPU моделей: средний размер почти в 100 раз больше, чем у CPU моделей

Для контроля над размером разработчики советуют крутить параметр **max_ctr_complexity** (сколько комбинаций категориальных фичей можно использовать для поиска сплита)

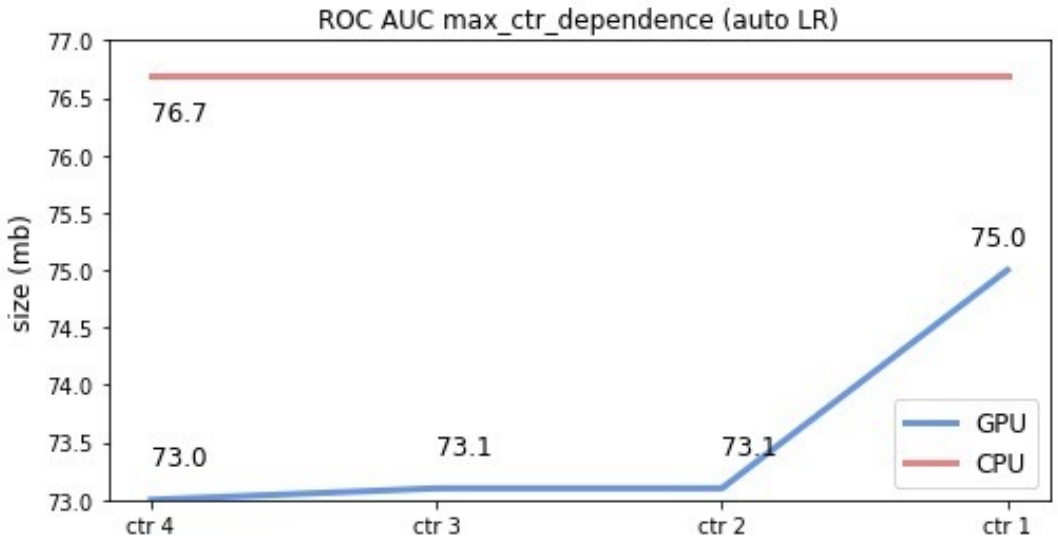


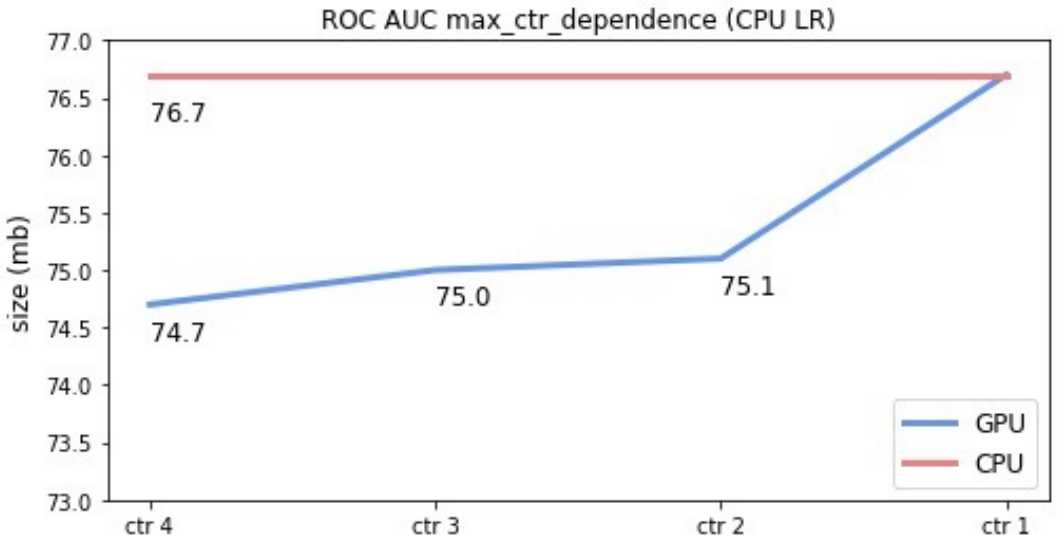




Куда делись еще 2 AUC?

- Внезапно оказалось, что max_ctr_complexity – ключевой параметр и для качества

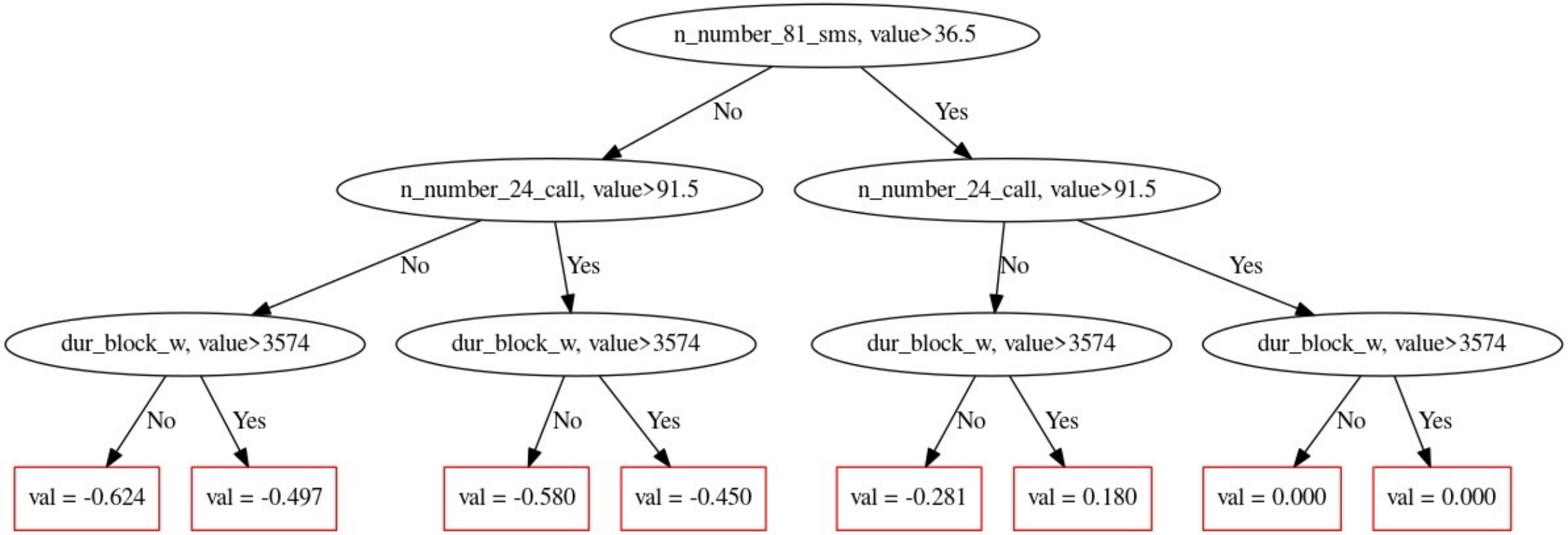




- Хотя max_ctr_complexity = 4 стоит по умолчанию при обучении и CPU и GPU моделей, по факту он работает только на GPU: регуляризатор размера CPU моделей не дает ему включиться
- Разрыв в метриках был получен и на других данных – Kaggle HomeCredic Risk, где уникальных категорий сильно меньше

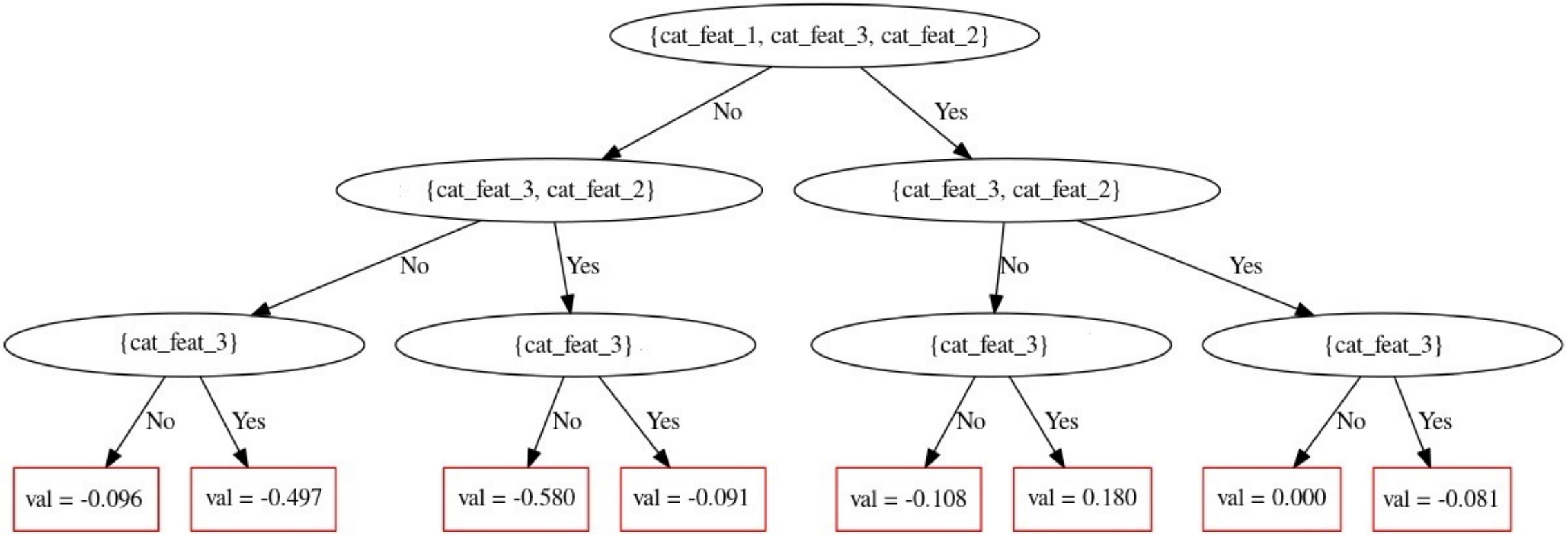
Первое дерево модели CPU

- В разбиениях участвуют вещественные признаки
- По итогу категориальные признаки не входят в топ признаков ансамбля



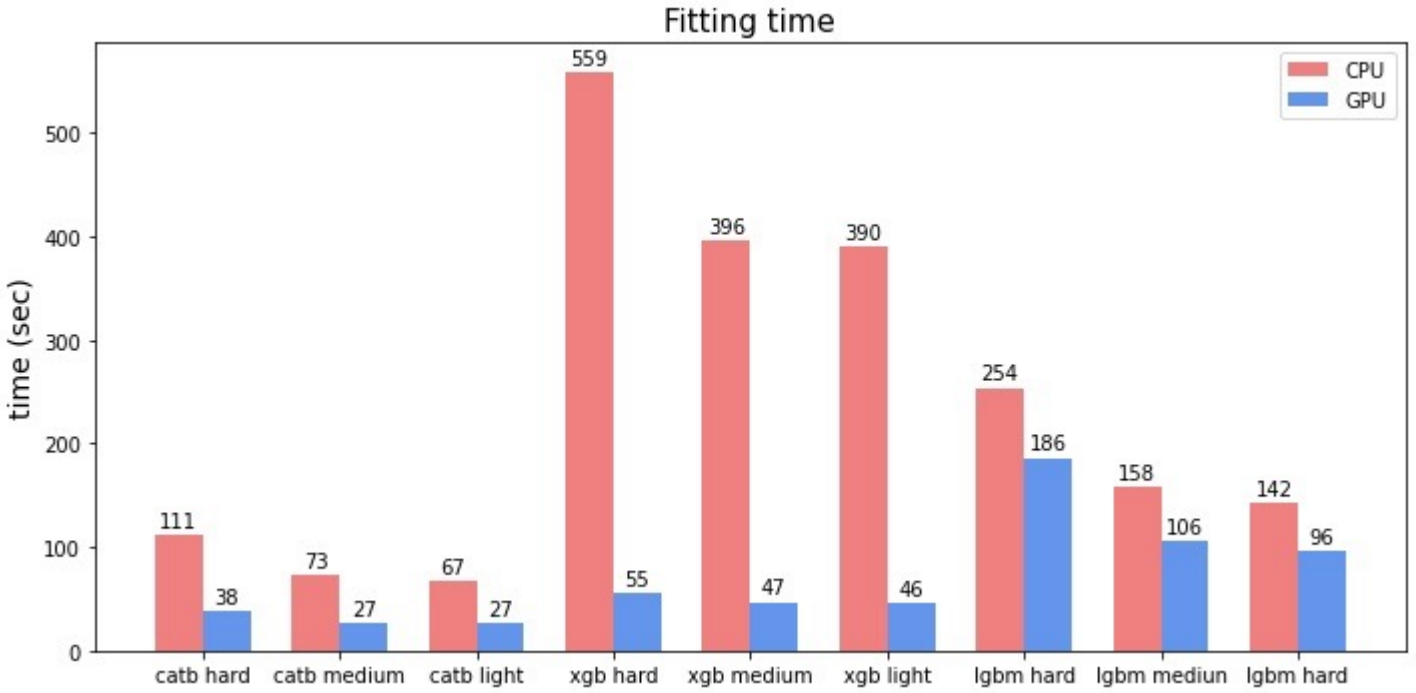
Первое дерево модели GPU

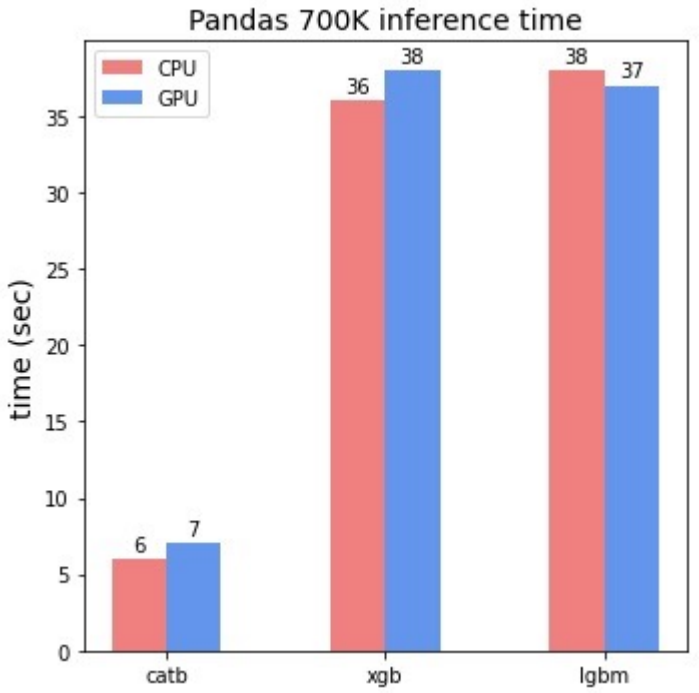
- В разбиениях участвуют участвуют категориальные признаки и их комбинации
- По итогу 3 категориальных признака входят в топ 5 признаков ансамбля



Время обучения и inference

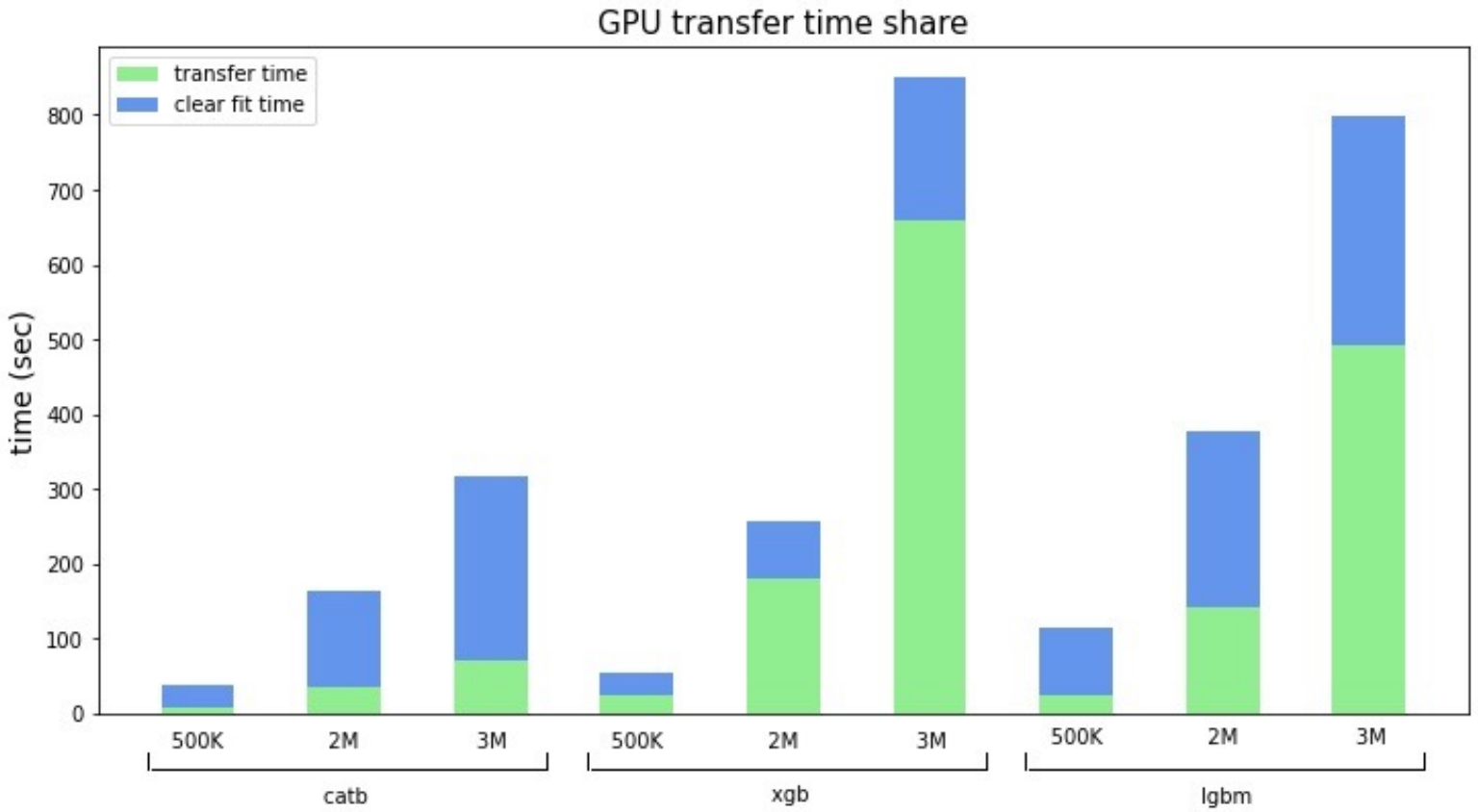
- Быстрее всего обучается catboost – и на CPU и на GPU
- У xgboost наибольший выигрыш в скорости от GPU + он растет с усложнением параметров, у lightgbm наименьший
- Catboost inference намного быстрее (~6 раз)





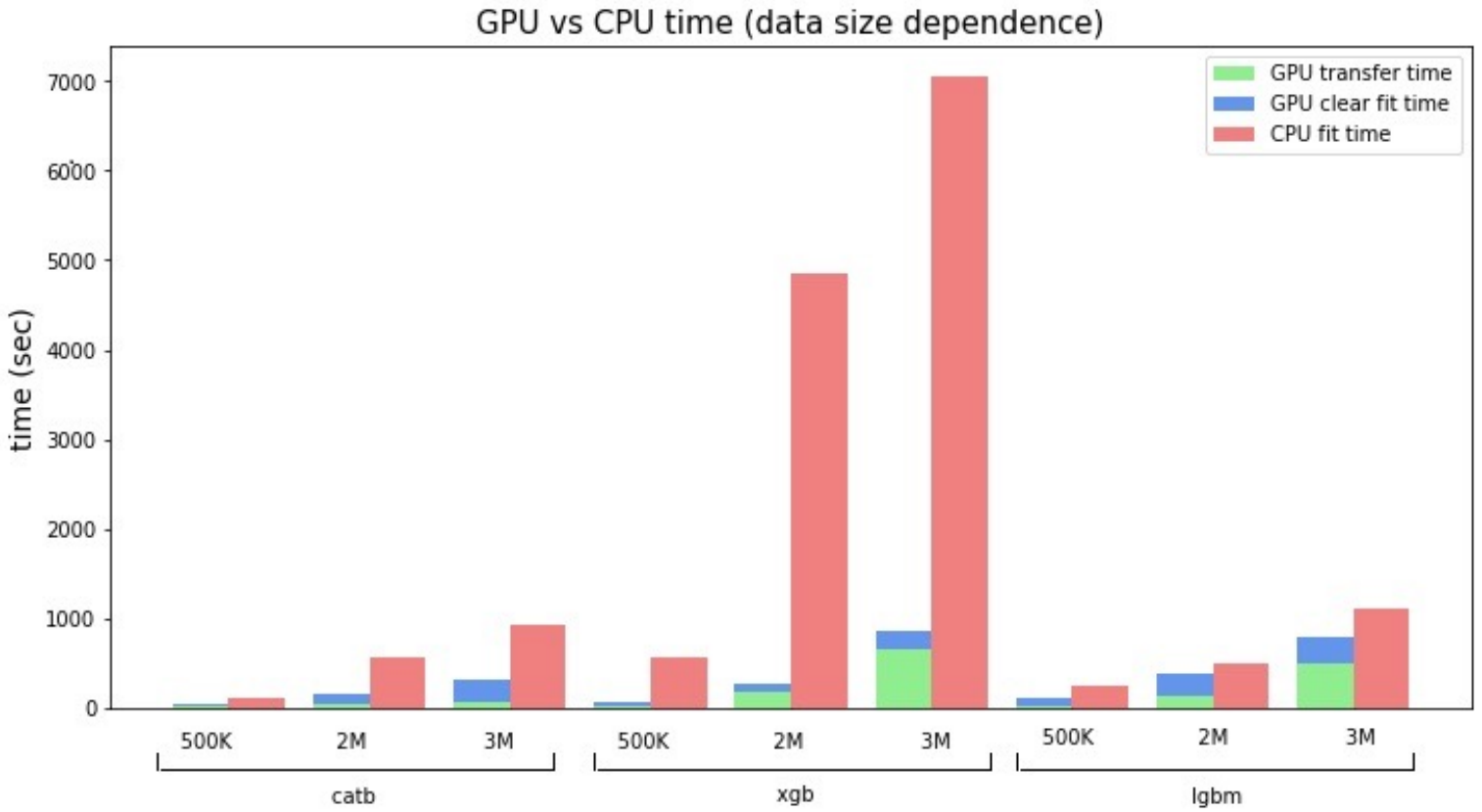
Время перекладки данных на GPU

- Catboost намного быстрее перекладывает данные, за счет чего выигрывает по параметру общего времени обучения на GPU



Время перекладки данных на GPU

- Три датасета: 500К x 300f, 2M x 300f, 3.2M x 600f
- Наибольший отрыв у GPU по-прежнему у xgboost



Выводы

- Метрики качества (CPU vs GPU) сильно отличаются у catboost и почти идентичны у xgboost и lightgbm
- Seed сильно влияет на дисперсию предиктов и метрик catboost CPU моделей и почти не влияет на xgboost и lightgbm –> seed лучше всегда фиксировать
- Распределения предиктов catboost CPU vs catboost GPU сильно отличаются, ни одна пара моделей не прошла теста на равенство распределений
- Catboost GPU модели зануляют от 1/2 до 2/3 признаков
- Catboost CPU модели могут очень отличаться по топу признаков
- Топ признаков между CPU и GPU моделями также может почти не совпадать – отбор фичей на GPU не лучшая идея
- Самые стабильные с точки зрения порядка признаков – модели lightgbm

Выводы

- Дефолтные параметры не отличаются у xgboost и lightgbm, но отличаются у catboost
- Из отличающихся признаков для качества модели важен только learning rate. Он слишком мал на GPU и модели недообучаются. Без учета этого факта подбор параметров на GPU – также не самая лучшая идея
- Лайфхак – взять learning_rate из обученной на CPU модели
- max_ctr_complexity сильно влияет как на размер моделей, так и на их качество
- если в датасете достаточное количество категорий, то дефолтный max_ctr_complexity = 4 не работает на CPU, но работает на GPU, приводя к сильной подстройке под категориальные признаки и худшему перфомансу
- На наших данных сравнять качество позволил только max_ctr_complexity = 1
- Catboost с большим отрывом выигрывает в скорости обучения и применения