**Описание и ход эксперимента 1**

Описание гиперпараметров: <https://xgboost.readthedocs.io/en/latest/parameter.html#parameters-for-tree-booster>

Для начала было принято решение сравнить несколько методов оптимизации гиперпараметров алгоритма XGBoost: поиск по сетке, случайный поиск и случайный поиск по группам. Для этого были выделены основные группы гиперпараметров каждого метода. Для эксперимента были взяты данные Facebosok Comment Volume Dataset [ <https://archive.ics.uci.edu/ml/datasets/Facebook+Comment+Volume+Dataset> ], на этих данных были проведены отбор и проектирование признаков.

Для проведения эксперимента было принято решение взять в рассмотрение основные гиперпараметры деревьев (max\_depth - максимальная глубина дерева, min\_child\_weight - минимальное количество экземпляров, которое должно быть в каждом узле), регуляризации (alpha - гиперпараметр регуляризации L1, lambda - гиперпараметр регуляризации L2), и гиперпараметры, отвечающие за подвыборки при построении деревьев (subsample - соотношение подвыборок для обучения деревьев; colsample\_bytree - процент подвыборки столбцов для построения дерева).

Для случайного поиска по группам гиперпараметры были разделены на 3 группы на основании статьи Complete Guide to Parameter Tuning in XGBoost with codes in Python [ <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/> ]:

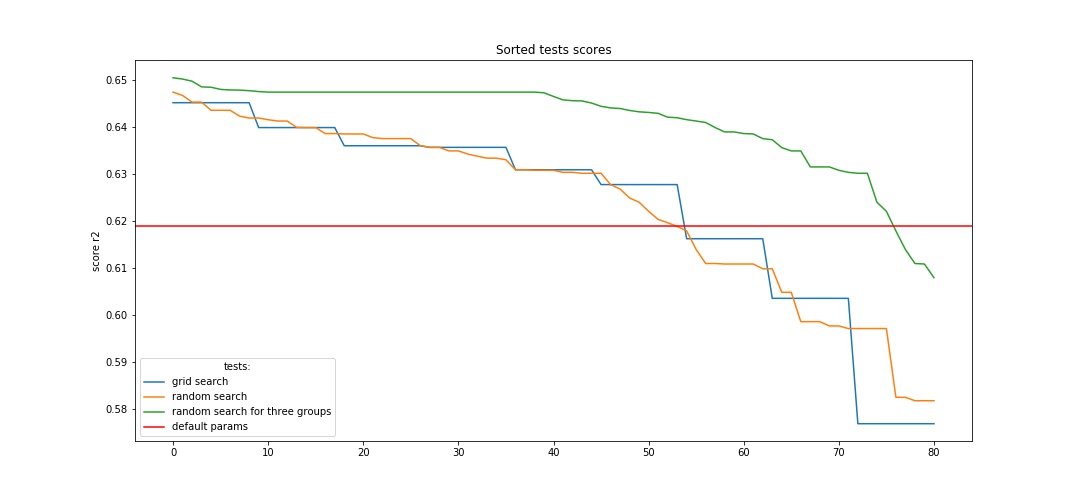
1. гиперпараметры деревьев (max\_depth, min\_child\_weight)
2. гиперпараметры регуляризации (alpha, lambda)
3. гиперпараметры подвыборок (subsample, colsample\_bytree)

Оптимизация происходит в рамках каждой группы, учитывая наилучший результат из предыдущей группы.

Для поиска по сетке было сделано 729 запусков для разных комбинаций гиперпараметров. Для случайного поиска и поиска по группам трижды были сделаны 729 запусков для оценки доверительных интервалов полученных результатов.

Также для определения эффективности оптимизаци гиперпараметров были вычислены результаты работы для XGBoost с гиперпараметрами по умолчанию.

В ходе выполнения эксперимента были получены следующие результаты:



**Рис. 1.** График оценки r2 для каждого запуска XGBoost при использовании разных методов оптимизации.

Для каждого набора гиперпараметров были сохранены результаты работы алгоритма. На рисунке 1 изображены графики оценок r2, отсортированных по убыванию для каждого метода. Очевидно, что в данном случае методы поиска применять целесообразно, потому что они дают большой прирост оценки r2 относительно гиперпараметров по умолчанию. Также можно заметить, что поиск по группам в среднем работает лучше, чем другие методы, графики оценок качества для случайного поиска и поиска по сетке выглядят очень похоже.

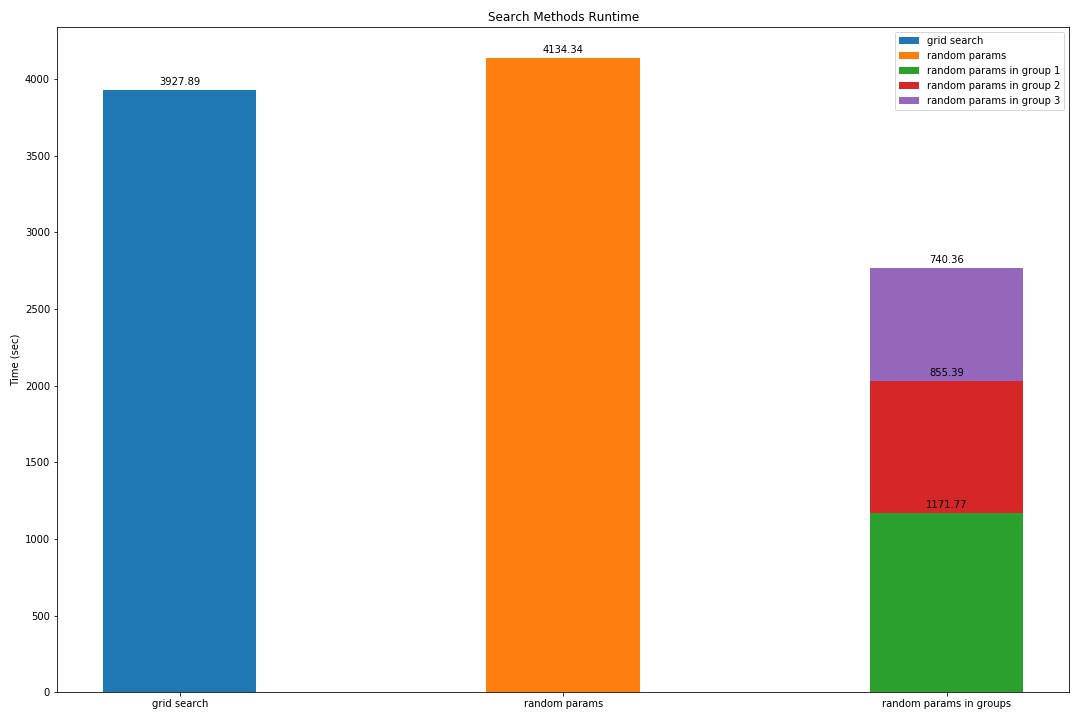
При этом для каждого эксперимента были достигнуты следующие наилучшие оценки r2 на следующих гиперпараметрах при указаных диапазонах поиска:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Максимально достигнутая оценка r2 | Гиперпараметры, на которых была достигнута эта оценка | Значения поиска |
| Параметры по-умолчанию | ~~0,619~~ | colsample\_bytree=1,  gamma=0,  max\_depth=3, min\_child\_weight=1,  reg\_alpha=0,  reg\_lambda=1,  subsample=1 | - |
| Поиск по сетке | ~~0,645~~ | ~~'max\_depth': 5, 'min\_child\_weight': 6,~~  ~~'alpha': 0.25,~~  ~~'lambda': 0.25,~~ | 'max\_depth' : [2, 5, 8], 'min\_child\_weight' :  [2, 5, 8],  'alpha' : [0.25,0.5,0.75], 'lambda' : [0.25,0.5,0.75],  'subsample' :  [0.625, 0.75, 0.875], 'colsample\_bytree' :  [0.625, 0.75, 0.875] |
| Случайный поиск | ~~0,647~~ | ~~'max\_depth': 4, 'min\_child\_weight': 5,~~  ~~'alpha': 0.576,~~  ~~'lambda': 0.783~~ | ‘max\_depth’:  sp\_randint(1, 10),  ‘min\_child\_weight’: sp\_randint(1, 10),  ‘alpha’:  uniform(loc=0, scale=1),  ‘lambda’:  uniform(loc=0, scale=1),  "subsample":  uniform(loc=0.5, scale=0.4), "colsample\_bytree":  uniform(loc=0.5, scale=0.4) |
| Поиск по группам | **~~Группа 1:~~**  ~~0,647~~  **~~Группа 2:~~**  ~~0,647~~  **~~Группа 3:~~**  ~~0,651~~ | ~~'max\_depth': 4, 'min\_child\_weight': 5~~  ~~'alpha': 0.245,~~  ~~'gamma': 0.096, 'lambda': 0.862~~  ~~'subsample': 0.826~~  ~~'colsample\_bytree': 0.766~~ | ‘max\_depth’: sp\_randint(1, 10),  ‘min\_child\_weight’: sp\_randint(1, 10)  ‘alpha’: uniform(loc=0, scale=1),  ‘lambda’: uniform(loc=0, scale=1)  ‘subsample’:  uniform(loc=0.5, scale=0.4),  ‘colsample\_bytree’:  uniform(loc=0.5, scale=0.4) |

**Табл. 1.** В таблице указаны результаты оптимизации для максимальной оценки r2 для каждого эксперимента. Если максимальная оценка была получена на нескольких наборах гиперпараметров. то в таблицу записывается набор, на котором максимальная оценка была получена раньше. Указаны значения только тех гиперпараметров, которые участвовали в оптимизации. Для поиска по сетке указываются значения гиперпараметров, а для случайного поиска и поиска по группам указаны распределения гиперпараметров, на которых выполнялся поиск. Округление значений произведено до тысячных.

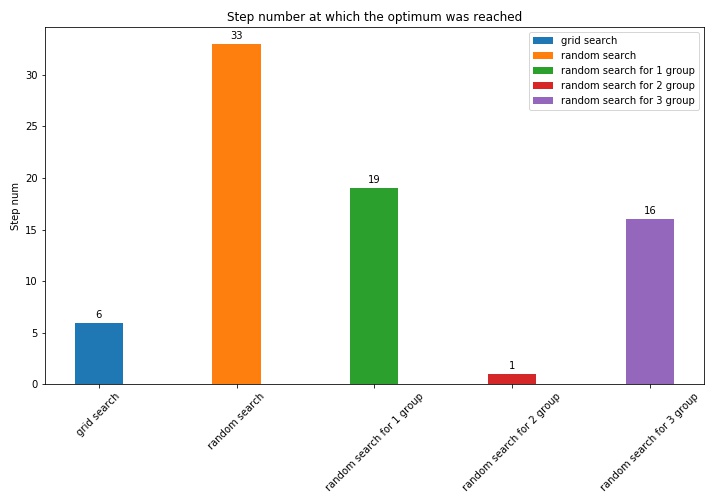
Из таблицы 1 видно, что поиск по группам достигает наилучшей оценки, причем оптимизация по второй группе не оказывает никакого эффекта на оценку.

Также виден эффект переобучения: поиск значения для max\_depth происходит в диапазоне от 3 до 10, а наилучшее значение достигается при значении 4.



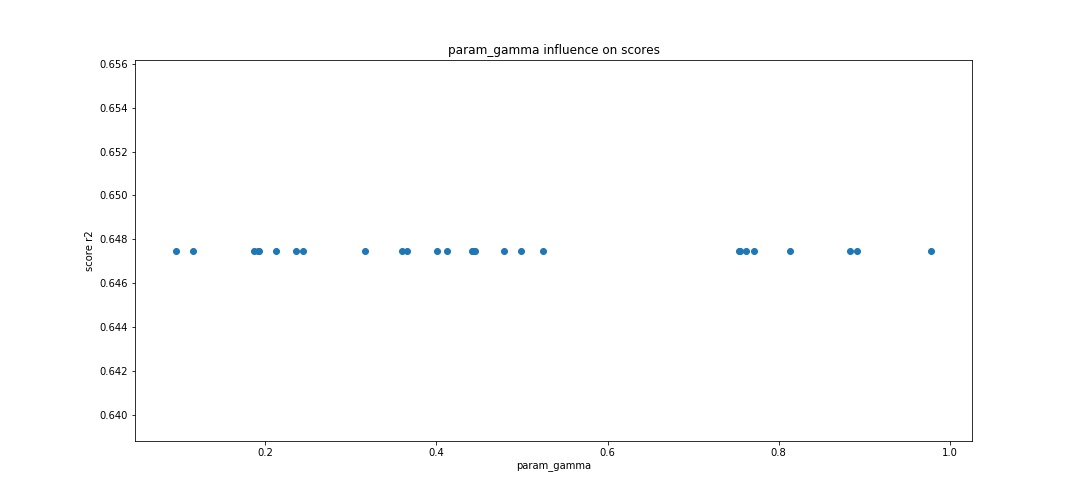
**Рис. 2.** График времени выполнения каждого метода оптимизации. Время указано в секундах.

На рисунке 2 можно увидеть количество времени, потраченного на 81 запуск для каждого метода. Для случайного поиска по группам было сделано по 27 запусков для каждой группы. Из графика на рисунке 2 видно, что случайный поиск по группам занял гораздо меньше времени, чем другие методы при том, что он достиг лучших результатов. Рассмотрим теперь, сколько итераций было сделано, чтобы получить эти результаты.



**Рис. 3.** График номера запуска, на котором был получен наилучший результат для каждого метода.

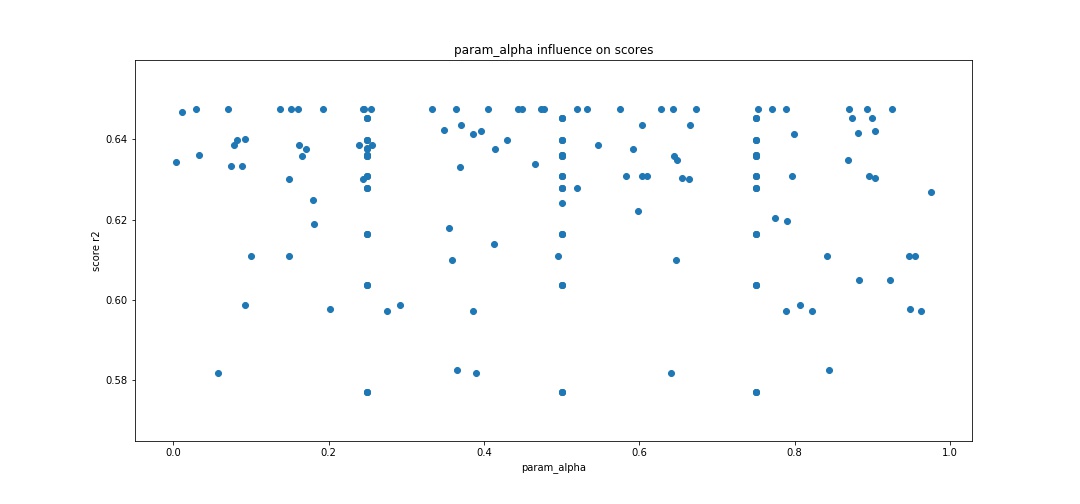
Из графика на рисунке 3 видно, что быстрее всего наилучший для поиска по сетке результат был достигнут уже на 6 итерации. Наилучший результат для случайного поиска по трем группам был получен уже на 70 наборе гиперпараметров, причем 27 итераций, сделанных на 2 группе гиперпараметров, не оказали никакого влияния на оценку качества.

После этого было рассмотрено влияние гиперпараметров на оценку качества работы алгоритма. Для этого были построены диаграммы рассеяния оценки r2 при разных значениях гиперпараметров.

**Рис. 4.** Диаграмма рассеяния оценки r2 относительно значений гиперпараметра gamma.

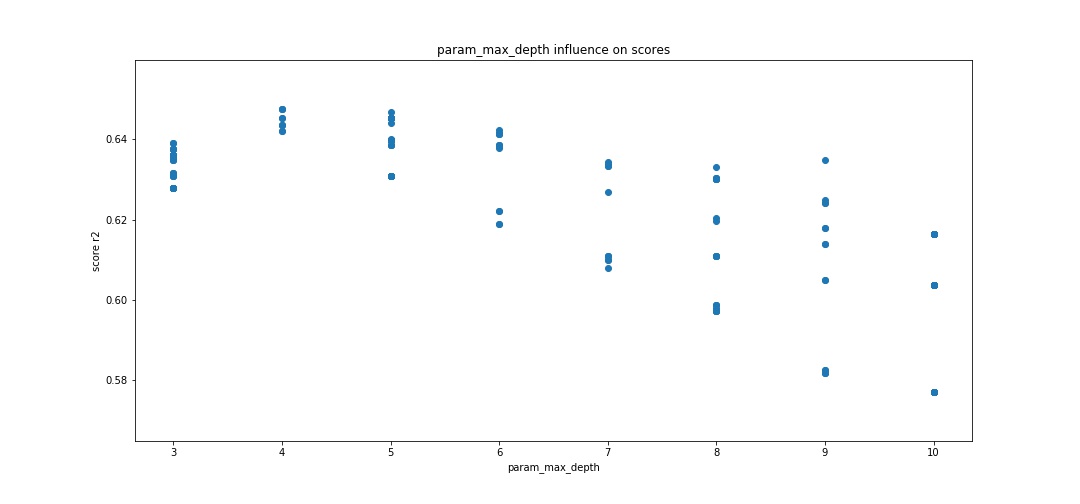
Из графика на рисунке 4, очевидно, что настройка гиперпараметра gamma почти не влияет на оценку качества в данном эксперименте, поэтому, возможно, что настройка этого гиперпараметра не так важна, как настройка других.

Параметры lambda и alpha имеет похожие друг на друга графики рассеяния оценки r2, поэтому здесь рассмотрен только график для гиперпараметра alpha.



**Рис. 5.** Диаграмма рассеяния оценки r2 относительно значений гиперпараметра alpha.

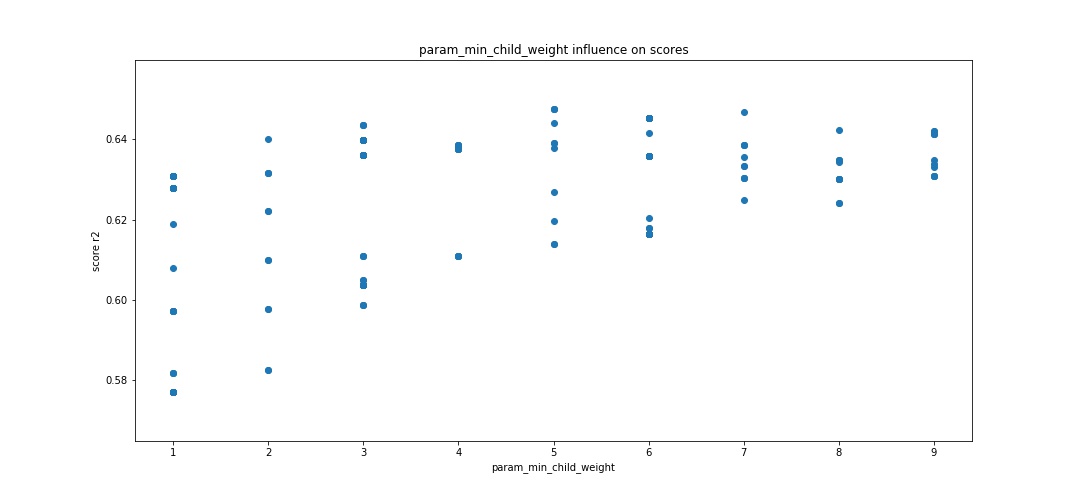
Очевидно, что на графике на рисунке 5 значение r2 имеет очень большой размах. Для поиска по сетке были выбраны значения 0,25, 0,5 и 0,75. Рассеяние r2 для этих значений выглядит примерно одинаково, значит в этой области значений гиперпараметра alpha, скорее всего основное влияние оказывают другие гиперпараметры. При этом можно заметить на графике точки, которые находятся в окрестности найденного оптимума, при любом значении гиперпараметра alpha из рассматриваемого диапазона. Это может означать, что основное влияние на оценку в данном эксперименте оказывают другие гиперпараметры. График для lambda выглядит похоже и все выводы для alpha справедливы и для lambda.



**Рис. 6.** Диаграмма рассеяния оценки r2 относительно значений гиперпараметра max\_depth.

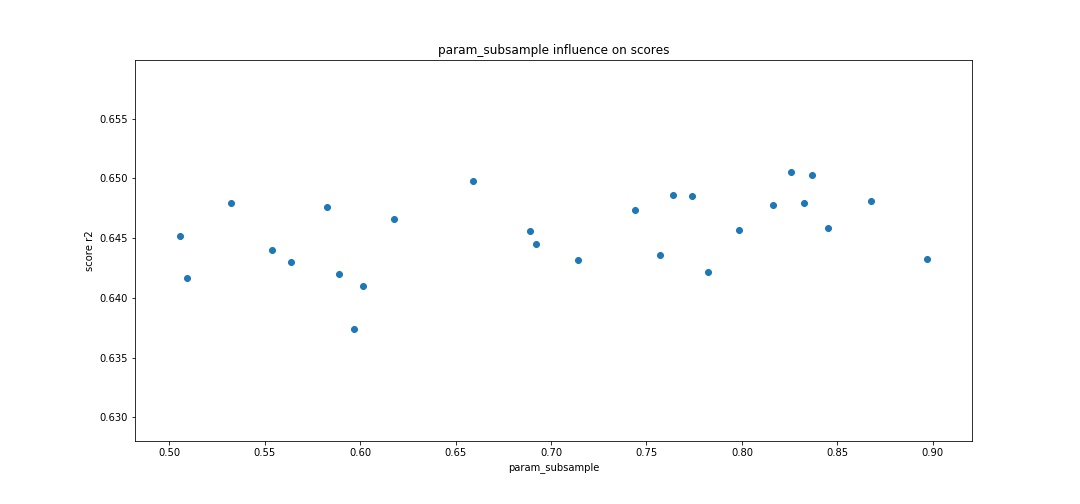
Если рассмотреть график для гиперпараметра max\_depth (рис. 6), то можно заметить описанный выше эффект переобучения: разброс оценки r2 увеличивается по мере увеличения значений гиперпараметра max\_depth и при этом верхняя граница разброса постепенно снижается после преодоления значения 4.

Схожий эффект наблюдается для гиперпараметра min\_child\_weight (рис. 7), при этом очевидно, что при переходе через наилучшее значение наблюдается сужение разброса оценки.

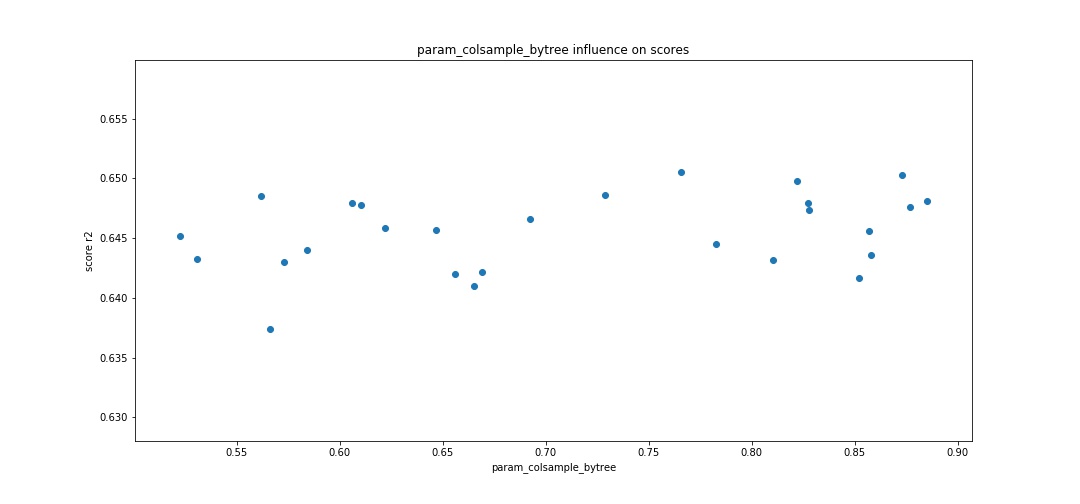


**Рис. 7.** Диаграмма рассеяния оценки r2 относительно значений гиперпараметра min\_child\_weight.

Если обратить внимание на графики рассеяния гиперпараметров subsample (рис. 8) и colsample bytree (рис. 9), то можно заметить, что графики напоминают полиномиальную кривую, что может сигнализировать о некой зависимости оценки от гиперпараметров.



**Рис. 8.** Диаграмма рассеяния оценки r2 относительно значений гиперпараметра subsample.



**Рис. 9.** Диаграмма рассеяния оценки r2 относительно значений гиперпараметра сolsample\_bytree.

Из всех этих наблюдений можно сделать следующие выводы для проведенного эксперимента:

* Случайный поиск по группам работает оптимальнее и быстрее с точки зрения затраченного времени, чем поиск по сетке и случайный поиск, но при этом наилучшего значения достигает довольно-таки поздно.
* Параметр gamma почти не оказывает влияние на оценку качества алгоритма.
* Вполне вероятно, что гиперпараметры alpha и lambda не оказывают сильного влияния на оценку качества работы алгоритма.
* Для гиперпараметров max\_depth и min\_child\_weight наблюдается эффект переобучения, это означает, что области поиска значений этих гиперпараметров в данном эксперименте были выбраны верно. Эти гиперпараметры однозначно стоит оптимизировать.
* Гиперпараметры subsample и colsample bytree стоит оптимизировать, исходя из структуры графиков рассеяния, на которых проглядывается зависимость оценки от значений гиперпараметров.