Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)
Факультет управления и прикладной математики
Кафедра «Интеллектуальные системы»
при Вычислительном центре им. А. А. Дородницына РАН

Гущин Александр Евгеньевич

Методы ансамблирования обучающихся алгоритмов

010900 — Прикладные математика и физика

Выпускная квалификационная работа бакалавра

Научный руководитель:
д.ф.-м.н.
Дьяконов Александр Геннадьевич

Москва
2015 г.

Содержание

Аннотация

В задачах обучения с учителем, в которых требование качества решения превалирует над ограничением его вычислительной сложности, применяются различные способы ансамблирования обучающихся алгоритмов. Одним из наиболее общих и эффективных в смысле достигаемого качества методов ансамблирования является стекинг, идея которого состоит в использовании предсказаний базовых алгоритмов в качестве признаков для некоторого метаалгоритма. Использование предложенных ранее модификаций стекинга для небольших по объему выборок имеет различные ограничения, по большей части связанные с недостаточно эффективным использованием обучающей выборки. В данной работе предлагается модификация стекинга, стремящаяся компенсировать эти недостатки.

1 Введение

При решении сложных задач классификации, регрессии, прогнозирования часто оказывается, что ни один из алгоритмов не обеспечивает желаемого качества восстановления зависимости. В таких случаях имеет смысл строить композиции алгоритмов, в которых ошибки отдельных алгоритмов взаимно компенсируются.

Наиболее общее определение алгоритмической композиции даётся в алгебраическом подходе Ю. И. Журавлёва [8]. Наряду с множеством объектов и множеством соответствующих им значений целевой функции вводится вспомогательное множество, называемое пространством оценок. Рассматриваются алгоритмы, в которых функция, называющаяся алгоритмическим оператором, устанавливает соответствие между множеством объектов и пространством оценок, а функция, называющаяся решающим правилом, устанавливает соответствие между пространством оценок и множеством значений целевой функции. Таким образом, рассматриваемые алгоритмы имеют вид суперпозиции алгоритмического оператора и решающего правила. Многие алгоритмы классификации имеют именно такую структуру: сначала вычисляются оценки принадлежности объекта классам, затем решающее правило переводит эти оценки в номер класса. Значением оценки может быть вероятность принадлежности объекта классу, расстояние от объекта до разделяющей поверхности, степень уверенности классификации и т. п.

Существует несколько наиболее известных методов объединения базовых алгоритмов в композиции: голосование, взвешенное голосование, смесь экспертов (Mixture of Experts [19]). Эти методы часто применяются, когда базовые алгоритмы существенно отличаются друг от друга.

В случаях, когда необходимо построить композицию используя один базовый алгоритм, широко применяется бэггинг (bagging, boostrap aggregation) [4]. Идея бэггинга состоит в том, что базовый алгоритм многократно обучается на случайных подвыборках с повторениями из обучающей выборки. Такой метод генерации подвыборок принято называть бутсрап (bootsrap).

Похожим на бэггинг методом является метод случайных подпространств (random subspace method, RSM [10]). Его идея заключается в создании вариативности при обучении с помощью выбора случайных подмножеств признаков. Широко известным примером использования бэггинга и RSM является случайный лес [5].

Другим известным способом ансамблирования базовых алгоритмов является бустинг. Идея бустинга состоит в жадном выборе очередного алгоритма для добавления в композицию так, чтобы он лучшим образом компенсировал имеющиеся на этом шаге ошибки. Широко известные примеры бустинга — AdaBoost [6] и градиентный бустинг (Gradient boosting [7]).

Стекинг (Stacked generalization) был впервые предложен Д. Волпертом в 1992 году в работе [2] в достаточно общем виде. Основная идея стекинга заключается в использовании базовых классификаторов для получения предсказаний (метапризнаков) и использовании их как признаков для некоторого "обобщающего" алгоритма (метаалгоритма). Иными словами, основной идеей стекинга является преобразование исходного пространства признаков задачи в новое пространство, точками которого являются предсказания базовых алгоритмов. Предлагается сначала выбрать набор пар произвольных подмножеств из обучающей выборки, затем для каждой пары обучить базовые алгоритмы на первом подмножестве и предсказать ими целевую переменную для второго подмножества. Предсказанные значения и становятся объектами нового пространства. В частности, автором рассматривается случай выбора всевозможнных пар подмножеств, в которых второе подмножество состоит из единственного объекта, а первое множество из всей обучающей выборки кроме этого объекта (leave-one-out). Очевидно, что такой способ позволяет перевести каждую точку исходного пространства признаков в точку нового пространства. Автор обобщает идею стекинга тем, что предлагает, обучая базовые классификаторы (первого уровня) над метапризнаками (первого уровня), получать метапризнаки второго уровня и так далее.

Развитие идеи стекинга для задач многоклассовой классификации происходит в работах [17], [16]. В первой работе предлагается свести задачу мноклассовой классификации к набору задач одноклассовой классификации, которые решают базовые классификаторы, и обучения метаклассификатора над их предсказаниями. Во второй работе предлагается усложнить эту схему с помощью использования дополнительного уровня базовых классификаторов, построенных на предсказаниях базовых классификаторов первого уровня.

Стекинг являлся основным способом ансамблирования базовых алгоритмов команд-победителей известного соревнования по машинному обучению Netflix [12]. Командой BigChaos [15] было использовано разбиение обучающей выборки на K непересекающихся блоков (K-fold). В этом случае формирование метапризнаков осуществляется с помощью всевозможных пар подмножеств, где вторым подмножеством является один из блоков, а первым - все оставшиеся блоки. Очевидно, что использование такого разбиения позволяет породить метапризнаки для всех объектов в обучающей выборке. Недостатком такого подхода является то, что распределение значений метапризнаков для разных блоков будут различаться из-за того, что для обучения соответствующих базовых классификаторов обучающие выборки (первое подмножество в паре) будут отличаться друг от друга. В этом же соревновании был успешно использован другой вариант стекинга, названный блендингом (blending, stacking ensembling) [13]. Он заключается в выполнении всего одной пары разбиений обучающей выборки (hold-out). Преимуществами такого подхода являются меньшая вычислительная сложность решения и отсутствие риска переобучения, недостатком - неэффективное использование обучающей выборки и необходимость подбора параметров разбиения.

Другими участниками этого же соревнования был предложен Feature-Weighted Linear Stacking [14], идея которого заключается в том, чтобы настраивать веса взвешенной суммы ответов базовых алгоритмов как линейную комбинацию нескольких специально отобранных признаков.

Оригинальным использованием стекинга в нейронных сетях является архитектура Deep Stacking Network, предложенная в работе [18]. Предлагается одновременно использовать упорядоченный набор однослойных нейронных сетей в качестве базовых алгоритмов. Каждая сеть в наборе обучается на исходных признаках и на метапризнаках, предсказанных всеми предыдущими сетями, таким образом образуя в совокупности глубокую архитектуру.

В данной работе предлагается модификация идеи стекинга, которую можно рассматривать как частный случай предложенного Д. Волпертом общего подхода к формированию обучающей выборки с помощью базовых алгоритмов. Модернизация заключается в том, чтобы получить метапризнак усреднением предсказаний, полученных с помощью различных разбиений на K непересекающихся блоков (K-fold). На реальных данных продемонстрировано, что полученные с усреднением метапризнаки лучше метапризнаков без усреднений в смысле получаемого метаклассификатором качества.

2 Постановка задачи и используемые обозначения

Пусть $\mathcal{X}$ — множество описаний объектов, $\mathcal{Y}$ — множество ответов, и существует неизвестная "целевая зависимость" — отображение $f: \mathcal{X} \to \mathcal{Y}$ значения которой известны только на объектах конечной обучающей выборки $(X, Y) = \{(x_1, y_1), \ldots, (x_{|(X,Y)|}, y_{|(X,Y)|})\}$. Требуется построить алгоритм $a: \mathcal{X} \to \mathcal{Y}$, аппроксимирующий целевую зависимость на всем множестве $\mathcal{X}$.

Наряду с множествами $\mathcal{X}$ и $\mathcal{Y}$ вводится вспомогательное множество R, называемое пространством оценок. Рассматриваются алгоритмы, имеющие вид суперпозиции $a(x) = C(b(x))$, где функция $b: \mathcal{X} \to R$ называется алгоритмическим оператором, функция $C: R \to \mathcal{Y}$ — решающим правилом. Многие алгоритмы классификации имеют именно такую структуру: сначала вычисляются оценки принадлежности объекта классам, затем решающее правило переводит эти оценки в номер класса. Значением оценки $b(x)$ может быть вероятность принадлежности объекта x классу, расстояние от объекта до разделяющей поверхности, степень уверенности классификации и т. п.

Определение 1 *Композицией T алгоритмов $a_t(x) = C(b_t(x)), t = 1, \ldots, T$ называется суперпозиция алгоритмических операторов $b_t: \mathcal{X} \to R$, корректирующей операции $F: R^T \to R$ и решающего правила $C: R \to \mathcal{Y}$:*

$$a(x) = C(F(b_1(x), \ldots, b_T(x)), x \in X$$

Алгоритмы a_t, а иногда и операторы b_t, называют базовыми алгоритмами.

Суперпозиции вида $F(b_1, \ldots, b_T)$ являются отображениями из $\mathcal{X}$ в R, то есть, опять-таки, алгоритмическими операторами. Это позволяет строить иерархические композиции, применяя определение 1 рекурсивно.

Обозначим множество базовых классификаторов как $\mathcal{A}$. Используя сформулированные выше термины, идеей стекинга является использование в качестве алгоритмических операторов b_t базовых классификаторов $A \in \mathcal{A}$, а в качестве корректирующей операции F некоторого метаклассификатора M.

3 Обзор известных методов ансамблирования

3.1 Голосование и смесь экспертов

Существует несколько наиболее известных корректирующих операций:

- простое голосование (Simple Voting):

$$b(x) = F(b_1(x), \ldots, b_T(x)) = \frac{1}{T}\sum_{t=1}^{T} b_t(x);$$

- взвешенное голосование (Weighted Voting):

$$b(x) = F(b_1(x), \ldots, b_T(x)) = \sum_{t=1}^{T} w_t b_t(x),$$

$$\sum_{t=1}^{T} w_t = 1,\ w_t \geqslant 0;$$

- смесь экспертов (Mixture of Experts [19]):

$$b(x) = F(b_1(x), \ldots, b_T(x)) = \sum_{t=1}^{T} w_t(x) b_t(x),$$

$$\sum_{t=1}^{T} w_t(x) = 1,\ \forall x \in X.$$

Очевидным является факт, что простое голосование — это лишь частный случай взвешенного голосования, а взвешенное голосование является частным случаем смеси экспертов. Также стоит отметить, что из-за наличия большого числа степеней свободы обучение смеси экспертов происходит значительно дольше других алгоритмов построения композиции, поэтому их практическая применимость обоснована только в случае априорной информации о функциях компетенции $g_t(x)$, которые чаще всего определяются:

- признаком $f(x)$:

$$w_t(x; \alpha, \beta) = \sigma(\alpha f(x) + \beta),\ \alpha, \beta \in \mathbb{R};$$

- направлением $\alpha \in \mathbb{R}^n$:

$$w_t(x; \alpha, \beta) = \sigma(x^T \alpha + \beta),\ \alpha \in \mathbb{R}^n, \beta \in \mathbb{R};$$

- расстоянием до $\alpha \in \mathbb{R}^n$

$$w_t(x; \alpha, \beta) = \exp(-\beta||x - \alpha||^2),\ \alpha \in \mathbb{R}^n, \beta \in \mathbb{R};$$

- более сложными способами (при помощи оценки плотности распределения данных и т.п.).

В приведенных выше формулах функций компетенции $\sigma(z) = \dfrac{1}{1 + e^{-z}}$ — сигмоида.

3.2 Бустинг

Рассмотрим задачу классификации на два класса, $Y = \{-1, +1\}$. Допустим, что решающее правило фиксировано, $C(b) = sign(b)$, базовые алгоритмы возвращают ответы -1, 0, +1. Ответ $b_t(x) = 0$ означает, что базовый алгоритм b_t отказывается от классификации объекта x, и ответ $b_t(x)$ не учитывается в композиции. Искомая алгоритмическая композиция имеет вид:

$$a(x) = C(F(b_1(x), \ldots, b_T(x)) = sign(\sum_{t=1}^{T} \alpha_t b_t P_n), x \in \mathcal{X}$$

Пусть теперь при добавлении в композицию слагаемого $\alpha_t b_t(x)$ оптимизируется только базовый алгоритм b_t и коэффициент при нём α_t, а все предыдущие слагаемые $\alpha_1 b_1(x), \ldots, \alpha_{t-1} b_{t-1}(x)$ полагаются фиксированными.

Определим функционал качества композиции как число ошибок, допускаемых ею на обучающей выборке:

$$Q_t = \sum_{i=1}^{l} [y_i \sum_{t=1}^{T} \alpha_t b_t(x_i) < 0]$$

Пороговая функция потерь в функционале Q_t аппроксимируется (заменяется) непрерывно дифференцируемой оценкой сверху.

Аппроксимируя функцию потерь экспонентой, получим AdaBoost [6].

3.3 Бэггинг

Метод бэггинга (bagging, bootstrap aggregation) был предложен Л. Брейманом в 1996 году [4]. Из исходной обучающей выборки длины l формируются различные обучающие подвыборки той же длины l с помощью бутстрепа — случайного выбора с возвращениями. При этом некоторые объекты попадают в подвыборку по несколько раз, некоторые — ни разу. Можно показать, что доля объектов, оказавшихся в каждой подвыборке, стремится к $1 - e^{-1} \approx 0.632$ при $l \to \inf$. Базовые алгоритмы, обученные по подвыборкам, объединяются в композицию с помощью простого голосования.

Эффективность бэггинга объясняется двумя обстоятельствами. Во-первых, благодаря различности базовых алгоритмов, их ошибки взаимно компенсируются при голосовании. Во-вторых, объекты-выбросы могут не попадать в некоторые обучающие подвыборки. Тогда алгоритм, построенный по подвыборке, может оказаться

даже точнее алгоритма, построенного по полной выборке. Бэггинг особенно эффективен на малых выборках, когда исключение даже небольшой доли обучающих объектов приводит к построению существенно различных базовых алгоритмов. В случае сверхбольших избыточных выборок приходится строить подвыборки меньшей длины $l_0 \ll l$, при этом возникает задача подбора оптимального значения l_0.

3.4 Метод случайных подпространств

В методе случайных подпространств (random subspace method, RSM) базовые алгоритмы обучаются на различных подмножествах признакового описания, которые также выделяются случайным образом [10]. Этот метод предпочтителен в задачах с большим числом признаков и относительно небольшим числом объектов, а также при наличии избыточных неинформативных признаков. В этих случаях алгоритмы, построенные по части признакового описания, могут обладать лучшей обобщающей способностью по сравнению с алгоритмами, построенными по всем признакам.

Широко известным алгоритмом, использующим одновременно метод случайных подпространств и бэггинг, является случайный лес. Разбиение объектов в вершине случайного леса ищется среди случайного подмножества признаков, а обучение каждого дерева в композиции происходит на выборке, полученной с помощью операции бутстрапа.

3.5 Стекинг

Для определенности будем рассматривать задачу классификации (задача регрессии рассматривается аналогично). Постановка задачи классификации звучит следующим образом.

Пусть $\mathcal{X}$ — множество описаний объектов, $\mathcal{Y}$ — конечное множество номеров (имён, меток) классов. Существует неизвестная "целевая зависимость" — отображение $f: \mathcal{X} \to \mathcal{Y}$ значения которой известны только на объектах конечной обучающей выборки $(X, Y) = \{(x_1, y_1), \ldots, (x_{|(X,Y)|}, y_{|(X,Y)|})\}$. Требуется построить алгоритм $a: \mathcal{X} \to \mathcal{Y}$, способный классифицировать произвольный объект $x \in \mathcal{X}$.

Для начала введем дополнительные обозначения:

(X_0, Y_0) — валидационная выборка;

A — базовый классификатор, использующийся для построения метапризнака;

$A.\text{fit}(X, Y)$ — функция обучения классификатора A на (X, Y);

$A.\text{predict}(X)$ — функция, предсказывающая целевую переменную для X классификатором A;

M — некоторый метаклассификатор;

$MF(X, A)$ — метапризнак, полученный классификатором A для выборки X;

P — финальное предсказание стекинга для валидационной выборки;

$\text{concatV}(X_i, X_j)$ — операция конкатенирования X_i и X_j по столбцам;

$\text{concatH}(X_i, X_j)$ — операция конкатенирования X_i и X_j по строкам.

Идея стэкинга состоит в том, чтобы обучить метаклассификатор M на (1) исходных признаках, матрице X, и (2) на предсказаниях (метапризнаках), полученных с помощью базовых классификаторов (для простоты здесь мы часто будем рассматри-

вать единственный классификатор A). Метапризнаки, полученные с помощью классификатора A для выборки X будем обозначать $MF(X, A)$. Принятым также является представление стекинга в виде многоуровневой схемы, где признаки обозначаются как "уровень 0", метапризнаки, полученные с помощью обучения базовых классификаторов на признаках, как "уровень 1", и так далее.

В простейшем виде [13] получение предсказания для тестовой выборки P с помощью стекинга выглядит следующим образом (Рисунок 1):

Алгоритм 1

разбить обучающую выборку (X, Y) на две части: (X_1, Y_1) и (X_2, Y_2).

A.fit(X_1,Y_1)
$MF(X_2, A) :=$ A.predict(X_2)
$MF(X_0, A) :=$ A.predict(X_0)

M.fit(concatV($X_2, MF(X_2, A)$), Y_2)
$P := M$.predict(concatV($X_0, MF(X_0, A)$))

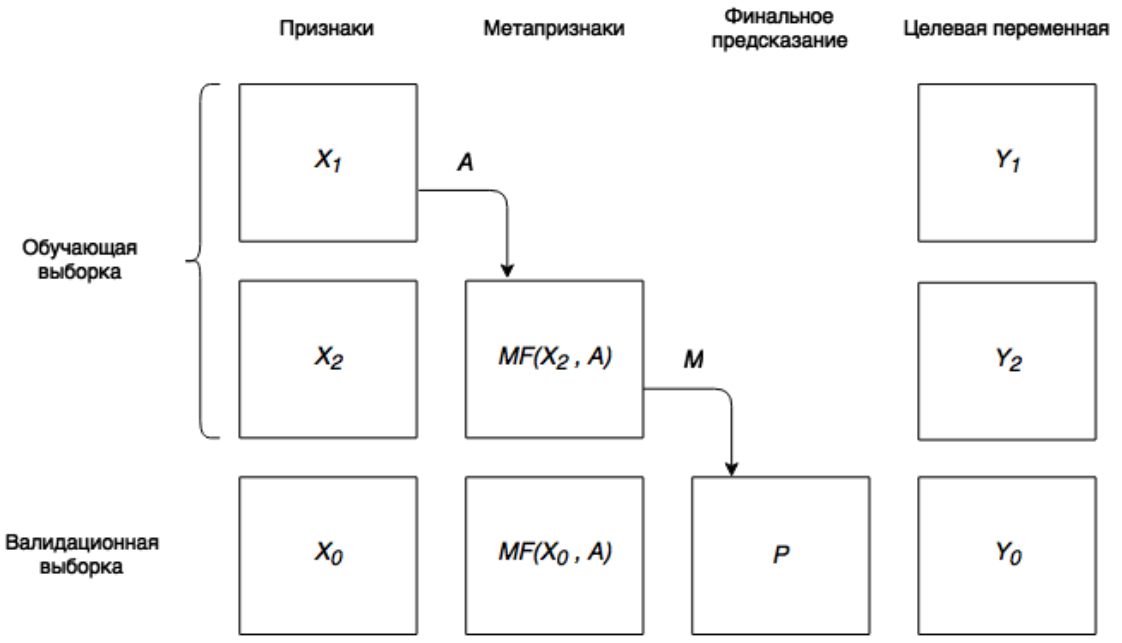

Рис. 1: Алгоритм 1, стекинг по схеме hold-out

Недостатком этого варианта стекинга является то, что M обучается только на части X_2 обучающей выборки, а другая часть X_1 им не используется. Чтобы избежать этого, мы можем повторить алгоритм, поменяв местами (X_1, Y_1) и (X_2, Y_2). В таком случае мы получим два предсказания для валидационной выборки, которые мы можем усреднить. Более эффективным в смысле достижимого качества решением этой проблемы является следующая идея: мы можем повторить **Алгоритм 1**,

каждый раз используя различные разбиения и затем усреднить полученные предсказания:

Алгоритм 2

сделать разбиения $\{\{(X_{n1}, Y_{n1}), (X_{n2}, Y_{n2})\}, n = 1 \dots N\}$

для n=1...N
$A_n := A$
$A_n.\text{fit}(X_{n1}, Y_{n1})$
$MF(X_{n2}, A_n) := A_n.\text{predict}(X_{n2})$
$MF(X_0, A_n) := A_n.\text{predict}(X_0)$
$M_n := M$
$M_n.\text{fit}(\text{concatV}(X_{n2}, MF(X_{n2}, A_n)), Y_2)$
$P_n := M_n.\text{predict}(\text{concatV}(X_0, MF(X_0, A_n)))$
$P := \frac{1}{N} \sum_{n=1}^{N} P_n$

Можно сделать **Алгоритм 2** более эффективным, если осуществлять разбиения по принципу бутстрапа. Логика для использования такого подхода следующая: мы знаем, что эффективно использовать бэггинг для классификаторов. Однако, мы также можем использовать объекты, которые при этом не попадают в обучающую выборку для каждого классификатора. Таким образом, при построении очередного классификатора в бэггинге из обучающей выборки формируется выборка X_2 той же длины с возвращениями. Все объекты, которые не попали в эту обучающую выборку, помещаются в множество X_1 [11].

Отметим, что **Алгоритм 1** и **Алгоритм 2** с легкостью модернизировать для любого множества базовых классификаторов $\mathcal{A}$.

Однако, в любом случае описанные выше алгоритмы не решают проблемы уменьшения размера обучающей выборки с каждым следующим уровнем стекинга, а также возникающей дополнительной вычислительной сложности при повторном обучении метаклассификатора N раз при добавлении нового базового алгоритма в множество $\mathcal{A}$.

Очевидным способом решения возникшей проблемы является получение метапризнака $MF(X, A)$ для всей обучающей выборки [3]:

Видно, что описанный алгоритм (как и те, которые будут описаны ниже) на самом деле состоит из двух частей: получения метапризнака $MF(X, A)$ и использования метаклассификатора M для осуществления предсказания P для валидационной выборки.

Поскольку для получения мета-признака мы обучали классификатор A только на признаках, будем называть пару $MF(X, A)$, $MF(X_0, A)$ метапризнаком первого уровня.

Преимуществом этого способа является также и то, что мы можем включить полученный метапризнак в обучающую выборку в качестве обычного признака. По-

Алгоритм 3

разбить обучающую выборку (X, Y) на две части: (X_1, Y_1) и (X_2, Y_2).

$A_1 := A$
$A_1.\text{fit}(X_1, Y_1)$
$MF(X_2, A_1) := A_1.\text{predict}(X_2)$
$MF(X_0, A_1) := A_1.\text{predict}(X_0)$

$A_2 := A$
$A_2.\text{fit}(X_2, Y_2)$
$MF(X_1, A_2) := A_2.\text{predict}(X_1)$
$MF(X_0, A_2) := A_2.\text{predict}(X_0)$

$MF(X, A) := \text{concatH}(MF(X_1, A_2), MF(X_2, A_1))$
$MF(X_0, A) := (MF(X_0, A_1) + MF(X_0, A_2)) \ / \ 2$

$M.\text{fit}(\text{concatV}(X, MF(X, A)), Y)$
$P := M.\text{predict}(\text{concatV}(X_0, MF(X_0, A)))$

вторив **Алгоритм 3** над такой выборкой, мы получим мета-признак второго уровня, и так далее.

Другой логичной модернизацией идеи стекинга может стать число разбиений обучающей выборки на первом шаге. Модифицируем последний алгоритм (Рисунок 2):

Алгоритм 4

сделать разбиения $\{(X_k, Y_k), k = 1 \dots K\}$

для k=1...K
$A_k := A$
$A_k.\text{fit}(X \setminus X_k, Y \setminus Y_k)$
$MF(X_k, A_k) := A_k.\text{predict}(X_k)$
$MF(X_0, A_n) := A_k.\text{predict}(X_0)$

$MF(X, A) := \text{concatH}(MF(X_k, A_k), k = 1 \dots K)$
$MF(X_0, A) := \frac{1}{K} \sum_{k=1}^{K} MF(X_0, A_k)$

$M.\text{fit}(\text{concatV}(X, MF(X, A)), Y)$
$P := M.\text{predict}(\text{concatV}(X_0, MF(X_0, A)))$

Назовём такой способ получения метапризнаков out-of-fold(K) или, для краткости, oof(K).

При $K = |X|$ получаем способ, изначально рассмотренный Д. Волпертом в его статье [2] (leave-one-out).

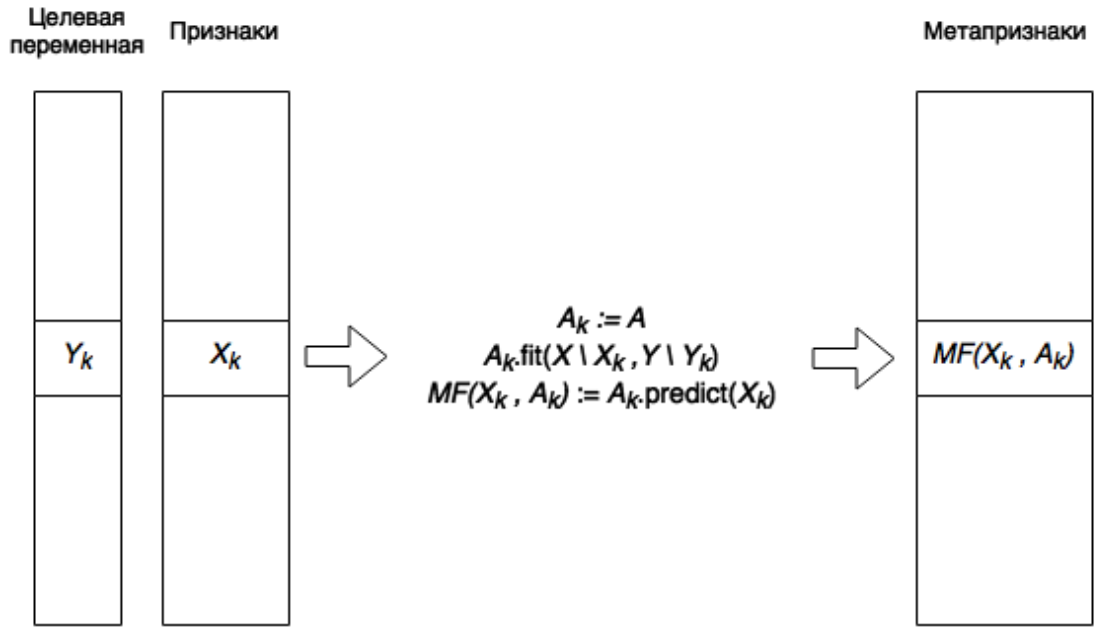

Рис. 2: Алгоритм 4, создание метапризнака по схеме K-fold

При $K \ll |X|$ осуществлять разбиения представляется логичным таким образом, чтобы в каждой части оказалось равное количество объектов, а также, чтобы для каждого фиксированного класса доля объектов в каждой из K частей была одинакова. Такой способ принято называть разбиением на K блоков со стратификацией классов (Stratified K-fold). В дальнейшем будем предполагать именно его.

Таким образом, oof(K) позволяет метаклассификатору M использовать для обучения всю выборку X. Однако, у него имеется следующий недостаток: полученные распределения значений метапризнаков для обучающей $MF(X, A)$ и валидационной выборки $MF(X_0, A)$ часто оказываются различны в силу того, что метапризнак для валидационной выборки был получен усреднением K предсказаний. Очевидно, что это приводит к ухудшению качества финального предсказания P.

В качестве простой иллюстрации такой возможности рассмотрим следующий пример.

Пример 1

Рассматривается задача двухклассовой классификации. A — классификатор, предсказывающий класс объекта как класс его ближайшего соседа из обучающей выборки. В этом случае $MF(X, A)$ может иметь только значения $\{0, 1\}$, в отличии от $MF(X_0, A)$, которая, в результате усреднения K предсказаний, может иметь значения $\{\frac{k}{K}, k = 0 \dots K\}$.

Чтобы избежать этого, мы могли бы предсказывать $MF(X, A)$ с помощью одного классификатора, обученного на всей обучающей выборке. Модернизированный алгоритм:

Предложенный последним алгоритм будем обозначать oof(K, test_averaging=False), а предыдущий, соответственно, oof(K, test_averaging=True).

Однако, и oof(K, test_averaging=False) обладает своим недостатком: $MF(X_0, A)$

Алгоритм 5

сделать разбиения $\{(X_k, Y_k), k = 1 \dots K\}$

для k=1...K
  $A_k := A$
  A_k.fit$(X \setminus X_k, Y \setminus Y_k)$
  $MF(X_k, A_k) := A_k$.predict(X_k)

$MF(X, A) = \text{concatH}(MF(X_k, A_k), k = 1 \dots K)$

$A_0 := A$
A_0.fit(X, Y)
$MF(X_0, A) := A_0$.predict(X_0)

M.fit$(\text{concatV}(X, MF(X, A)), Y)$
$P := M$.predict$(\text{concatV}(X_0, MF(X_0, A)))$

может существенно отличаться от $MF(X, A)$, поскольку для объектов из разных блоков обучение базовых классификаторов происходило на различных подмножествах обучающей выборки.

В работе предлагается уменьшить отрицательный эффект этого с помощью внесения следующей модернизации в идею стекинга out-of-fold(K): классификатором A получить один и тот же метапризнак для различных разбиений обучающей выборки, затем усреднить. Выдвигается гипотеза, что достаточное количество усреднений позволяет уменьшить отрицательный эффект описанной выше вариативности значений метапризнаков. Модифицированный алгоритм:

Если результаты классификатора A при фиксированной обучающей выборке фиксированны, то есть не зависят от некоторых стохастических параметров, разумным упрощением то $N_{modified} = 1$, иначе $N_{modified} = \text{N}$.

Будем обозначать такой способ получения метапризнаков out-of-fold(K)*N или, для краткости, oof(K)*N. Очевидно, что oof(K) является частным случаем oof(K)*N.

Также отметим, что при получении $MF(X, \mathcal{A})$ фиксированное для всех классификаторов разбиение обучающей выборки на $\{(X_k, Y_k), k = 1 \dots K\}$ является более предпочтительным, чем различные разбиения для разных классификаторов, поскольку в таком случае для фиксированного объекта из обучающей выборки значения метапризнаков от различных классификаторов будут более похожи друг на друга в силу того, что классификаторы $A \in \mathcal{A}$ будут обучены на одной и той же части обучающей выборки.

Таким же образом, при получении метапризнаков oof(K)*N с помощью различных классификаторов логичным представляется фиксированный набор разбиений $\{\{(X_{nk}, Y_{nk}), k = 1 \dots K\}, n = 1 \dots N\}$, что и будет использовано в данной работе.

Алгоритм 6

зафиксировать разбиения $\{\{(X_{nk}, Y_{nk}), k = 1 \dots K\}, n = 1 \dots N\}$

для n=1...N
  для k=1...K
    $A_{nk} := A$
    A_{nk}.fit$(X \setminus X_{nk}, Y \setminus Y_{nk})$
    $MF(X_{nk}, A_{nk}) := A_{nk}$.predict$(X_{nk})$
    $MF(X_0, A_{nk}) := A_{nk}$.predict$(X_0)$
  $MF(X_n, A_n) =$ concatH$(\{MF(X_{nk}, A_{nk}), k = 1 \dots K\})$
  $MF(X, A_n) := \frac{1}{K} \sum_{k=1}^{K} MF(X_0, A_{nk})$

если test_averaging==True **то**
  $MF(X_0, A) := \frac{1}{N} \sum_{n=1}^{N} MF(X_0, A_n)$
иначе
  для n=1...$N_{modified}$
    $A_{0n} := A$
    A_{0n}.fit(X, Y)
    $MF(X_0, A_{0n}) = A_{0n}$.predict(X_0)
  $MF(X_0, A) := \frac{1}{N_{modified}} \sum_{n=1}^{N_{modified}} MF(X_0, A_{0n})$

M.fit(concatV$(X, MF(X, A)), Y)$
$P := M$.predict(concatV$(X_0, MF(X_0, A)))$

4 Экспериментальная часть

4.1 Необходимые пояснения

Для удобства описания результатов экспериментов введём следующие обозначения:

- Будем называть качеством (quality) метапризнака качество метаклассификатора, обученного на признаках и этом метапризнаке, на валидационной выборке.
- Будем называть непосредственным качеством (direct quality) метапризнака $MF(X, A)$ качество, которое мы получаем, предсказывая целевую переменную непосредственно этим метапризнаком. Вид рассматриваемого функционала качества совпадает с функционалом, который оптимизирует классификатор A.

Поскольку качество метапризнака может быть не связано с его непосредственным качеством, предлагается сравнивать метапризнаки по результатам метаклассификаторов, построенных на них.

Ясно, что для oof(K)*N качество метапризнака зависит от числа усреднений и, если классификатор A может давать различные результаты для одной и той же обучающей выборки в силу зависимости от некоторых случайных параметров, для oof(K) тоже. В таком случае, чтобы сравнить различные способы получения метапризнака, необходимо зафиксировать одинаковую вычислительную сложность для различных K в oof(K)*N и oof(K). Разумным представляется следующее предположение. Для того, чтобы построить метапризнак oof(K)*1, необходимо обучить K классификаторов на доле выборки, равной (K-1)/K. Пусть время обучения классификатора прямо пропорционально объему выборки, тогда время обучения oof(K)*1 будет пропорционально K*(K-1)/K = K-1, а время обучения oof(K)*N будет пропорционально (K-1)*N. В таком случае для сравнения качества метапризнаков с различным K предлагается зафиксировать (K-1)*N. Во всех проведенных в данной работе экспериментах (K-1)*N = 100.

Обозначим цели вычислительного эксперимента:

- сравнить качество метапризнаков, полученных с помощью oof(K) и oof(K)*N;
- показать, что улучшение непосредственного качества метапризнака при измении способа его построения не обязательно ведёт к улучшению качества метапризнака;
- показать, что уменьшение разности непосредственного качества метапризнака на обучающей и валидационной выборке при измении способа его построения не обязательно ведёт к улучшению качества метапризнака.

Для удобства в указании способа получения метапризнака иногда будем опускать "test_averaging". Например, вместо "oof(K), test_averaging=True" будем записывать "oof(K), True".

Также для удобства будем обозначать схему стекинга следующим образом: $M(X, A)$, например, если метаклассификатор M - градиентный бустинг (XGBoost) [21], а базовый классификатор A - ExtraTreesClassifier (ET) [20], то обозначение схемы стекинга будет выглядеть так: XGBoost(X,ET).

ExtraTreesClassifier является классификатором, построенным по аналогии с случайным лесом. Разница между случайным лесом и ET состоит в стратегии выбора разделения подвыборки в вершине дерева для фиксированного подмножества признаков. Случайный лес проверяет всевозможные варианты разбиений для каждого признака, тогда как ET выбирает лишь среди случайно выбранных по одному на признак разбиений.

Используемый XGBoost оптимизирует многоклассовый logloss (multiclass logloss), ExtraTreesClassifier оптимизирует аккуратность (accuracy), логистическая регрессия (Logistic Regression [20]) решает задачу многоклассовой классификации методом один-против-всех (One-vs-rest, OVR), поэтому оптимизируется средний logloss (mean logloss).

4.2 Поставленные эксперименты

Для эксперимента выбраны два датасета с задачами мультиклассовой классификации:

- UCI, Forest Cover Type Prediction;
- Kaggle, Otto Group Product Classification Challenge.

Все результаты приводятся для усреднения по 10 случайным разбиениям на обучающую и тестовую выборки размером по 15 тысяч объектов. Если не указано иначе, границами ошибки на графиках изображено среднеквадратичное отклонение.

Эксперименты 1-3: UCI Forest Cover Type Prediction. Используется один и тот же набор метапризнаков, отличаются только метаклассификаторы: ET, LogisticRegression, XGBoost соответственно. Метапризнаки получены с помощью базового классификатора XGBoost с параметрами max_depth=15, eta=0.1, early_stopping=True. При получении метапризнака для валидационной выборки с помощью XGBoost и test_averaging=False здесь и далее будем использовать среднее число итераций XGBoost для K моделей, полученных во время генерации метапризнака для обучающей выборки. Результат обучения базового классификатора зависел только от обучающей выборки, поэтому вычислительная сложность oof(K) была пропорциональна K-1.

Эксперимент 1: Схема ET(X, XGBoost(X)). Параметры метаклассификатора: n_estimators=10000.

K	N	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False
2	100	0.8300	0.8443	0.8463	0.8488
3	50	0.8371	0.8432	0.8489	0.8488
5	25	0.8431	0.8447	0.8502	0.8469
7	16	0.8443	0.8444	0.8502	0.8468
10	11	0.8470	0.8445	0.8492	0.8457

Таблица 1: Эксперимент 1. ET(X,XGBoost). В таблице приведены значения качества для метапризнаков, построенных различными способами. ET без метапризнаков даёт accuracy 0.8414.

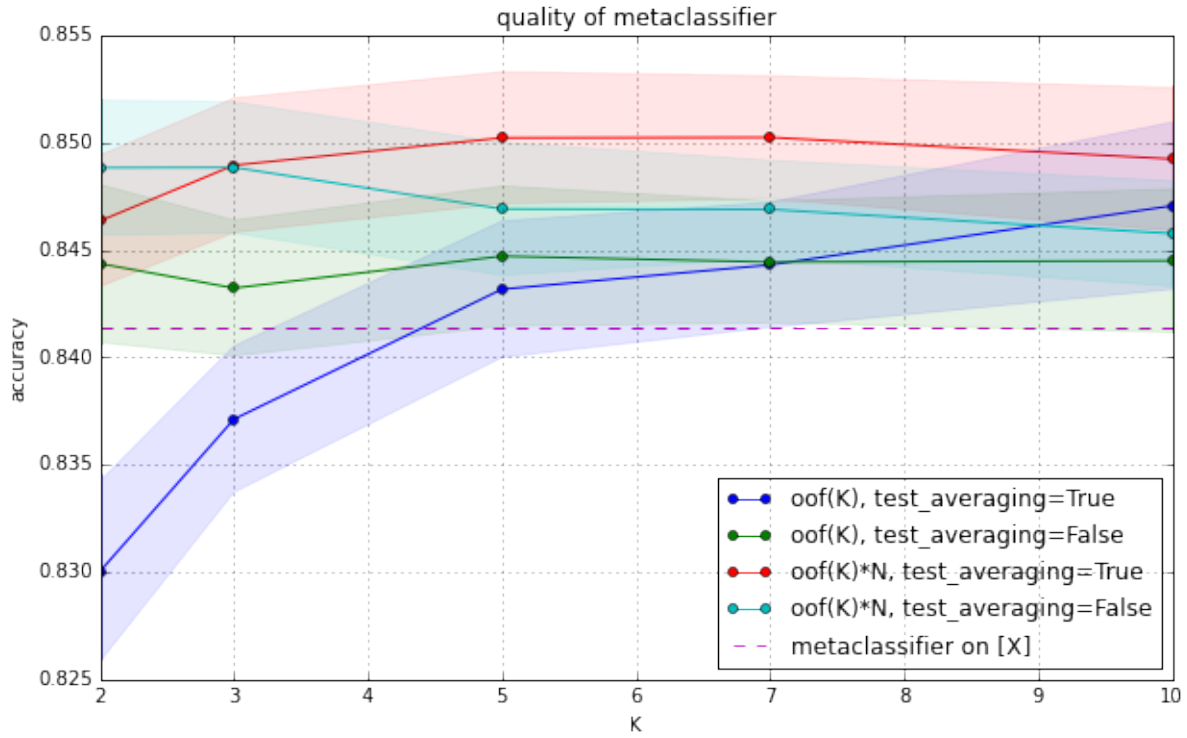

Рис. 3: Эксперимент 1. Качество метапризнаков, полученных различными способами

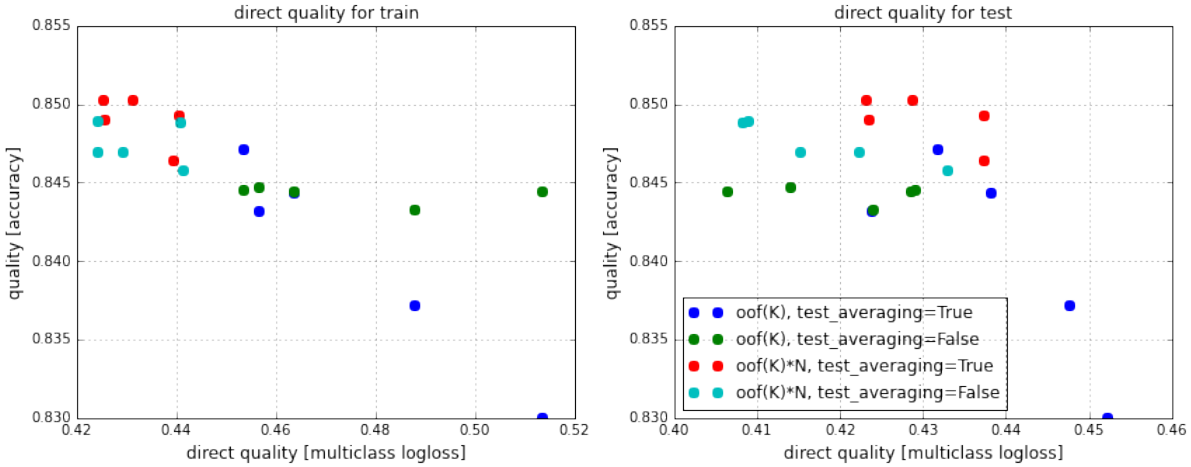

Рис. 4: Эксперимент 1. Зависимость качества метаклассификатора от непосредственного качества метапризнака для обучающей и тестовой выборки.

Эксперимент 2: Схема LogisticRegression(X, XGBoost(X)). Параметры метаклассификатора: C=1.

Эксперимент 3: Схема XGBoost(X, XGBoost). Параметры метаклассификатора: max_depth=15, eta=0.1, early_stopping=True, colsample_bytree=0.75, subsample=0.75. Происходит усреднение предсказаний метаклассификаторов с параметром seed=1...6.

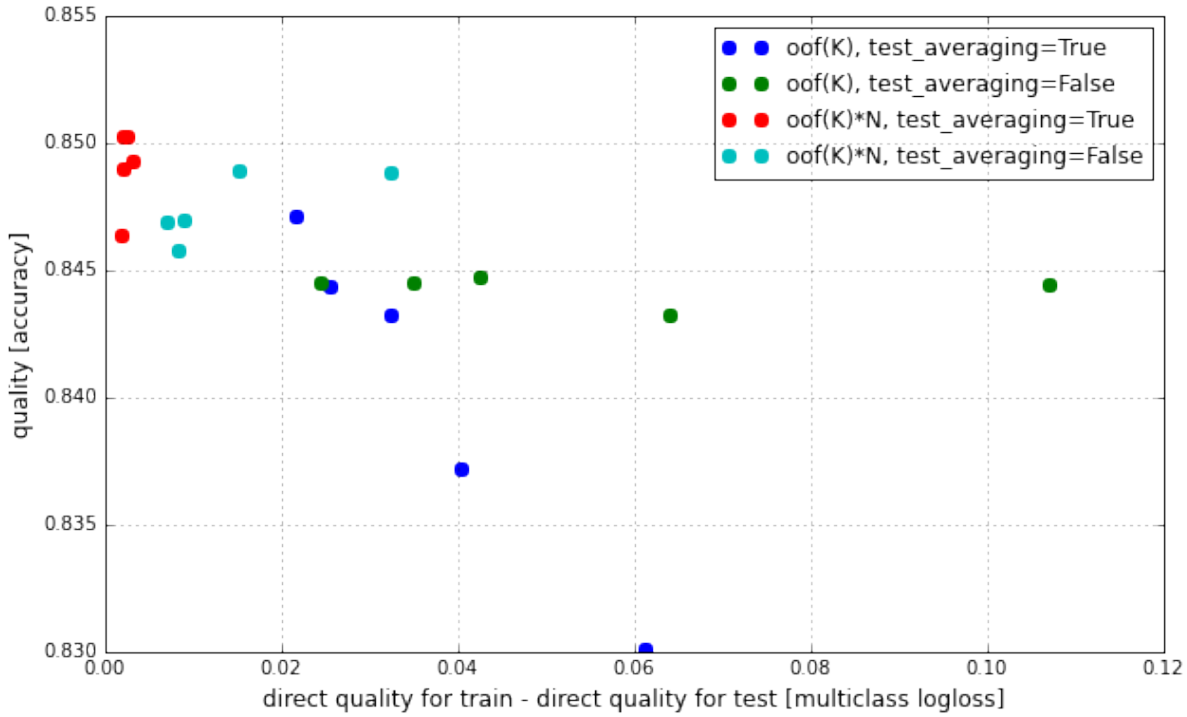

Рис. 5: Эксперимент 1. Зависимость качества метаклассификатора от разности между непосредственным качеством метапризнака для обучающей и тестовой выборки.

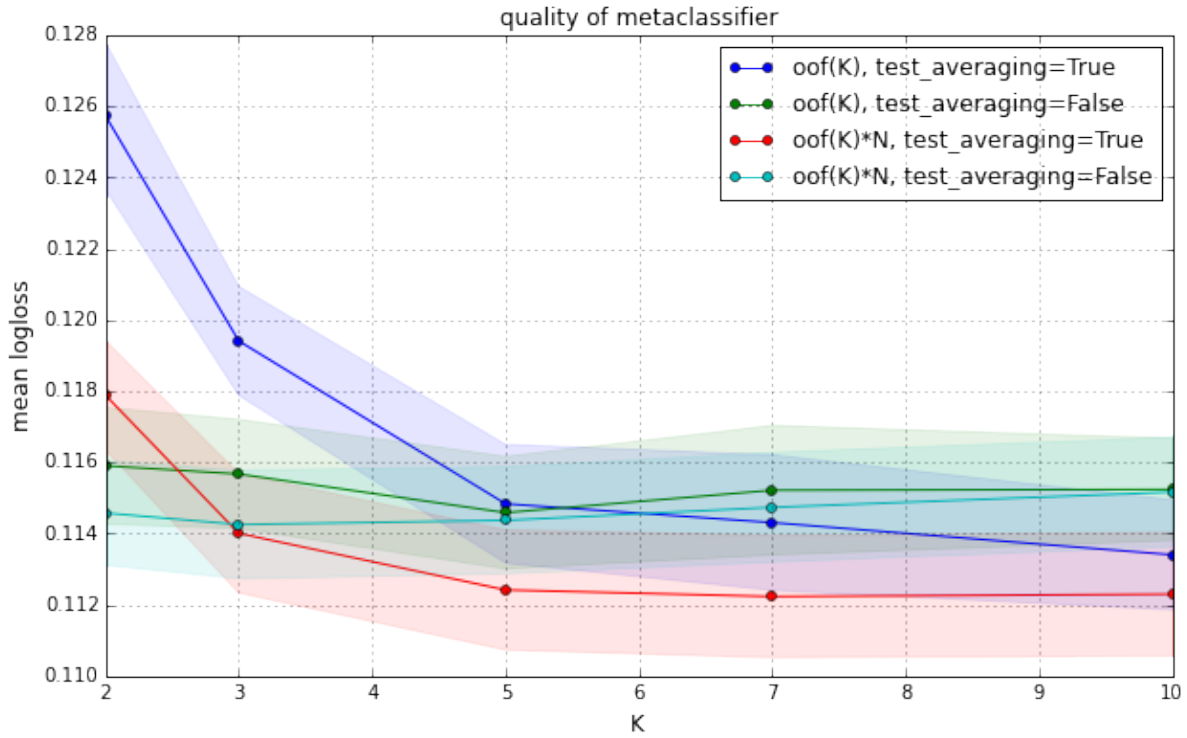

Рис. 6: Эксперимент 2. Качество метапризнаков, полученных различными способами

K	N	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False
2	100	0.1257	0.1159	0.1178	0.1145
3	50	0.1194	0.1156	0.1140	0.1140
5	25	0.1148	0.1145	0.1124	0.1143
7	16	0.1143	0.1152	0.1122	0.1147
10	11	0.1134	0.1152	0.1122	0.1151

Таблица 2: Эксперимент 2. LogisticRegression(X,XGBoost). В таблице приведены значения качества для метапризнаков, построенных различными способами. LogisticRegression без метапризнаков даёт средний logloss 0.1771.

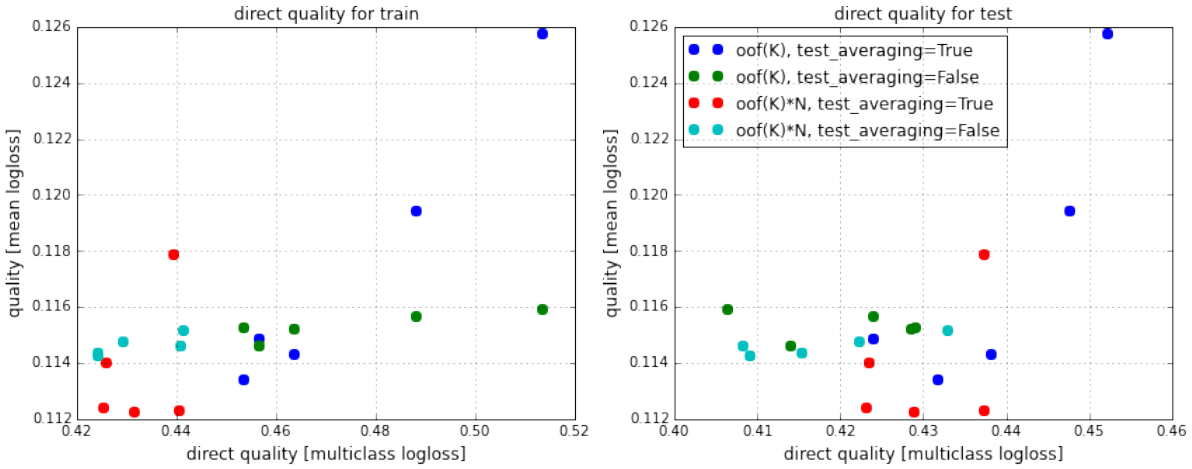

Рис. 7: Эксперимент 2. Зависимость качества метаклассификатора от непосредственного качества метапризнака для обучающей и тестовой выборки.

K	N	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False
2	100	0.4361	0.4146	0.4180	0.4327
3	50	0.4261	0.4248	0.4139	0.4320
5	25	0.4180	0.4248	0.4154	0.4341
7	16	0.4189	0.4305	0.4154	0.4354
10	11	0.4170	0.4315	0.4184	0.4392

Таблица 3: Эксперимент 3. XGBoost(X,XGBoost). В таблице приведены значения качества для метапризнаков, построенных различными способами. XGBoost без метапризнаков даёт мультиклассовый logloss 0.3906.

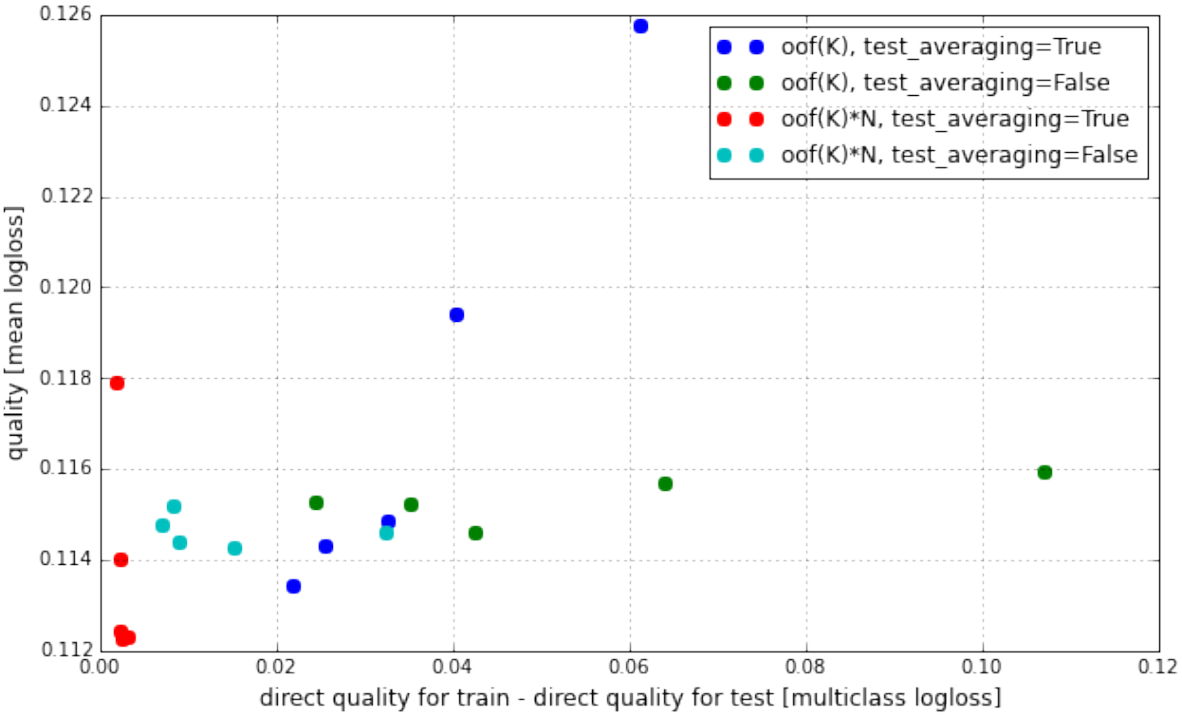

Рис. 8: Эксперимент 2. Зависимость качества метаклассификатора от разности между непосредственным качеством метапризнака для обучающей и тестовой выборки.

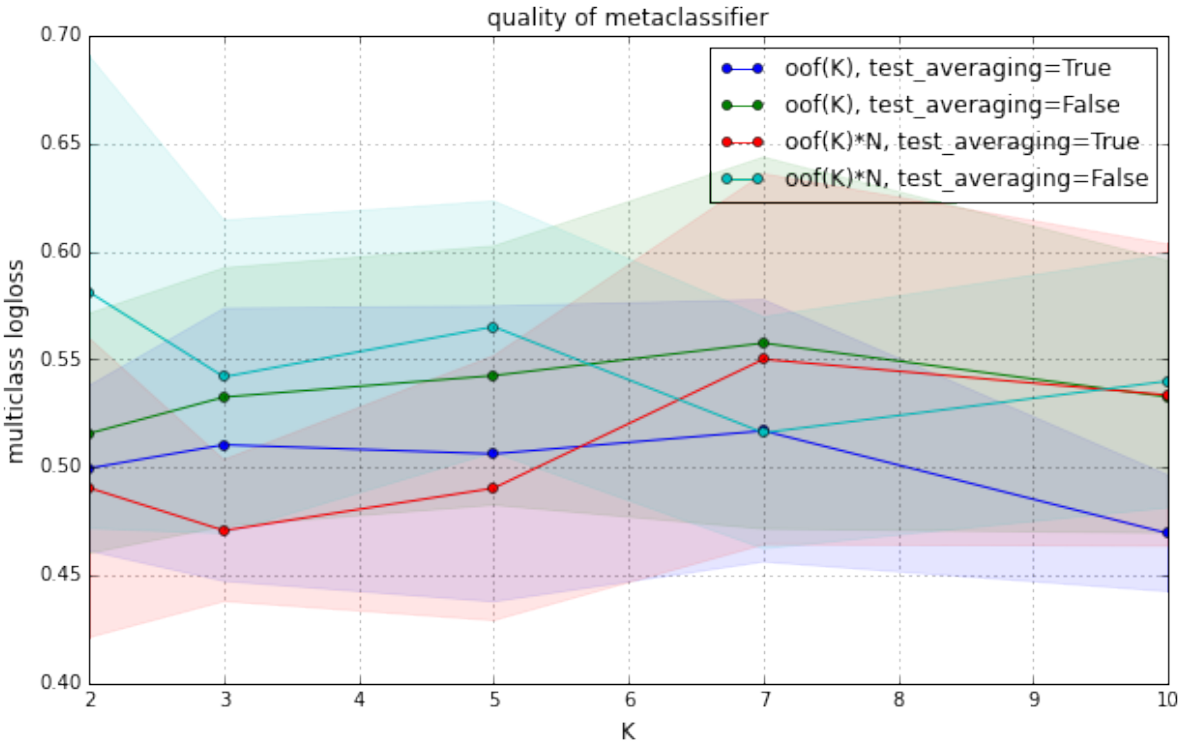

Рис. 9: Эксперимент 3. Качество метапризнаков, полученных различными способами

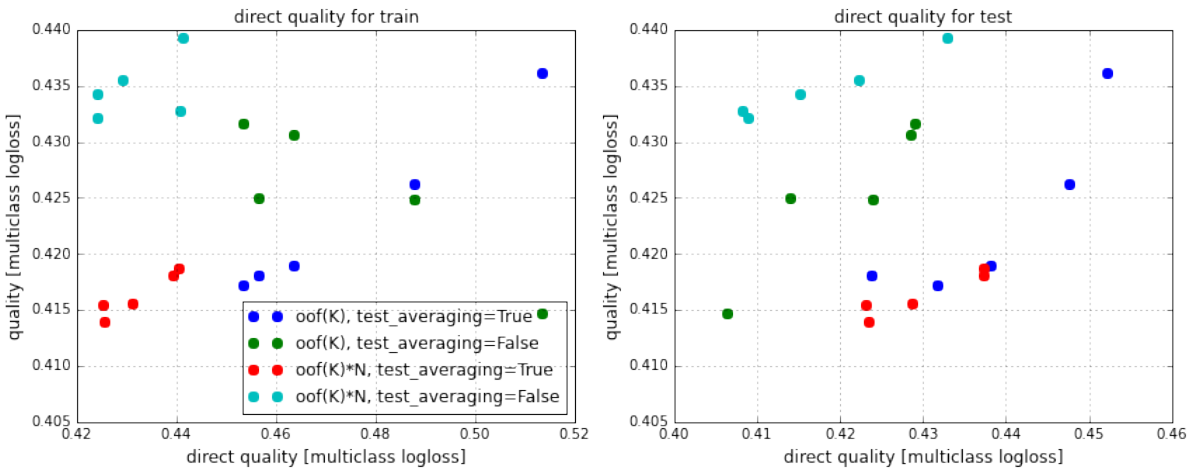

Рис. 10: Эксперимент 3. Зависимость качества метаклассификатора от непосредственного качества метапризнака для обучающей и тестовой выборки.

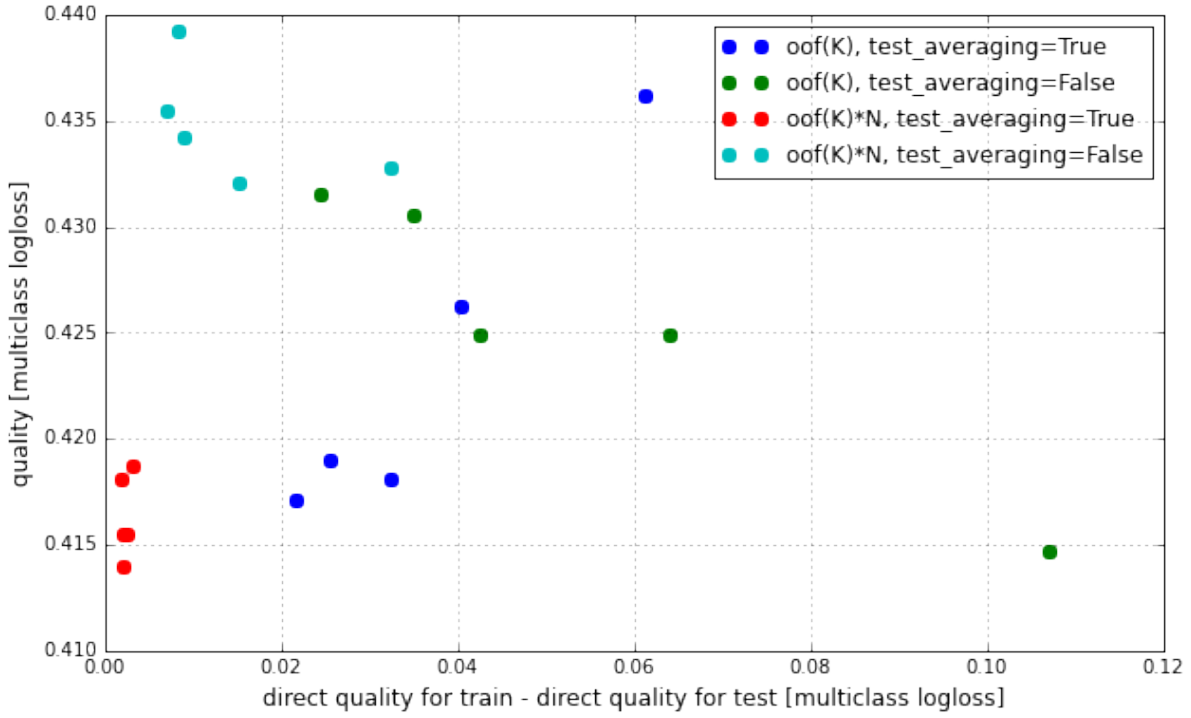

Рис. 11: Эксперимент 3. Зависимость качества метаклассификатора от разности между непосредственным качеством метапризнака для обучающей и тестовой выборки.

Эксперименты 4-5: UCI Forest Cover Type Prediction. Используется один и тот же набор метапризнаков, отличаются только метаклассификаторы: ET, XGBoost соответственно. Метапризнаки получены с помощью базового классификатора LogisticRegression с параметрами C=1. Результат обучения базового классификатора зависел только от обучающей выборки, поэтому вычислительная сложность oof(K) была пропорциональна K-1.

Эксперимент 4: Схема ET(X, LogisticRegression(X)). Параметры метаклассификатора: n_estimators=10000.

K	N	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False
2	100	0.8382	0.8274	0.8459	0.8313
3	50	0.8405	0.8350	0.8465	0.8387
5	25	0.8432	0.8408	0.8463	0.8437
7	16	0.8446	0.8428	0.8464	0.8451
10	11	0.8451	0.8442	0.8465	0.8455

Таблица 4: Эксперимент 4. ET(X,LogisticRegression). В таблице приведены значения качества для метапризнаков, построенных различными способами. ET без метапризнаков даёт accuracy 0.8414.

Эксперимент 5: Схема XGBoost(X, LogisticRegression). Параметры метаклассификатора: max_depth=15, eta=0.1, early_stopping=True.

K	N	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False
2	100	0.4229	0.4353	0.4104	0.4289
3	50	0.4184	0.4253	0.4113	0.4196
5	25	0.4168	0.4196	0.4125	0.4154
7	16	0.4156	0.4174	0.4119	0.4146
10	11	0.4148	0.4159	0.4115	0.4143
20	5	0.4128	0.4133	-	-
50	2	0.4127	0.4128	-	-
100	1	0.4130	0.4131	-	-

Таблица 5: Эксперимент 5. XGBoost(X,LogisticRegression). В таблице приведены значения качества для метапризнаков, построенных различными способами. XGBoost без метапризнаков даёт мультиклассовый logloss 0.4020.

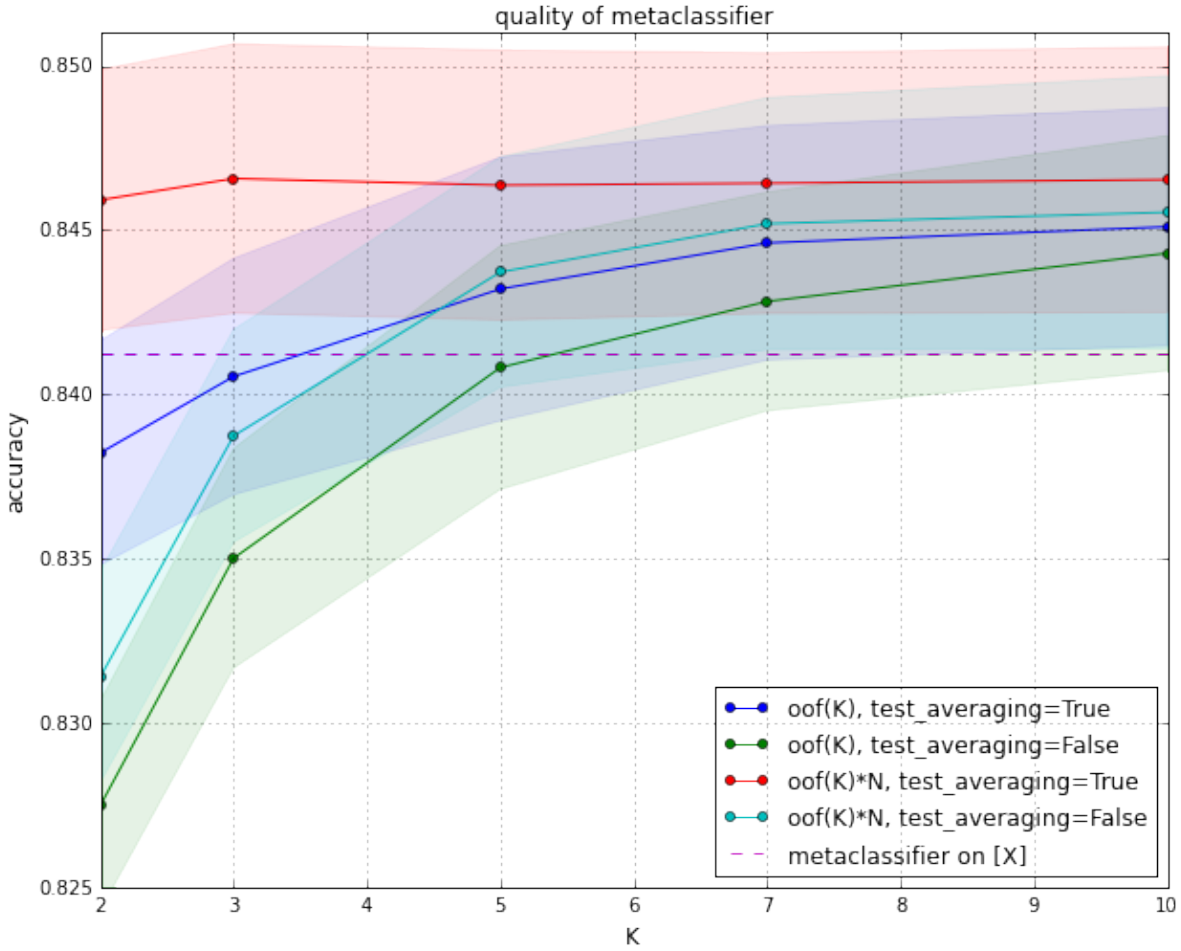

Рис. 12: Эксперимент 4. Качество метапризнаков, полученных различными способами

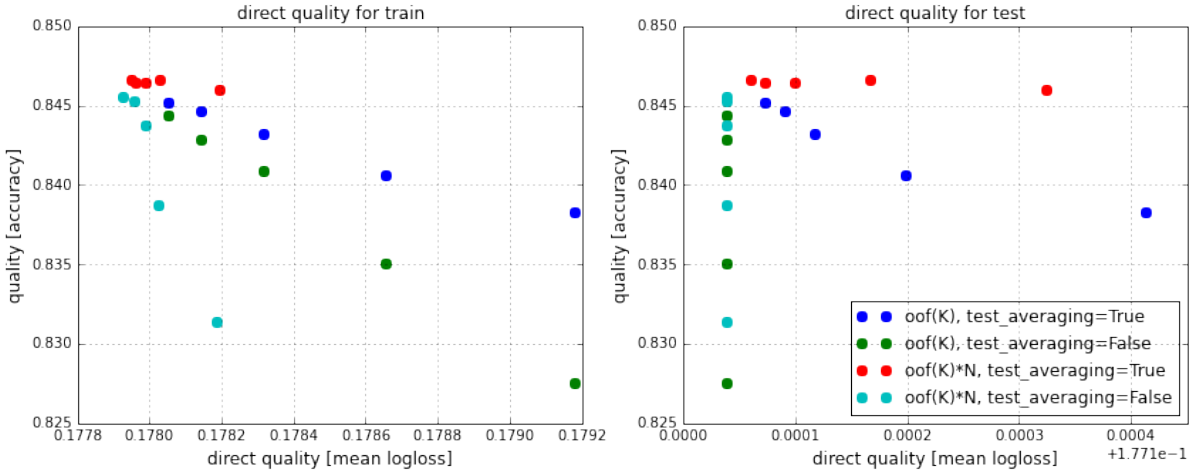

Рис. 13: Эксперимент 4. Зависимость качества метаклассификатора от непосредственного качества метапризнака для обучающей и тестовой выборки.

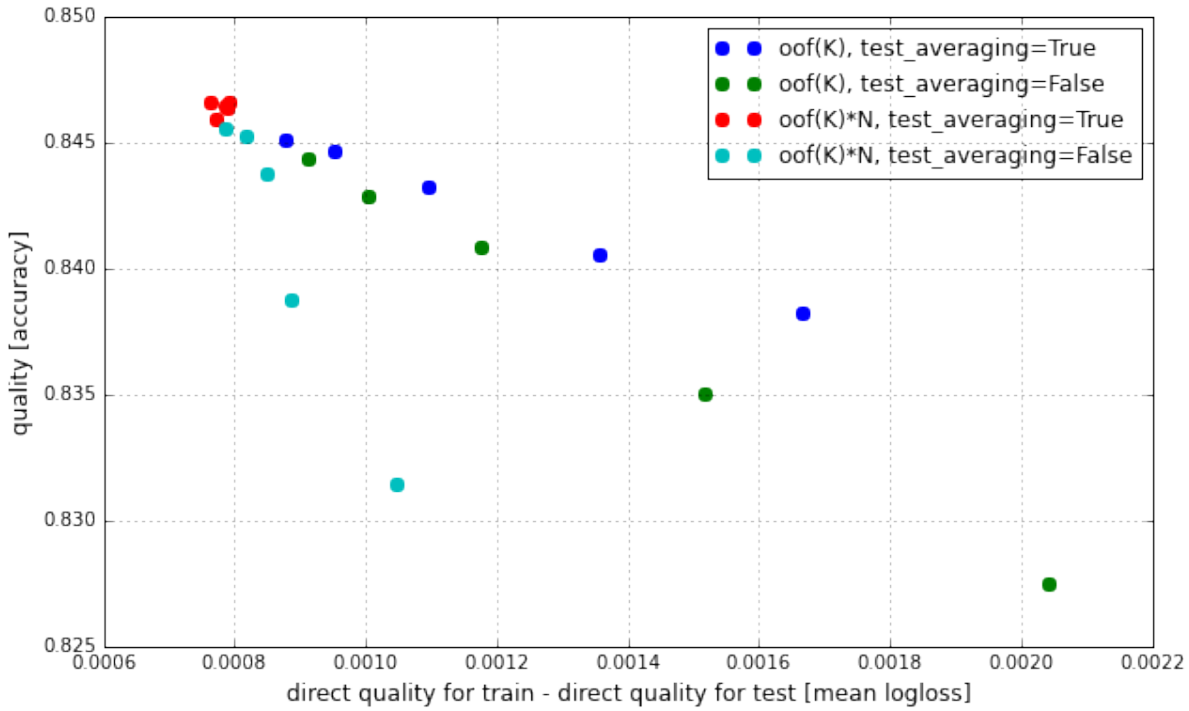

Рис. 14: Эксперимент 4. Зависимость качества метаклассификатора от разности между непосредственным качеством метапризнака для обучающей и тестовой выборки.

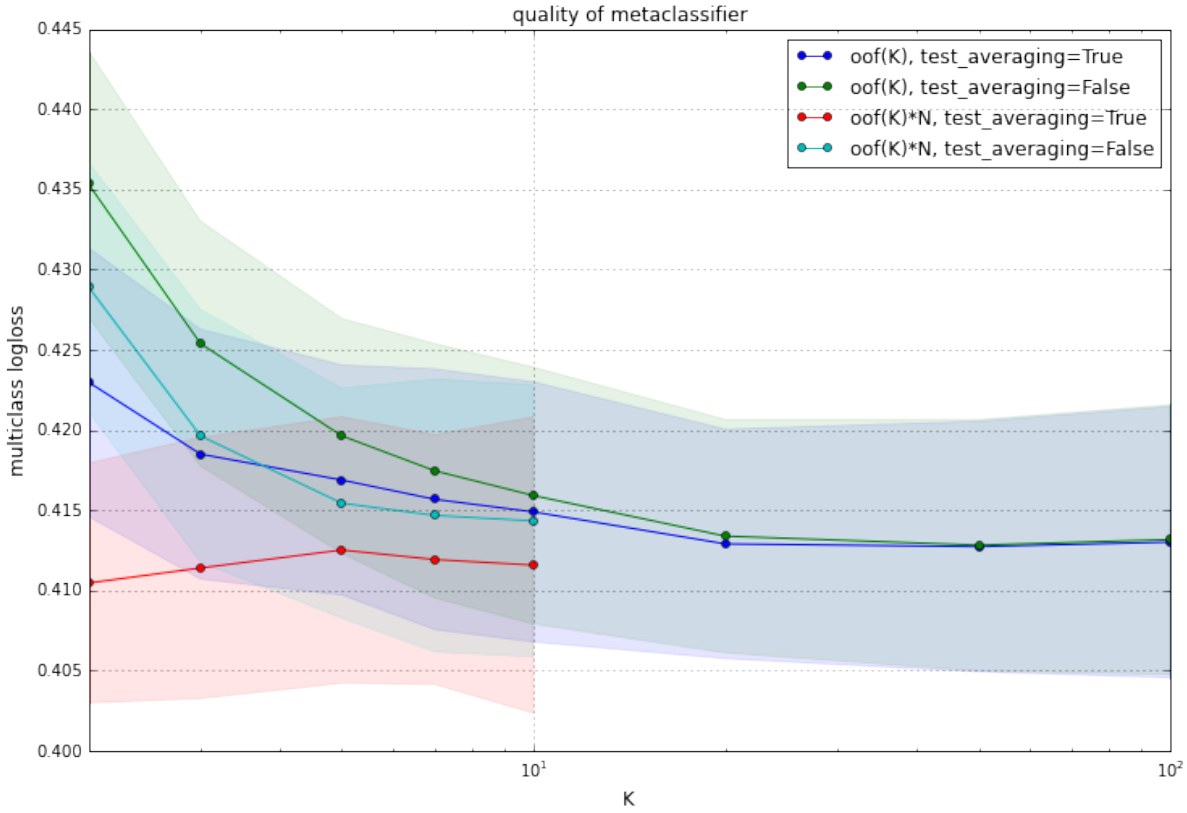

Рис. 15: Эксперимент 5. Качество метапризнаков, полученных различными способами

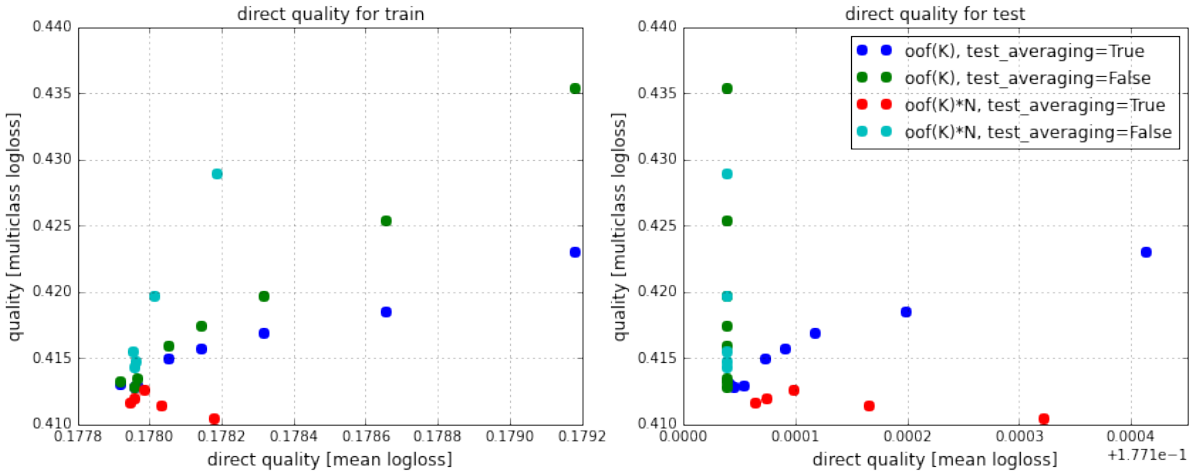

Рис. 16: Эксперимент 5. Зависимость качества метаклассификатора от непосредственного качества метапризнака для обучающей и тестовой выборки.

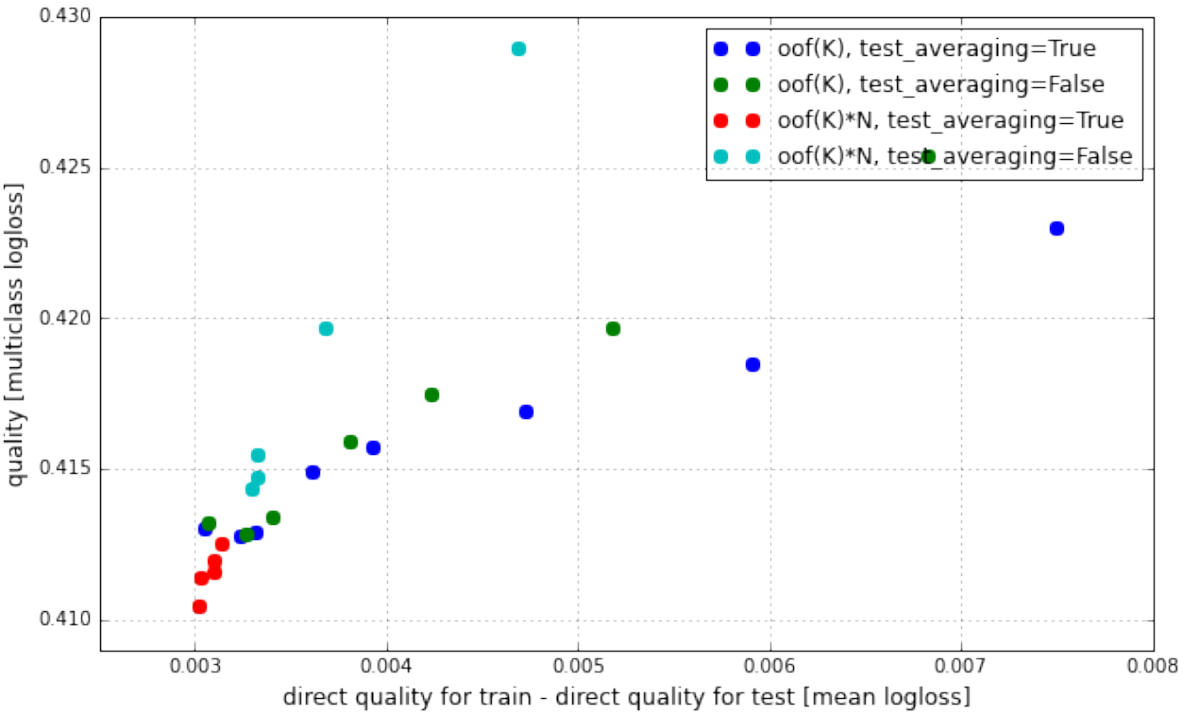

Рис. 17: Эксперимент 5. Зависимость качества метаклассификатора от разности между непосредственным качеством метапризнака для обучающей и тестовой выборки.

Эксперименты 6-7: Kaggle, Otto Group Product Classification Challenge. Используется один и тот же набор метапризнаков, отличаются только метаклассификаторы: ET, XGBoost соответственно. Метапризнаки получены с помощью базового классификатора LogisticRegression с параметрами C=1. Результат обучения базового классификатора зависел только от обучающей выборки, поэтому вычислительная сложность oof(K) была пропорциональна K-1. В качестве границ ошибки изображено среднеквадратичное отклонение, делённое на 5.

Эксперимент 6: Схема ET(X, LogisticRegression(X)). Параметры метаклассификатора: n_estimators=10000.

K	N	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False
2	100	0.7940	0.7939	0.7949	0.7950
3	50	0.7942	0.7942	0.7952	0.7947
5	25	0.7942	0.7943	0.7949	0.7948
7	16	0.7947	0.7946	0.7950	0.7953
10	11	0.7945	0.7945	0.7948	0.7948
20	5	0.7946	0.7946	0.7950	0.7949
50	2	0.7949	0.7949	0.7951	0.7949
100	1	0.7950	0.7951	-	-

Таблица 6: Эксперимент 6. ET(X,LogisticRegression). В таблице приведены значения качества для метапризнаков, построенных различными способами. ET без метапризнаков даёт accuracy 0.7862.

Эксперимент 7: Схема XGBoost(X, LogisticRegression). Параметры метаклассификатора: max_depth=15, eta=0.1, early_stopping=True, colsample_bytree=0.25, subsample=0.65. Происходит усреднение предсказаний метаклассификатора с параметром seed=1...11.

K	N	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False
2	100	0.5374	0.5389	0.5379	0.5407
3	50	0.5375	0.5386	0.5382	0.5398
5	25	0.5379	0.5387	0.5388	0.5395
7	16	0.5386	0.5390	0.5390	0.5391
10	11	0.5383	0.5387	0.5387	0.5391
20	5	0.5386	0.5388	0.5387	0.5391
50	2	0.5391	0.5391	0.5391	0.5392
100	1	0.5389	0.5388	0.5389	0.5390

Таблица 7: Эксперимент 7. XGBoost(X,LogisticRegression). В таблице приведены значения качества для метапризнаков, построенных различными способами. XGBoost без метапризнаков даёт мультиклассовый logloss 0.5388.

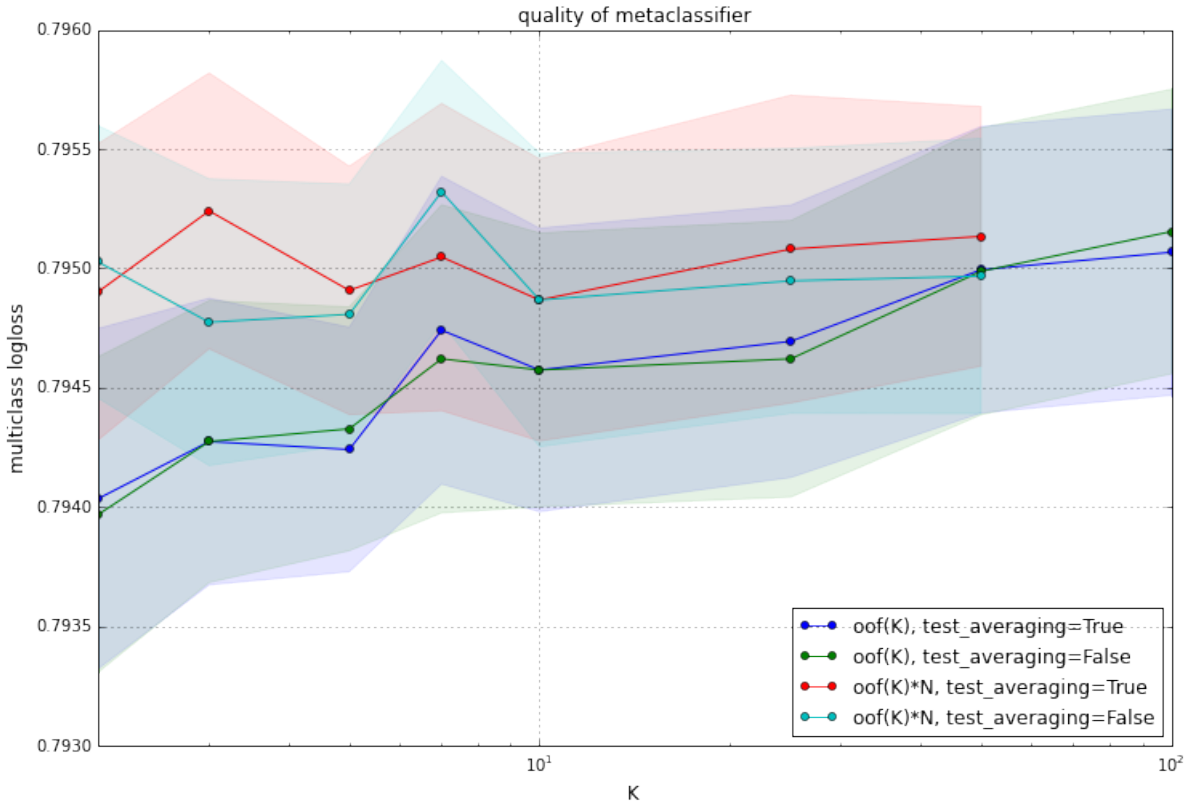

Рис. 18: Эксперимент 6. Качество метапризнаков, полученных различными способами

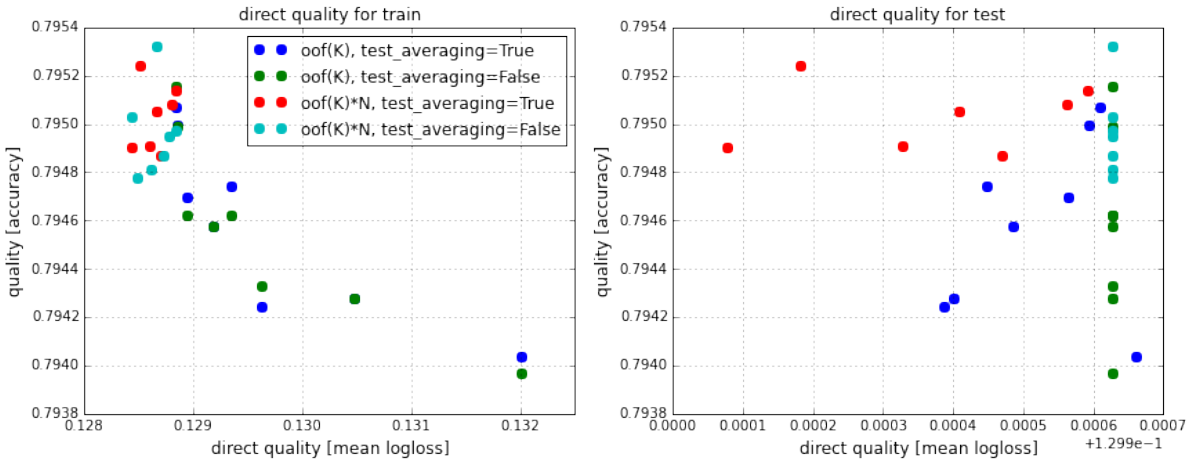

Рис. 19: Эксперимент 6. Зависимость качества метаклассификатора от непосредственного качества метапризнака для обучающей и тестовой выборки.

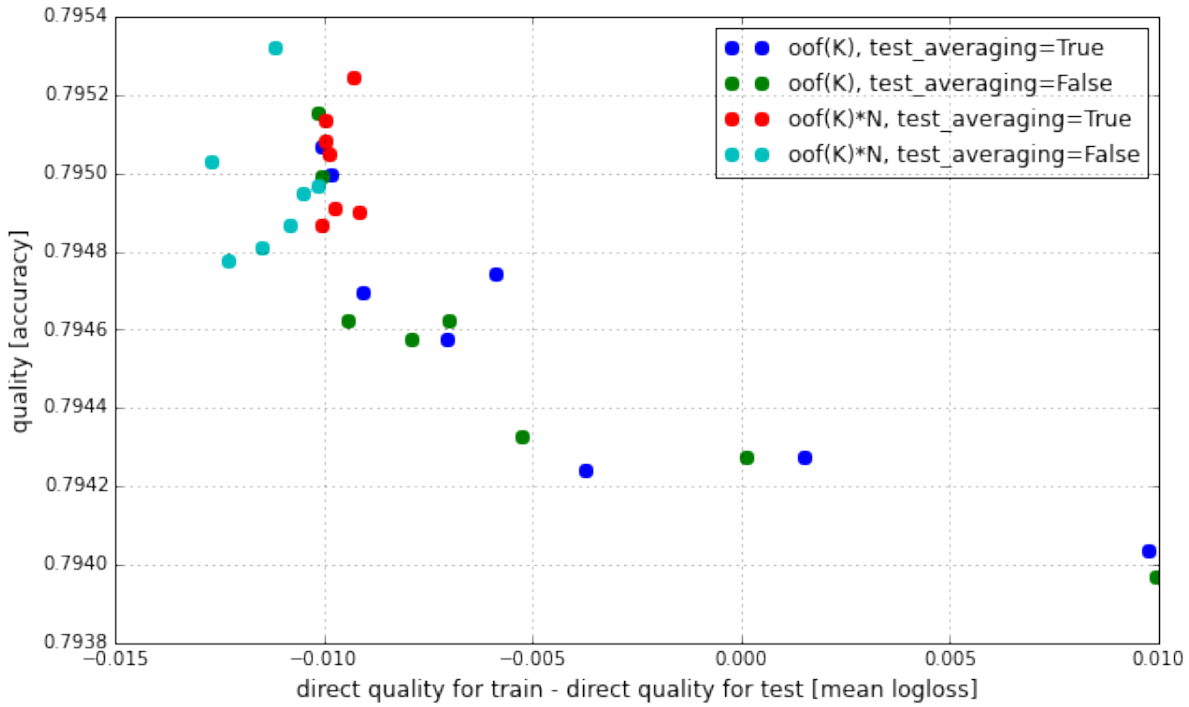

Рис. 20: Эксперимент 6. Зависимость качества метаклассификатора от разности между непосредственным качеством метапризнака для обучающей и тестовой выборки.

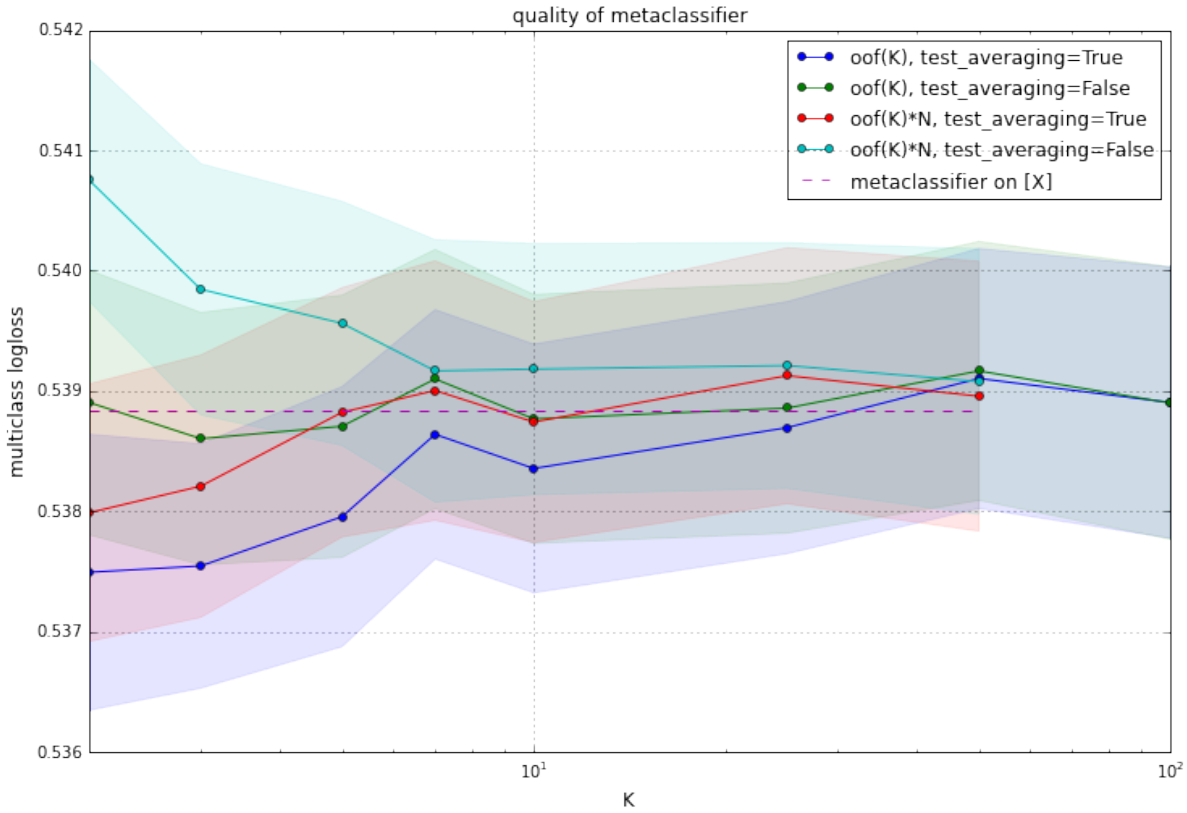

Рис. 21: Эксперимент 7. Качество метапризнаков, полученных различными способами

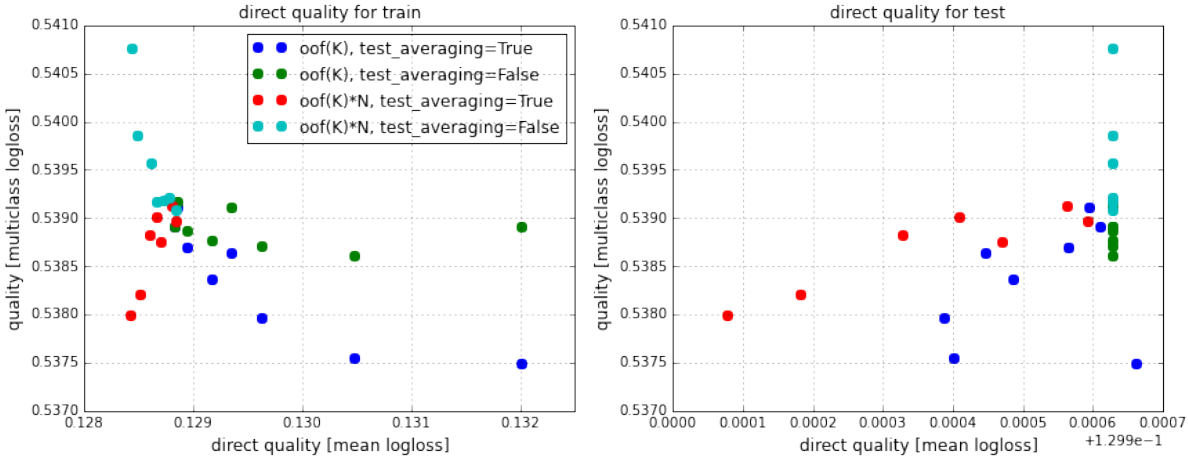

Рис. 22: Эксперимент 7. Зависимость качества метаклассификатора от непосредственного качества метапризнака для обучающей и тестовой выборки.

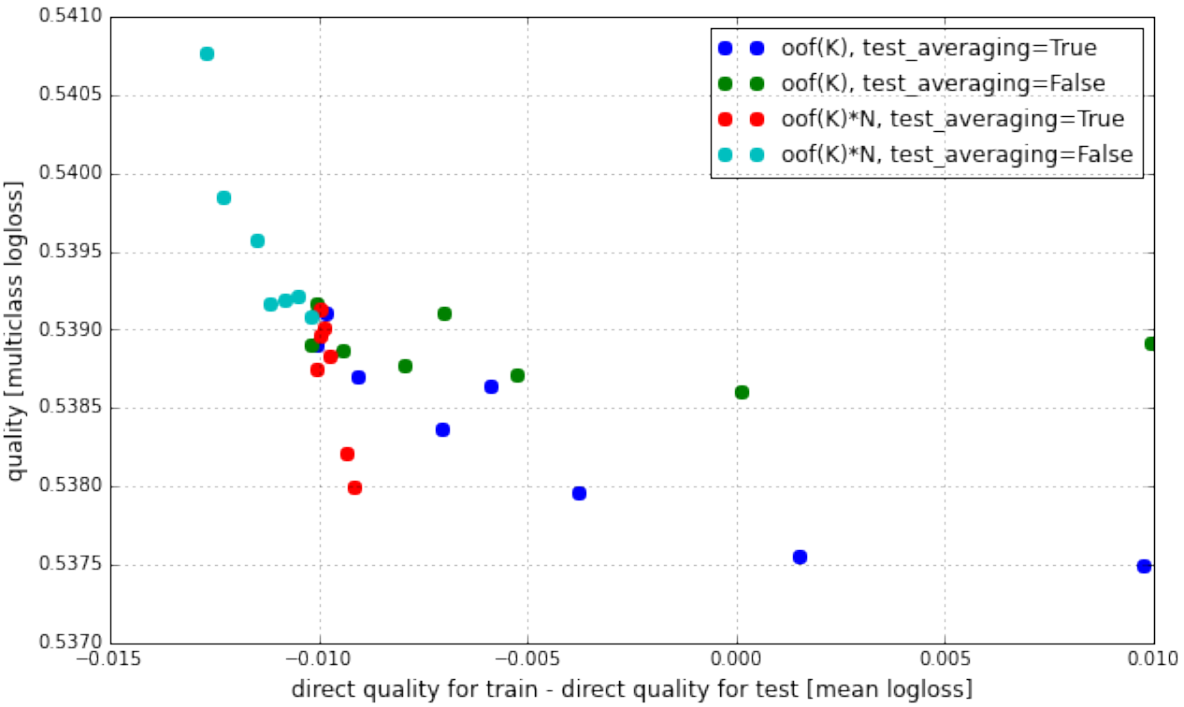

Рис. 23: Эксперимент 7. Зависимость качества метаклассификатора от разности между непосредственным качеством метапризнака для обучающей и тестовой выборки.

Эксперименты 8-10: Kaggle, Otto Group Product Classification Challenge. Используется один и тот же набор метапризнаков, отличаются только метаклассификаторы: ET, XGBoost, LogisticRegression соответственно. Метапризнаки получены с помощью базового классификатора XGBoost с параметрами max_depth=15, eta=0.1, early_stopping=True, colsample_bytree=0.75, subsample=0.75. При получении метапризнака с помощью XGBoost и test_averaging=False мы используем среднее число итераций K моделей, полученных во время генерации метапризнака для обучающей выборки, для установки количества итераций в модели, которой мы предсказываем метапризнак для тестовой выборки. Результат обучения базового классификатора зависел не только от обучающей выборки, но и от случайного параметра, поэтому вычислительная сложность oof(K) была пропорциональна (K-1)*N. В качестве границ ошибки изображено среднеквадратичное отклонение, делённое на 2.

Эксперимент 8: Схема ET(X, XGBoost(X)). Параметры метаклассификатора: n_estimators=10000.

K	N	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False
2	100	0.8019	0.8036	0.8024	0.8054
3	50	0.8036	0.8051	0.8036	0.8045
5	25	0.8034	0.8041	0.8048	0.8048
7	16	0.8038	0.8037	0.8039	0.8047
10	11	0.8046	0.8038	0.8040	0.8035

Таблица 8: Эксперимент 8. ET(X,XGBoost). В таблице приведены значения качества для метапризнаков, построенных различными способами. ET без метапризнаков даёт аккуратность 0.7862.

Эксперимент 9: Схема XGBoost(X, XGBoost). Параметры метаклассификатора: max_depth=15, eta=0.1, early_stopping=True, colsample_bytree=0.25, subsample=0.65. Происходит усреднение предсказаний метаклассификатора с параметром seed=1...11.

K	N	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False
2	100	0.5533	0.5493	0.5560	0.5549
3	50	0.5529	0.5530	0.5540	0.5547
5	25	0.5540	0.5555	0.5548	0.5565
7	16	0.5554	0.5579	0.5561	0.5585
10	11	0.5551	0.5596	0.5566	0.5602

Таблица 9: Эксперимент 9. XGBoost(X,XGBoost). В таблице приведены значения качества для метапризнаков, построенных различными способами. XGBoost без метапризнаков даёт мультиклассовый logloss 0.5388.

Эксперимент 10: Схема LogisticRegression(X, XGBoost(X)). Параметры метаклассификатора: C=1.

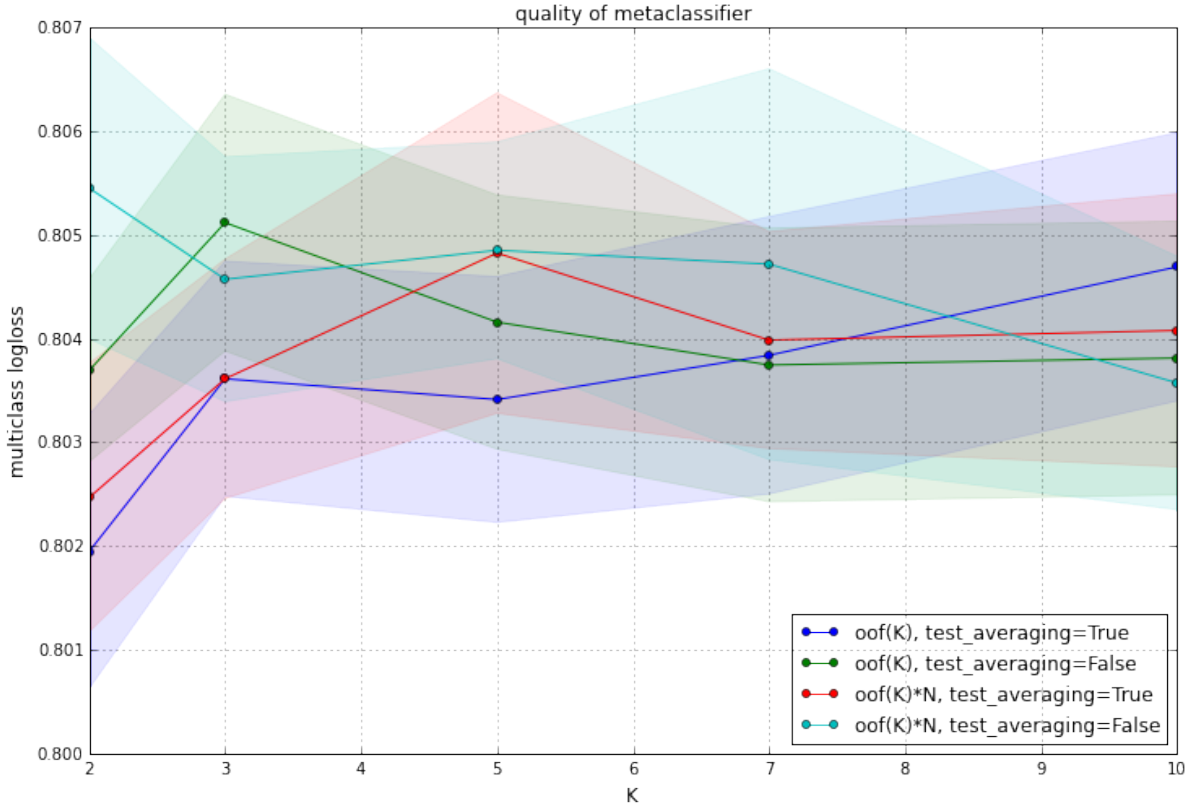

Рис. 24: Эксперимент 8. Качество метапризнаков, полученных различными способами

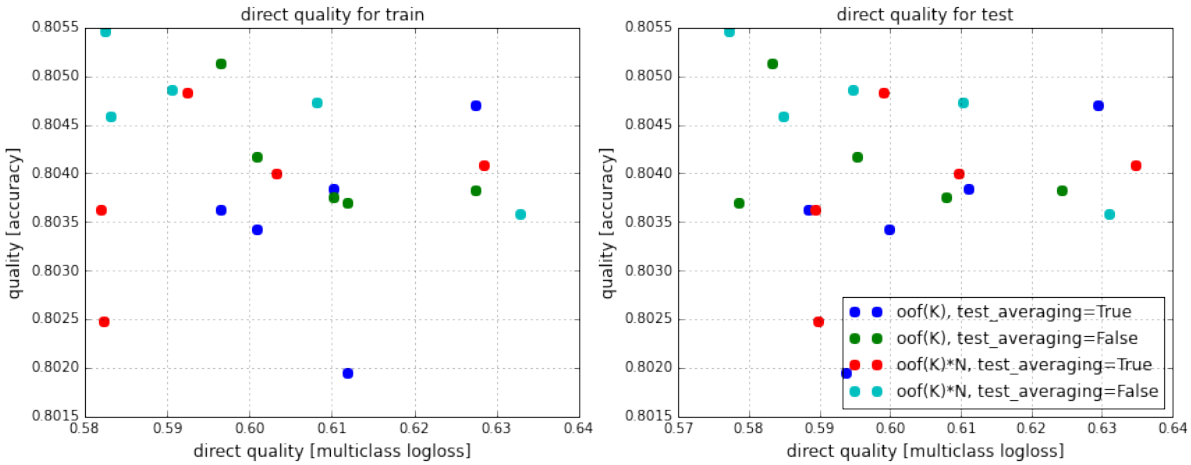

Рис. 25: Эксперимент 8. Зависимость качества метаклассификатора от непосредственного качества метапризнака для обучающей и тестовой выборки.

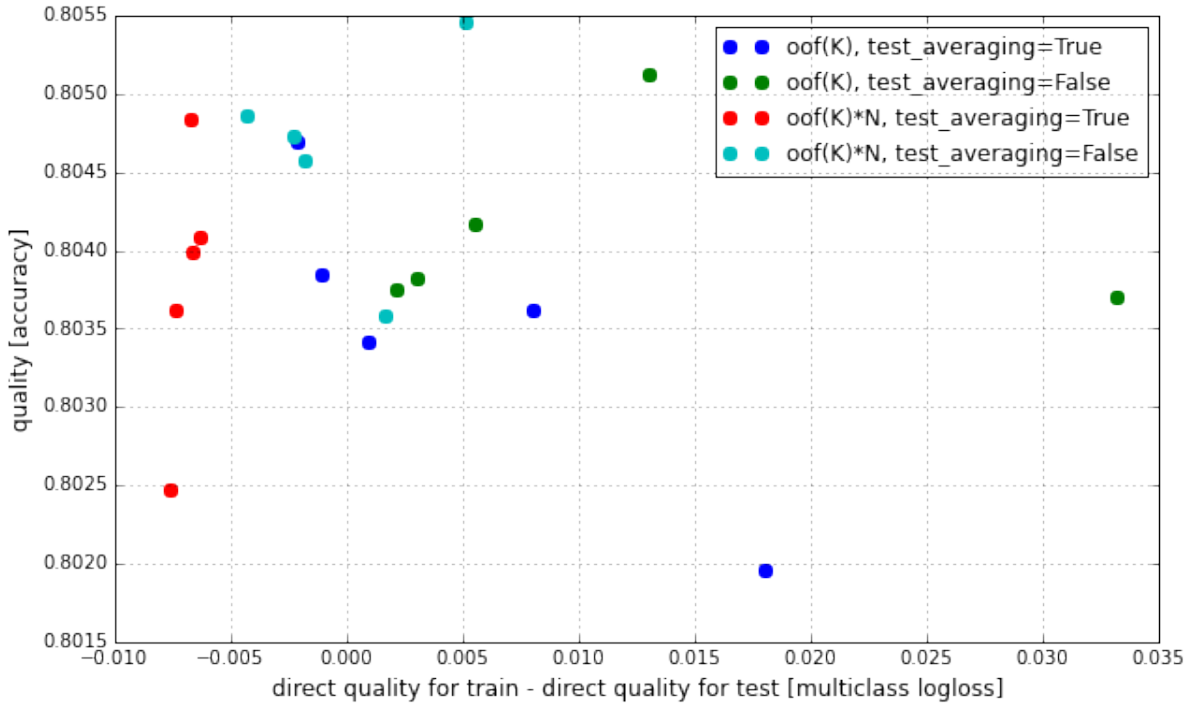

Рис. 26: Эксперимент 8. Зависимость качества метаклассификатора от разности между непосредственным качеством метапризнака для обучающей и тестовой выборки.

K	N	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False
2	100	0.1086	0.1067	0.1073	0.1059
3	50	0.1074	0.1065	0.1067	0.1060
5	25	0.1068	0.1064	0.1065	0.1062
7	16	0.1067	0.1065	0.1065	0.1064
10	11	0.1067	0.1067	0.1066	0.1066

Таблица 10: Эксперимент 10. LogisticRegression(X,XGBoost). В таблице приведены значения качества для метапризнаков, построенных различными способами. LogisticRegression без метапризнаков даёт средний logloss 0.1304.

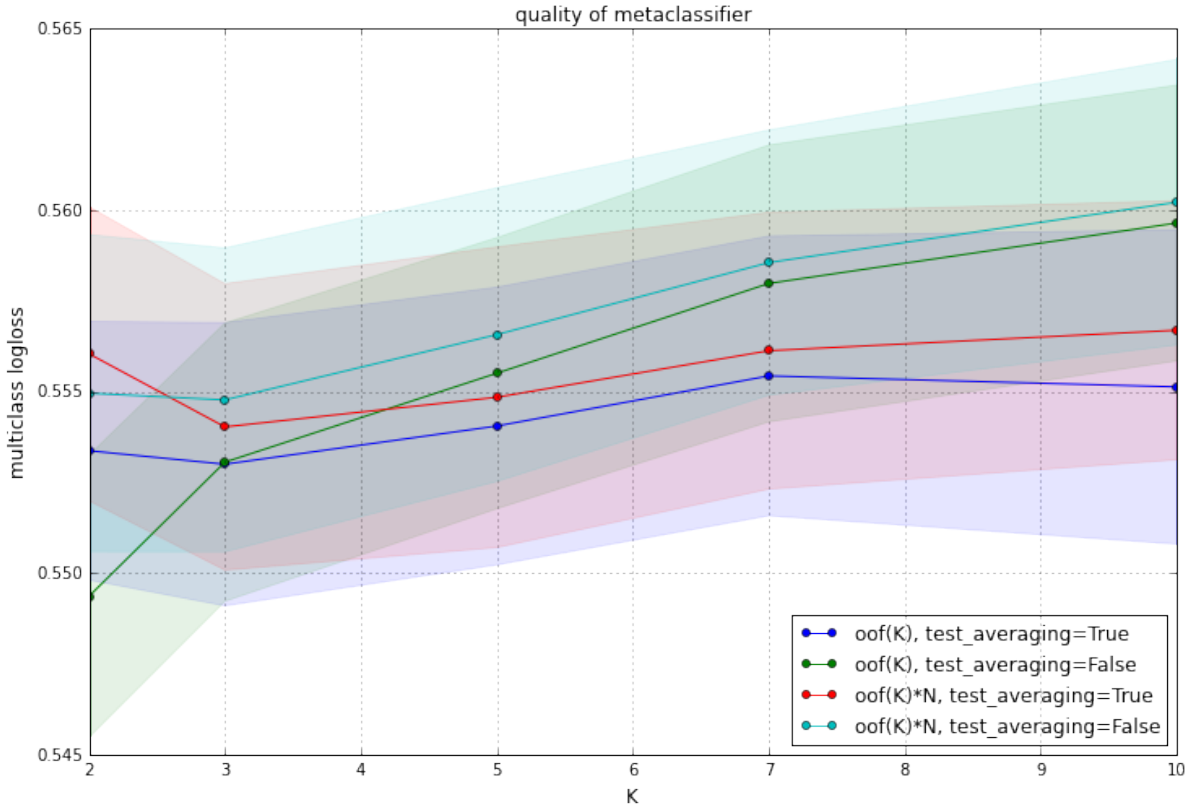

Рис. 27: Эксперимент 9. Качество метапризнаков, полученных различными способами

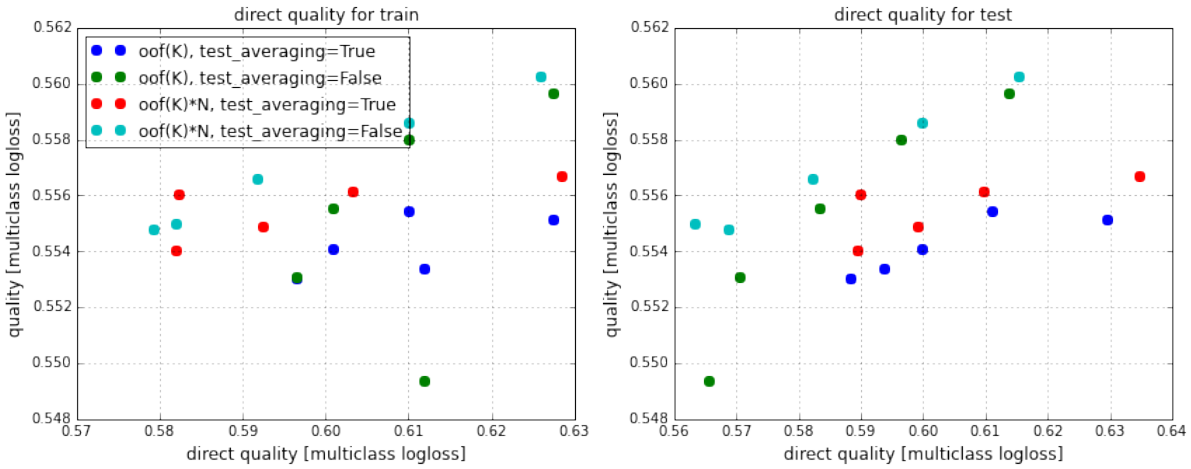

Рис. 28: Эксперимент 9. Зависимость качества метаклассификатора от непосредственного качества метапризнака для обучающей и тестовой выборки.

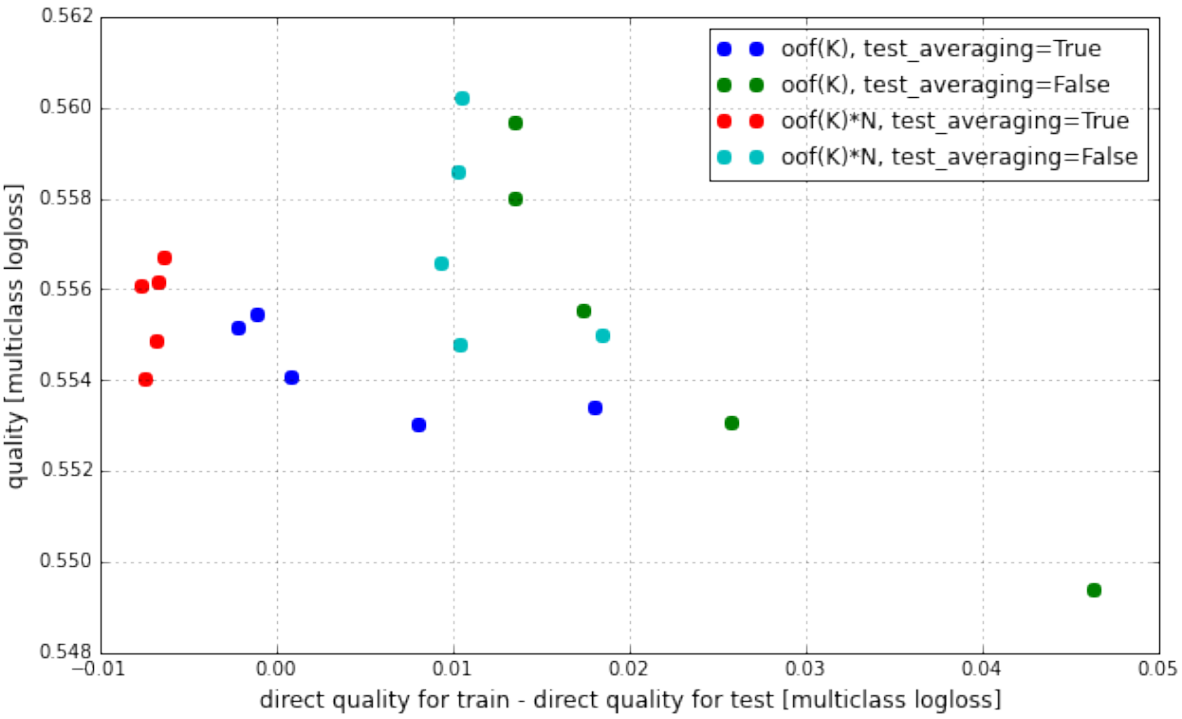

Рис. 29: Эксперимент 9. Зависимость качества метаклассификатора от разности между непосредственным качеством метапризнака для обучающей и тестовой выборки.

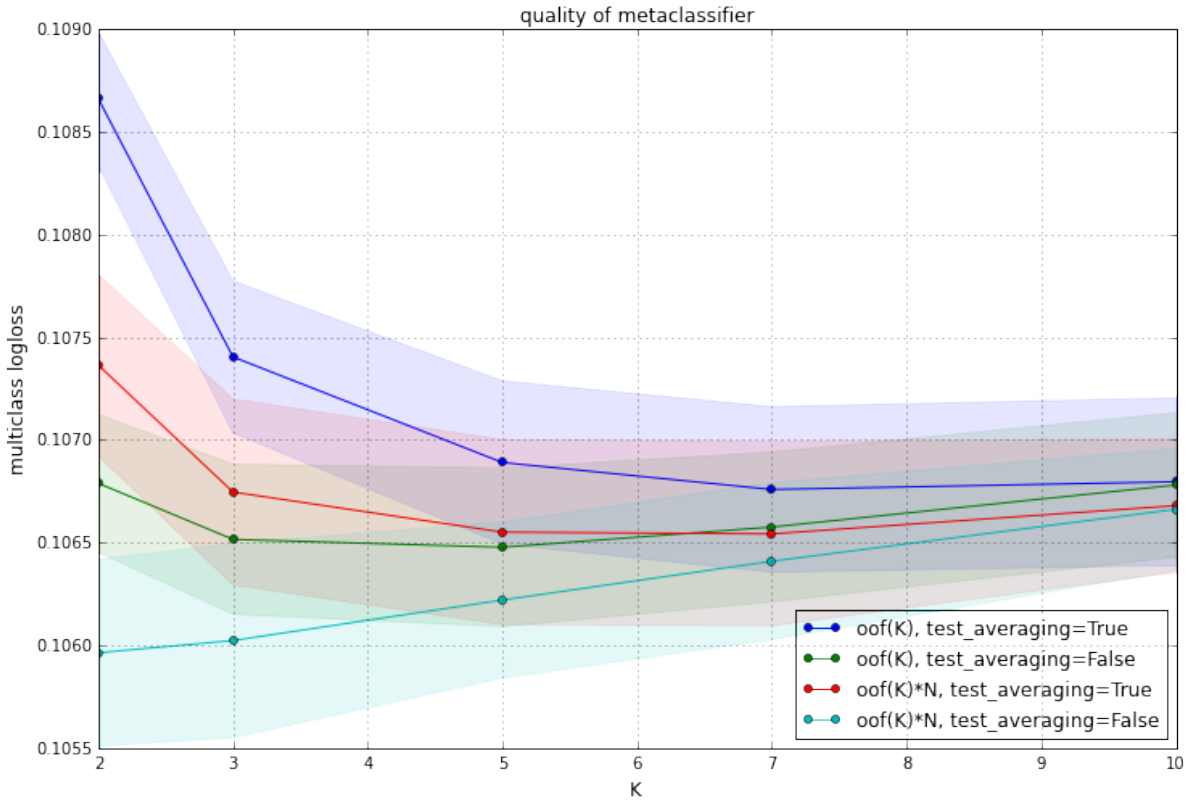

Рис. 30: Эксперимент 10. Качество метапризнаков, полученных различными способами

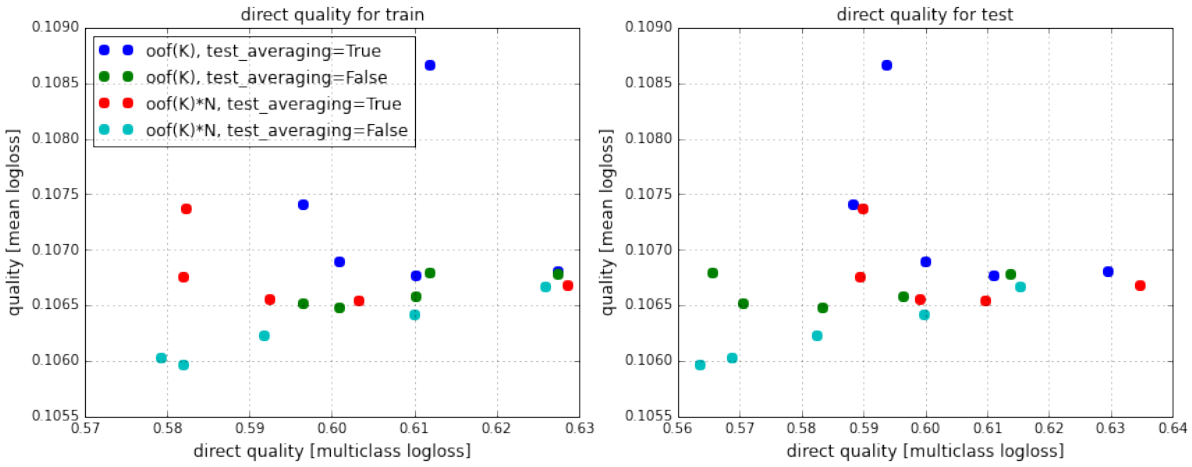

Рис. 31: Эксперимент 10. Зависимость качества метаклассификатора от непосредственного качества метапризнака для обучающей и тестовой выборки.

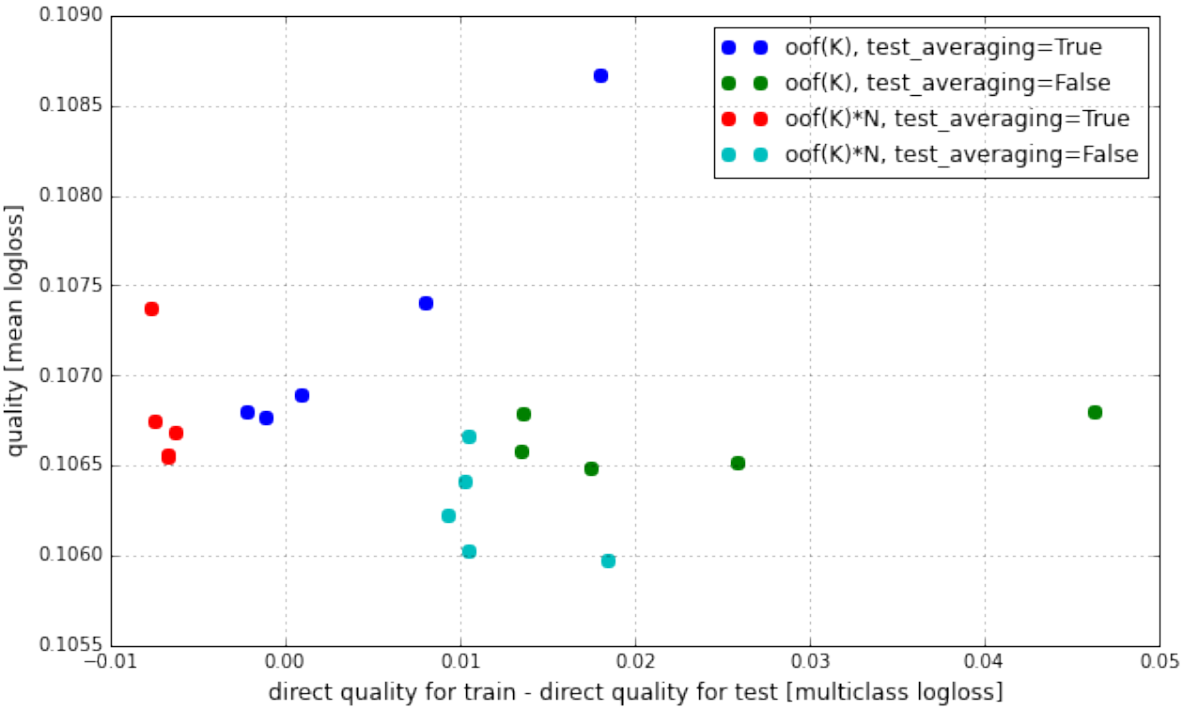

Рис. 32: Эксперимент 10. Зависимость качества метаклассификатора от разности между непосредственным качеством метапризнака для обучающей и тестовой выборки.

4.3 Обобщение результатов экспериментов

Сделаем обобщение результатов проведенных экспериментов. Для каждого способа получения метапризнака посчитаем лучшее качество (или минимальную ошибку) по всем K. Сравним их для разных способов и составим таблицу, в строках и столбцах которой указаны способы получения метапризнаков, а в ячейках стоит число экспериментов, в которых способ по горизонтали дал метапризнак лучшего качества, чем способ по вертикали. Так как всего было поставлено десять экспериментов, идеальный способ получения метапризнака мог бы получить в сумме 30 вдоль соответствующей строки.

	oof(K), True	oof(K), False	oof(K)*N, True	oof(K)*N, False	всего
oof(K), True	0	6	3	5	14/30
oof(K), False	4	0	3	4	11/30
oof(K)*N, True	7	7	0	7	21/30
oof(K)*N, False	5	6	3	0	14/30

Таблица 11: Результаты экспериментов

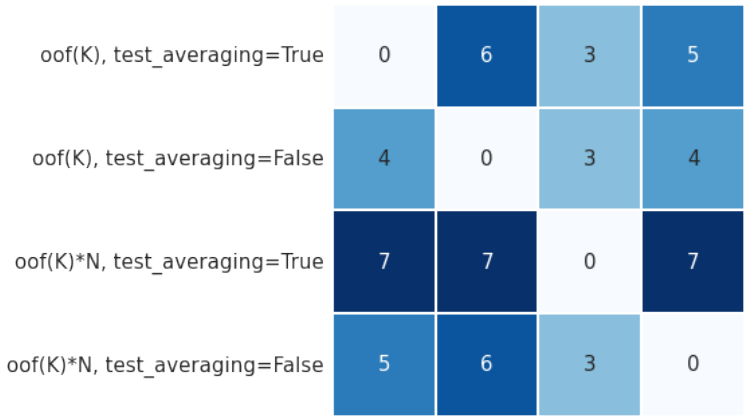

Рис. 33: Таблица 11. Результаты экспериментов.

5 Заключение

В работе предложен метод ансамблирования обучающихся алгоритмов, являющийся модификацией алгоритма стекинга. Поставлены эксперименты на реальных данных, показавшие, что предложенный алгоритм в большинстве случаев превосходит предложенные ранее аналогичные алгоритмы.

Предложенный алгоритм обладает следующими преимуществами:

- происходит эффективное использование обучающей выборки
- осуществляется борьба с "неоднородностью"метапризнаков

Список литературы

[1] Воронцов К.В. Лекции по алгоритмическим композициям // http://www.machinelearning.ru/wiki/images/0/0d/Voron-ML-Compositions.pdf

[2] Wolpert, David H. Stacked generalization // Neural networks 5.2 (1992): 241-259

[3] Breiman, Leo. Stacked regressions // Machine learning 24.1 (1996): 49-64.

[4] Breiman, Leo. Bagging predictors // Machine learning 24.2 (1996): 123-140.

[5] Breiman, Leo. Random Forests // Machine Learning, 45(1), 5-32, 2001

[6] Freund, Yoav, Robert Schapire, and N. Abe. A Short Introduction to Boosting // Journal-Japanese Society For Artificial Intelligence 14.771-780 (1999): 1612

[7] Friedman, Jerome H. Stochastic gradient boosting // Computational Statistics and Data Analysis, 2002

[8] Журавлёв Ю. И. Об алгебраическом подходе к решению задач распознавания или классификации // Проблемы кибернетики. 1978

[9] Рыжков А. М. Композиции алгоритмов, основанные на случайном лесе // Дипломная работа, ВМК МГУ, 2015.

[10] Skurichina M., Duin R. P. W. Limited bagging, boosting and the random subspace method for linear classifiers // Pattern Analysis & Applications. 2002. Pp. 121–135.

[11] Kim, Mike. // 2015, https://www.kaggle.com/c/otto-group-product-classification-challenge/forums/t/14295/41599-via-tsne-meta-bagging

[12] Netflix Prize // http://www.netflixprize.com

[13] Jahrer, Michael. Netflix Prize report 2009 // http://elf-project.sourceforge.net/CombiningPredictionsForAccurateRecommenderSystems.pdf

[14] Sill, Joseph, et al. Feature-weighted linear stacking // arXiv preprint arXiv:0911.0460 (2009).

[15] Toscher, Andreas and Jahrer, Michael. The BigChaos Solution to the Netflix Grand Prize // 2009, http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

[16] Menahem, Eitan, Lior Rokach, and Yuval Elovici. Troika–An improved stacking schema for classification tasks // Information Sciences 179.24 (2009): 4097-4122.

[17] Seewald, A. How to Make Stacking Better and Faster While Also Taking Care of an Unknown Weakness // Nineteenth International Conference on Machine Learning (pp. 554-561) (2002)

[18] Deng, Li, Dong Yu, and John Platt. Scalable stacking and learning for building deep architectures // Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. IEEE, 2012.

[19] Jordan, Michael I., and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm // Neural computation 6.2 (1994): 181-214.

[20] Scikit-learn, http://scikit-learn.org

[21] XGBoost, https://github.com/dmlc/xgboost