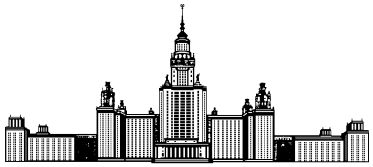
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

Медведев Дмитрий Владимирович

Методы повышения обобщающей способности, основанные на различных способах построения ансамблей

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

д.ф-м.н., профессор

Сенько Олег Валентинович

Москва, 2019

Содержание

Аннотация

Целью работы является разработка и исследование нового метода ансамблей решающих деревьев, максимально удалённых друг от друга. Разработанный метод сравнивается с известными моделями, основанными на ансамблях деревьев, включая: случайный лес и адаптивный бустинг.

1 Введение

Для решения многих задач классификации и восстановления регрессии часто используются ансамбли. Эти методы машинного обучения основаны на композиции нескольких, так называемых, "слабых" алгоритмов. В качестве базовой модели может выступать решающее дерево, нейронная сеть или другой обучаемый алгоритм.

Цель построения ансамбля — получить с помощью композиции нескольких более слабых, в смысле качества, алгоритмов, один сильный. Множество экспериментов показало, что коллективные методы позволяют увеличивать обобщающую способность, поэтому построение ансамблей становится важным направлением для решения практических задач.

Есть два основных шага для построения ансамбля: обучение базовых алгоритмов и объединение их ответов для получения результата. Последний шаг называется корректирующей операцией. Есть несколько видов таких операций: голосование, взвешенное голосование, смесь экспертов, стекинг и другие.

Считается, что на качество ансамбля ключевое влияние оказывают два фактора: точность каждого базового алгоритма и их различия между собой. Часто наблюдалось, что удаление даже самого слабого базового алгоритма негативно влияло на качество ансамбля. Этот факт показывает, что между качеством и различием алгоритмов существует связь. Поэтому обучение точных, но при этом разных алгоритмов является нетривиальной задачей.

В этой работе был предложен новый метод построения ансамбля, где базовыми алгоритмами выступили решающие деревья. И были проведены исследования объединения получающихся базовых алгоритмов с помощью простого и взвешенного голосования.

2 Постановка задачи

В данной работе рассматривается новый метод построения ансамбля для задачи классификации. Для обоснования интуиции предлагаемого метода, можно выписать разложения error-ambiguity decomposition и bias-variance-covariance decomposition. Эти разложения обосновывают верность предположения о том, что для построе-

ния хорошего ансамбля нужны точные и различные базовые алгоритмы. Вывод этих разложений можно также найти, например, в [1].

2.1 Error-Ambiguity Decomposition

Рассмотрим задачу регрессии, которую решает ансамбль с помощью взвешенного усреднения. Ниже $H(x)$ — ответ даваемый ансамблем, $h_i(x)$ — ответ даваемый базовым алгоритмом, w_i — вес с которым ответ базового алгоритма входит в ансамбль. При этом на веса введены ограничения: $w_i \geqslant 0$ и $\sum_{i=1}^{T} w_i = 1$. T — общее число базовых алгоритмов в ансамбле.

$$H(x) = \sum_{i=1}^{T} w_i h_i(x)$$

Вводится понятие неоднозначности (ambiguity) для одного базового алгоритма, а также для всего ансамбля. Для ансамбля это взвешенное усреднение по неоднозначностям каждого алгоритма. Неоднозначность показывает различие ответов базовых алгоритмов при предсказании объекта x.

$$\begin{aligned} ambi(h_i|x) &= (h_i(x) - H(x))^2 \\ \overline{ambi}(h|x) &= \sum_{i=1}^{T} w_i \cdot ambi(h_i|x) \end{aligned}$$

Пусть для измерения качества, будет использоваться квадратичная ошибка. Как $f(x)$ обозначается настоящее значение функции, которую необходимо восстановить, в точке x.

$$\begin{aligned} err(h_i|x) &= (f(x) - h_i(x))^2 \\ err(H|x) &= (f(x) - H(x))^2 \\ \overline{err}(h|x) &= \sum_{i=1}^{T} w_i \cdot err(h_i|x) \end{aligned}$$

Используя выражения для квадратичных ошибок, неоднозначность ансамбля можно представить в другом виде.

$$\begin{aligned}
\overline{ambi}(h|x) &= \sum_{i=1}^{T} w_i \left(h_i(x) - f(x) - H(x) + f(x)\right)^2 = \sum_{i=1}^{T} w_i \Big[err(h_i|x) + err(H|x) - \\
&- 2 \cdot (f(x) - h_i(x)) \cdot (f(x) - H(x)) \Big] = \overline{err}(h|x) + err(H|x) - \\
&- 2 \cdot (f(x) - H(x)) \cdot \left(\sum_{i=1}^{T} w_i \left(f(x) - h_i(x)\right) \right) = \overline{err}(h|x) + err(H|x) - \\
&- 2 \cdot (f(x) - H(x)) \cdot \left(f(x) - \sum_{i=1}^{T} w_i h_i(x) \right) = \overline{err}(h|x) + err(H|x) - \\
&- 2 \cdot err(H|x) = \overline{err}(h|x) - err(H|x)
\end{aligned}$$

Имея выражение, определенное для каждого объекта, можно найти его математическое ожидание. Ниже $p(x)$ — распределение на объекты.

$$\begin{aligned}
\int \overline{ambi}(H|x)p(x)dx &= \int \overline{err}(h|x)p(x)dx - \int err(H|x)p(x)dx \\
\int \left(\sum_{i=1}^{T} w_i \cdot ambi(h_i|x) \right) p(x)dx &= \int \left(\sum_{i=1}^{T} w_i \cdot err(h_i|x) \right) p(x)dx - \int err(H|x)p(x)dx \\
\sum_{i=1}^{T} w_i \int ambi(h_i|x)p(x)dx &= \sum_{i=1}^{T} w_i \int err(h_i|x)p(x)dx - \int err(H|x)p(x)dx
\end{aligned}$$

Известны выражения среднеквадратичного риска для ансамбля и для его базового алгоритма:

$$\begin{aligned}
err(h_i) &= \int err(h_i|x)p(x)dx \\
err(H) &= \int err(H|x)p(x)dx \\
ambi(h_i) &= \int ambi(h_i|x)p(x)dx \\
\overline{err}(h) &= \sum_{i=1}^{T} w_i \cdot err(h_i) \\
\overline{ambi}(h) &= \sum_{i=1}^{T} w_i \cdot ambi(h_i)
\end{aligned}$$

Таким образом, получается итоговое выражение:

$$err(H) = \overline{err}(h) - \overline{ambi}(h)$$

$\overline{ambi}(h)$ по определению является положительным и, при увеличении неоднозначности базовых алгоритмов, повышается качество итогового ансамбля. Также качество повышается, при уменьшении ошибки каждого базового алгоритма.

2.2 Bias-Variance-Covariance Decomposition

Запишем еще одно разложение для ошибки. Пусть h — алгоритм, решающий задачу регрессии, f — прогнозируемая величина. Дальше везде опускается внешнее математическое ожидание по объектам x. Оно всегда берется от всего выражения. Так например вместо выражения:

$$err(h_i) \;=\; \int \left(f(x) - h_i(x)\right)^2 p(x)dx$$

Записывается выражение:

$$err(h_i) \;=\; \left(f - h_i\right)^2$$

Далее знак $\mathbb{E}$ означает математическое ожидание по всевозможным обучающим выборкам фиксированного размера. Перейдём к разложению ошибки:

$$\begin{aligned} err(h) &= \mathbb{E}\left[(h-f)^2\right] = \\ &= \left(\mathbb{E}[h] - f\right)^2 + \mathbb{E}\left[\left(h - \mathbb{E}[h]\right)^2\right] = \\ &= bias(h)^2 + variance(h) \end{aligned}$$

Данное разложение имеет отличия от известного разложения ошибки на шум, смещение(bias) и разброс(variance). Так как шум сложно предсказывать, в этом выражении он был внесён в компоненту смещения, что упрощает запись формулы. Также не было введено совместное распределение $p(x, f)$. Но для демонстрации интуиции построения ансамблей такое разложение имеет место быть.

$$\begin{aligned} bias(h) &= \mathbb{E}[h] - f \\ variance(h) &= \mathbb{E}\left[\left(h - \mathbb{E}[h]\right)^2\right] \end{aligned}$$

Пусть ансамбль имеет, приведённый ниже, вид. Это не ограничивает общности, так как веса могут быть внесены внутрь h_i.

$$H(x) = \sum_{i=1}^{T} h_i(x)$$

Тогда:

$$\begin{aligned}
\overline{bias}(H) &= \frac{1}{T}\sum_{i=1}^{T}\left(\mathbb{E}[h_i] - f\right) \\
\overline{variance}(H) &= \frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\left[\left(h_i - \mathbb{E}[h_i]\right)^2\right] \\
\overline{covariance}(H) &= \frac{1}{T(T-1)}\sum_{i=1}^{T}\sum_{\substack{j=1 \\ j\neq i}}^{T}\mathbb{E}\left[\left(h_i - \mathbb{E}[h_i]\right)\left(h_j - \mathbb{E}[h_j]\right)\right]
\end{aligned}$$

Таким образом, ошибку ансамбля, с учетом всех введённых обозначений, можно представить так:

$$err(H) = \overline{bias}(H)^2 + \frac{1}{T}\overline{variance}(H) + \left(1 - \frac{1}{T}\right)\overline{covariance}(H)$$

В описанном разложении появляется другое описание различности базовых алгоритмов — ковариация между ними. Очевидно, что если два базовых алгоритма совершают одни и те же ошибки, то их ковариация велика. Чем меньше ковариация, тем меньше ошибка.

2.3 Связь двух разложений

Эти два разложения имеют общую связь. Одно может быть переписано через слагаемые другого. Так представление $err(H)$ можно записать двумя способами:

$$\begin{aligned}
\overline{err}(H) - \overline{ambi}(H) &= \mathbb{E}\left[\frac{1}{T}\sum_{i=1}^{T}(h_i - f)^2 - \frac{1}{T}\sum_{i=1}^{T}(h_i - H)^2\right] \\
&= \overline{bias}(H)^2 + \frac{1}{T}\overline{variance}(H) + \left(1 - \frac{1}{T}\right)\overline{covariance}(H)
\end{aligned}$$

После нескольких преобразований получаем итоговые выражения:

$$\begin{aligned}
\overline{err}(H) &= \mathbb{E}\left[\frac{1}{T}\sum_{i=1}^{T}(h_i - f)^2\right] = \overline{bias}(H)^2 + \overline{variance}(H) \\
\overline{ambi}(H) &= \mathbb{E}\left[\frac{1}{T}\sum_{i=1}^{T}(h_i - H)^2\right] = \overline{variance}(H) - variance(H) = \\
&= \overline{variance}(H) - \frac{1}{T}\overline{variance}(H) - \left(1 - \frac{1}{T}\right)\overline{covariance}(H)
\end{aligned}$$

Можно заметить, что средняя дисперсия входит и в $\overline{err}(H)$ и в $\overline{ambi}(H)$. При вычитании дисперсия сокращается. Вхождение средней дисперсии в обе составляющие

ошибки, показывает, что максимизация неоднозначности генерацией как можно более разных алгоритмов, без изменения средней ошибки базовых алгоритмов — непростая задача.

Последнее, рассматриваемое в данной работе, разложение ошибки может быть найдено в [2]. Если вернуться к исходному представлению ансамбля:

$$H = \sum_{i=1}^{T} w_i h_i$$

Задача минимизации ошибки можно свести к:

$$err(H) \quad \rightarrow \quad min \Leftrightarrow \sum_{i=1}^{T} w_i \cdot err(h_i) - \frac{1}{2} \sum_{i=1}^{T} \sum_{j=1}^{T} w_i w_j \mathbb{E}(h_i - h_j)^2 \rightarrow min$$

Для минимизации ошибки ансамбля, необходимо минимизировать ошибки каждого входящего в ансамбль алгоритма. Также для минимизации ошибки следует максимизировать каждую попарную разницу ответов этих алгоритмов.

3 Известные методы ансамблирования

Как уже упоминалось, в машинном обучении ансамбли основаны на объединении нескольких алгоритмов с целью повышения обобщающей способности итогового алгоритма. Достижение такого результата объясняется тем, что ошибки базовых классификаторов сделаны на различных областях признакового пространства, и алгоритмы при объединении компенсируют ошибки друг друга.

Выше было показано, что в ансамблях алгоритмы должны быть как можно более разными и как можно более точными. Чем более точные алгоритмы участвуют в построении ансамбля и чем меньше, наример, их корреляция, тем меньше общая ошибка ансамбля.

Обычно базовыми для ансамблирования выбирают быстро обучаемые модели, например деревья. В этом случае каждый отдельный алгоритм может предсказывать обучающую выборку с очень высокой точностью. Так, например, обучение каждого дерева до конца обеспечивает безошибочную классификацию всей обучающей выборки. Но деревья являются нестабильными алгоритмами и сильно переобучаются, имея большую компоненту разброса. Известно, что ансамблирование устраняет этот недо-

статок. Примером может служить случайный лес. Наиболее известными методами ансамблирования являются: бэггинг и бустинг.

3.1 Бэггинг

Бэггинг (bootstrap aggregating) - метод построения ансамбля, предложенный Лео Брейманом [3] в 1994 году. Суть метода состоит в обучении базовых алгоритмов на случайно сгенерированных из начальной выборки подвыборок. Каждая подвыборка получается с помощью бутстрапа.

Из обучающей выборки размера N выбираются N объектов с возвращением, в результате чего некоторые объекты дублируются несколько раз, а некоторые вообще не попадают в новую выборку ни разу. Затем базовые алгоритмы обучаются на созданных выборках и объединяют свои предсказания, например, с помощью голосования.

Замечено, что такой ансамбль, как случайный лес, построенный с помощью бэггинга на основе нестабильных, сильно переобучающихся алгоритмов, получается стабильным.

3.2 Случайный лес

Случайный лес (random forest) - алгоритм машинного обучения. Первая его версия была предложена Тин Кам Хо в работе [4], и была основана на методе случайных подпространств (random subspace method, RSM). Этот метод похож на бэггинг, только строятся подвыборки не из объектов, а из признаков.

В случайном лесе метод случайных подпространств используется при построении каждого узла. В каждом узле генерируется случайная подвыборка из m признаков, и из них выбирается оптимальный для разбиения по некоторому критерию.

Окончательный вид, участвующий в экспериментах в данной работе, был предложен Лео Брейманом в [5]. В этой версии случайный лес сразу использует при построении и бэггинг и метод случайных подпространств. Из описания следует, что одно из сильных преимуществ случайного леса — скорость и простота. Этот метод имеет мало гиперпараметров, а также каждое его базовое дерево может быть построено независимо от других, поэтому часто этот процесс распараллеливается.

Обычно в задачах классификации деревья доучивают до конца. Поэтому итоговая классификация решающего леса — простое голосование по всем базовым деревьям. Если же дерево не было дообучено до конца, то можно устроить взвешенное голосование, где вес каждого дерева — это уверенность алгоритма в своём ответе. Реализацию этого алгоритма можно найти в [9].

3.3 Бустинг

Бустинг [6] — это способ построения ансамбля алгоритмов машинного обучения, в котором каждый следующий алгоритм стремится компенсировать недостатки объединения всех предыдущих алгоритмов.

Общий вид градиентного бустинга, частным случаем которого является, описанный ниже, адаптивный бустинг, можно найти в [7]. Итоговый ансамбль можно описать выражением:

$$C^M(x) = \sum_{m=1}^{M} b_m h(x, a_m),\ b_m \in \mathbb{R},\ a_m \in \mathbb{A}$$

В этом выражении M — число базовых алгоритмов. Каждый базовый алгоритм, является элементом некоторого семейства моделей $h(x, a) \in H : X \to \mathbb{R}$ и задаётся вектором параметров $a \in \mathbb{A}$.

Нахождение параметров b_m, a_m осуществляется только после построения C^{m-1}. Для этого вводится функционал $Q = \sum\limits_{i=1}^{N} L\left(y_i, C^{m-1}(x_i)\right)$ и берётся градиент:

$$\nabla Q = \left[\frac{\partial Q}{\partial C^{m-1}(x_i)}\right]_{i=1}^{N} = \left[\frac{\partial\left(\sum\limits_{j=1}^{N} L\left(y_j, C^{m-1}(x_j)\right)\right)}{\partial C^{m-1}(x_i)}\right]_{i=1}^{N} = \left[\frac{\partial L\left(y_i, C^{m-1}(x_i)\right)}{\partial C^{m-1}(x_i)}\right]_{i=1}^{N}$$

В таком случае поиск a_m, b_m осуществляется так:

$$\begin{aligned} a_m &= \underset{a \in \mathbb{A}}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(\nabla Q_i, h(x_i, a)\right) \\ b_m &= \underset{b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, C^{m-1}(x_i) + b_m h(x_i, a_m)\right) \end{aligned}$$

AdaBoost (адаптивный бустинг) - один из первых алгоритмов бустинга, и один из самых популярных, он был предложен Йоавом Фройндом и Робертом Шапиро.Суть

AdaBoost, состоит во взвешивании объектов обучающей выборки таким образом, чтобы каждый следующий классификатор старался преимущественно как можно точнее классифицировать объекты, плохо классифицируемые предыдущими базовыми алгоритмами.

Также от ошибки базового классификатора зависит его вес при взвешенном голосовании, которое проводится при итоговом предсказании. В данной работе использовался AdaBoost с многоклассовым обобщением SAMME.R (Stagewise Additive Modeling using a Multi-class Exponential loss function Real) [8]. Реализацию этого алгоритма можно найти в [9].

Базовым алгоритмом AdaBoost может быть любой метод машинного обучения, поддерживающий взвешивание объектов. В данной работе для экспериментов базовым алгоритмом было выбрано решающее дерево.

4 Предложенный метод

Метод, предложенный для ансамблирования деревьев в данной работе, опирается на факт, что базовые алгоритмы ансамбля должны быть как можно более разными, совершая ошибки на разных объектах признакового пространства. Метод имеет две реализации с названиями **алгоритм 1** и **алгоритм 2**.

Реализации отличаются друг от друга правилами объединения деревьев в ансамбль и правилами нахождения пары признак-порог для узла дерева. Каждая пара признак-порог участвует в задании общего правила, по которому объекты, попадающие на вход дереву, распределяются по его листовым узлам.

Сначала для исследования был предложен **алгоритм 1**. Этот метод имел множество недостатков, которые были исправлены во втором алгоритме. **Алгоритм 2** можно считать улучшением **алгоритма 1**, а результаты сравнения с другими методами ансамблирования приведены только для **алгоритма 2**.

В отличие от случайного леса, построение деревьев предлагаемого алгоритма детерминировано, если не использовать процедуру бэггинга. В приведенных ниже экспериментах, предлагаемый метод использует процедуру бэггинга, так как это даёт небольшой прирост в качестве.

Обучение каждого следующего дерева опирается на уже построенные, но не прямо направлено на исправление их ошибок, в отличие от AdaBoost. Из приведенных ниже экспериментов следует, что различные в построении ансамбли, могут совершать различные ошибки, и также могут быть объеденины в один алгоритм, что может улучшить качество решения задачи.

4.1 Обозначения

- N — число объектов в выборке, D — размерность признакового пространства.
- $(x_1, y_1), ..., (x_N, y_N)$ — обучающая выборка.
- $x_i \in R^D$ — объект выборки, x_i^j — j-й признак объекта.
- y_i — метка класса для объекта x_i, принимает значения из множества $\{1, 2, ..., K\}$. Также используется выражение $y(x)$ — обозначает реальную метку, соответствующую объекту x в выборке.
- K — число классов в задаче классификации.
- $Node$ — множество объектов в текущем узле, для которого идёт поиск признака и порога по нему.
- T^M — дерево-классификатор, построенное на M-м шаге. Для дерева могут быть определены специальные веса. Вес можно интерпретировать как уверенность дерева в соответствии объекту x_i метки k. Пусть веса обозначаются $\hat{p}(k|T^M, x)$, и пусть $Leaf$ — множество объектов в листовом узле дерева T^M, в котором оказался объект x после построения. Тогда для вычисления веса используется следующая формула:

$$\hat{p}(k|T^M, x) = \frac{1}{|Leaf|} \sum_{\hat{x} \in Leaf} \mathbb{I}[y(\hat{x}) = k] \tag{1}$$

Такой же вес использует при голосовании случайный лес, если базовое дерево не было достроено до конца. Также далее используется выражение:

$$T^M(x) = \underset{k \in 1,...,K}{\operatorname{argmax}} \hat{p}(k|T^M, x) \tag{2}$$

- C^M — ансамбль построенный на M-м шаге. Объединяет голоса M деревьев с помощью простого (**алгоритм 1**) или взвешенного голосования (**алгоритм 2**).

  Для простого голосования используется выражение:

$$C^M(x) = \underset{k \in 1,\dots,K}{\operatorname{argmax}} \sum_{m=1}^{M} \mathbb{I}\left[T^m(x) = k\right] \tag{3}$$

  Для взвешенного вводятся веса:

$$\hat{p}(k|C^M, x) = \frac{1}{M} \sum_{m=1}^{M} \hat{p}(k|T^m, x) \tag{4}$$

  А также используется выражение:

$$C^M(x) = \underset{k \in 1,\dots,K}{\operatorname{argmax}} \hat{p}(k|C^M, x) \tag{5}$$

- λ — коэффициент "влияния" предыдущих деревьев на построение.

4.2 Критерий разбиения в узле

Основой алгоритма решающего дерева является критерий для нахождения оптимального разбиения в узле. Разбиение объектов, попадающих в узел, организуется по одному из признаков. Выбирается порог. Объекты со значением признака, большим значения порога, оказываются в правом поддереве, иначе в левом.

Классификация объекта, попавшего в один из листовых узлов дерева, происходит на основе меток объектов обучающей выборки, попавших в этот листвой узел при построении дерева.

Ниже, согласно введенным ранее обозначениям, приведена общая формула для задачи нахождения признака и его порога для алгоритмов: простого решающего дерева, **алгоритма 1** и **алгоритма 2**. По данному порогу объекты попавшие в узел будут разбиты на две части.

$$\begin{cases} S_l = \{x \in Node | x^d <= \tau\} \\ S_r = \{x \in Node | x^d > \tau\} \\ F(\tau, d) = \frac{|S_l|}{|S_l|+|S_r|} H(S_l) + \frac{|S_r|}{|S_l|+|S_r|} H(S_r) \\ (\tau, d) = \underset{\tau,d}{\operatorname{argmin}}(F(\tau, d)) \end{cases} \tag{6}$$

Отличие алгоритмов в реализации $H(S)$. Для решающего дерева это просто энтропия.

$$\begin{cases} p(k|S,Y) = \frac{1}{|S|} \sum\limits_{x \in S} \mathbb{I}[y(x) = k] \\ H(S) = -\sum_{k \in K} p(k|S,Y) \cdot \ln p(k|S,Y) \end{cases} \tag{7}$$

В формулах $H(S)$ **алгоритма 1** и **алгоритма 2** появляется второе слагаемое. Через это слагаемое на построение дерева влияет уже построенный ансамбль. Второе слагаемое можно назвать отрицанием энтропии ансамбля предыдущего шага.

Порог и признак, выбираются так, чтобы максимизировать энтропию построенного ансамбля и минимизировать энтропию реальных откликов. Интуиция такого критерия состоит в том, что разбиение объектов в узле, должно создать такой путь для конечного решения дерева, где построенный ансамбль плохо разделяет объекты. Первое же слагаемое направлено на создание новым деревом хорошего разделения объектов в этом пути. Понятно, что именно второе слагаемое, отвечает за различность базовых алгоритмов.

Алгоритм 1:

$$\begin{cases} p(k|S,Y) = \frac{1}{|S|} \sum\limits_{x \in S} \mathbb{I}[y(x) = k] \\ p(k|S,C^{m-1}) = \frac{1}{|S|} \sum\limits_{x \in S} \mathbb{I}[C^{m-1}(x) = k], \text{ где } C^{m-1} \text{ из (3)} \\ H(S) = -\sum\limits_{k \in K} p(k|S,Y) \cdot \ln p(k|S,Y) + \\ \qquad + \lambda \cdot \sum\limits_{k \in K} p(k|S,C^{m-1}) \cdot \ln p(k|S,C^{m-1}) \end{cases} \tag{8}$$

Алгоритм 2:

$$\begin{cases} p(k|S,Y) = \frac{1}{|S|} \sum\limits_{x \in S} \mathbb{I}[y(x) = k] \\ p(k|S,C^{m-1}) = \frac{1}{|S|} \sum\limits_{x \in S} \hat{p}(k|C^{m-1},x), \text{ где } \hat{p}(k|C^{m-1},x) \text{ из (4)} \\ H(S) = -\sum_{k \in K} p(k|S,Y) \cdot \ln p(k|S,Y) + \\ \qquad + \lambda \cdot \sum_{k \in K} p(k|S,C^{m-1}) \cdot \ln p(k|S,C^{m-1}) \end{cases} \tag{9}$$

4.3 Структура построения ансамбля

Для определенных выше формул критериев для **алгоритма 1** и **алгоритма 2**, можно написать общую структуру построения ансамбля. Первый алгоритм в ансамбле — обычное решающее дерево

Algorithm Общая структура алгоритмов 1 и 2

1: Подготовка с помощью бэггинга обучающей выборки X_1.

2: Обучение T^1 — обычного решающего дерева на выборке X_1.

3: $C^1 \leftarrow T^1$

4: **Для** $m = 2, ..., M$

5: Подготовка с помощью бэггинга обучающей выборки X_m.

6: Обучение T^m на X_m согласно соответствующему критерию $H(S)$ алгоритма.

7: $C^m \leftarrow T^m$

8: $C^M(x)$ — итоговое предсказание для объекта x, соответствует формуле (3) для **алгоритма 1** и формуле (5) для **алгоритма 2**.

4.4 Различия предложенных алгоритмов

Для обоих алгоритмов оказалось, что второе слагаемое позволяет строить до конца только очень глубокие деревья с малым числом объектов в листовых узлах. К примеру, если максимальная глубина нахождения листового узла стандартного решающего дерева не превышала 10, то глубина достроенного до конца с помощью **алгоритма 1** дерева второй итерации превышала 100. Тогда было предложено ограничивать глубину дерева с помощью гиперпараметра.

Алгоритм 1 оказался довольно нестабильным: его гиперпараметры было довольно сложно подобрать. Также возникла проблема из-за недостаточной глубины генерируемых деревьев. Например, средняя глубина дерева в случайном лесе, с обученными до конца деревьями, на выборке оказалась равной 10. **Алгоритм 1** на той же выборке достигал лучшего качества при гиперпараметре ограничения глубины равном 5. Генерируя неглубокие деревья, алгоритм значительно уступал в качестве случайному лесу. Необходимо было решить эту проблему.

Так как обучение базовых деревьев невозможно было довести до конца, в листовых узлах оставались объекты из разных классов. Это позволило перейти к взвешенному голосованию (5). Обучение стало более стабильным, слишком слабые классификаторы не так сильно мешали при итоговом голосовании. Но качество выросло не сильно и глубина деревьев по-прежнему оказывалась значительно меньше, чем у случайного леса.

Как упоминалось выше, второе слагаемое в выражении (8) для **алгоритма 1** является энтропией ансамбля предыдущего шага. Так как ансамбль начал использовать взвешенное голосование, было предложено поменять и формулу второго слагаемого. А точнее вычисление вероятностей классов, входящих в неё. Если в **алгоритме 1** для энтропии ансамбля вероятность класса $p(k|S, C^m)$— это отношение числа объектов помеченных ансамблем как класс k к общему числу объектов в узле. То в **алгоритме 2** для энтропии ансамбля (9) вероятность класса $p(k|S, C^m)$ — это отношение суммы весов $\hat{p}(k|C^m, x)$ из (4) по всем объектам в узле к общему числу объектов в узле.

Благодаря таким изменениям деревья получались глубже, и качество выросло. Ансамбль начал достигать наилучшего качества с гиперпараметром ограничения глубины примерно таким же, как средняя глубина неограниченных деревьев случайного леса. Ниже приведены эксперименты с участием **алгоритма 2**, AdaBoost и случайного леса.

5 Вычислительные эксперименты

Для экспериментов были выбраны шесть задач классификации. Ниже приведена информация о них. В последней задаче используется три различные выборки.

Название задачи	Число объектов тренировочной выборки	Число объектов тестовой выборки	Количество признаков	Количество классов
Классификация рукописных цифр	5620	–	64	10
Кредитный скоринг	1000	–	24	2
Классификация стекла	214	–	9	6
Распознование мин	208	–	60	2
Классификация силуэта машины	846	–	18	4
Систолическое давление (Выборка 1)	718	458	116	2
Систолическое давление (Выборка 2)	373	221	116	2
Систолическое давление (Выборка 3)	832	495	116	2

Таблица 1: Информация о выборках

Первые пять задач были взяты из репозитория UCI (University of California, Irvine) Machine Learning Repository [10]. И для сравнения качества на этих задачах запускалась перекрёстная проверка на поделенном пополам наборе данных с пятью перезапусками. Итоговое качество усреднялась по этим десяти проведенным измерениям.

В задаче систолического давления для измерения качества работы алгоритмов использовалось их качество на тестовой выборке. В задачах бинарной классификации помимо точности вычислялась ROC-AUC (area under receiver operating characteristic curve). В каждом эксперименте для объединения двух методов использовалось взвешенное голосование по объединению базовых алгоритмов. Каждая итерация — обучение и добавление нового дерева.

5.1 Классификация рукописных цифр

Первая задача — классификация изображений рукописных цифр. В этой задаче есть десять классов, каждый соответствует цифре от нуля до девяти.

Можно заметить, что лучшего качества достиг AdaBoost. Его объединение с предложенным алгоритмом не дает прироста в качестве. В отличие от объединения со случайным лесом. В данной задаче отключение процедуры бэггинга, давало значительный прирост качества случайному лесу, и наоборот негативно влияло на качество **алгоритма 2**. Это подтверждает серьёзные различия между этими двумя ансамблями.

Средняя глубина деревьев случайного леса составила 13, такое же ограничение на глубину оказалось оптимальным для **алгоритма 2**. В качестве значения для гиперпараметра "влияния" , было выбрано $\lambda = 2.7$. Что касается гиперпараметров алгоритма AdaBoost, интересным оказалось оптимальное значение ограничения глубины, оно приняло значение 9, что значительно отличается от остальных алгоритмов.

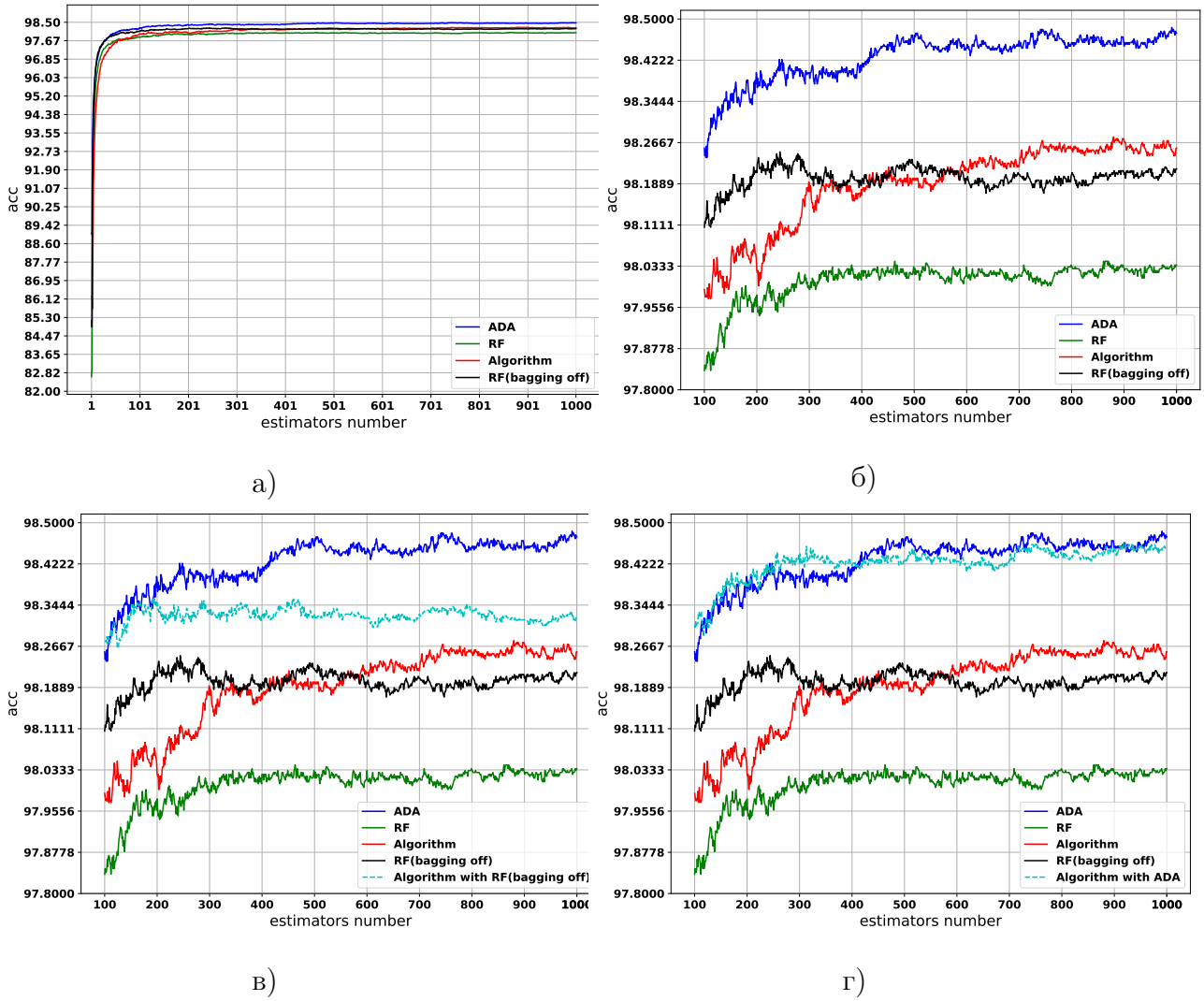


Рис. 1: Классификация рукописных цифр. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost и чёрный — случайный лес без процедуры bagging. **б)** Тот же график что и в пункте а), но начиная с итерации 100. **в)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и случайного леса без процедуры bagging. **г)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и AdaBoost.

5.2 Кредитный скоринг

Вторая задача — оценки кредитоспособности клиентов банка "Statlog (German Credit Data) Data Set". В этой задаче два класса: оценки "хорошо" и "плохо".

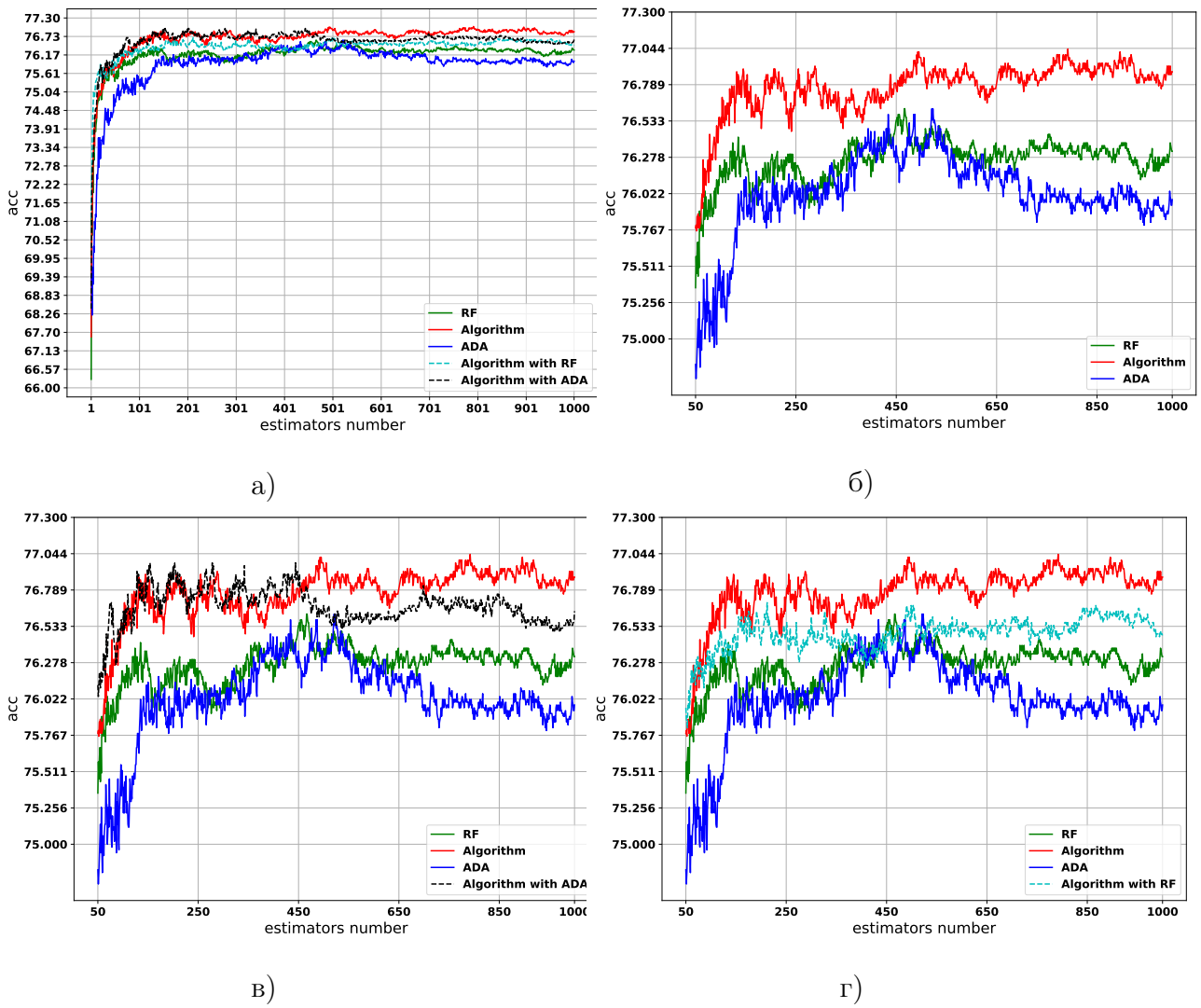


Рис. 2: Кредитный скоринг. Точность. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost, чёрный — объединение **алгоритма 2** и AdaBoost, голубой — объединение **алгоритма 2** и случайного леса. **б)**, **в)** и **г)** Тот же график, начиная с итерации 50 для разных ансамблей, отмеченных теми же цветами.

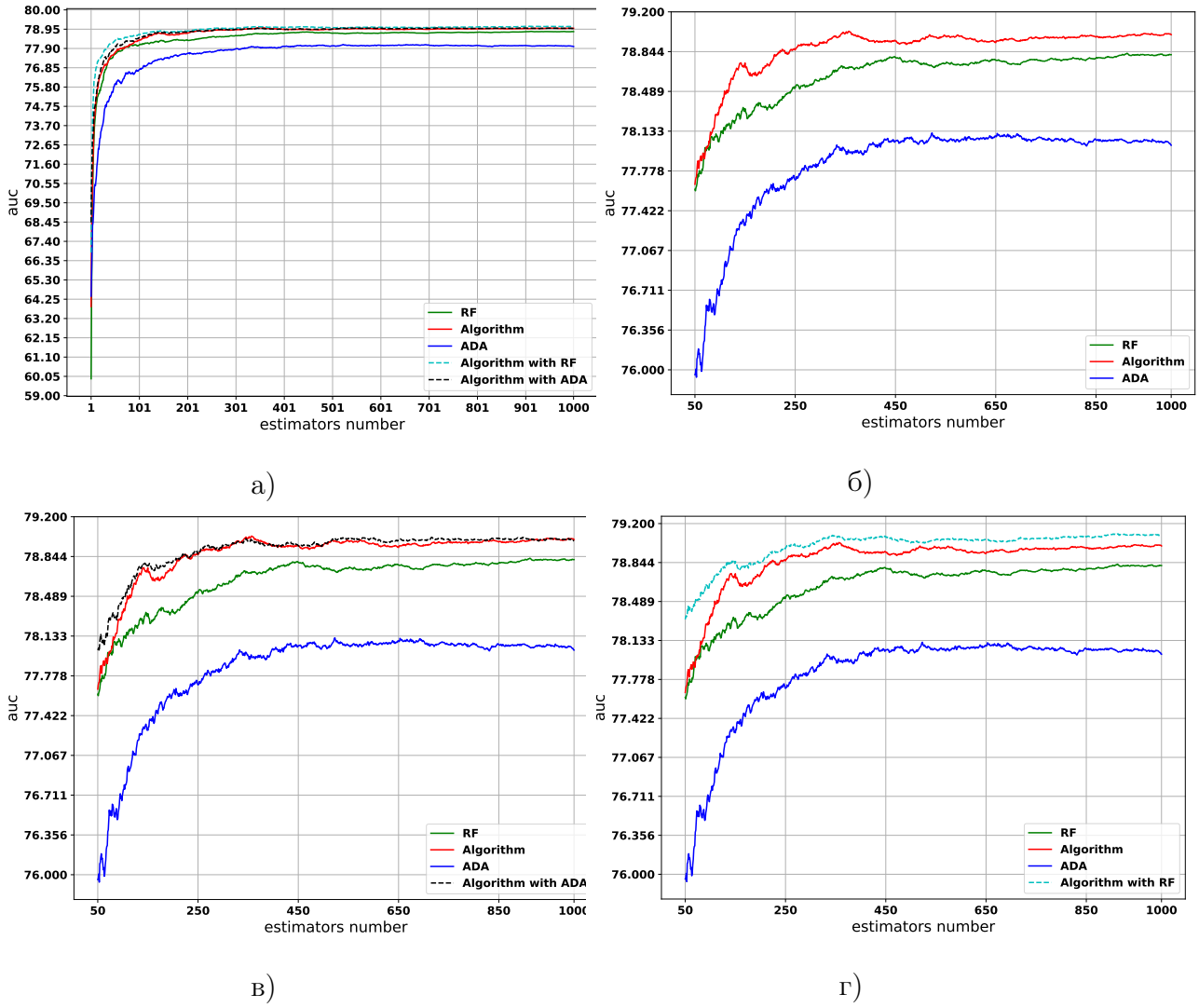


Рис. 3: Кредитный скоринг. AUC-ROC. **а)**Зависимость качества от числа итераций. AUC-ROC разных ансамблей обозначен разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost, чёрный — объединение **алгоритма 2** и AdaBoost, голубой — объединение **алгоритма 2** и случайного леса. **б)**, **в)** и **г)** Тот же график, начиная с итерации 50 для разных ансамблей, отмеченных теми же цветами.

5.3 Классификация стекла

Третья задача — задача классификации стекла по его составу.

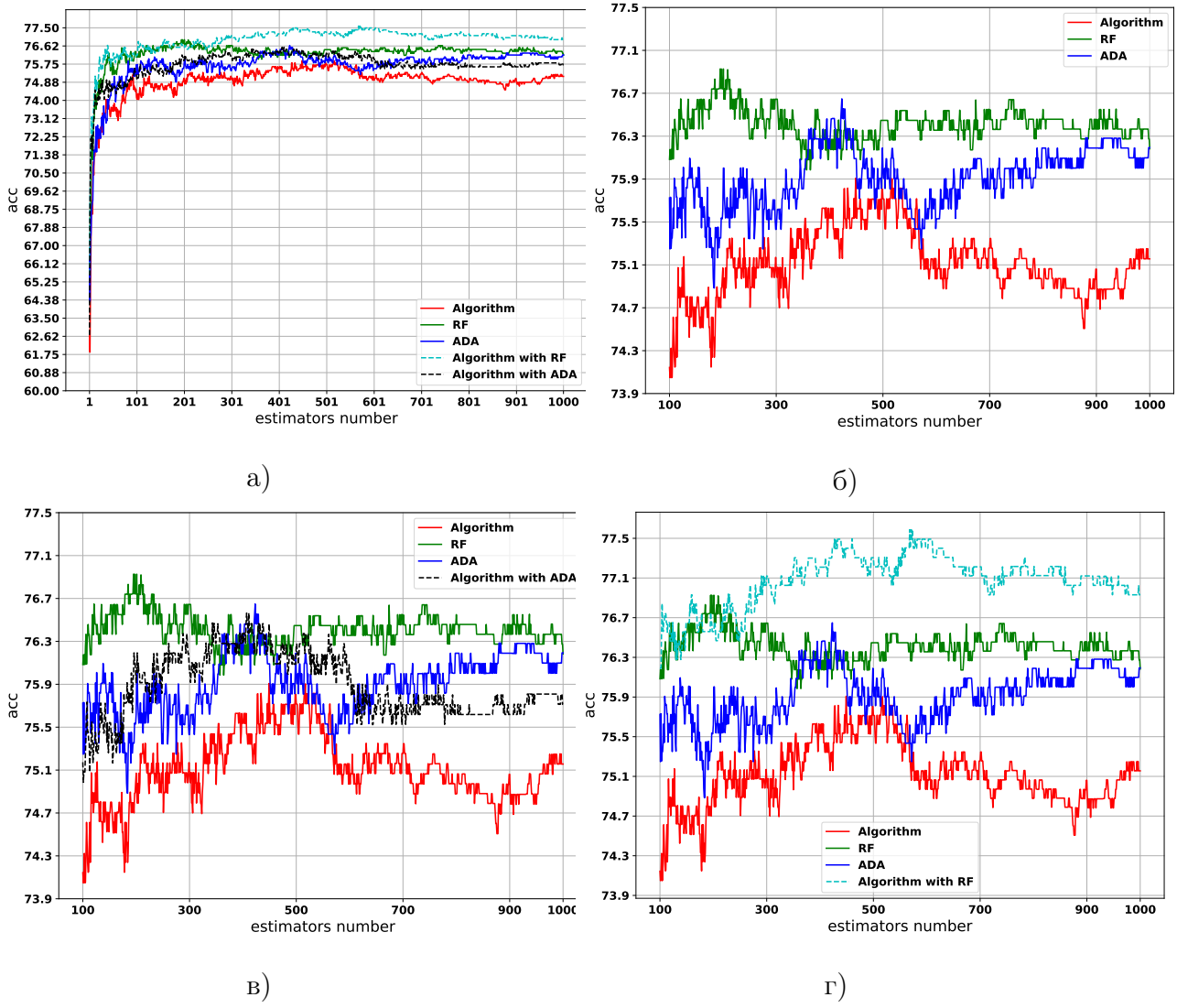


Рис. 4: Классификация стекла. Точность. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost, чёрный — объединение **алгоритма 2** и AdaBoost, голубой — объединение **алгоритма 2** и случайного леса. **б)**, **в)** и **г)** Тот же график, начиная с итерации 50 для разных ансамблей, отмеченных теми же цветами.

5.4 Распознование мин

Четвёртая задача — распознвания мин по сигналу гидролокатора. В этой задаче два класса: "камень" и "мина".

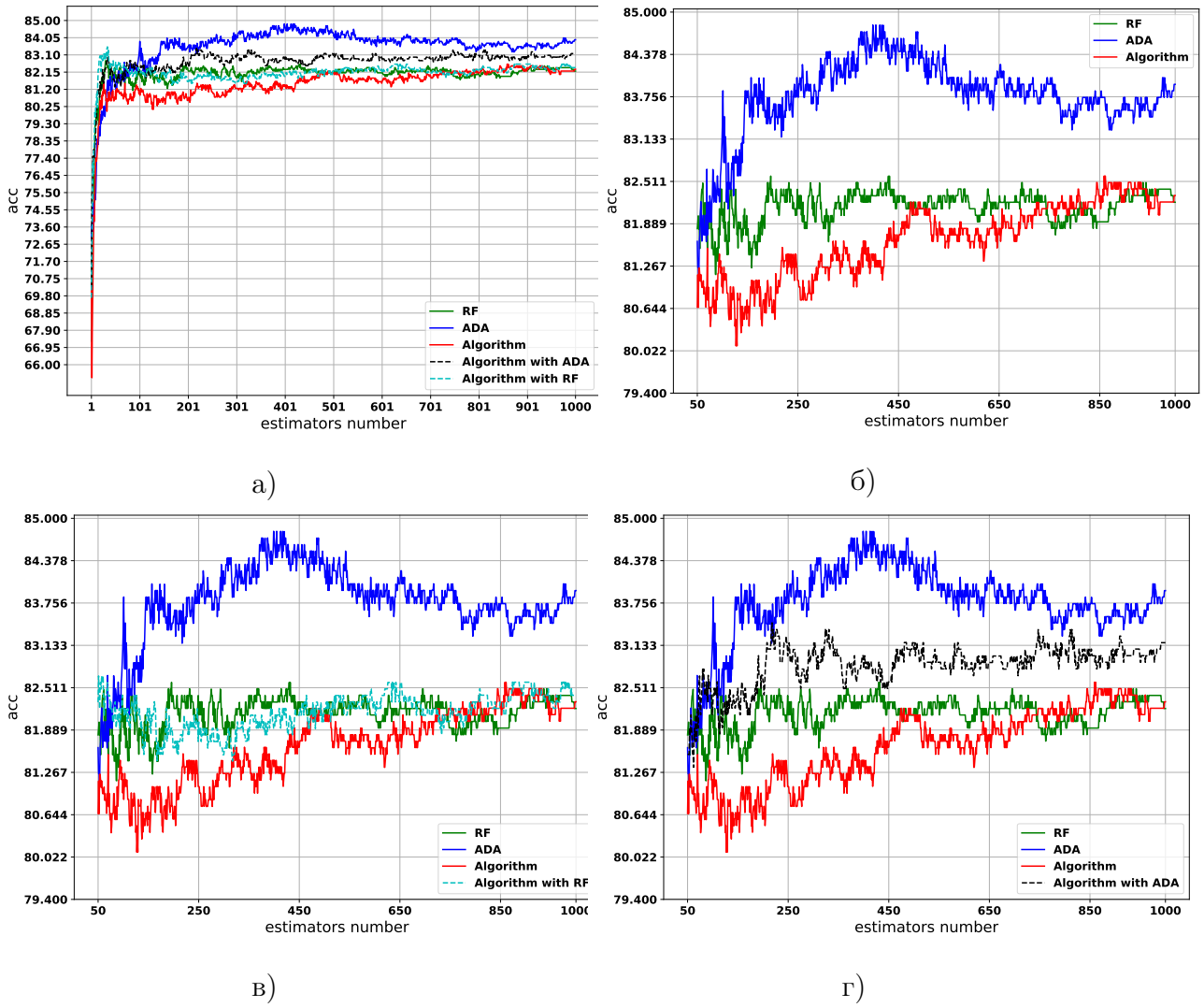


Рис. 5: Распознование мин. Точность. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost, чёрный — объединение **алгоритма 2** и AdaBoost, голубой — объединение **алгоритма 2** и случайного леса. **б)**, **в)** и **г)** Тот же график, начиная с итерации 50 для разных ансамблей, отмеченных теми же цветами.

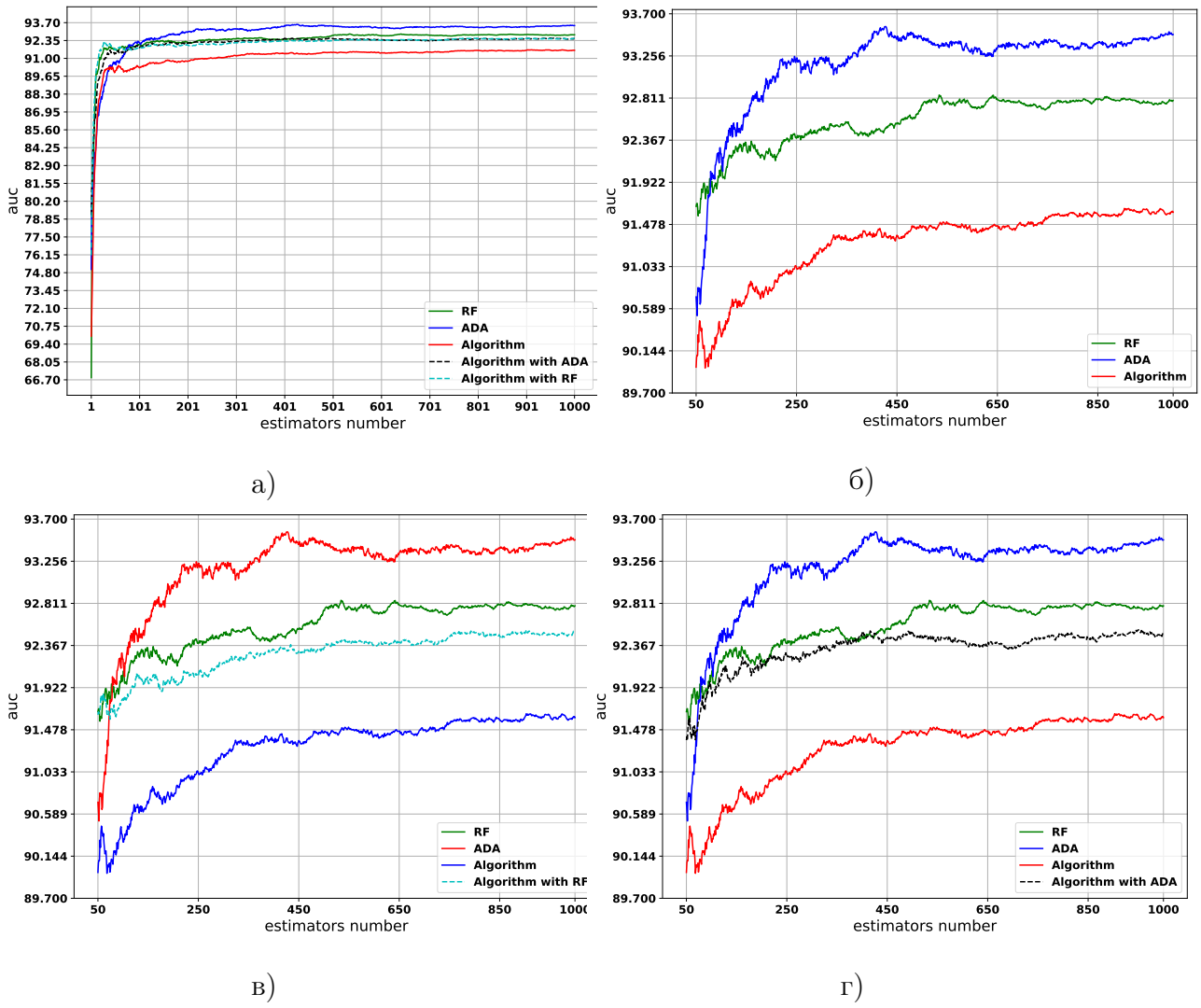


Рис. 6: Распознование мин. AUC-ROC. **а)**Зависимость качества от числа итераций. AUC-ROC разных ансамблей обозначен разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost, чёрный — объединение **алгоритма 2** и AdaBoost, голубой — объединение **алгоритма 2** и случайного леса. **б)**, **в)** и **г)** Тот же график, начиная с итерации 50 для разных ансамблей, отмеченных теми же цветами.

5.5 Классификация силуэта машины

Пятая задача — классификация типа машины по её силуэту.

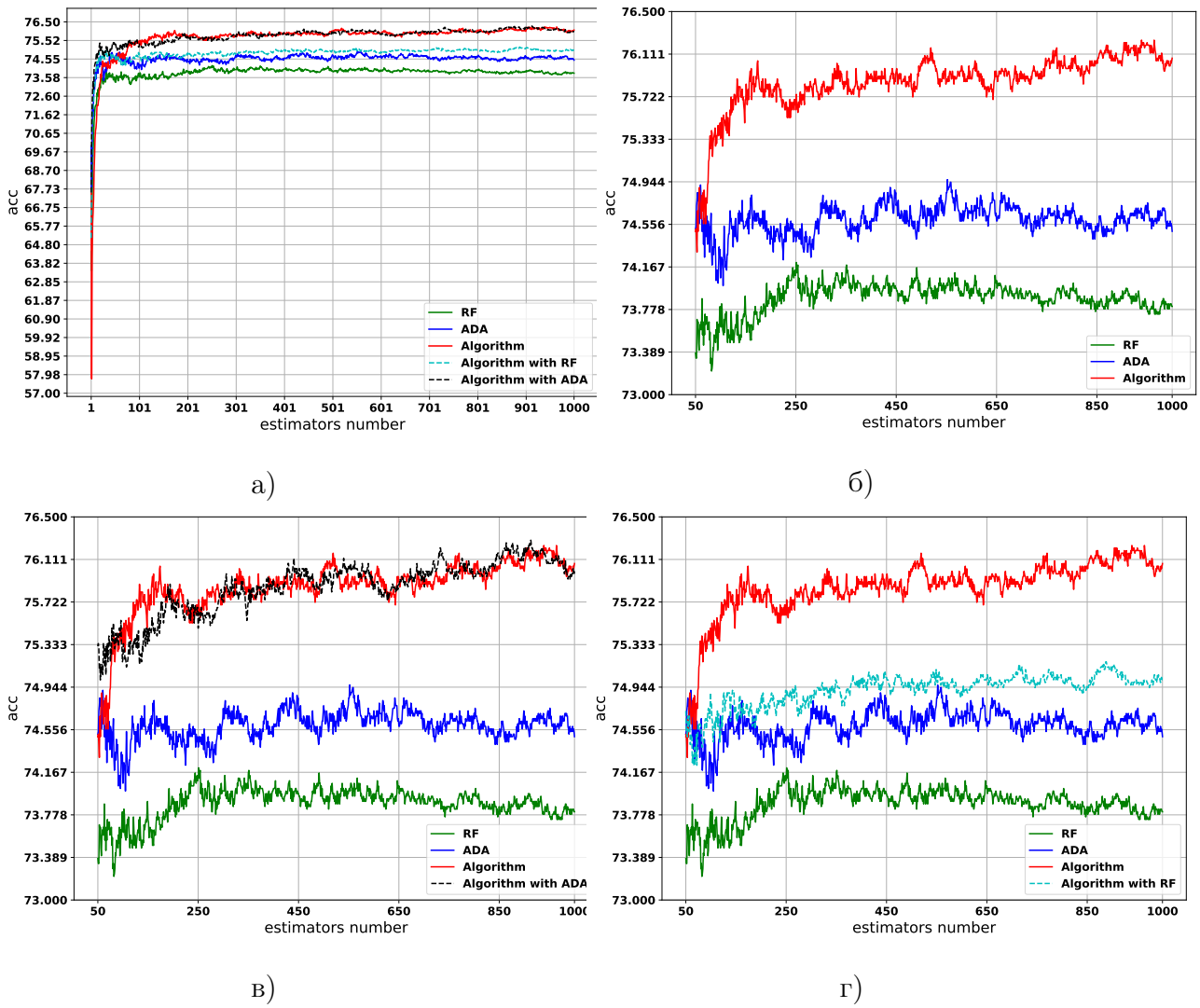


Рис. 7: Классификация силуэта машины. Точность. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost, чёрный — объединение **алгоритма 2** и AdaBoost, голубой — объединение **алгоритма 2** и случайного леса. **б)**, **в)** и **г)** Тот же график, начиная с итерации 50 для разных ансамблей, отмеченных теми же цветами.

5.6 Систолическое давление

Шестая задача — распознавания случаев повышенного систолического давления. Значение систолического давления показывает величину давления в артериях в момент, когда сердце сжимается и выталкивает кровь в артерии. В данной задаче в качестве признаков использовались коэффициенты спектрального разложения сигнала, а также другие признаки, характеризующие геометрическую форму сигнала пульсовой волны. В этой задаче имеется 3 различных выборки, на каждой решается бинарная задача классификации.

Ниже приведены результаты по каждой выборке.

Таблица 2: некоторые гиперпараметры алгоритмов и средняя глубина случайного леса.

номер выборки	Средняя глубина дерева в случайном лесе	Ограничение на глубину в **алгоритме 2**	Ограничение на глубину в AdaBoost	Значение λ в **алгоритме 2**
1	8	13	5	1.75
2	9	13	6	1.0
3	12	11	5	2.0

5.6.1 Выборка 1

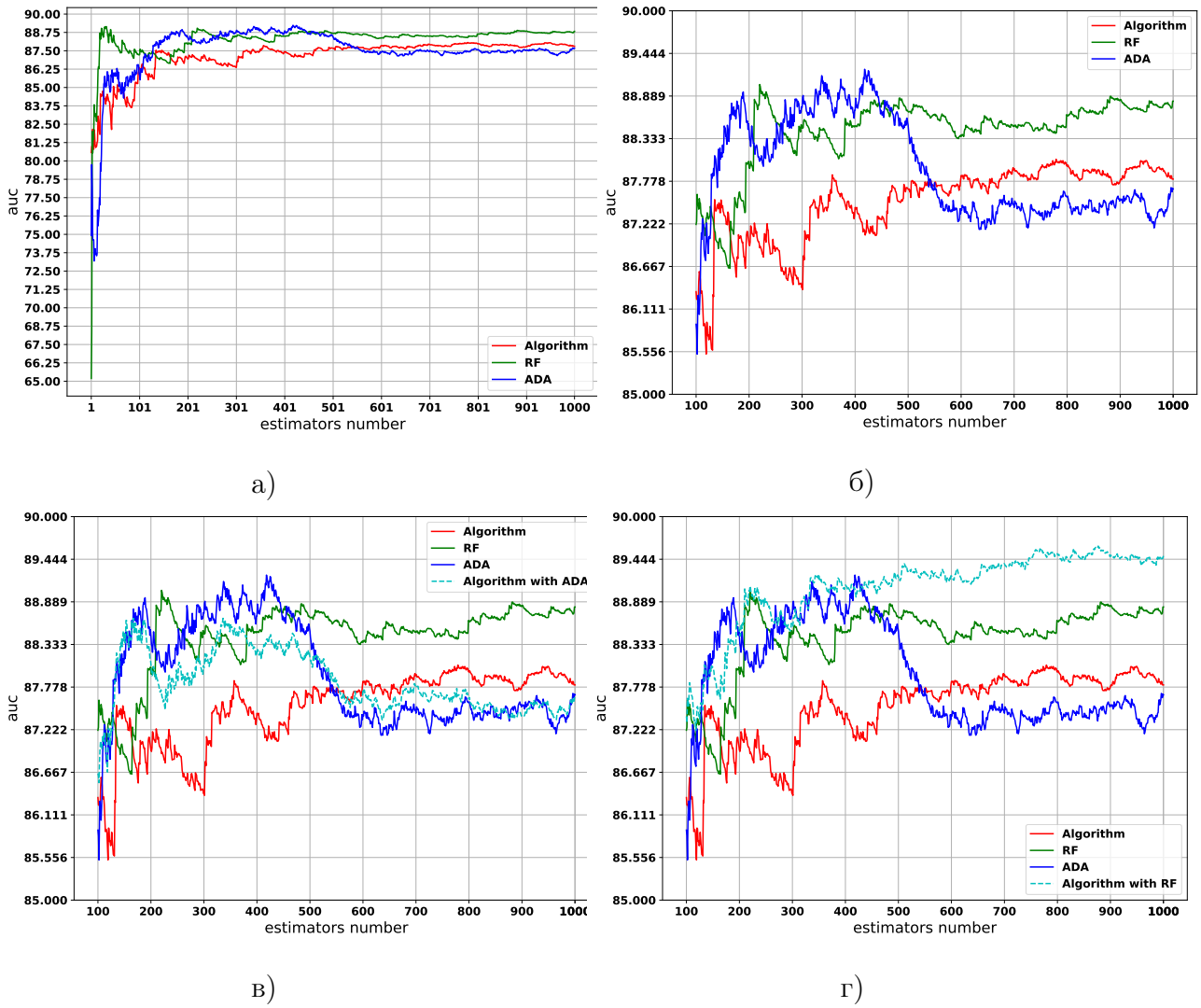


Рис. 8: Систолическое давление. Выборка 1. ROC-AUC. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost. **б)** Тот же график что и в пункте а), но начиная с итерации 100. **в)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и AdaBoost. **г)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и случайного леса.

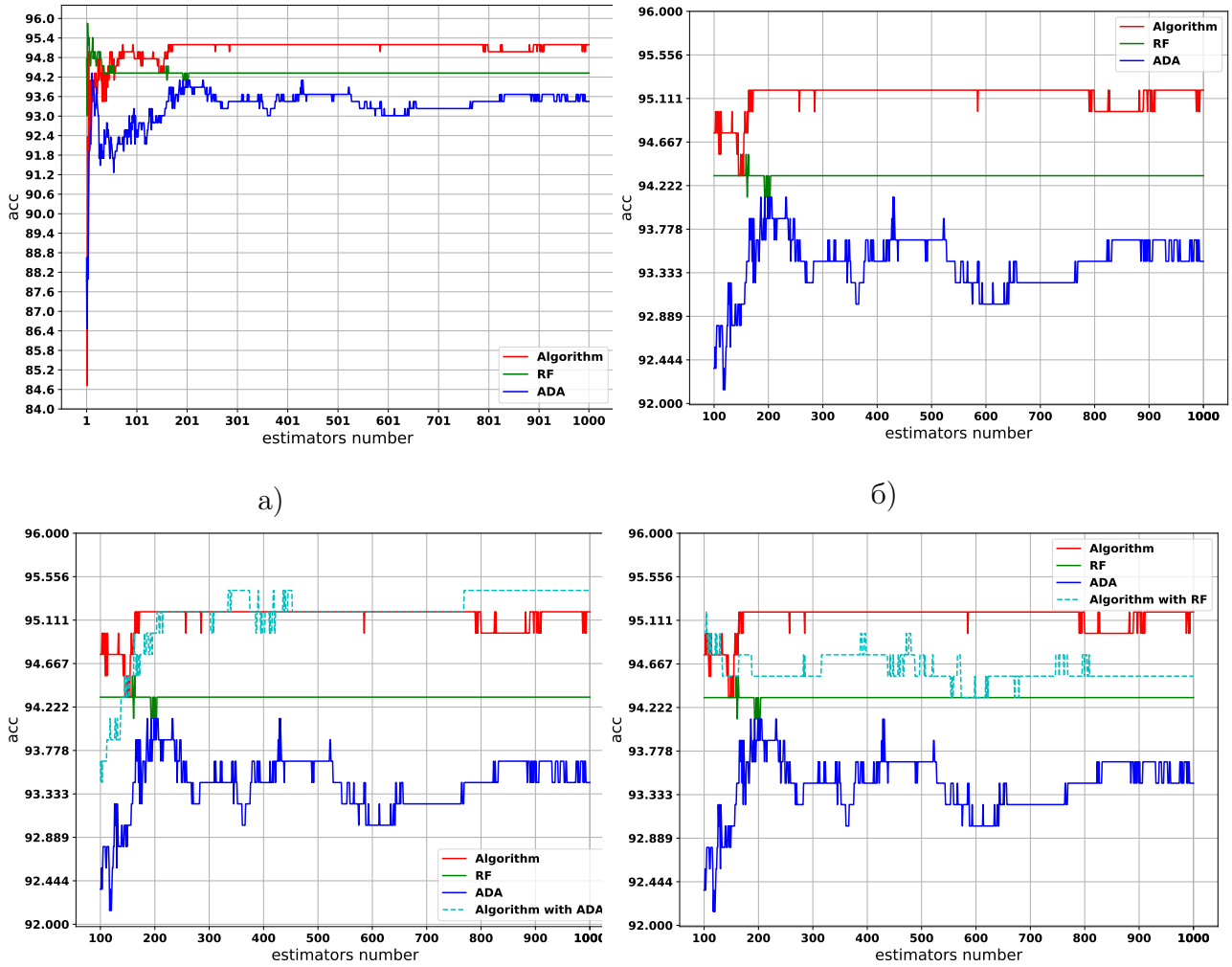


Рис. 9: Систолическое давление. Выборка 1. Точность. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost. **б)** Тот же график что и в пункте а), но начиная с итерации 100. **в)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и AdaBoost. **г)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и случайного леса.

5.6.2 Выборка 2

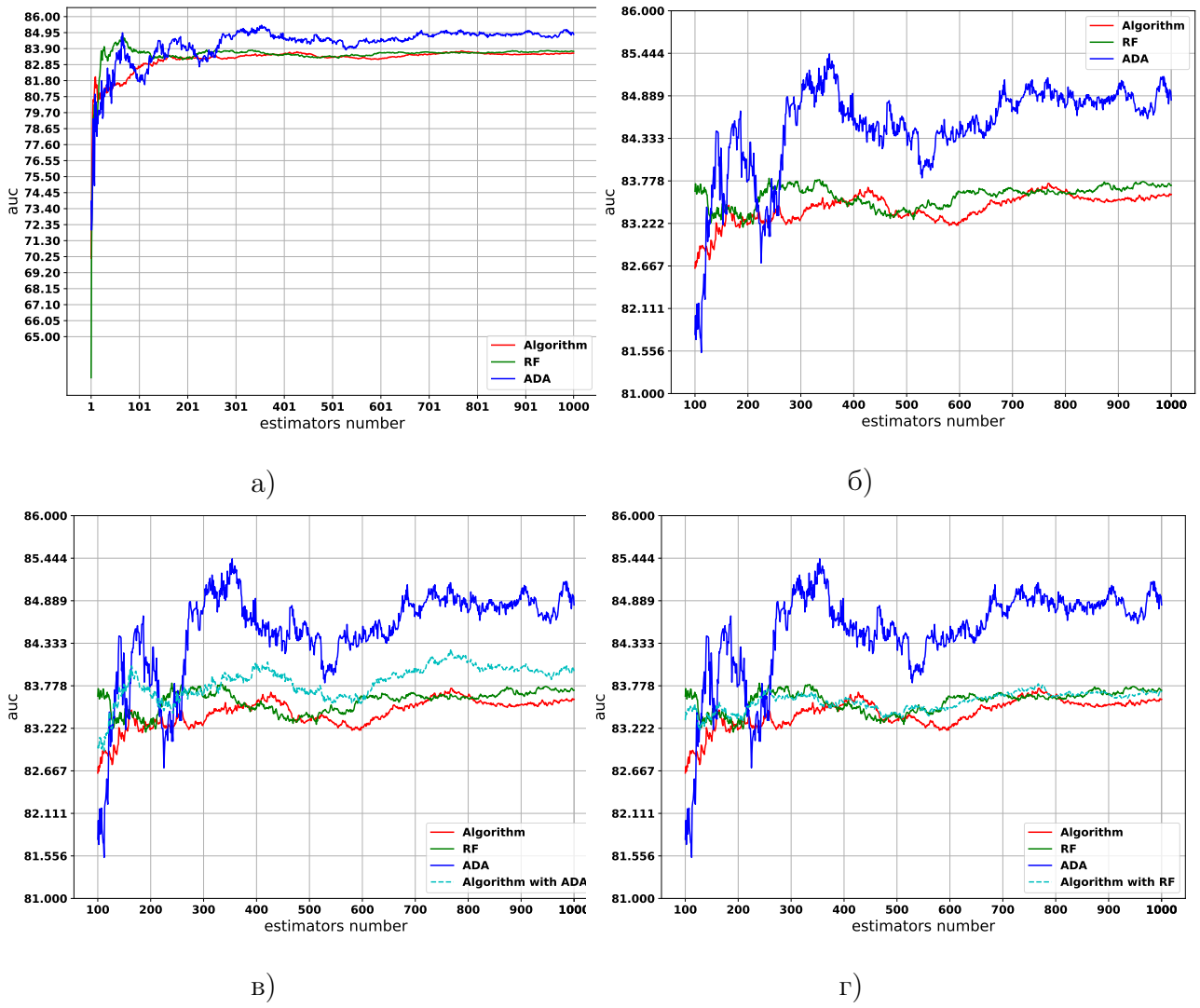


Рис. 10: Систолическое давление. Выборка 2. ROC-AUC. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost. **б)** Тот же график что и в пункте а), но начиная с итерации 100. **в)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и AdaBoost. **г)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и случайного леса.

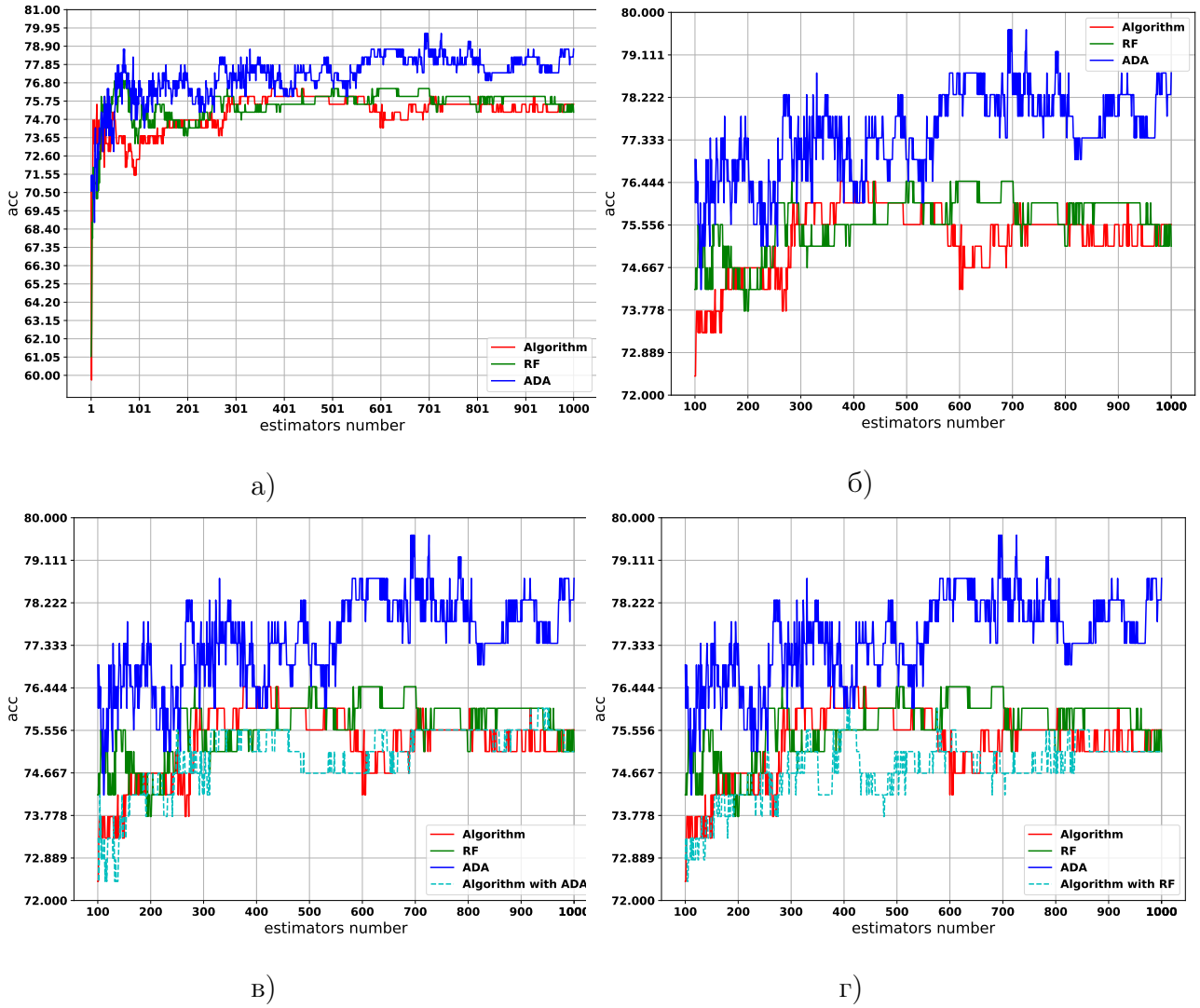


Рис. 11: Систолическое давление. Выборка 2. Точность. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost. **б)** Тот же график что и в пункте а), но начиная с итерации 100. **в)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и AdaBoost. **г)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и случайного леса.

5.6.3 Выборка 3

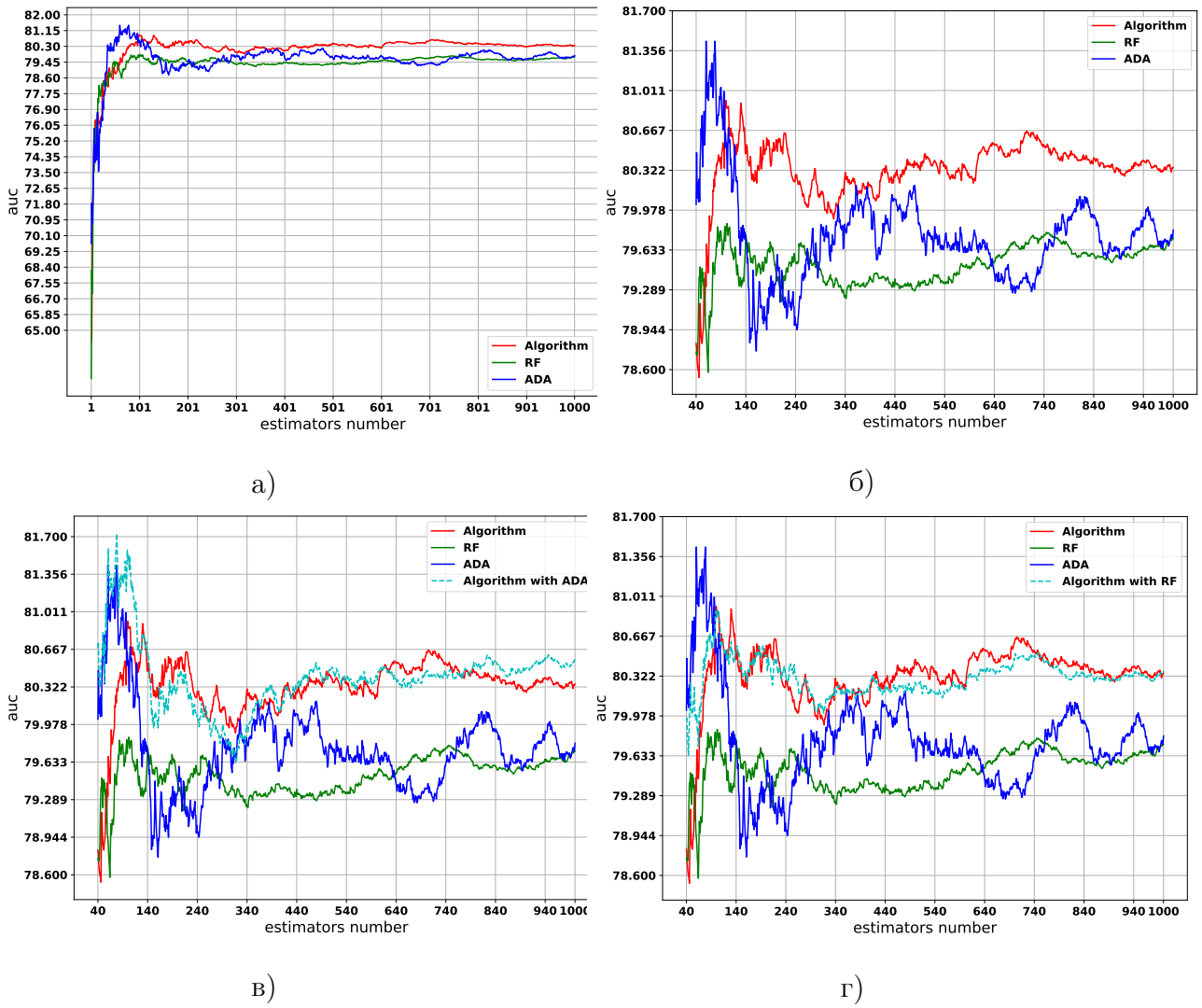


Рис. 12: Систолическое давление. Выборка 3. ROC-AUC. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost. **б)** Тот же график что и в пункте а), но начиная с итерации 40. **в)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и AdaBoost. **г)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и случайного леса.

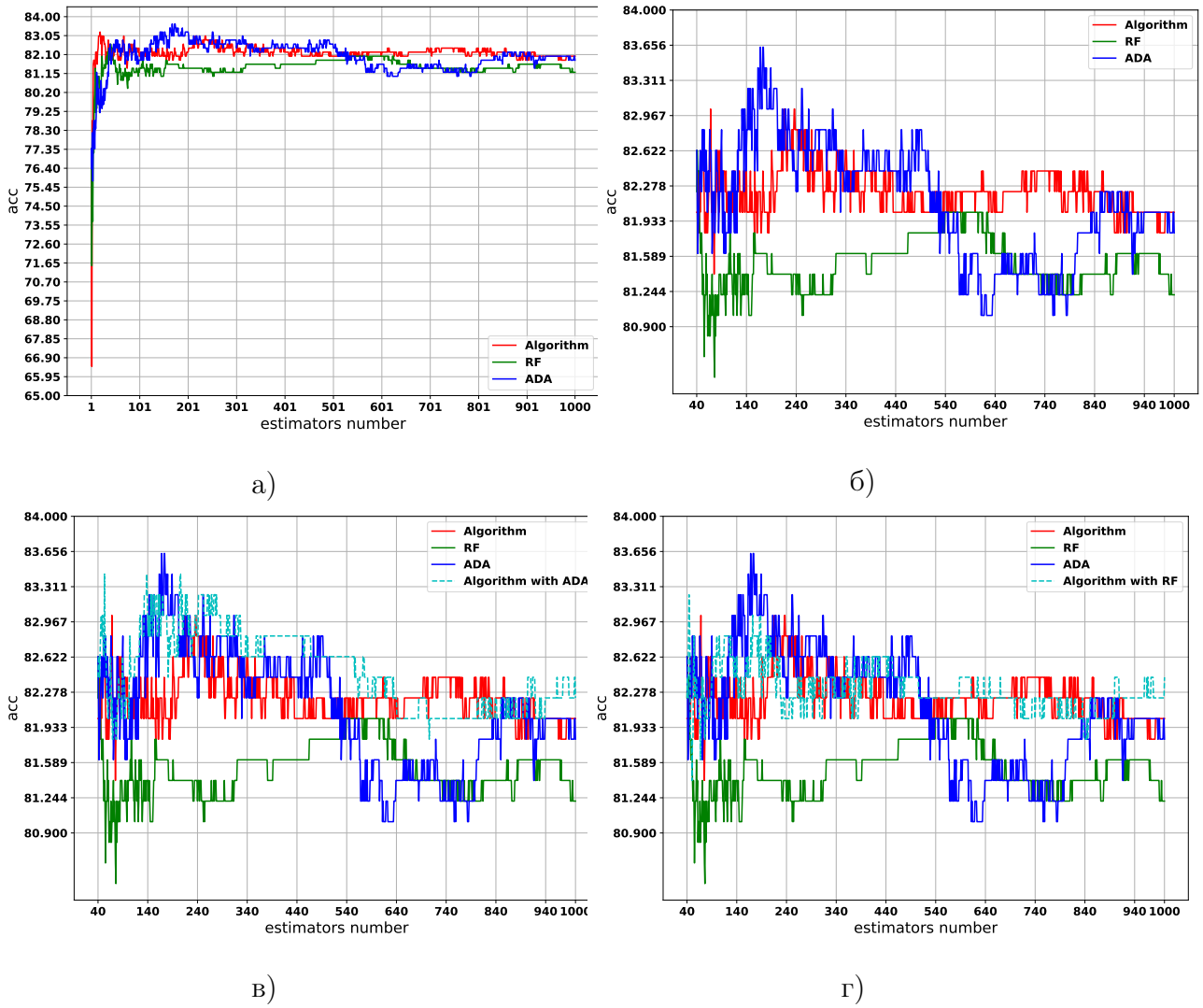
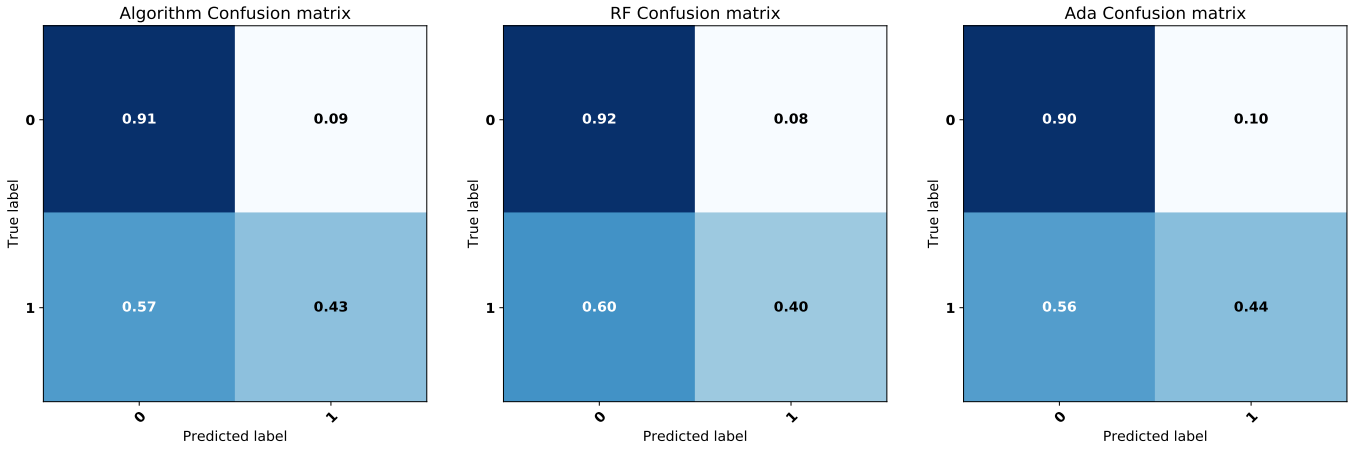
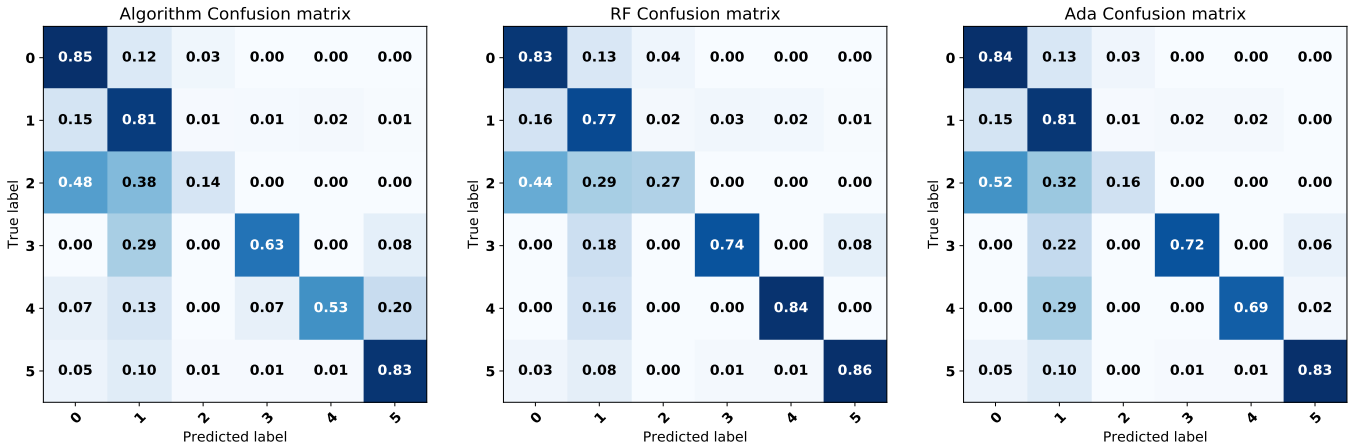


Рис. 13: Систолическое давление. Выборка 3. Точность. **а)**Зависимость качества от числа итераций. Точность разных ансамблей обозначена разным цветом: красный — **алгоритм 2**, зелёный — случайный лес, синий — AdaBoost. **б)** Тот же график что и в пункте а), но начиная с итерации 40. **в)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и AdaBoost. **г)** К перечисленным в пункте а) ансамблям добавляется голубой — объединение **алгоритма 2** и случайного леса.

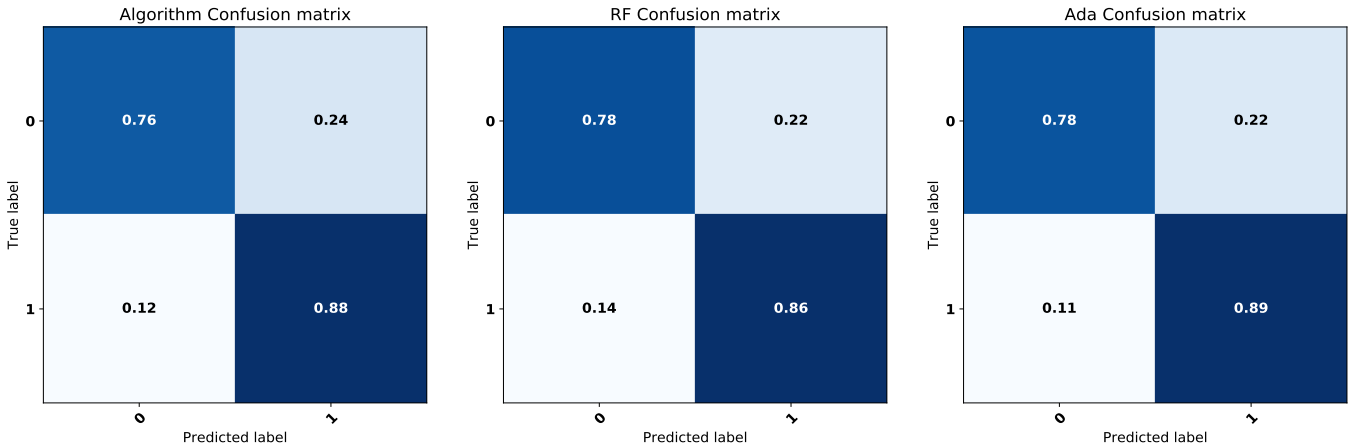
5.7 Матрицы ошибок и таблицы с качеством



а)

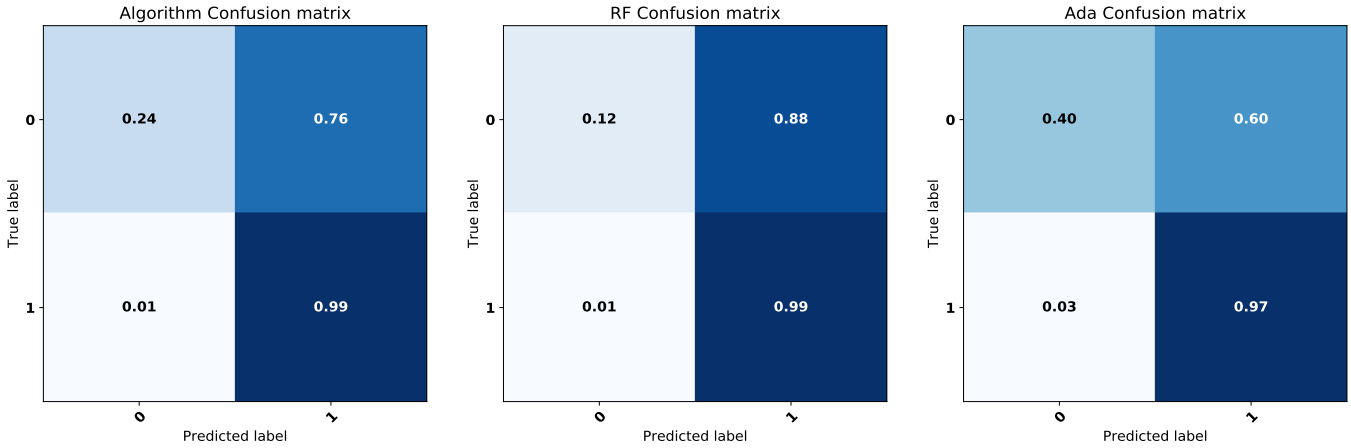


б)

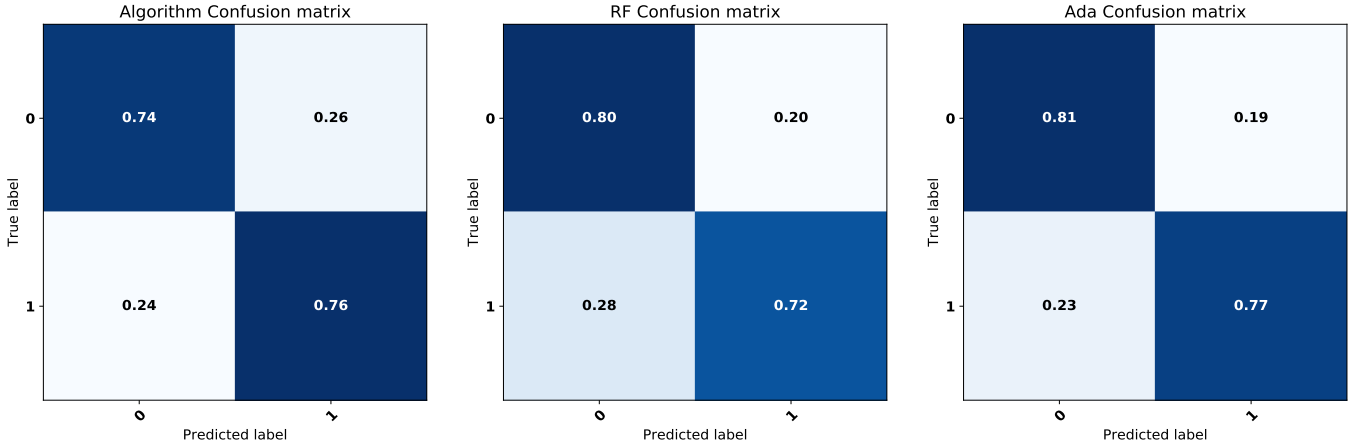


в)

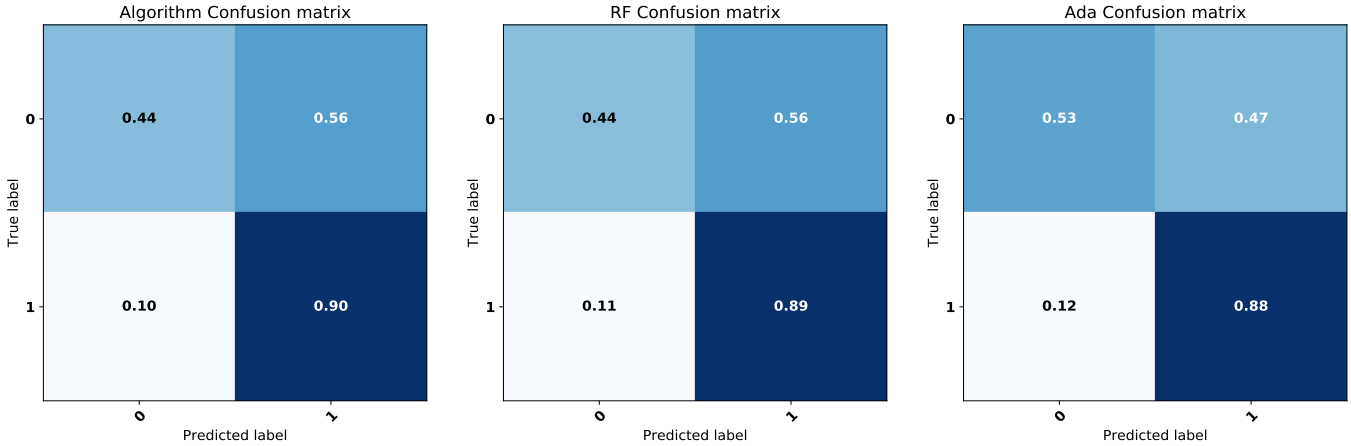
Рис. 14: Матрицы ошибок: **а)**кредитный скоринг, **б)**классификация стекла, **в)**распознование мин



а)



б)



в)

Рис. 15: Матрицы ошибок задачи систолического давления: **а)**для выборки 1, **б)**для выборки 2, **в)**для выборки 3

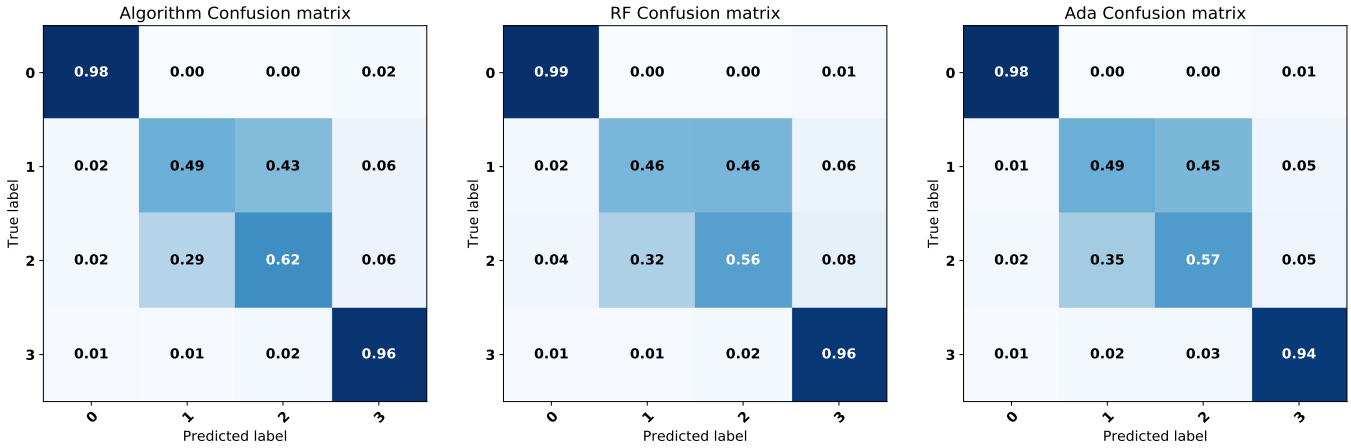


Рис. 16: Матрицы ошибок: классификация силуэта машины

Название метода	Точность на последней итерации	Максимальная точность	ROC-AUC на последней итерации	Максимальный ROC-AUC
RF	0.980	0.980	-	-
AdaBoost	0.984	0.984	-	-
Алгоритм 2	0.982	0.982	-	-
Алгоритм 2 + RF	0.982	0.982	-	-
Алгоритм 2 + AdaBoost	**0.984**	**0.984**	-	-

Таблица 3: Классификация рукописных цифр

Название метода	Точность на последней итерации	Максимальная точность	ROC-AUC на последней итерации	Максимальный ROC-AUC
RF	0.763	0.766	0.788	0.788
AdaBoost	0.759	0.766	0.780	0.781
Алгоритм 2	**0.768**	**0.770**	0.789	0.790
Алгоритм 2 + RF	0.764	0.767	**0.790**	**0.791**
Алгоритм 2 + AdaBoost	0.766	0.769	0.789	0.790

Таблица 4: Кредитный скоринг

Название метода	Точность на последней итерации	Максимальная точность	ROC-AUC на последней итерации	Максимальный ROC-AUC
RF	0.761	0.769	-	-
AdaBoost	0.761	0.766	-	-
Алгоритм 2	0.751	0.759	-	-
Алгоритм 2 + RF	**0.769**	**0.775**	-	-
Алгоритм 2 + AdaBoost	0.757	0.765	-	-

Таблица 5: Классификация стекла

Название метода	Точность на последней итерации	Максимальная точность	ROC-AUC на последней итерации	Максимальный ROC-AUC
RF	0.822	0.827	0.927	0.928
AdaBoost	**0.839**	**0.848**	**0.934**	**0.935**
Алгоритм 2	0.823	0.825	0.916	0.916
Алгоритм 2 + RF	0.823	0.835	0.925	0.925
Алгоритм 2 + AdaBoost	0.831	0.834	0.924	0.925

Таблица 6: Распознование мин

Название метода	Точность на последней итерации	Максимальная точность	ROC-AUC на последней итерации	Максимальный ROC-AUC
RF	0.738	0.742	-	-
AdaBoost	0.744	0.749	-	-
Алгоритм 2	**0.760**	**0.762**	-	-
Алгоритм 2 + RF	0.750	0.751	-	-
Алгоритм 2 + AdaBoost	0.759	0.762	-	-

Таблица 7: Классификация силуэта машины

Название метода	Точность на последней итерации	Максимальная точность	ROC-AUC на последней итерации	Максимальный ROC-AUC
RF	0.888	0.943	0.891	0.958
AdaBoost	0.876	0.934	0.892	0.943
Алгоритм 2	0.878	0.951	0.880	0.951
Алгоритм 2 + RF	**0.894**	0.945	**0.896**	**0.958**
Алгоритм 2 + AdaBoost	0.876	**0.954**	0.886	0.954

Таблица 8: Систолическое давление. Выборка 1

Название метода	Точность на последней итерации	Максимальная точность	ROC-AUC на последней итерации	Максимальный ROC-AUC
RF	0.751	0.773	0.837	0.847
AdaBoost	**0.787**	**0.796**	**0.848**	**0.854**
Алгоритм 2	0.755	0.764	0.836	0.837
Алгоритм 2 + RF	0.751	0.760	0.836	0.838
Алгоритм 2 + AdaBoost	0.751	0.760	0.839	0.842

Таблица 9: Систолическое давление. Выборка 2

Название метода	Точность на последней итерации	Максимальная точность	ROC-AUC на последней итерации	Максимальный ROC-AUC
RF	0.812	0.826	0.797	0.798
AdaBoost	0.818	**0.836**	0.798	0.814
Алгоритм 2	0.820	0.832	0.803	0.809
Алгоритм 2 + RF	**0.824**	0.832	0.803	0.808
Алгоритм 2 + AdaBoost	0.822	0.834	**0.805**	**0.817**

Таблица 10: Систолическое давление. Выборка 3

6 Заключение

В работе предложен новый метод ансамблирования с деревьями в качестве базовых моделей. Также были проведены эксперименты на реальных данных, показавшие, что предложенный алгоритм иногда достигает лучшего качества, чем другие популярные алгоритмы. Было показано, что предложенный метод отличается от других рассмотренных, и во множестве случаев объединение других методов с предложенным приводит к улучшению качества.

Список литературы

[1] Zhi-Hua Zhou. Ensemble Methods: Foundations and Algorithms. — Chapman and Hall/CRC, 2012.

[2] А. А. Докукин, О. В. Сенько, “Оптимальные выпуклые корректирующие процедуры в задачах высокой размерности”, Ж. вычисл. матем. и матем. физ., 51:9 (2011), 1751–1760; Comput. Math. Math. Phys., 51:9 (2011), 1644–1652

[3] Breiman L. Bagging Predictors. Technical Report No. 421. — 1994.

[4] Ho, Tin Kam. Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

[5] Breiman L. Random Forests. Machine Learning. — 2001.

[6] Yoav Freund, Robert E. Schapire. A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September, 1999

[7] Friedman J. Greedy Function Approximation: A Gradient Boosting Machine. — IMS 1999 Reitz Lecture.

[8] J.Zhu, H. Zou, S. Rosset, T. Hastie, “Multi-class AdaBoost”, 2006.

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

[10] UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets