ОГЛАВЛЕНИЕ

ГЛАВА 1

ВВЕДЕНИЕ

Машинное обучение все больше и больше начинает использоваться в различных областях нашей жизнедеятельности. Оно уже есть в недвижимости, общественном питание, финансах, сельском хозяйстве, медицине, биологии и многих других сферах. И поэтому все больше людей начинают проявлять к машинному обучению интерес. Но у них возникает следующая проблема, для этого им требуется достаточно хорошая подготовка в области программирования и математики, чтобы иметь возможность собрать какую-нибудь простейшую модель для решения нужной им задачи. Именно поэтому актуально следующие решение, создается сайт, где пользователь имея собственный набор данных может их загрузить, выполнить небольшую предварительную обработку и после чего компьютер сам найдет наилучшую последовательность алгоритмов для восстановления зависимости.

1.1 Основные понятия

1.1.1 Обучение с учителем

Обучение с учителем (англ. Supervised learning) [1] является одним из способов машинного обучения, построенного для решения следующий задачи.

Имеется множество объектов, которое делится на два множества: X, которое является множеством признаков, Y, которое является множеством целевых переменных. Между этими множествами существует некая зависимость, но она не известна.

Запишем данную задачу в следующем виде $f^* : X \to Y$, где значениями функции f^* является $y_i = f^*(x_i)$. Пусть l - это количество объектов в

множестве, тогда $X^l = (x_i, y_i)_{i=1}^l$ будет обучающей выборкой.

Задача обучения с учителем состоит в том, что требуется на основании обучающей выборки X^l восстановить зависимость f^*, то есть требуется построить решающую функцию $a : X \to Y$, которая бы не только приближала функцию f^* на обучающей выборке, но и хорошо предсказывала значения для остальной части множества X.

1.1.2 Автоматическое машинное обучение

Процесс машинного обучения имеет следующую последовательность действий [7]:

1) Предварительная обработка данных для анализа
2) Создание, если требуется, новых признаков на основе имеющихся
3) Выбор метрик для оценки качества алгоритма
4) Выбор алгоритма машинного обучения
5) Оптимизация гиперпараметров выбранного алгоритма

Задача автоматического машинного обучения в том, чтобы данные процессы компьютер мог выполнять самостоятельно.

1.2 Цели работы

Целью работы является разработать распределенное приложение, где пользователь сможет провести в режиме онлайн анализ загруженных им данных.

Для достижения данной цели ставятся следующие задачи:

1) Изучить способы автоматического машинного обучения;
2) Изучение и реализация способов предварительной обработки данных;
3) Выбрать алгоритмы машинного обучения, которые будут задействованы;

4) Выбрать наборы значений возможных значений для гиперпараметров алгоритмов машинного обучения;
5) Автоматизировать процесс машинного обучения с учителем;
6) Создание веб-приложения для онлайн-анализа данных загруженных пользователем

1.3 Существующие работы

На сегодняшний день существует множество фреймворков автоматического машинного обучения [2] и каждый из них базируется на разных идеях. Чаще всего они строятся на основании Байсовской оптимизации [3] [4], но сейчас также развиваются фреймворки на базе глубоких нейронных сетей [5].

Также есть ещё эффективный способ автоматического построения моделей машинного обучения - это использование генетического алгоритма. Он предложен в статье [6], где последовательность применения алгоритмов машинного представляется в виде древовидной структуры, после чего каждая такая структура улучшается с помощью применения генетического алгоритма. Этот фреймворк является надстройкой над библиотекой **scikit-learn**. Он показывает очень хорошую скорость работы на маленьких и средних наборах данных, но на больших он начинает работать очень долго. Также ещё одним минусом является то, что он без предварительной обработки не может работать с категориальными признаками.

Если рассматривать веб-приложения, которые позволяют обрабатывать пользовательские данные машинным обучением, то популярным и профессиональным решением является Microsoft Azure Machine Learning. Плюсом данной системы является её широкие возможности, то есть там имеется множество алгоритмов как для предварительной обработки, так и для машинного обучения, также это всё происходит на удаленных серверах, в

следствии чего это работает очень быстро. Но как минус стоит заметить эта система подходит только для подготовленных пользователей, потому что модели машинного обучения нужно строить в ручную.

Научная новизна данной работы будет заключаться в том, чтобы создать такой алгоритм автоматического машинного обучения, который имел бы в своей основе генетический алгоритм и работал бы быстро на различных объемах данных, и при этом точность предсказаний целевых значений была как можно ближе к алгоритму предложенному в статье [6].

В то время как практическая новизна данной работы будет состоять в том, чтобы создать веб-приложение, которые позволит не подготовленным пользователям, как минимум в области программирования, получить хорошее решение предсказания целевых значений посредством машинного обучения.

1.4 Используемые термины

Пайплайн Датасет Выбросы Индивид Популяция Поколение Терминал Примитив Гиперпараметр Активация пайплайна

ГЛАВА 2

РЕАЛИЗАЦИЯ SELECTOR OF ALGORITHM MODELS

Веб-приложение Selector of Algorithm Models можно разделить на две большие части: клиентская часть, которая будет видна пользователю, и серверная часть, она же будет называться алгоритмической. Элементы клиентской части будут показаны в разделе [??]. Остальные разделы будут посвящены алгоритмической части программы.

2.1 Схема работы программы

Как показано на рисунке программа имеет следующие этапы:

- **Загрузка пользовательских данных:**

  В рамках данного блока программа получает на вход файл, который загрузил пользователь и отправляет его на сервер. Более подробно об этом будет сказано в разделе 2.2.

- **Предварительная обработка данных:**

  Здесь программа позволяет выполнить предварительную обработку данных и увидеть их обновленный вид, после чего их можно отправить на анализ. Более подробно об этом будет сказано в разделе 2.3.

- **Автоматическое машинное обучение:**

  В данном блоке происходит построение различных пайплайнов из алгоритмов на основании данных загруженных пользователем, после чего пользователю предлагаются лучшие из полученных пайплайнов. Более подробно об этом будет сказано в разделе 2.4.

- **Работа с полученными результатами:**

  В рамках данного блока пользователь сначала должен выбрать один из трёх лучших пайплайнов предложенных ему, после чего он может загрузить данные, которые уже не имеют множества целевых значений

Y, а имеют только множество признаков X, после чего, с помощью выбранного пайплайна, будут получены значения для множества Y. Более подробно об этом будет сказано в разделе 2.5.

2.2 Загрузка пользовательских данных

В данной части приложения пользователь должен подать на вход файл, который имеет расширение **csv**, и после чего отправить его на сервер. На сервере создается временный файл, в который копируется вся информация из переданного **csv** файла. После загрузки данных, сервер считывает все имена столбцов, если они не имеют имен, то он дает им имена **Feature i**, где i это порядковый номер столбца, а последнему столбцу, в случае отсутствия имени, он присваивает ему имя **Target**.

2.3 Предварительная обработка данных

В данной части приложения пользователь будет иметь возможность внести следующие изменения в файл перед тем как его отправить на анализ данных [8] [9]:

- **Выбор множества признаков и множества допустимых значений:** Пользователю предложен список, состоящий из названий признаков, и он сможет самостоятельно исключить те, которые он не хочет использовать при анализе.

  Последний признак по умолчанию превращается в множество целевых значений, но если пользователь хочет, чтобы им был какой-то другой признак, то он может выбрать его из предложенного ему списка.
- **Выбор типа задачи:** Пользователю предложен выбор между задачей регрессии и задачей классификации. По умолчанию ничего не выбра-

но.

- **Обработка пропущенных значений:** Пользователь может выбрать обработать пропущенные значений в нужных ему признаках следующими методами:
  - **auto** – Если количество пропущенных значений в признаке больше 80%, то он удаляется, иначе пропущенные значения заменяются на средние арифметическое, а если это столбец с категориями, то замена на наиболее часто встречаемое значение.
  - **mean** – Замена пропущенных значения признака на средние арифметическое столбца, а если это столбец с категориями, то замена на наиболее часто встречаемое значение.
  - **median** – Замена пропущенного значения признака на медиану столбца, а если это столбец с категориями, то замена на наиболее часто встречаемое значение.
  - **most_frequent** – Замена пропущенного значения признака на наиболее часто встречаемое значение в данном столбце.
  - **del** – Если объект содержит пропущенное значение, то он удаляется.
- **Обработка выбросов:** Пользователь может обработать выбросы в нужных ему признаках следующими методами:
  - **replace** – Если значение признака больше процентиля 0.95, по всему столбцу, или меньше процентиля 0.05, то происходит замена этого значения на соответствующий процентиль.
  - **std** – Если значение признака имеет отклонение превышающие стандартное отклонение в несколько раз, то объект удаляется.
  - **percentile** – Если значение признака больше процентиля 0.95, по всему столбцу, или меньше процентиля 0.05, то объект удаляется.
- **Категоризация:** Пользователь может представить выбранные им признаки в виде категорий с помощью следующих методов:

  - **equal** – Распределяет значения признака равномерно по заданному количеству категорий.
  - **entropy** – Разделяет признаки на категории на основании максимальной информативность по отношению к столбцу целевых значений. Это осуществляется с помощью Information Gain.
  - **quantile** – Разделяет на категории с помощью квантилей.
- **Трансформация:** Пользователь может применить к выбранным признакам следующие математические функции:
  - **log** – К значениям признака применяется функция натурального логарифма.
  - **box-cox** – К значениям признака применяется метод Бокса-Кокса.
- **Нормализация:** Пользователь может нормализовать выбранные им признаки следующими способами:
  - **norm** – Нормализация значений признака с помощью максимального и минимального значения в столбце.
  - **stand** – Нормализация признака с помощью его среднего значения и стандартного отклонения.
  - **l2-norm** – Делит все значения признака на его норму.

2.4 Построение моделей машинного обучения

После того как пользователь отправил датасет на анализ запускается алгоритм автоматического машинного обучения.

В статье [6] предлагается фреймворк для автоматического машинного обучения основанный на древовидной структуре пайплайна, которые будут улучшаться с помощью генетического алгоритма. Плюсом такой структуры является то, что она является очень гибкой для различных преобразований данных . Минусом является сложность подобной структур, которая сказывается на скорости работы. Здесь будет предложена более простая структура

пайплайнов, которая работает по принципу очереди.

2.4.1 Структура пайплайна

Пусть длина пайплайна равняется трём, то есть он будет иметь три алгоритма. Так как алгоритмы машинного обучения являются терминальным алгоритмами, то он может быть только один, соответственно остальные алгоритмы будут примитивными. Теперь в пустую очередь длины три, помещаются алгоритмы в следующем порядке, сначала туда помещается первый примитивный алгоритм, потом второй примитивный алгоритм, после чего туда помещается терминальный алгоритм. Активация данного пайплайна будет происходить сначала очереди.

2.4.2 Примитивные и терминальные алгоритмы

Примитивные алгоритмы можно разделить на два вида: трансформирующие и модифицирующие. Трансформирующими являются те алгоритмы, которые каким-то образом преобразуют значения, которые есть в датасете. А модифицирующими являются те, которые могут исключить какие-то признаки.

Список применяемых трансформирующих алгоритмов:

- PolynomialFeatures (добавляет новые признаки через полиномиальную трансформацию)
- StandardScaler (стандартизация)
- RobustScaler (устранение выбросов через квантили)
- MaxAbsScaler (нормализация с максимальным по модулю значением)
- MinMaxScaler (линейная нормализация)
- Binarizer (превращает значения либо в ноль, либо в единицу на основании заданного порога)

Список применяемых модифицирующих алгоритмов:

- VarianceThreshold (удаляет объекты с низкой дисперсией)
- SelectKBest (выбирает k лучших признаков)
- SelectPercentile (выбирает лучшие признаки по заданному процентилю)
- SelectFwe (выбирает признаки на основании определенного уровня значимости)
- RFE (рекурсивное исключение признаков с помощью заданного алгоритма машинного обучения)
- SelectFromModel (выбирает признаки на основании их оценки заданным алгоритмом машинного обучения)

Терминальные алгоритмы тоже делятся на два вида, одни для задачи восстановления регрессии, а другие для задачи классификации.

Список применяемых алгоритмов для задачи классификации:

- GaussianNB, BernoulliNB, MultinomialNB (Наивный байесовский классификаторы)
- DecisionTreeClassifier (Дерево решений)
- ExtraTreesClassifier (Сверх-случайные леса)
- RandomForestClassifier (Случайные леса)
- GradientBoostingClassifier (Градиентный бустинг)
- LogisticRegression (Логистическая регрессия)
- KNeighborsClassifier (Метод K-ближайших соседей)
- LinearSVC (Метод опорных векторов)
- XGBClassifier
- AdaBoostClassifier
- SGDClassifier (Стохастический градиентный спуск)

Список применяемых алгоритмов для задачи восстановления регрессии:

- ElasticNetCV (Эластичная сеть)
- DecisionTreeRegressor (Дерево решений)

- ExtraTreesRegressor (Сверх-случайные леса)
- RandomForestRegressor (Случайные леса)
- AdaBoostRegressor
- GradientBoostingRegressor (Градиентный бустинг)
- LassoLarsCV (Метод Лассо)
- KNeighborsRegressor (Метод K-ближайших соседей)
- LinearSVR (Метод опорных векторов)
- RidgeCV (Ридж-регрессия)
- XGBRegressor
- SGDRegressor (Стохастический градиентный спуск)

Все перечисленные выше алгоритмы берутся из библиотеки scikit-learn.

2.4.3 Заполнение пайплайна алгоритмами

Заполнение пайплайна алгоритмами происходит следующем образом:

1) Выбирается случайная длина пайплайна l в диапозоне от возможно минимально значения до максимально возможного.
2) Если $l \neq 1$, то добавляем в пайплайн случайно выбранные примитивные алгоритмы в количестве $l - 1$, для гиперпараметров алгоритма случайным образом берутся значения из множества возможных значений этого гиперпарметра.
3) В пайплайн добавляется терминальный алгоритм, который выбирается случайным образом, и также как в примитивных алгоритмах для гиперпараметров берутся значения.
4) Производится подсчёт уже готовых пайплайнов, имеющих те же самые алгоритмы, если оно превышает 5 %, то пайплайн создается заново, иначе он добавляется в множество готовых пайплайнов.

О множестве возможных значений гиперпараметров для терминальных алгоритмов будет сказано в разделе 3.

2.4.4 Структура генетического алгоритма

Как было уже сказано, для поиска наилучшего пайплайна используется генетических алгоритм. Рассмотрим то, как устроен данный алгоритм конкретно для задачи автоматического машинного обучения.

Как показано на рисунке [!!!] генетических алгоритм в данной задаче состоит из трех основных этапов и двух дополнительных, которые используется для создания самой первой популяции.

- **Создание первой популяции:** В данном этапе создается заданное количество индивидов, которые конкретно здесь будут являться пайплайнами из алгоритмов. Значит создание первой популяции будет происходить с помощью многократного вызова функции заполнения пустого пайплайна.
- **Оценка первой популяции:** В данном этапе с помощью функции оценки популяции, измеряется длина пайплайна и его точность предсказания.
- **Создание потомства:** В данном этапе с помощью различных мутаций и скрещиваний создается потомство популяции.
- **Оценка потомства:** В данном этапе вызывается функция оценки, которая замеряет длину пайплайна каждого индивида и его точность предсказания.
- **Селекция:** В данном этапе происходит отбор лучших индивидов из популяции и его потомства для создания новой популяцию.

2.4.5 Создание потомства

Потомство может создаваться двумя способами:

1) Индивид из популяции скрещиваются и получается новый потомок, который будет иметь какие-то параметры от первого индивида, а какие-

то от второго.

2) Индивид мутирует, то есть происходят какие-то изменения в его пайплайне.

В алгоритме 1 приведен псевдокод создания нового потомства. В данном примере вероятность мутации имеет значение 0.7, а вероятность скрещивания 0.3. Вероятность мутации имеет более высокое значение, потому что существует несколько видов мутаций. Также задается ограничение на количество одинаковых пайплайнов, чтобы избежать переполнения популяции однотипным вариантами решения.

Algorithm 1 Создание потомства

INPUT $F_{population}$ – популяция индивидов, $Size_{offspring}$ – размер потомства

OUTPUT $F_{offspring}$

$F_{offspring} \leftarrow \{\}$, $Number_{offspring} \leftarrow 0$;

$Probability\_mutation \leftarrow 0.7, Probability\_mate \leftarrow 0.3$;

while $Number_{offspring} \leq Size_{offspring}$ **do**

  $p \leftarrow random\_value(0, 1)$;

  if $p <= Probability\_mate$ **then**

    $new\_individual \leftarrow mate(random\_two\_ind\_for\_mate(F_{population}))$;

  else if $p <= Probability\_mate + Probability\_mutation$ **then**

    $new\_individual \leftarrow mutation(random\_choice(F_{population}))$;

  end if

  if $number\_identical\_ind(new\_individual) <= Size_{offspring} * 0.1$ **then**

    $F_{offspring} \leftarrow F_{offspring} \cup new\_individual$;

    $Number_{offspring} + +$;

  end if

end while

Мутации

В рамках данной задачи мутации бывают трёх видов:

- **Replacement_mutation:** На вход данная мутация получает пайплайн индивида и после чего случайным образом оттуда выбирает алгоритм. С данным алгоритмом может произойти две вещи: замена значения гиперпараметра данного алгоритма на другое или замена самого алгоритма на другой (оба события равновероятны).
- **Shrink_mutation:** На вход данная мутация получает пайплайн индивида и после чего удаляет из него случайный примитивный алгоритм.
- **Insert_mutation:** На вход данная мутация получает пайплайн индивида и после чего добавляет в него случайный примитивный алгоритм.

Все вышеперечисленные мутации равновероятны.

Скрещивание

На вход функции скрещивания передаются два случайных индивида, но при условие что у них есть хотя бы один общий алгоритм. После чего случайным образом выбирается общий алгоритм у этих индивидом, и аналогичным способом выбирается гиперпараметр, который будет заменен.

Построение нового индивида происходит следующем образом, копируется первый инвидивид и на место значения выбранного гиперпараметра вставляет значение гиперпараметра из второго индивида.

Также была рассмотрена возможность выбора гиперпараметра для замены, в том случае если это терминальный алгоритм, с помощью **GridSearchCV** (функция из библиотеки **scikit-learn**, которая выбирает лучшие значения гиперпараметров из тех, которые были переданы в нее), то есть передаются значения гиперпарметров из терминального алгоритма первого индивида и второго, а после данная функция возвращает те значения для гиперпараметров, которые дали наилучший результат и после чего они записывались в

нового индивида. Но точность получаемого пайплайна оставалась примерно той же, но время работы алгоритма значительно увеличивалось. Поэтому данный способ замены не используется.

2.4.6 Функция оценки

Данная функция используется для оценки индивидов из популяции или потомства. Она использует обычную кросс-валидацию по пайплайну каждого индивида на обучающей выборки. После чего в атрибут индивида записываются данные об его оценки в формате (длина пайплайна, оценка качества пайплайна).

В зависимости от задачи оценка качества происходит по разным метрикам. Метрикой по умолчанию для задачи восстановления регрессии является отрицательная среднеквадратичная ошибка, а для задачи классификации F-мера.

2.4.7 Селекция

Данная функция на вход получает популяцию, потомство и каким должен быть размер новой популяции после селекции. Для селекции используется алгоритм **NSGA-II** [10], которые является алгоритмом многокритериальной оптимизации. Он выполняет сортировку на основании Рангов границы Парето и достижения максимальной разреженности. Что позволяет конкретно в рамках данной задачи достичь того, чтобы в приоритете при селекции были те пайплайны, которые имеют наименьшую длину и имеют наибольшую оценку, но при этом они должны быть максимально разрежены в рамках своих критериев, что позволяет сохранять многообразие пайплайнов при селекции.

2.4.8 Алгоритм автоматического машинного обучения

В подразделе 2.4.4 было рассказано о структуре генетического алгоритма. В алгоритме 2 будет представлен псевдокод функции обучения для переданных в нее данных.

Algorithm 2 Обучение

INPUT $Size_{offspring}, Size_{population}, N_{generations}, Features, Targets$

$F_{population} \leftarrow create\_population(Size_{population})$;

$evaluate(F_{population}, Features, Targets)$;

for $generation \leftarrow 1, N_{generations}$ **do**

$F_{offspring} \leftarrow create\_offspring(F_{population}, Size_{offspring})$;

$evaluate(F_{offspring}, Features, Targets)$;

$F_{population} \leftarrow select(F_{population}, F_{offspring}, Size_{population})$;

end for

После завершения работы данной функции информация о последнем популяции сохраняется как атрибут класса.

Для того чтобы получить итоговый пайплайн требуется вызвать функцию получения наилучшего индивида последний популяции. Функция на вход получает тестовую выборку, после чего эта тестовая выборка разбивается на валидационную и тестовую. Происходит валидация и после чего пайплайн с наилучшим результатом на валидационной выборке, проверяется на тестовой. И результатом возвращаемым функцией является пайплайн с самой высокой точностью на валидационной выборке и его результат на тестовой.

2.4.9 Использование алгоритма в веб-приложении

Сервер вызывает описанный выше алгоритм со следующими параметрами:

- $Size_{population} \leftarrow 50$
- $Size_{offspring} \leftarrow 50$
- $N_{generations} \leftarrow 5$

Также функция получения наилучшего пайплайна имеет небольшую надстройку, после замера на валидационной выборке три лучших пайплайна сохраняются и далее пользователю будет предложено самому выбрать какой из них использовать.

2.5 Работа с полученными результатами

После того как алгоритм автоматического обучения закончиться пользователю будет предложено выбрать из трёх лучших пайплайнов для дальнейший работы, но это возможно после того, как пользователь загрузит датасет, в котором отсутствует множество целевых значений.

После этого пользователь сможет запустить выбранный пайплайн для предсказания множества целевых значений у переданного датасета.

ГЛАВА 3

ВЫБОР НАБОРОВ ГИПЕРПАРАМЕТРОВ

TODO: Доделать таблицы

В данной главе будет показан процесс выбора набора возможных значений для каждого гиперпараметра алгоритмов машинного обучения, которые используются в данный программе.

Для того чтобы программа находила более оптимальные значения гиперпараметров как можно быстрее требуется взять для каждого гиперпараметра такие наборы возможных значений, которые будут давать наилучшие результаты. Но некоторые алгоритмы имеют слишком много комбинаций значений, поэтому нужно каким-то образом их сократить.

Для этого будет проделан следующий порядок действий:

1) Берётся набор датасетов определённого класса задач, на которых будут проверяться разные комбинации гиперпараметров.
2) Выбираем алгоритм машинного обучения для данного класса задач.
3) После чего берётся набор гиперпараметров, которые наиболее значимы для хорошего результата этого алгоритма.
4) Для каждого гиперпараметра из этого набора, берётся множество возможных значений, которое может принимать этот гиперпараметр, назовём его начальным набором значений.
5) После чего мы получаем множество всех возможных комбинаций значений гиперпараметров для выбранного алгоритма.
6) Через функцию **BayesSearchCV()** из библиотеки **scikit-optimize**, которая работает на принципах Байсовской оптимизации, на каждом датасеты будет испробовано 50 различных комбинаций гиперпараметров, через которые выше названная функция на каждой итерации будут стремиться улучшить точность алгоритма.
7) После того как результаты работы функции **BayesSearchCV()** на каж-

дом датасеты будут получены, для каждой комбинации возможных значений гиперпараметров будет взято среднее арифметическое от их точности на каждом датасете.

8) Получив среднею точность для испробованных комбинаций строится график, где по оси X будет номер комбинации, а по оси Y точность.

Получив такой график нужно снизить количество возможных комбинаций и при этом чтобы этих точек на графике осталось как можно больше и у них была хорошая точность, потому что эти точки, которые обозначают точность комбинации гиперпараметров, были получены на основании Байсовской оптимизации из-за чего можно предположить что эти комбинации гиперпараметров являются наиболее выгодными для получения высокой точности.

Данную задачу можно решить следующим образом, посмотреть для каждого гиперпараметра какие его значения встречаются реже всего или они показывают плохую точность, и после чего их убрать. Таким образом получается итоговый набор значений гиперпараметров. После этого будет построен график с точками, которые остались(они будут обозначены синим), и точками,которые были потеряны(они будут обозначены красным).

3.1 Алгоритмы регрессии

Для вычисления оптимальных возможных значений гиперпараметров алгоритмов регрессии используется 20 датасетов, таблица с ними будет приведена ниже.

Метрикой качества здесь будет являться коэффициент детерминации, или коэффициент R^2: $R^2 = 1 - \frac{\sum_{i=1}^{l}(a(x_i)-y_i)^2}{\sum_{i=1}^{l}(y_i-\overline{y})^2}$

Название датасета	Признаки	Объекты
Automobile	24	205
BlogFeedback	280	60 021
BostonHousing	13	405
Communities and Crime	127	1 994
Daily Demand Forecating	12	60
Facebook	18	500
Diamonds	10	54 000
Fish	6	159
Forestfires	12	517
Fundamentals	77	1 562
HousePrices_1	79	1 460
HousePrices_2	9	20 640
Parkinsons	22	5 875
Smoker	0	0
StudentsMarks	32	395
Tracks	68	1058
TracksPlus	116	1 058
WinequalityRed	11	1 599
WinequalityWhite	11	4 898
Winequality	12	6 497

3.1.1 Дерево решений (Decision Tree)

Рассмотрим для данного алгоритма следующий начальный набор значений:

- *max_depth:* [1, 29]
- *min_samples_split:* [2, 29]
- *min_samples_leaf:* [1, 29]

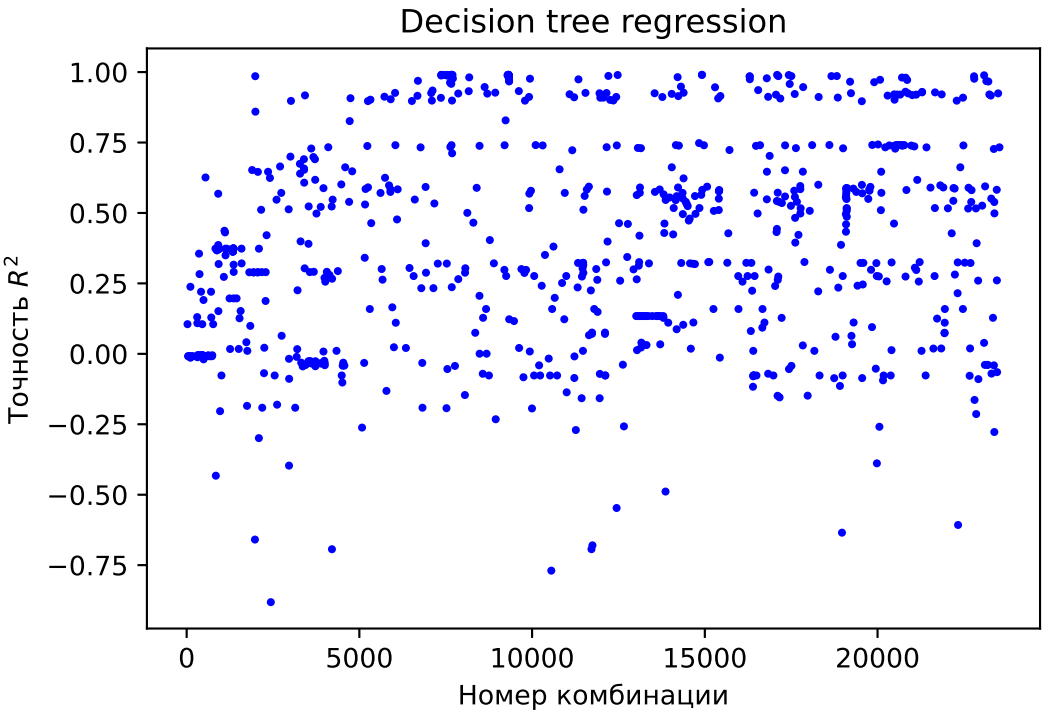

Рис. 3.1: Начальный набор значений гиперпараметров для алгоритма Decision tree regressor.

Соответственно получая 23 548 возможных комбинаций значений гиперпараметров.

После того как мы избавимся от значений гиперпараметров, которые встречаются реже всего или имеют низкий показатель точности, получим следующие:

- *max_depth:* [6, 19]
- *min_samples_split:* [2, 29]
- *min_samples_leaf:* [1, 23]

Теперь данный алгоритм имеет 15 456 возможных комбинации значений гиперпараметров. 60% точек осталось.

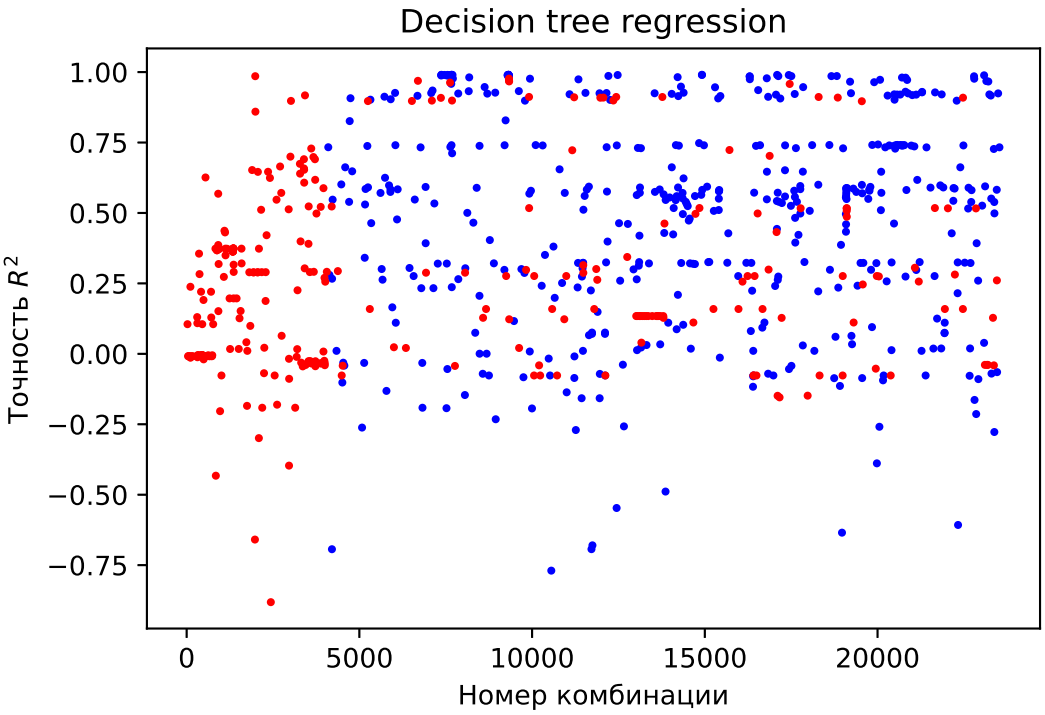

Рис. 3.2: Итоговый набор значений гиперпараметров для алгоритма Decision tree regressor.

3.1.2 Случайные и сверхслучайные деревья решения (Random Forest and Extra Trees)

Здесь будет рассмотрено сразу два алгоритма Random Forest и Extra Trees, так как это схожие алгоритмы и начальный набор значений гиперпараметров для них был один тот же. Для данных алгоритмов следующий начальный набор значений:

- *max_features:* [0.10, 1.0], с шагом 0.05
- *min_samples_split:* [2, 29]
- *min_samples_leaf:* [1, 29]

Соответственно получая 15 428 возможных комбинаций значений гиперпараметров.

После того как мы избавимся от значений гиперпараметров, которые встречаются реже всего или имеют низкий показатель точности, получим

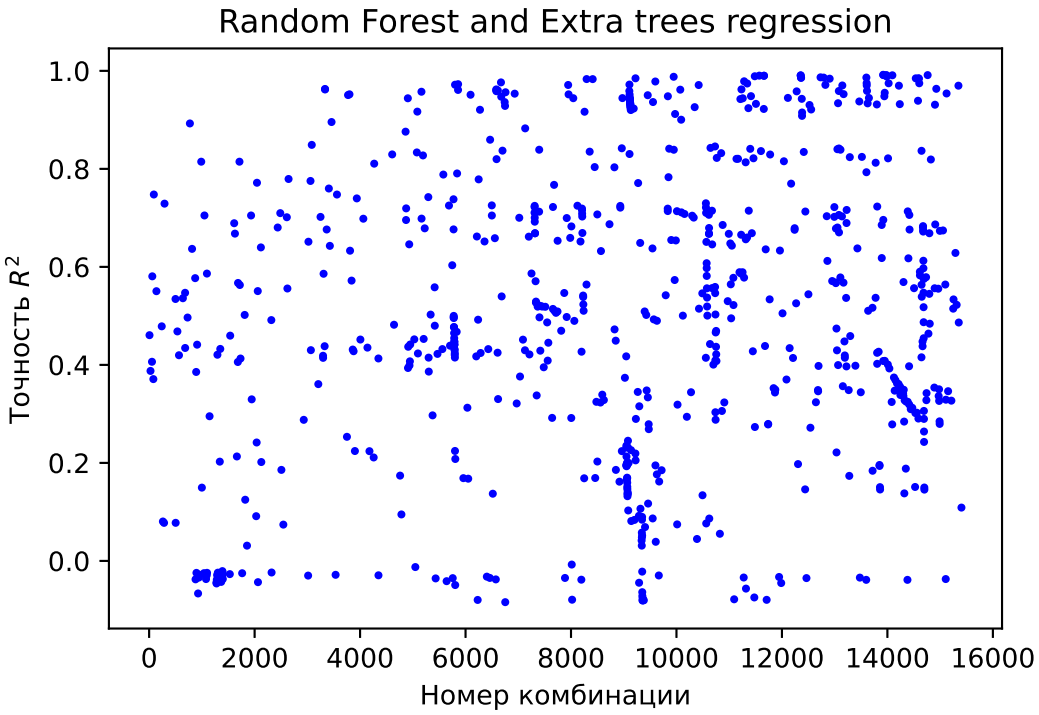

Рис. 3.3: Начальный набор значений гиперпараметров для алгоритмов Random forest regressor и Extra trees regressor.

следующие возможные значения:

- *max_features:* [0.4, 1.0], с шагом 0.05
- *min_samples_split:* [2, 25]
- *min_samples_leaf:* [1, 22]

Теперь данные алгоритмы имеют 6 864 возможных комбинации значений гиперпараметров. 60% точек осталось.

3.1.3 Градиентный бустинг (Gradient boosting)

Рассмотрим для данного алгоритма следующий начальный набор значений гиперпараметров:

- *learning_rate:* {1e-4, 1e-3, 1e-2, 1e-1, 0.5, 1}
- *max_depth:* [1, 19]
- *min_samples_split:* [2, 19]

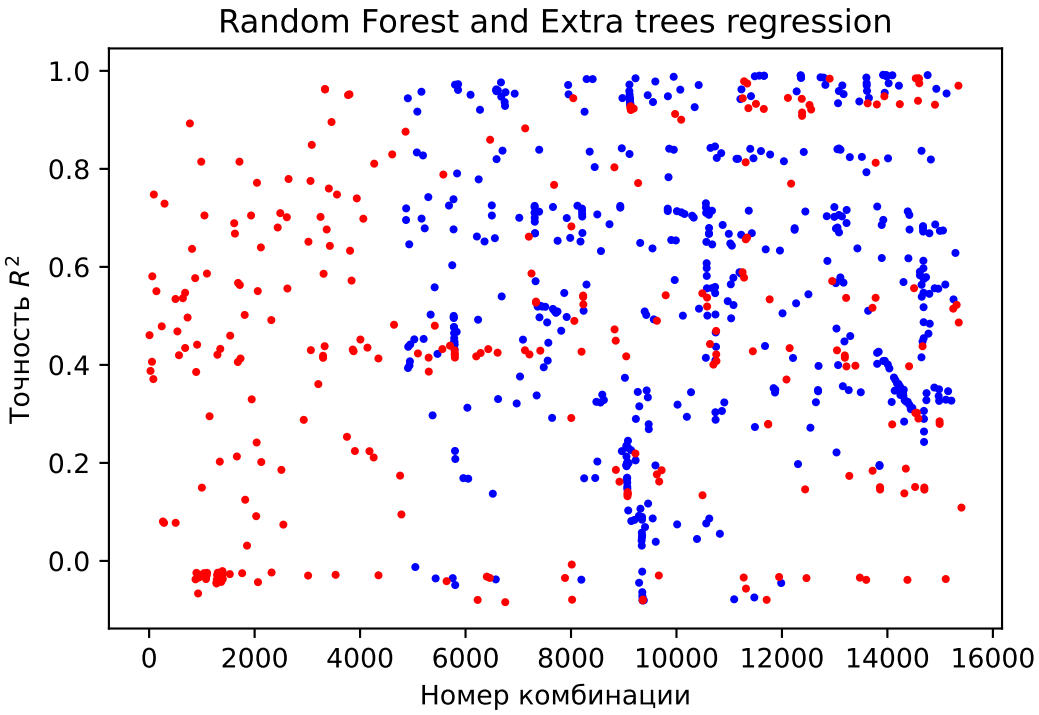

Рис. 3.4: Итоговый набор значений гиперпараметров для алгоритмов Random forest regressor и Extra trees regressor.

- *min_samples_leaf:* [1, 24]
- *subsample:* [0.2, 1.00], с шагом 0.05
- *max_features:* [0.20, 1.0], с шагом 0.05

Соответственно получая 13 395 456 возможных комбинаций значений гиперпараметров. На Рис.3.5 показаны не все точки, это связанно с тем, что имеются выбросы с большими отрицательными значениями.

После того как мы избавимся от значений гиперпараметров, которые встречаются реже всего или имеют низкий показатель точности, получим следующие возможные значения:

- *learning_rate:* {1e-2, 1e-1, 0.5}
- *max_depth:* [1, 19]
- *min_samples_split:* [2, 19]
- *min_samples_leaf:* [1, 24]
- *subsample:* [0.2, 1.00], с шагом 0.05

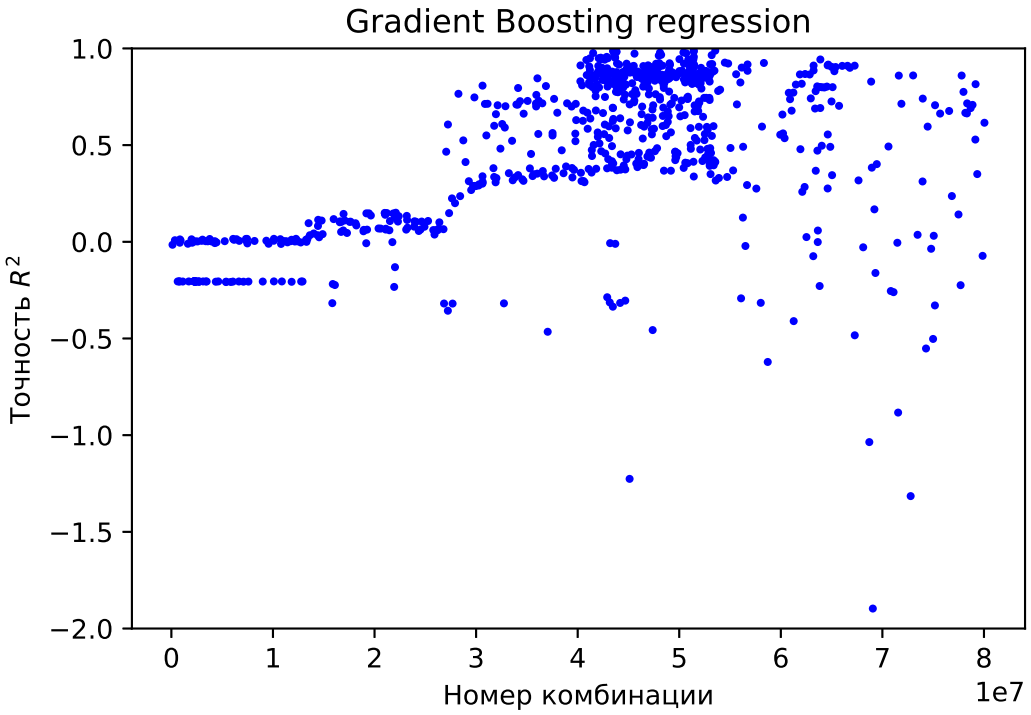

Рис. 3.5: Начальный набор значений гиперпараметров для алгоритма Gradient Boosting regression.

- *max_features:* [0.20, 1.0], с шагом 0.05

Теперь данный алгоритм имеет 6 697 728 возможных комбинации значений гиперпараметров. 73% точек осталось.

3.1.4 XGBoost

Рассмотрим для данного алгоритма следующий начальный набор значений гиперпараметров:

- *max_depth:* [1, 20]
- *learning_rate:* {0.00001, 0.0001, 0.001, 0.01, 0.1, 0.5, 1}
- *subsample:* [0.05, 1.00], с шагом 0.05
- *min_child_weight:* [1, 20]

Соответственно получая 56 000 возможных комбинаций значений гиперпараметров.

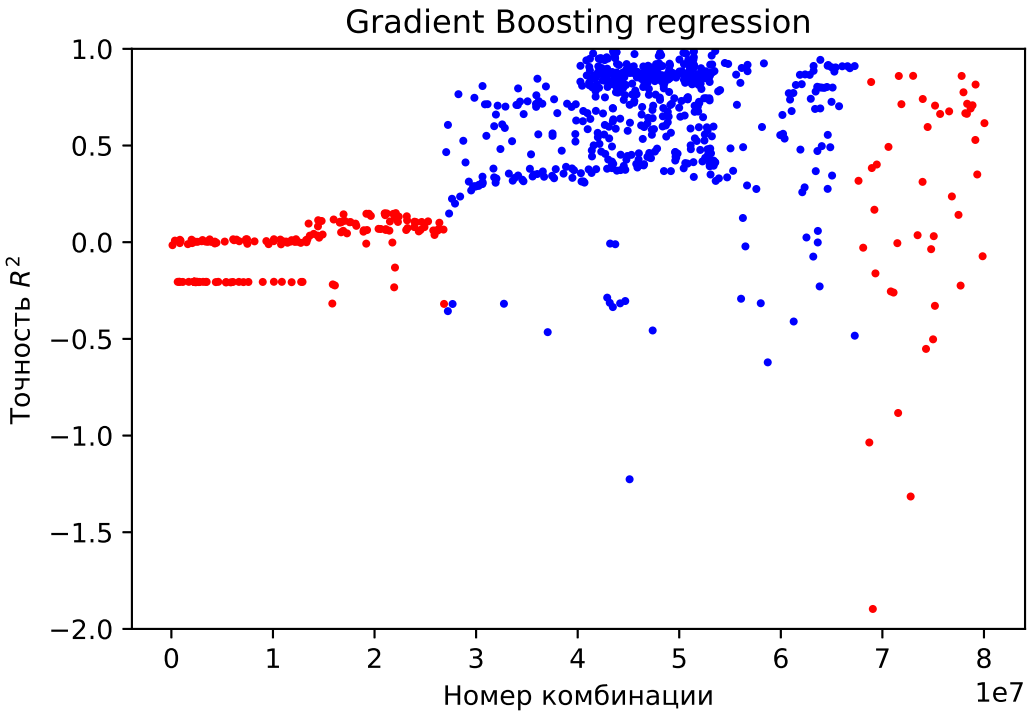

Рис. 3.6: Итоговый набор значений гиперпараметров для алгоритма для алгоритма Gradient Boosting regression.

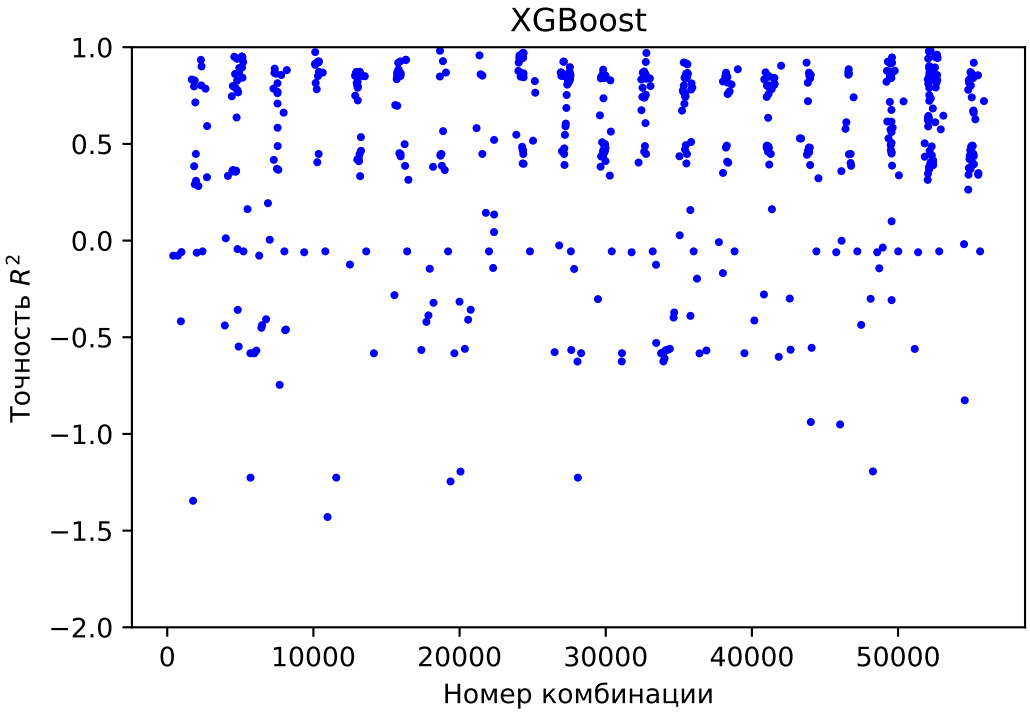

Рис. 3.7: Начальный набор значений гиперпараметров для алгоритма XGBoost regression.

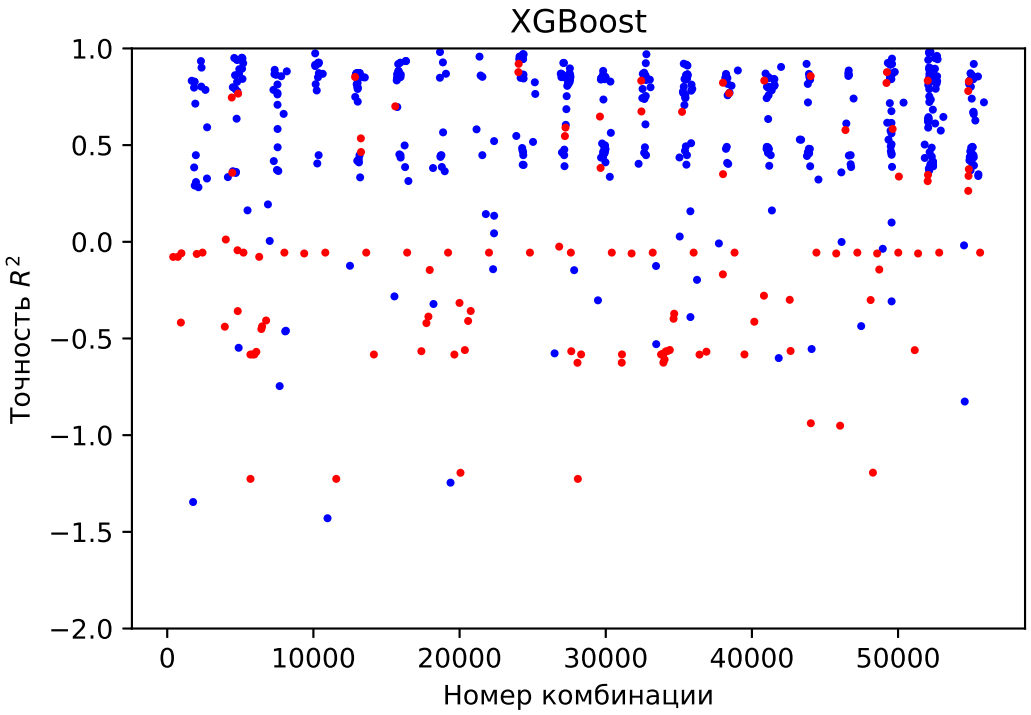

Рис. 3.8: Итоговый набор значений гиперпараметров для алгоритма для алгоритма XGBoost regression.

После того как мы избавимся от значений гиперпараметров, которые встречаются реже всего или имеют низкий показатель точности, получим следующие возможные значения:

- *max_depth:* [1, 20]
- *learning_rate:* {0.01, 0.1, 0.5, 1}
- *subsample:* [0.2, 1.00], с шагом 0.05
- *min_child_weight:* [1, 20]

Теперь данный алгоритм имеет 27 200 возможных комбинации значений гиперпараметров. 74% точек осталось.

Следующие алгоритмы имеют относительно маленькое число возможных значений гиперпараметров, поэтому для них не будет выполняться поиск наиболее оптимальных возможных значений гиперепараметров.

3.1.5 Метод k-ближайших соседей (k-nearest neighbors)

Набор возможных значений для данного алгоритма является следующим:

- *n_neighbors:* [1, 100]
- *weights:* {uniform, distance}
- *p:* {1, 2}

3.1.6 Метод Лассо с использованием метода наименьших углов (LassoLars

Набор возможных значений для данного алгоритма является следующим:

- *normalize:* {True, False}

3.1.7 Эластичная сеть (ElasticNet)

Набор возможных значений для данного алгоритма является следующим:

- *l1_ratio:* [0, 1], с шагом 0.5
- *tol:* {1e-5, 1e-4, 1e-3, 1e-2, 1e-1}

3.1.8 AdaBoost

Набор возможных значений для данного алгоритма является следующим:

- *learning_rate:* {1e-3, 1e-2, 1e-1, 0.5, 1}
- *loss:* {linear, square, exponential}

3.1.9 Метод опорных векторов для регрессии (Support Vector Regression)

Набор возможных значений для данного алгоритма является следующим:

- *loss:* {epsilon_insensitive, squared_epsilon_insensitive}
- *tol:* {1e-5, 1e-4, 1e-3, 1e-2, 1e-1}
- *C:* {1e-4, 1e-3, 1e-2, 1e-1, 0.5, 1, 5, 10, 15, 20, 25, 50, 70}
- *epsilon:* {1e-4, 1e-3, 1e-2, 1e-1, 1}

3.1.10 Стохастический градиентный спуск (Stochastic Gradient Descent)

Набор возможных значений для данного алгоритма является следующим:

- *loss:* {squared_loss, huber, epsilon_insensitive}
- *penalty:* {elasticnet}
- *alpha:* {0.0, 0.01, 0.001}
- *learning_rate:* {invscaling, constant}
- *fit_intercept:* {True, False}
- *l1_ratio:* {0.25, 0.0, 1.0, 0.75, 0.5}
- *eta0:* {0.1, 1.0, 0.01}
- *power_t:* {0.5, 0.0, 1.0, 0.1, 100, 10, 50}

3.1.11 Ридж-регрессия (Rigde-regression)

Набор возможных значений для данного алгоритма является следующим:

- *alphas:* {1e-4, 1e-3, 1e-2, 1e-1, 0.5, 1, 5, 10, 15, 20, 25, 50, 70}

3.2 Алгоритмы классификации

Для вычисления оптимальных возможных значений гиперпараметров алгоритмов классификации используется 20 датасетов, таблица с ними будет приведена ниже.

Название датасета	Признаки	Объекты	Классы
Abalone	7	4 177	3
Bank Note	4	1 372	2
Breast Cancer	10	699	2
Cargo 2000 Freight	98	3 942	3
Dermatology	34	366	4
Haberman	3	306	2
Ionosphere	34	351	2
Letter Recognition	16	20 000	26
Leukemia	72	7130	2
Lung Cancer	56	42	2
Movement Libras	90	360	15
Mushrooms	22	8 124	6
Optical Digits	64	5 620	10
Pima Diabetes	8	768	2
Poker	10	25 010	10
Sonar	60	208	2
Spambase	57	4 601	2
Tic-Tac-Toe	9	958	2
Waveform	21	5 000	3
Winequality	17	101	7

Метрикой качества здесь будет являться F-мера: $F = \frac{2*precision*recall}{precision+recall}$

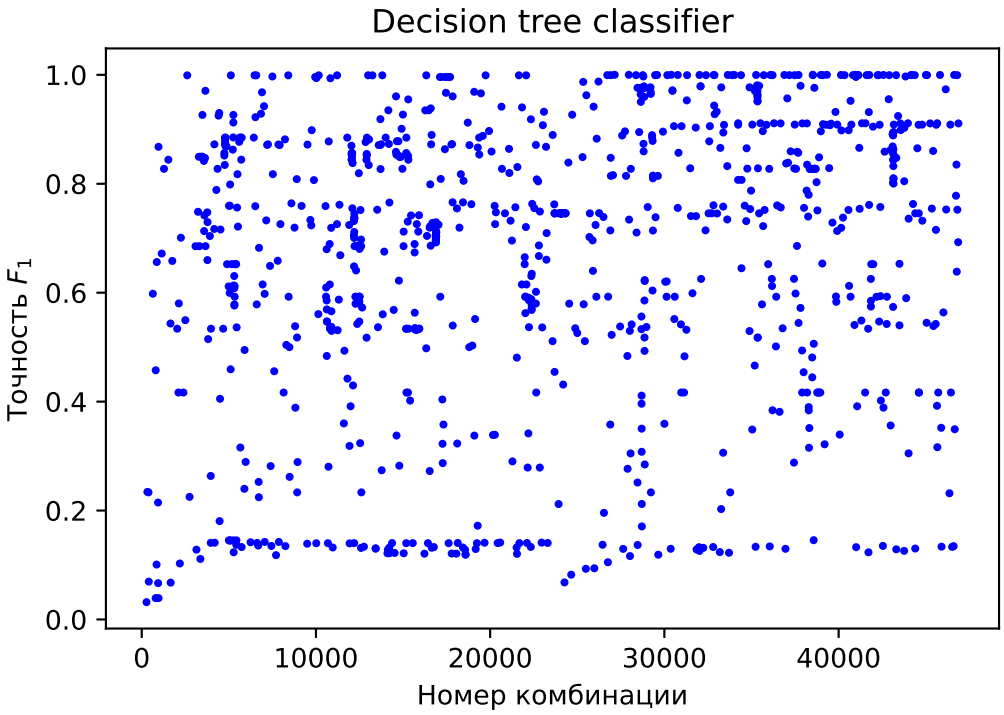

Рис. 3.9: Начальный набор значений гиперпараметров для алгоритма Decision tree classifier.

3.2.1 Дерево решений (Decision Tree)

Рассмотрим для данного алгоритма следующий начальный набор значений:

- *criterion:* {gini, entropy}
- *max_depth:* [1, 29]
- *min_samples_split:* [2, 29]
- *min_samples_leaf:* [1, 29]

Соответственно получая 47 096 возможных комбинаций значений гиперпараметров.

После того как мы избавимся от значений гиперпараметров, которые встречаются реже всего или имеют низкий показатель точности, получим следующие:

- *criterion:* {gini, entropy}

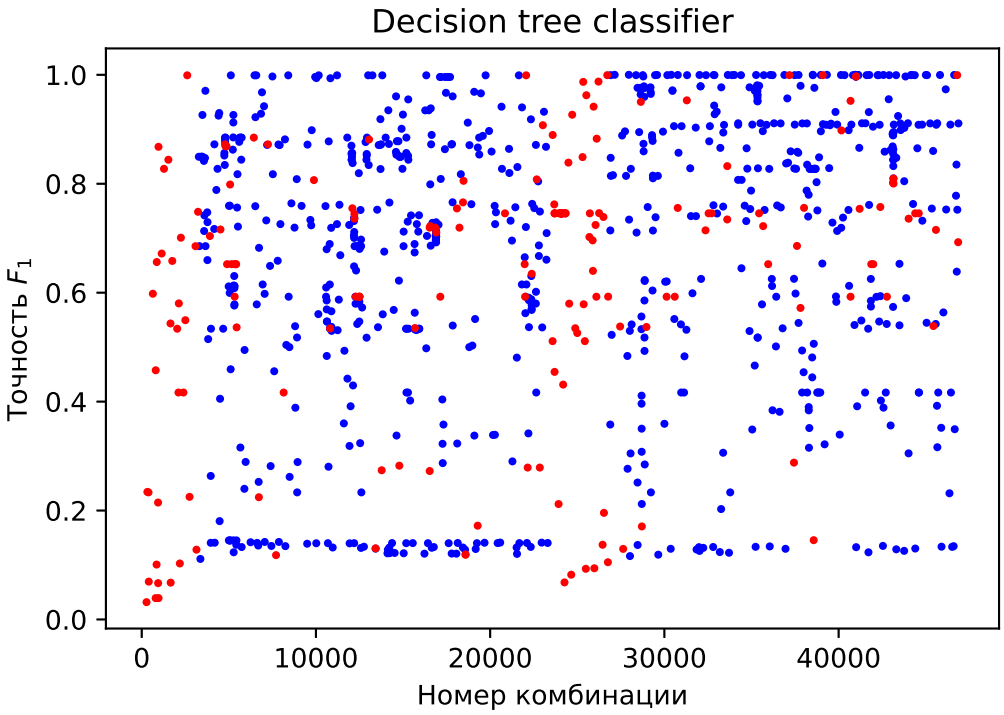

Рис. 3.10: Итоговый набор значений гиперпараметров для алгоритма Decision tree classifier.

- *max_depth:* [5, 29]
- *min_samples_split:* [1, 29]
- *min_samples_leaf:* [1, 25]

Теперь данный алгоритм имеет 35000 возможных комбинации значений гиперпараметров. 79% точек осталось.

3.2.2 Случайные и сверхслучайные деревья решения (Random Forest and Extra Trees)

Здесь будет рассмотрено сразу два алгоритма Random Forest и Extra Trees, так как это схожие алгоритмы и начальный набор значений гиперпараметров для них был один тот же. Для данных алгоритмов следующий начальный набор значений:

- *criterion:* {gini, entropy}

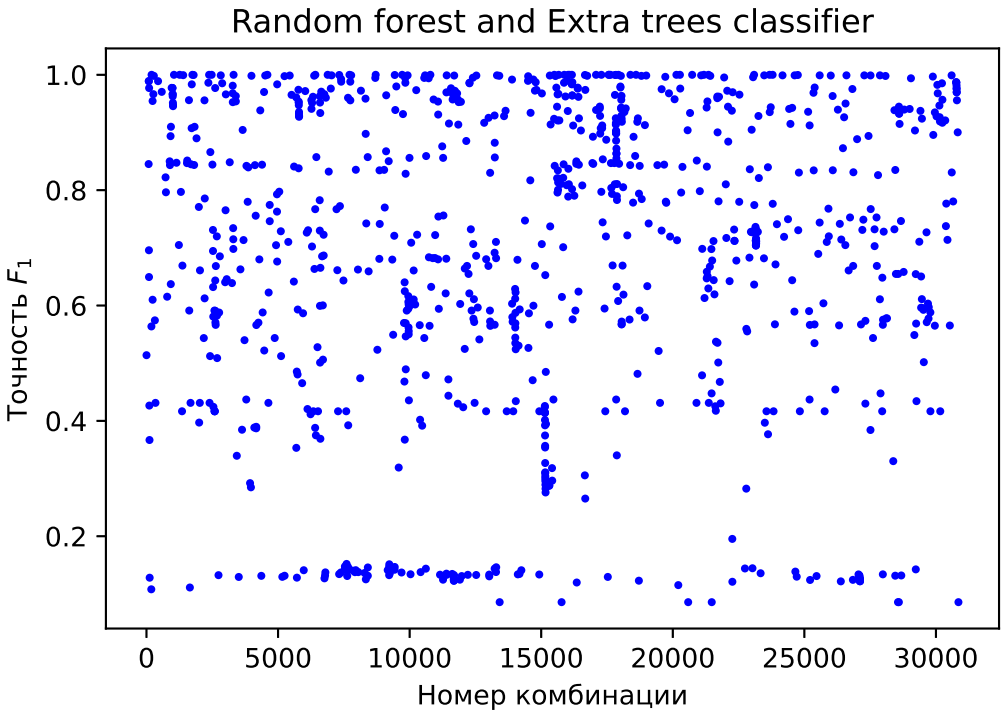

Рис. 3.11: Начальный набор значений гиперпараметров для алгоритмов Random forest classifier и Extra trees classifier.

- *max_features:* [0.1, 1.0], с шагом 0.05
- *min_samples_split:* [2, 29]
- *min_samples_leaf:* [1, 29]

Соответственно получая 30 856 возможных комбинаций значений гиперпараметров.

После того как мы избавимся от значений гиперпараметров, которые встречаются реже всего или имеют низкий показатель точности, получим следующие возможные значения:

- *criterion:* {gini, entropy}
- *max_features:* [0.1, 1.0], с шагом 0.05
- *min_samples_split:* [1, 29]
- *min_samples_leaf:* [1, 20]

Теперь данные алгоритмы имеют 21 280 возможных комбинации значений гиперпараметров. 82% точек осталось.

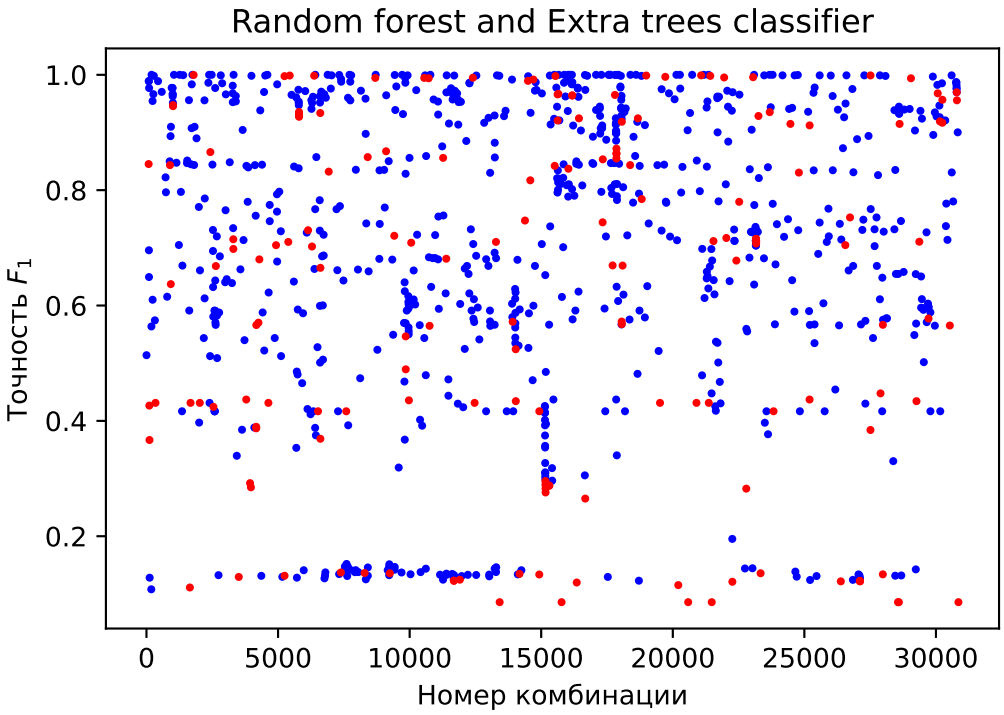

Рис. 3.12: Итоговый набор значений гиперпараметров для алгоритмов Random forest classifier и Extra trees classifier.

3.2.3 Градиентный бустинг (Gradient boosting)

Рассмотрим для данного алгоритма следующий начальный набор значений гиперпараметров:

- *learning_rate:* {1e-4, 1e-3, 1e-2, 1e-1, 0.5, 1}
- *max_depth:* [1, 19]
- *min_samples_split:* [2, 30]
- *min_samples_leaf:* [1, 30]
- *subsample:* [0.1, 1.00], с шагом 0.05
- *max_features:* [0.05, 1.0], с шагом 0.05

Соответственно получая 35 175 840 возможных комбинаций значений гиперпараметров.

После того как мы избавимся от значений гиперпараметров, которые встречаются реже всего или имеют низкий показатель точности, получим

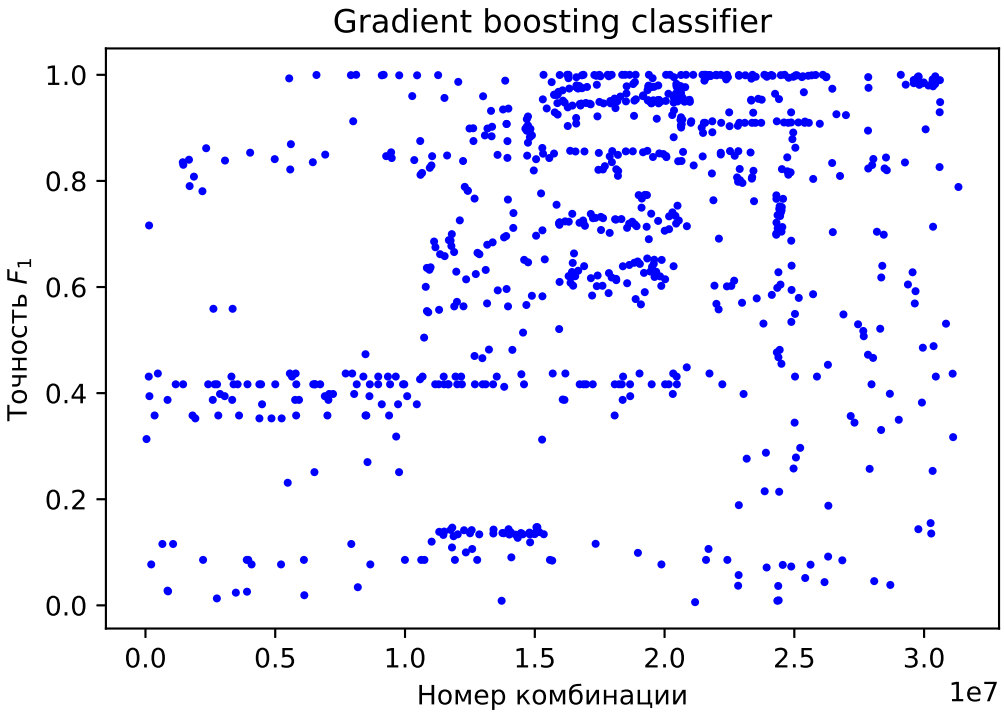

Рис. 3.13: Начальный набор значений гиперпараметров для алгоритма Gradient Boosting classifier.

следующие возможные значения:

- *learning_rate:* {1e-3, 1e-2, 1e-1, 0.5}
- *max_depth:* [1, 19]
- *min_samples_split:* [5, 30]
- *min_samples_leaf:* [1, 30]
- *subsample:* [0.1, 1.00], с шагом 0.05
- *max_features:* [0.05, 1.0], с шагом 0.05

Теперь данный алгоритм имеет 20 938 000 возможных комбинации значений гиперпараметров. 77% точек осталось.

3.2.4 XGBoost

Рассмотрим для данного алгоритма следующий начальный набор значений гиперпараметров:

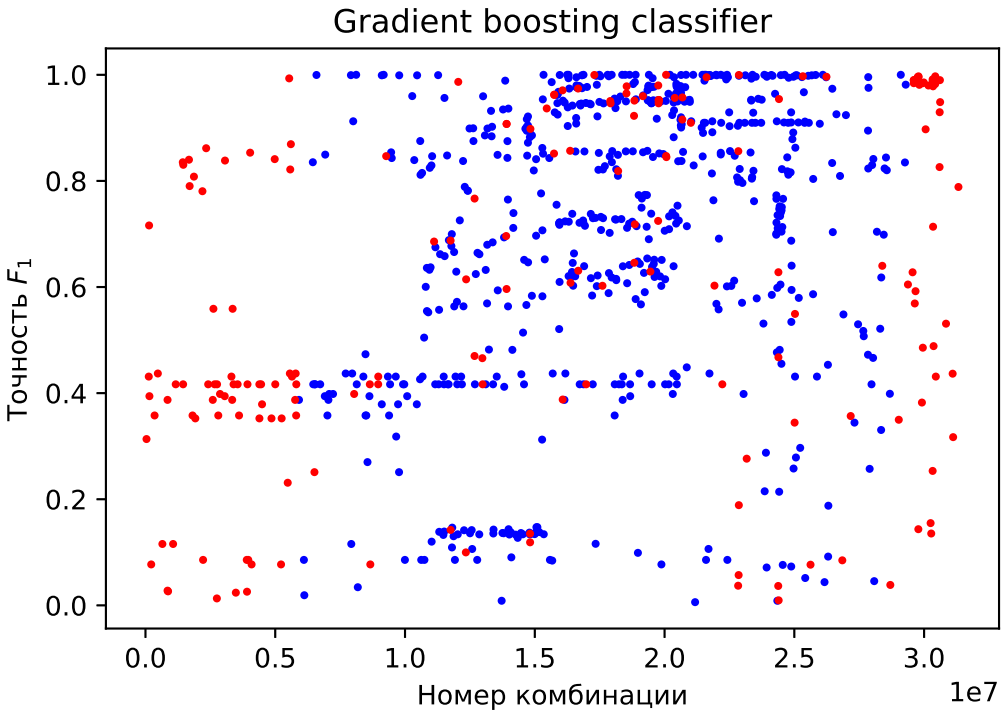

Рис. 3.14: Итоговый набор значений гиперпараметров для алгоритма для алгоритма Gradient Boosting classifier.

- *learning_rate:* {0.0001, 0.001, 0.01,, 0.1, 0.5, 1}
- *max_depth:* [1, 20]
- *subsample:* [0.05, 1.00], с шагом 0.05
- *min_child_weight:* [1, 29]

Соответственно получая 66 120 возможных комбинаций значений гиперпараметров.

После того как мы избавимся от значений гиперпараметров, которые встречаются реже всего или имеют низкий показатель точности, получим следующие возможные значения:

- *learning_rate:* {0.01, 0.1, 0.5, 1}
- *max_depth:* [1, 20]
- *subsample:* [0.05, 1.00], с шагом 0.05
- *min_child_weight:* [1, 20]

Теперь данный алгоритм имеет 30 400 возможных комбинации значений ги-

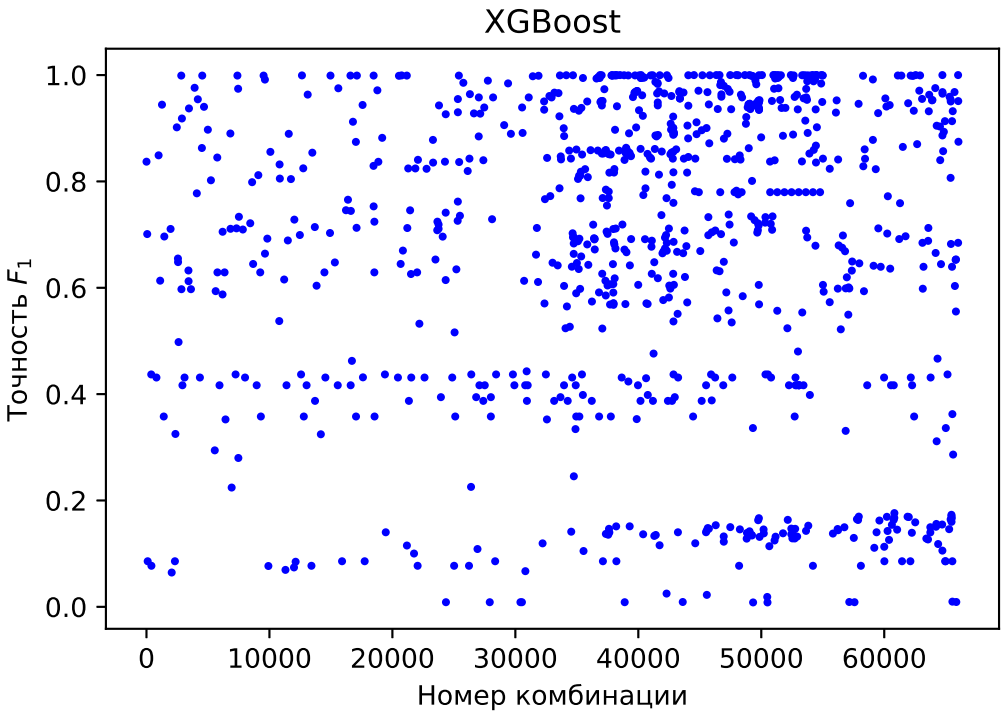

Рис. 3.15: Начальный набор значений гиперпараметров для алгоритма XGBoost classifier.

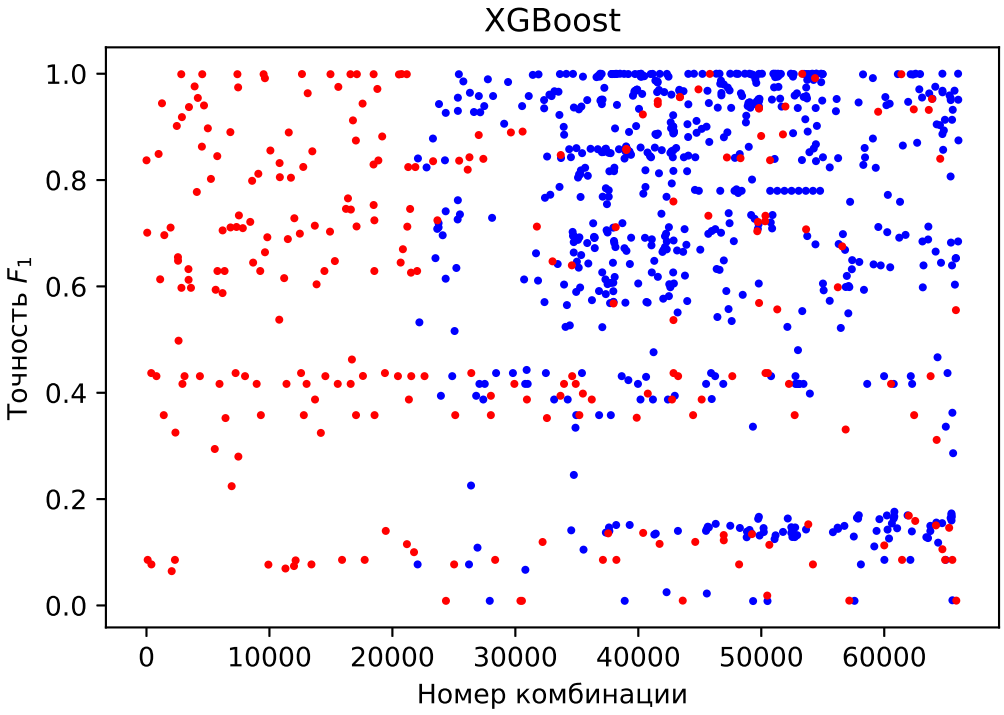

Рис. 3.16: Итоговый набор значений гиперпараметров для алгоритма для алгоритма XGBoost classifier.

перпараметров. 70% точек осталось.

Следующие алгоритмы имеют относительно маленькое число возможных значений гиперпараметров, поэтому для них не будет выполняться поиск наиболее оптимальных возможных значений гиперепараметров.

3.2.5 Метод k-ближайших соседей (k-nearest neighbors)

Набор возможных значений для данного алгоритма является следующим:

- *n_neighbors:* [1, 100]
- *weights:* {uniform, distance}
- *p:* {1, 2}

3.2.6 Логистическая регрессия (Logistic Regression)

Набор возможных значений для данного алгоритма является следующим:

- *penalty:* {elasticnet}
- *l1_ratio:* {0.25, 0.0, 1.0, 0.75, 0.5}
- *C:* {1e-4, 1e-3, 1e-2, 1e-1, 0.5, 1, 5, 10, 15, 20, 25, 50, 70}

3.2.7 AdaBoost

Набор возможных значений для данного алгоритма является следующим:

- *learning_rate:* {1e-3, 1e-2, 1e-1, 0.5, 1}

3.2.8 Стохастический градиентный спуск (Stochastic Gradient Descent)

Набор возможных значений для данного алгоритма является следующим:

- *loss:* {log, hinge, modified_huber, squared_hinge, perceptron}
- *penalty:* {elasticnet}
- *alpha:* {0.0, 0.01, 0.001}
- *learning_rate:* {invscaling, constant}
- *fit_intercept:* {True, False}
- *l1_ratio:* {0.25, 0.0, 1.0, 0.75, 0.5}
- *eta0:* {0.1, 1.0, 0.01}
- *power_t:* {0.5, 0.0, 1.0, 0.1, 100, 10, 50}

3.2.9 Метод опорных векторов для классификации (Support Vector Classification)

Набор возможных значений для данного алгоритма является следующим:

- *penalty:* { l1, l2 }
- *loss:* { hinge, squared_hinge }
- *tol:* {1e-5, 1e-4, 1e-3, 1e-2, 1e-1}
- *C:* {1e-4, 1e-3, 1e-2, 1e-1, 0.5, 1, 5, 10, 15, 20, 25, 50, 70}

3.2.10 Наивный байсовский классификатор (Naive Bayes)

Здесь будут приведены различные байсовские алгоритмы классификации и также значениях их гиперпараметров:

- GaussianNB
- BernoulliNB

  - *alpha:* { 1e-3, 1e-2, 1e-1, 1, 10, 100 }
- MultinomialNB
  - *alpha:* { 1e-3, 1e-2, 1e-1, 1, 10, 100 }

ГЛАВА 4

РЕЗУЛЬТАТЫ

В данной главе будет приведены результаты сравнения качества и скорости работы алгоритмов SAM и TPOT. Все измерения происходили на сайте https://colab.research.google.com.

Название датасета	Алгоритм	I	II	III
	SAM	97.81%	86.85%	89.3%

Automobile

ЛИТЕРАТУРА

[1] К. В. Воронцов. Математические методы обучения по прецедентам (теория обучения машин). http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf

[2] F. Hutter, L. Kotthoff, J. Vanschoren. Automated Machine Learning // Methods, Systems, Challenges // Springer, 2019

[3] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2013. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms.

[4] B. Komer, J. Bergstra, C. Eliasmith. Hyperopt-Sklearn // 2019

[5] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, M. Urban, M. Burkart, M. Dippel, M. Lindauer, and F. Hutter. Towards Automatically-Tuned Deep Neural Networks

[6] R. S. Olson and J. H. Moore. TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning

[7] F. Hutter, R. Caruana, R. Bardenet,M. Bilenko, I. Guyon, B. Kegl, and H. Larochelle. AutoML 2014 @ ICML

[8] Zheng A. Feature Engineering for Machine Learning [Text] / Zheng A., Casari A. // O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. – 2018.

[9] Emre R. Fundamental Techniques of Feature Engineering for Machine Learning [Электронный ресурс]. – 2019. – URL: https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114

[10] Deb K. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II [Текст] / Deb K. Pratap A., Agarwal S., Meyarivan T. // IEEE Transactions on evolutionary computation, VOL. 6, NO. 2, APRIL 2002. – P. 182–197.