

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**АЛГОРИТМЫ НАСТРОЙКИ ГИПЕРПАРАМЕТРОВ НА ОСНОВЕ
ОБЪЕДИНЕНИЯ АПРИОРНЫХ И АПОСТЕРИОРНЫХ ЗНАНИЙ О
ЗАДАЧЕ**

Автор: Смирнова Валентина Сергеевна _____

Направление подготовки: 01.03.02 Прикладная
математика и информатика

Квалификация: Бакалавр

Руководитель: Фильченков А.А., к.ф.-м.н. _____

К защите допустить

Руководитель ОП Парфенов В.Г., проф., д.т.н. _____

« ____ » _____ 20__ г.

Санкт-Петербург, 2020 г.

Обучающийся Смирнова В.С.

Группа М3435 Факультет ИТиП

Направленность (профиль), специализация

Математические модели и алгоритмы в разработке программного обеспечения

ВКР принята « ____ » _____ 20 ____ г.

Оригинальность ВКР ____ %

ВКР выполнена с оценкой _____

Дата защиты « ____ » _____ 20 ____ г.

Секретарь ГЭК Павлова О.Н. _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

УТВЕРЖДАЮ

Руководитель ОП
проф., д.т.н. Парфенов В.Г. _____
« ____ » _____ 20__ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Обучающийся Смирнова В.С.

Группа М3435 **Факультет/институт/кластер** ИТиП

Квалификация Бакалавр

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) образовательной программы Математические модели и алгоритмы в разработке программного обеспечения

Специализация

Тема ВКР Алгоритмы настройки гиперпараметров на основе объединения априорных и апостериорных знаний о задаче

Руководитель Фильченков А.А., к.ф.-м.н, доцент факультета информационных технологий и программирования, Университет ИТМО

2 Срок сдачи студентом законченной работы до « ____ » _____ 20__ г.

3 Техническое задание и исходные данные к работе

Требуется разработать алгоритм настройки гиперпараметров на основе объединения априорных и апостериорных знаний о задаче

4 Содержание выпускной работы (перечень подлежащих разработке вопросов)

В работе должна быть показана эффективность разработанного решения по сравнению с существующими

5 Перечень графического материала (с указанием обязательного материала)

Графические материалы и чертежи работой не предусмотрены

6 Исходные материалы и пособия

- а) Hutter, F., Hoos, H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. // LION'11. — 2011
- б) Efficient and robust automated machine learning / M. Feurer // Advances in neural information processing systems. — 2015.
- в) Leite R., Brazdil P. Active Testing Strategy to Predict the Best Classification Algorithm via Sampling and Metalearning. // ECAI. — 2010.

7 Дата выдачи задания « ____ » _____ 20__ г.

Руководитель ВКР _____

Задание принял к исполнению _____

« ____ » _____ 20__ г.

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Обучающийся Смирнова Валентина Сергеевна

Наименование темы ВКР Алгоритмы настройки гиперпараметров на основе объединения априорных и апостериорных знаний о задаче

Наименование организации, где выполнена ВКР: Национальный Исследовательский Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: Разработать алгоритм настройки гиперпараметров на основе объединения априорных и апостериорных знаний о задаче

2 Задачи, решаемые в ВКР:

- а) изучить существующие решения поставленной задачи;
- б) предложить и реализовать новый алгоритм;
- в) провести эксперименты, показывающие эффективность решения.

3 Число источников, использованных при составлении обзора: 0

4 Полное число источников, использованных в работе: 21

5 В том числе источников по годам:

Отечественных			Иностраннх		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	0	14	4	3

6 Использование информационных ресурсов Internet: нет

7 Использование современных пакетов компьютерных программ и технологий:

Пакеты компьютерных программ и технологий	Раздел работы
Пакет numpy	2, 3
Пакет pandas	2, 3
Пакет robo	2, 3
Пакет matplotlib	2, 3
Пакет george	2, 3

8 Краткая характеристика полученных результатов

 Был предложен и реализован эффективный алгоритм, решающий поставленную задачу.

9 Гранты, полученные при выполнении работы

 Отсутствуют

10 Наличие публикаций и выступлений на конференциях по теме работы

а) IX Конгресс Молодых Учёных

Обучающийся Смирнова В.С. _____

Руководитель ВКР Фильченков А.А. _____

« _____ » _____ 20__ г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	6
1. Обзор существующих решений	8
1.1. Определения и ключевые понятия	8
1.2. Обзор классификаторов	8
1.3. Меры оценки качества классификации	10
1.4. Обзор подходов к оптимизации гиперпараметров	11
1.4.1. Сравнение существующих подходов к оптимизации	13
1.4.2. Байесовская оптимизация	13
Выводы по главе 1	15
2. Предложенное решение	16
2.1. Параметры для оптимизации	16
2.1.1. Модель оптимизации	16
2.1.2. Классификатор	16
2.1.3. Целевая функция	18
2.2. Метрика для определения подобия задач	19
2.3. Расширение Байесовской оптимизации	20
2.3.1. Апостериорная информация	20
2.3.2. Расстановка начальных конфигураций	22
Выводы по главе 2	25
3. Анализ полученных результатов	26
3.1. Подробности реализации	26
3.2. Статистические результаты, дисперсия	26
3.3. Результаты классической Байесовской оптимизации	27
3.4. Сравнение результатов классической и расширенной Байесовской оптимизации	29
3.4.1. Оценка качества оптимизации	30
3.4.2. Оценка результатов	30
Выводы по главе 3	34
ЗАКЛЮЧЕНИЕ	35
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	36
ПРИЛОЖЕНИЕ А. Информация об отобранных датасетах	39
ПРИЛОЖЕНИЕ Б. Результаты классической Байесовской оптимизации ..	42
ПРИЛОЖЕНИЕ В. Результаты расширенной Байесовской оптимизации ..	45

ПРИЛОЖЕНИЕ Г. Дисперсии для классической Байесовской оптимизации	48
ПРИЛОЖЕНИЕ Д. Дисперсии для расширенной Байесовской оптимизации	51
ПРИЛОЖЕНИЕ Е. Значимость результатов	54

ВВЕДЕНИЕ

Задача классификации – построить алгоритм (классификатор), который по набору признаков вернул бы метку класса или вектор оценок принадлежности (апостериорных вероятностей) к каждому из классов. Основная её цель – максимально точно определить метку класса для заданного объекта. Задача широко применяется во многих областях:

- Медицинская диагностика: по набору медицинских характеристик требуется поставить диагноз
- Геологоразведка: по данным зондирования почв определить наличие полезных ископаемых
- Оптическое распознавание текстов: по отсканированному изображению текста определить цепочку символов, его формирующих
- Кредитный скоринг: по анкете заемщика принять решение о выдаче/отказе кредита
- Синтез химических соединений: по параметрам химических элементов спрогнозировать свойства получаемого соединения

Найти и обучить эффективный алгоритм классификации – трудоёмкая задача, которая включает в себя выбор самого классификатора, сбор и разметку данных (в случае обучения с учителем), и непосредственно настройку гиперпараметров классификатора. Так как гиперпараметры задаются до начала обучения и не изменяются в его ходе, а при этом могут существенно влиять на результат обучения, то появляется задача оптимизации гиперпараметров.

Оптимизация гиперпараметров – одна из задач машинного обучения, которая занимается выбором набора оптимальных гиперпараметров для обучающего алгоритма. В качестве гиперпараметров могут выступать различные предположения, веса, скорости обучения, а из разные значения могут давать различный результат для различных видах данных. Гиперпараметры следует настраивать так, чтобы модель могла оптимально решать поставленную задачу обучения. Это может быть кластеризация, классификация и т. д. Для этого находится набор гиперпараметров, который даёт оптимальную модель, оптимизирующую заданную функцию потерь на заданных независимых данных[4]. Такая задача несёт название AutoML [7], основная её задача – сделать процесс машинного обучения доступным не только для экспертов в области ML, но и для любого пользователя.

Интерес к задаче AutoML растет, проводятся конференции и соревнования по её решению. В частности, каждые два года проходит соревнование AutoML Challenge. В августе 2019 года происходило мероприятие, посвящённое этой теме – The Third International Workshop on Automation in Machine Learning[1]. Согласно выпуску Forbes за декабрь 2018 года, это один из пяти трендов в развитии машинного обучения в 2019 году. Тема AutoML появляется все чаще и чаще в дискуссиях и публикациях. Решения этой задачи уже используются в автономных машинах, предсказании цен и многих других областях.

Существующие решения [5, 6, 9, 13] для задачи оптимизации гиперпараметров основываются на случайной расстановке гиперпараметров и дальнейшей их настройке. Цель настоящей работы – расширить существующий алгоритм до получения лучших результатов при таком же условно временном лимите на обучение по средствам расстановки гиперпараметров не случайным образом, а с помощью особого подхода, основанного на результатах обучения смежных задач.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Более подробно рассмотрим задачу классификации, алгоритмы ее решения и способы оценки качества полученной классификации. Также опишем понятие модели алгоритма классификации и основные методы настройки её гиперпараметров, применимые ко многим алгоритмам машинного обучения. Заметим, что на сегодняшний момент не предложено способов не случайной расстановки гиперпараметров и дальнейшей их настройки, на основе результатов обучения на схожих задачах.

1.1. Определения и ключевые понятия

Для начала введем определения и ключевые понятия, которые будут использоваться в дальнейшей работе.

- классификатор – параметризованный алгоритм, решающий задачу классификации
- конфигурация – фиксированный набор параметров классификатора
- алгоритм – пара из классификатора и его конфигурации
- оптимизатор – алгоритм, который используется для оптимизации гиперпараметров
- решаемая задача – алгоритм с настроенными гиперпараметрами на конкретном датасете
- решённая задача – алгоритм с оптимизированными гиперпараметрами на конкретном датасете
- соседняя задача – ближайшая задача к решаемой
- текущее решение (текущая задача) – решаемая задача, для которой могут использоваться сведения из решённых (соседних) задач

1.2. Обзор классификаторов

В данной части рассмотрим популярные модели для решения задачи классификации. Вспомним, что цель задачи классификации – наиболее точно определять метку класса по заданному объекту.

Итак, наиболее используемые на сегодняшний момент модели классификаторов:

Линейная регрессия Можно представить в виде уравнения, которое описывает прямую, наиболее точно показывающую взаимосвязь между входными переменными X и выходными переменными Y

Логистическая регрессия По аналогии с линейной регрессией требуется найти коэффициенты для входных данных, но уже с помощью нелинейной или логистической функции.

Линейный дискриминантный анализ (LDA) Состоит из статистических свойств данных, рассчитанных для каждого класса

- Среднее значение для каждого класса
- Дисперсия, рассчитанная по всем классам

Деревья принятия решений [16] Представима в виде бинарного дерева, где каждый узел представляет собой входную переменную и точку разделения для этой переменной (при условии, что переменная — число)

Наивный Байесовский классификатор Состоит из двух типов вероятностей, которые рассчитываются с помощью тренировочных данных:

- а) Вероятность каждого класса
- б) Условная вероятность для каждого класса при каждом значении x

К-ближайших соседей (KNN) Предсказание метки класса делается на основе меток k ближайших соседей

Метод опорных векторов (SVM) Суть метода заключается в построении гиперплоскости, разделяющей классы

Бэггинг и случайный лес (RandomForest) Один из наиболее эффективных алгоритмов классификации, берётся множество подвыборок из данных, считается среднее значение для каждой, а затем усредняются результаты для получения лучшей оценки действительного среднего значения

Бустинг и AdaBoost : принадлежит семейству ансамблевых алгоритмов, суть которых заключается в создании сильного классификатора на основе нескольких слабых

Многослойный персептрон (Multilayered perceptron) Класс искусственных нейронных сетей прямого распространения, состоящих как минимум из трех слоёв: входного, скрытого и выходного

В работе мы заострим внимание на модели случайного леса как самой эффективной на сегодняшний момент и модели многослойного персептрона, так как он обладает наибольшим количеством гиперпараметров и также является достаточно эффективным.

1.3. Меры оценки качества классификации

Кроме выбора алгоритма классификации и его гиперпараметров, обучить модель, нужно ещё каким-то образом оценить качество работы обученного алгоритма. Для этого датасет делится на 2 части: *train* (на которой модель обучается) и *test* или *validate* (на которой оценивается качество классификации). В данной части мы рассмотрим существующие меры оценки качества классификации и выделим существенные для нашей задачи. Для этого сначала вспомним базовые понятия[3]:

- Верно-положительными (TP) называются объекты, которые были классифицированы как положительные и действительно являются таковыми
- Верно-отрицательными (TN) называются объекты, которые были классифицированы как отрицательные и действительно таковые
- Ложно-положительными (FP) называются объекты, которые были классифицированы как положительные, но фактически отрицательные
- Ложно-отрицательными (FN) называются объекты, которые были классифицированы как отрицательные, но фактически положительные

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Одна из наиболее простых и популярных мер оценки качества – *F-мера* или *F-score*[19]. Считается она следующим образом:

$$F - score = 2 * \frac{precision * recall}{precision + recall} \quad (4)$$

Приемущество данной меры в том, что её достаточно просто считать и результат отлично подходит для целевой функции оптимизации, о которой мы поговорим в следующей части.

Также существует *кривая ошибки* или *ROC-curve (Receiver Operating Characteristic)*. Суть данной меры состоит в том, что считается площадь кривой под графиком уровня верно-положительных предсказанных экземпляров от уровня ложно-положительных. Не будем заострять на ней внимание, так как к нашей задаче она не подходит из-за трудоёмкости расчёта.

1.4. Обзор подходов к оптимизации гиперпараметров

Существует несколько подходов к задаче оптимизации гиперпараметров. В данной части рассмотрим наиболее популярные из них, оценим преимущества и недостатки.

Поиск по решётке По сути, данный алгоритм делает полный перебор всех возможных моделей и конфигураций.

Доступные реализации

- LIBSVM
- scikit-learn
- Talos

Достоинства Гарантирована будет найдена наилучшая конфигурация.

Недостатки Слишком большие затраты на обучение.

Случайный поиск Отличается от поиска по решётке тем, что идёт не полный перебор всех конфигураций, а случайная их выборка.

Доступные реализации

- hyperopt
- scikit-learn

- H2O AutoML
- Talos

Достоинства Меньшие затраты на обучение. Существует вероятность нахождения наилучшей конфигурации за наименьшее время.

Недостатки Неопределённое количество времени на поиск наилучшей конфигурации.

Байесовская оптимизация Метод, основанный на обращении к функции «чёрного ящика» с шумом. Для задачи оптимизации гиперпараметров строит стохастическую модель из отображения из конфигурации в целевую функцию, применённую на валидационном наборе данных.

Доступные реализации

- Spearmint
- Bayesopt
- MOE
- Auto-WEKA
- Auto-sklearn
- mlrMBO
- tuneRanger
- BOCS
- SMAC

Достоинства Небольшие затраты на обучение, количество итераций задаётся вручную, адаптируется под значимость каждого гиперпараметра для конкретной задачи.

Недостатки Сложность реализации и использования.

Оптимизация на основе градиентов Для определённых алгоритмов обучения вычисляется градиент гиперпараметров и оптимизируется с помощью градиентного спуска.

Доступные реализации

— hypergrad

Достоинства Небольшие затраты на обучение, неплохой результат.

Недостатки Сложность понимания и реализации, небольшое количество доступных реализаций.

Эволюционная оптимизация Также, как и Байесовская оптимизация, основывается на обращениях к функции «чёрного ящика» с шумом, однако для поиска гиперпараметров для заданного алгоритма использует эволюционные подходы (алгоритмы)[2].

Доступные реализации

— TROT
— devol
— deap

Достоинства Небольшие затраты на обучение, неплохой результат.

Недостатки Используется только для статистических алгоритмов. Давайте немного подробнее рассмотрим Байесовскую оптимизацию.

1.4.1. Сравнение существующих подходов к оптимизации

Не смотря на то, что задача оптимизации гиперпараметров достаточно нова и не имеет под собой десятки лет исследований, уже сегодня существует множество подходов и алгоритмов для её решения. В одном из исследований[21] проведено достаточно подробное и качественное сравнение существующих алгоритмов и подходов. По результатам данного исследования можно заключить, что самый эффективный подход к задаче оптимизации гиперпараметров – Байесовская оптимизация. Рассмотрим данный подход поподробнее.

1.4.2. Байесовская оптимизация

Как уже говорилось в предыдущей части 1.4, Байесовская оптимизация основывается на обращении к функции «чёрного ящика» с шумом. Кроме того, подход требует определить модель, на которой будет основываться сама

Байесовская оптимизация, максимизатор, способ задания первичных гиперпараметров и непосредственно целевую функцию. Сам алгоритм представляет из себя следующую цепочку действий:

Листинг 1 – Байесовская оптимизация

```

задать первичные гиперпараметры
for количество итераций do
    выбрать следующую точку для рассмотрения
    получить результат целевой функции в этой точке
    сохранить лучшее значение и конфигурацию
end for

```

Точка в алгоритме определяется конфигурацией гиперпараметров. Лучшее значение – минимальное значение, выданное целевой функцией в данной точке. Наиболее интересная и значимая часть в алгоритме – выбор следующей для рассмотрения точки. Именно в этой части модель обучается на уже рассмотренных точках и обновляет так называемую *функцию выгоды* (*acquisition function*), после чего в максимуме функции выгоды будет место с наибольшей неопределённостью, значит, там нав и необходимо будет «смотреть» на точку. Подробнее про подход описано в AutoML Book[[automlbook19a](#)].

Модель Определяет, как будет обновляться пространство гиперпараметров модели оптимизатора. Важно отметить, что гиперпараметры оптимизатора никаким образом не связаны с гиперпараметрами классификатора (которые мы пытаемся оптимизировать). Наиболее распространённые модели:

- а) GaussianProcess
- б) GaussianProcessMCMC
- в) RandomForest
- г) WrapperBohamiann
- д) DNGO

У каждой модели есть свои преимущества и свои недостатки, например, в Гауссовском процессе нет возможности работать с категориальными признаками, DNGO же обладает наименее точным результатом.

Максимизатор В Байесовской оптимизации работает с функцией выгоды, которая находит следующую наиболее интересную точку. Максимизаторы бывают:

- а) RandomSampling
- б) SciPyOptimizer
- в) DifferentialEvolution

Функция выгоды Служит для определения наименее информативной области. В этой области меньше всего информации о значении функции, которую мы пытаемся аппроксимировать, поэтому в экстремуме функции выгоды будет находиться наиболее интересная на данный момент конфигурация гиперпараметров. Популярные функции выгоды:

- а) EI
- б) LogEI
- в) PI
- г) LCB

Целевая функция В общем случае принимает на вход конфигурацию гиперпараметров и отдаёт значение той самой функции «чёрного ящика»[12], значение которой минимизируется в процессе Байесовской оптимизации. В случае рассматриваемой нами задачи классификации, целевая функция будет обучать классификатор с полученными гиперпараметрами на третировочной выборке и вернёт обратное значение (так как функция минимизируется) меры оценки качества классификации, посчитанной на валидационной выборке.

Выводы по главе 1

В первой главе рассмотрены ключевые определения и понятия, необходимые для понимания решаемой задачи. Оценены преимущества и недостатки параметров для оптимизации гиперпараметров, такие как классификаторы, меры оценки качества классификации, подходы к задаче оптимизации и их параметры. Также подробно рассмотрен один из основных и наиболее популярных подходов – Байесовская оптимизация. Описаны функции и задачи модели оптимизатора, его целевой функции, функции выгоды и максимизатора.

ГЛАВА 2. ПРЕДЛОЖЕННОЕ РЕШЕНИЕ

За основу предложенного решения была выбрана Байесовская оптимизация, а именно реализация *RoBO (Robust Bayesian Optimization Framework)*[20]. Для однозначного понимания, далее будем называть её «классической Байесовской оптимизацией», чтобы иметь возможность отличать от непосредственно предложенного решения. Задача данного исследования – добиться лучших результатов оптимизации. Под «лучшими» результатами стоит понимать лучшее (минимальное) значение *incumbent*¹, полученное на более ранней итерации оптимизации. Поэтому достаточно задать фиксированные параметры классического решения и сравнивать его результаты с результатами доработанного решения с теми же параметрами. Далее будут рассмотрены выбранные в работе параметры для классической Байесовской оптимизации.

2.1. Параметры для оптимизации

В первой главе 1.4.2 сравнили параметры, необходимые для запуска классической Байесовской оптимизации. В текущей главе будут выбраны наиболее подходящие нам параметры.

2.1.1. Модель оптимизации

В качестве модели оптимизации в работе используется Гауссовский процесс. Как известно, оптимизаторы, построенные на модели случайного леса пользуются бóльшим спросом по причине того, что даёт лучшие результаты, однако, как очевидно из названия, данная модель основана на случайности, что не несёт под собой точного математического обоснования, в отличие от Гауссовского процесса. Именно по этой причине в работе используется именно Гауссовский процесс.

2.1.2. Классификатор

2.1.2.1. Обоснование

При выборе классификатора, стоит помнить, что мы решаем задачу оптимизации гиперпараметров, значит, набор гиперпараметров классификатора должен соответствовать следующим свойствам:

- а) иметь ненулевой конечный набор гиперпараметров
- б) настраиваемые гиперпараметры должны быть вещественными числами

¹значение целевой функции

- в) не иметь категориальных гиперпараметров, так как в качестве модели оптимизатора выступает Гауссовский процесс, который не предусматривает работу с категориальными гиперпараметрами

К сожалению, не существует модели, удовлетворяющей всем перечисленным свойствам, поэтому было принято решение игнорировать (фиксировать определенное значение и не изменять в процессе всей оптимизации) категориальные гиперпараметры (представлять их в виде чисел было бы некорректно) и приводить числа с плавающей точкой к целым на местах гиперпараметров, где требуются целочисленные значения (данное пренебрежение корректно, так как значения гиперпараметров, требующие целочисленности, располагаются в диапазоне, много большем, чем потеря при округлении).

Кроме того, стоит помнить, что необходимо подобрать классификатор с как можно большим количеством подходящих гиперпараметров, чтобы была возможность корректно оценить работу оптимизатора. Более подробно: гиперпараметры бывают более значимы для классификатора или менее значимы, их значимость определяет именно оптимизатор в ходе оптимизации. Поэтому в выбранном классификаторе должны присутствовать и те, и другие.

Проанализировав все требования к классификатору, был выбран многослойный парцептрон Румельхарта[8, 14] (Рисунок 1).

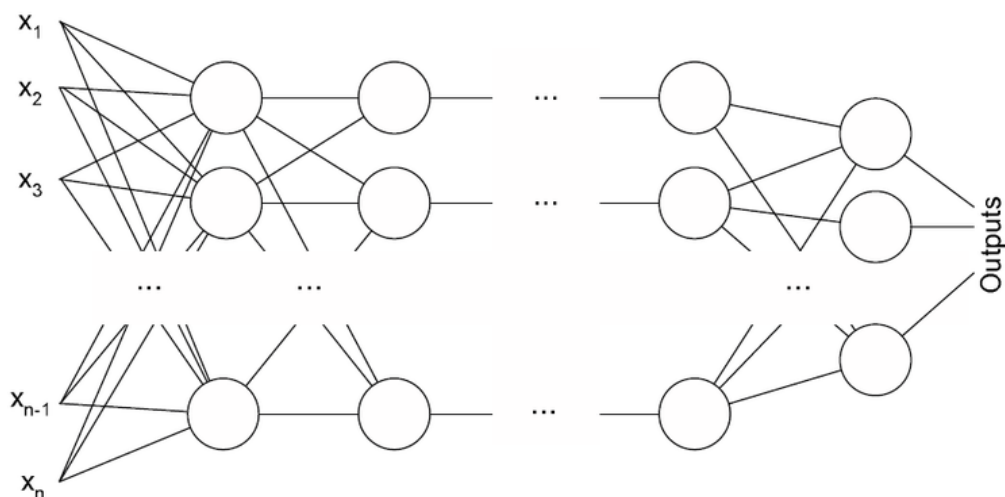


Рисунок 1 – Многослойный парцептрон Румельхарта[8]

2.1.2.2. Набор гиперпараметров

Гиперпараметры многослойного парцептрона Румельхарта, которые будут использованы для оптимизации (в круглых скобках указан тип гиперпараметра, в квадратных – диапазон его значений):

- hidden layer sizes (*int*) – число скрытых слоёв [1, 150]
- α (*float*) – L2 штраф [0,00001, 0,01]
- learning rate initial (*float*) – начальная скорость обучения [0,0001, 0,1]
- max iterations (*int*) – максимальное число итераций [50, 300]
- validation fraction (*float*) – доля данных обучения, отведенных в качестве проверки для преждевременной остановки [0,01, 0,9]
- β_1 (*float*) – скорость экспоненциального убывания для оценок вектора первого момента [0,09, 0,9]
- β_2 (*float*) – Скорость экспоненциального убывания для оценок вектора второго момента [0,0999, 0,999]
- n iterations no change (*int*) – максимальное число эпох, пройденных без прогресса [5, 15]

2.1.3. Целевая функция

Как было упомянуто в части 1.4.2, целевая функция в нашем случае должна отдавать значение меры качества классификации, вернее обратное её значение, так как идёт процесс минимизации. Поэтому вернёмся к части 1.3 и выберем подходящую нам меру оценки качества классификации.

Мера оценки качества классификации Наиболее подходящей мерой в нашем случае является *F-score*, так как она проста в вычислении и от неё легко берётся обратное значение: $(1 - F\text{-score})$ – именно это значение будет выдаваться целевой функцией при каждом обращении к ней. Более детально рассмотрим процесс, происходящий внутри целевой функции. До запуска оптимизации, функция инициализируется выборкой, разделённой на 2 части: *train* и *validate*. Далее запускается оптимизационный процесс, где вызывается наша целевая функция. При каждом обращении, на вход функции подаётся конфигурация гиперпараметров, сгенерированная алгоритмом оптимизации для классификатора. После чего запускается процесс обучения классификатора с заданной конфигурацией на тренировочной части выборки, далее запускается процесс предсказания меток классов на валидационной части выборки, считается значение *F-score* и возвращается описанное ранее значение целевой функции. Затем оптимизатор анализирует полученный результат, перестраивает конфигурацию и повторяет до тех пор, пока не достигнет последней итерации.

2.2. Метрика для определения подобия задач

В работе планируется использовать не только априорные знания по решаемой задаче, но и апостериорные знания, полученные при решении других задач. Но так как задач (датасетов) существует бесконечное множество, то возникает вопрос: каким образом выбирать подходящие для нашей задачи и какую информацию из обучения использовать. Фактически у всех датасетов разное число признаков, классов и радикально разные диапазоны значений. А нам необходимо выяснить не только насколько та или зая задача похожа на решаемую, но и дать количественную оценку подобию задач. в этой части мы рассмотрим меру оценки подобию задач.

Первое, что приходит на ум – рассчитать расстояние между задачами, однако для расчёта расстояний между задачами, необходимо, чтобы задачи имели одинаковую размерность. Для этого их необходимо привести к общему виду.

Один из способов – для каждого датасета рассчитать метапризнаки[10] и уже по ним считать расстояния. Для этого для каждого признака возьмём некую характеристику его связи с классом, попарные корреляции между признаками и характеристики структуры дерева принятия решений. После чего на каждом полученном массиве посчитаем статистики (минимум, максимум, среднее и т.д.). полученный массив и будет массивом метапризнаков. Однако этого ещё не достаточно для расчёта расстояния, так как значения по-прежнему довольно различаются. Поэтому необходимо также нормализовать массивы полученных метапризнаков. После нормализации мы получим не только подходящие значения, но и устраним ковариации и уравним дисперсию. В качестве нормализации используем Расстояние Махаланобиса – это мера расстояния между векторами случайных величин, обобщающая понятие евклидова расстояния. Именно данный подход к нормализации учитывает в себе ковариации. Формально расстояние Махаланобиса от рассматриваемого вектора $x = (x_1, x_2, x_3 \dots x_N)^T$ до множества со средним значением $\mu = (\mu_1, \mu_2, \mu_3 \dots \mu_N)^T$ матрицей ковариации S определяется следующим образом:

$$D_M(x) = \sqrt{(x - \mu)^T S^{-1} (x - \mu)} \quad (5)$$

Для удобства изначально домножим на $S^{-1/2}$, чтобы сократить затраты на расчёт. После того, как мы получим набор нормализованных признаков,

посчитаем обыкновенное Евклидово расстояние между задачами:

$$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \quad (6)$$

Теперь для каждой пары задач мы знаем расстояние между ними и можем использовать это расстояние для оценки подобия задач. Однако в реальности у нас не фиксированный набор задач, постоянно добавляются новые датасеты, а так как нормализация производится по фиксированному множеству задач, то добавление нового датасета может исказить уже посчитанные расстояния и придётся пересчитывать всё с самого начала. Казалось бы это не является универсальным решением, однако если учесть, что предпоставленных задач у нас много больше, чем приходящих (например, имеется 100 предпросчитанных задач и добавляется одна новая), то на значения факт добавления задачи повлияет незначительно и можно считать необходимые значения только для новой задачи.

2.3. Расширение Байесовской оптимизации

В параграфе 1.4 описан классический подход к Байесовской оптимизации, цель настоящей работы – расширить существующий алгоритм до получения лучших результатов при таком же условно временном лимите на обучение. За временной лимит стоит считать количество итераций оптимизатора. Для начала давайте определим, какая информация из соседних задач нам доступна и какая представляет для нас ценность в решении текущей задачи

2.3.1. Апостериорная информация

На первом этапе для каждого датасета запустим классическую Байесовскую оптимизацию, на выходе которой получим следующую апостериорную информацию:

- X_{opt} – оптимальная конфигурация гиперпараметров
- y_{opt} – оптимальное значение целевой функции
- X – массив рассмотренных конфигураций в порядке итераций
- y – массив значений целевых функций в порядке итераций
- *incumbents* – массив оптимальных конфигураций, известных на момент текущей итерации (значение обновляется только в случае получения лучшего результата)

- *incumbent_vvalues* – массив соответствующих значений целевой функции
- *runtime* – массив значений времени, затраченного на соответствующую итерацию (включает в себя время обращения к целевой функции и расчёт значений *incumbent*)
- *overhead* – массив значений времени, затраченного на оптимизацию сверх функции

Значения *runtime* и *overhead* могут быть полезны для оценки времени работы модифицированного алгоритма. По значениям *incumbent values* можно оценить на каких итерациях мы получаем улучшения, а значения оптимальных конфигураций использовать как полезную нам апостериорную информацию для решаемой задачи.

Рассмотрим наглядный пример, что происходит на каждой итерации. На рисунке 2 представлен пример Байесовской оптимизации на основе гауссовского процесса. В качестве примера взята одномерная функция для упрощения понимания. Точки на графике определяют гиперпараметры, в данном случае гиперпараметр. Оранжевой пунктирной линией показано настоящее значение

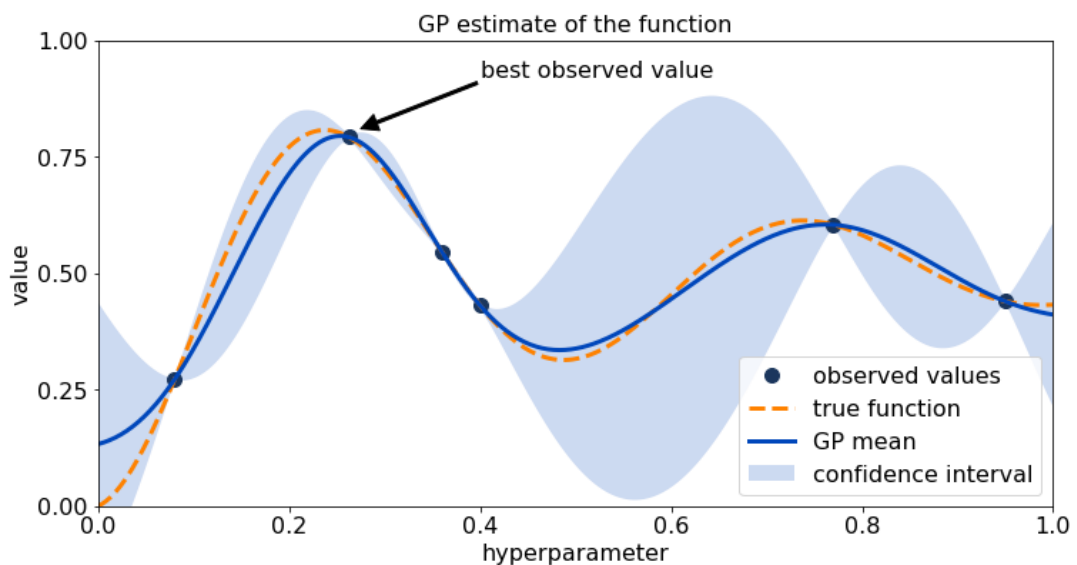


Рисунок 2 – Пример Байесовской оптимизации на основе Гауссовского процесса для одномерной функции [11].

функции «чёрного ящика», которое на самом деле нам не известно, онако мы пытаемся с течением каждой итерации оптимизации приблизиться к её значению. Как уже говорилось, на каждой итерации мы получаем значение целевой функции (это и есть наша функция «чёрного ящика») и всё больше приближаемся к действительному значению.

Для старта модели необходимо иметь 3 точки, чтобы понимать, от чего отталкиваться в ходе решения. В общем случае эти точки задаются случайно, как говорилось в главе 1, однако в нашем случае мы знаем результаты обучения на подобных задачах, а также меру подобия задач и можем расставить начальные точки не случайно.

2.3.2. Расстановка начальных конфигураций

Так как для старта модели необходимо 3 точки, имеет смысл рассмотреть 3 ближайшие задачи и получить оптимальный набор гиперпараметров от каждой. Сам такой подход уже может дать преимущество перед случайной расстановкой, однако этого не достаточно для получения инновационного результата. Кроме того мы обладаем полной информацией об обучении подобных задач, что также можно использовать.

Для начала давайте определим понятие *достоверности (reliability)*, которое будет показывать, на сколько можно «доверять» апостериорным знаниям той или иной задачи. Посчитанное в части 2.2 значение расстояния было бы использовать некорректно как минимум по причине того, что оно не нормализовано для модели оптимизации и значения могут значительно изменяться с добавлением новых задач и пересчётом метрики. Поэтому введём следующее определение достоверности:

$$R^t = (\alpha^t - K(d_i) + (1 - \alpha^t)K(\zeta_i)) \quad (7)$$

где t – номер итерации оптимизатора, α – мера «забывания» апостериорной информации (от 0 до 1), d_i – расстояние до задачи i , ζ_i – глобальная мера (имеется в виду ζ_{global} для задачи i , индекс *global* опущен для упрощения записи), на сколько отличается конфигурация задачи i от решаемой, $K(d_i)$, $K(\zeta_i)$ – функции ядра. Для d_i функция ядра – это обыкновенное евклидово расстояние, описанное в части 2.2, а для ζ_i – разница между значениями гиперпараметров для задачи i и решаемой.

Так как диапазоны значений для каждого гиперпараметра могут существенно отличаться, то нас интересует не только глобальное значение параметра ζ , которое и определяет подобие конфигураций, но и полная его характеристика для каждой задачи (то есть на сколько отличается каждое из значений гиперпараметров для пары задач). Формально у нас для каждой задачи (от-

носителем решаемой) будет считаться 2 характеристики: глобальное число и локальный массив. Связаны эти значения следующим образом:

$$\zeta_{local} = \{\Delta x_1, \Delta x_2 \dots \Delta x_N\}, \zeta_{global} = \text{avg}_i(\Delta x_i) \quad (8)$$

где avg – среднее значение.

Теперь рассмотрим описанный процесс на примере: предположим, что у нас есть решаемая задача, обозначим её *current* и мы подобрали к ней 2 подобные задачи – d_1 и d_2 . На рисунке 3 представлено состояние оптимизатора по

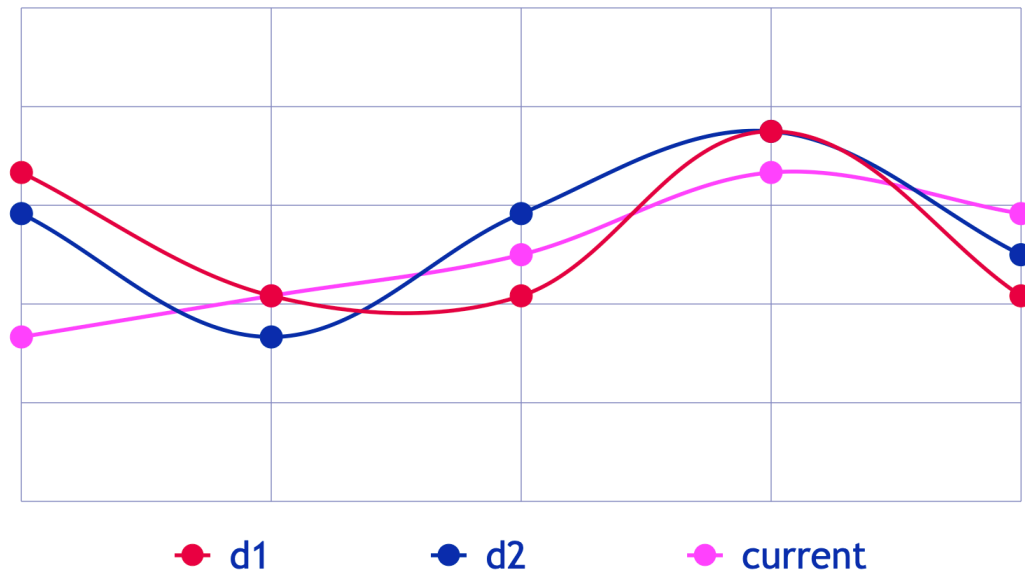


Рисунок 3 – Функция, определяющая гиперпараметры для решаемой задачи и двух подобных.

прохождении 5 итераций (на графике – 5 точек, определяющих конфигурации гиперпараметров, наш пример предполагает модель с одним гиперпараметром для упрощения визуализации). По горизонтали располагается значение гиперпараметра. Кривой представлена аппроксимация апостериорной функции для каждой из задач. Глобально ζ определяет расстояние по вертикали между точками итерации для каждой пары задач, то есть разницу значений функций.

Локальное значение ζ будет проще рассмотреть на примере с большим количеством гиперпараметров. Предположим, мы взяли конкретную итерацию для модели с 7 гиперпараметрами (обозначим НР и каждому гиперпараметру присвоим свой цвет для визуализации) и посчитали разницу значений каждого, на рисунке 4 представлены модули разности значений каждого ги-

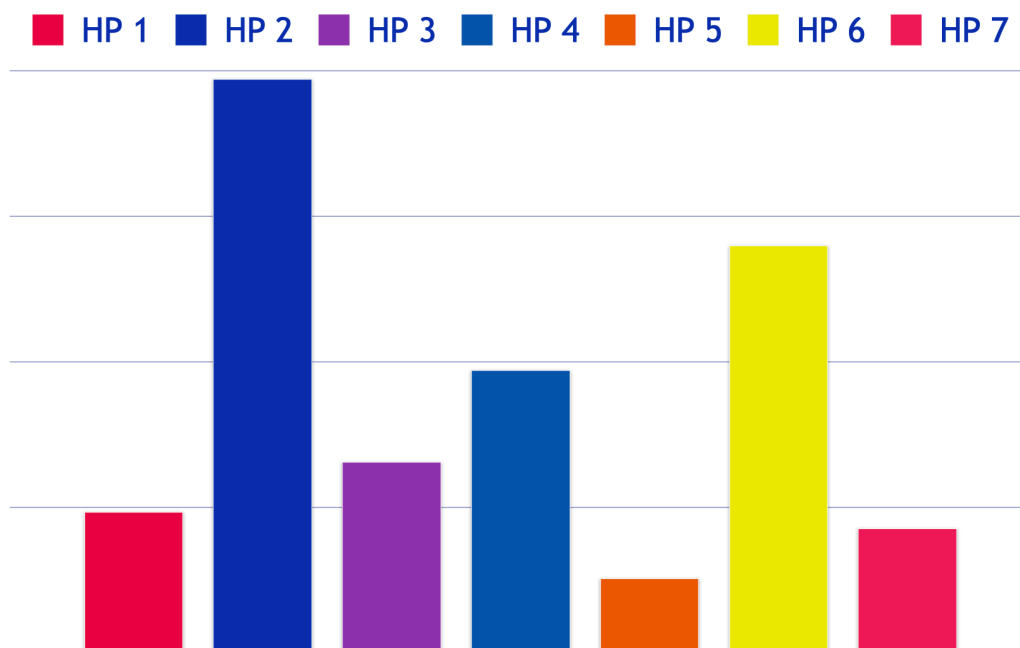


Рисунок 4 – Модуль разницы значений гиперпараметров для решаемой задачи и подобной.

перпараметра для пары задач: решаемой и подобной. Данная диаграмма наглядно представляет значения ζ_{local} , из которого по формуле 8 можно получить глобальное значение – ζ_{global} и с его помощью посчитать достоверность 7 подобной задачи, относительно решаемой.

Таким образом, на каждой итерации оптимизатора мы будем получать новое значение достоверности для решаемой задачи и подобной, однако теперь встаёт вопрос: как его применить к решению. Как очевидно из формулы 7, на первой итерации поправка по ζ уйдёт (что логично, так как на первой итерации мы не можем сравнить конфигурацию ренённой задачи и решаемой, так как для решаемой ещё не посчитано ни одно значение целевой функции) и останется только ядро по расстоянию до подобной задачи. Как сописывалось ранее, первые 3 итерации для решаемой задачи грубо посчитаны на значениях лучших конфигураций трёх подобных задач. На каждой следующей итерации уже начинается непосредственно сама оптимизация с использованием функции выгоды. Формально, на каждой итерации мы получаем число ζ_{global} и массив ζ_{local} , из которых считается достоверность и эта достоверность отправляется в качестве поправки для значения целевой функции. Более подробно: в глобальном цикле Байесовской оптимизации выбирается самая «интересная» на текущий момент точка (конфигурация гиперпараметров) и считается зна-

чение целевой функции для этой точки, после чего модель (в нашем случае гауссовский процесс) перетренировывается на новых данных и цикл продолжается. Сам гауссовский процесс тренируется как раз в момент выбора следующего кандидата (следующей наиболее интересной оптимизатору точки), в качестве кандидата возвращается набор координат максимума функции выгоды. Непосредственно поправка, посчитанная относительно подобной задачи, добавляется в момент обучения модели, а именно возводится в квадрат и добавляется по диагонали ковариационной матрицы для гауссовского процесса. Таким образом, значение «неопределённости» в конкретных точках будет изменяться пропорционально значению достоверности γ подобной задачи.

Выводы по главе 2

В данной главе было описано предложенное решение по расширению Байесовской оптимизации. Суть решения заключается в том, что для каждой задачи находятся подобные, между парами рашемая-подобная рассчитывается понятие достоверности. В качестве метрики для определения подобия задач предложено Евклидово расстояние, рассчитанное на нормализованных мета-признаках по всем датасетам. В качестве меры достоверности предложена собственная мера, которая основывается на ядерной функции от расстояния до датасета и разницы по каждому гиперпараметру в конфигурации. В качестве расширения классической Байесовской оптимизации предложено добавлять значение достоверности по диагонали матрицы гауссовского процесса в качестве меры неопределённости в рассматриваемой точке.

ГЛАВА 3. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

3.1. Подробности реализации

Описанный алгоритм реализован на языке `python3` с использованием сторонних пакетов: `numpy`, `pandas`, `george`, за основу для расширенной Байесовской оптимизации взята реализация из пакета `robo` (не находится в списке общедоступных пакетов, ставится отдельно), графики нарисованы с помощью пакета `matplotlib`.

Расмеченные данные в количестве 76 датасетов взяты с открытого ресурса OpenML [18]. Датасеты выбраны исходя из соображений максимального разнообразия, так как необходимо подтвердить эффективность решения на обширном диапазоне задач. В качестве разнообразия имеется в виду различное количество классов (3-1000), количество признаков (3-857), количество самих объектов (61-5000). Полные характеристики для всех выбранных датасетах представлены в приложении А.

3.2. Статистические результаты, дисперсия

Так как и алгоритм оптимизации, и сама модель частично основываются на случайных значениях, то будет некорректно сравнивать ответы по одному запуску. Поэтому полный цикл из 100 итераций Байесовской оптимизации запускался на каждом датасете по 10 раз. Сравнивать мы будем непосредственно средние значения по 10 запускам.

Для того, чтобы оценить статистические результаты, используем критерий Вилкоксона[17]: на основе нулевой гипотезы критерий считает попарную значимость результатов, то есть действительно ли один подход лучше другого, либо различие результатов в данном случае незначительно. В нашем случае будет присутствовать 2 выборки: первая – статистические результаты по 10 запускам классической Байесовской оптимизации, вторая – статистические результаты по 10 запускам расширенной Байесовской оптимизации. между этими парными выборками и будет считаться критерий Вилкоксона или, как его называют для независимых выборок, критерий Манна-Уитни[15].

Если оценить результаты, то можно заметить, что для некоторых датасетов разница между худшим и лучшим значениями в ходе оптимизации (имеется в виду значение целевой функции то есть $1 - F-score$) колеблется в пределах нескольких десятых, для других же – в пределах тысячных и то меньше, поэтому можно заключить, что для каждого датасета при оценке результатов будет

учитываться разное количество значимых цифр. Поэтому дисперсия также будет посчитана по 10 запускам, что позволит выяснить количество значащих цифр для сравнения результатов.

3.3. Результаты классической Байесовской оптимизации

Чтобы оценить эффективность предложенного решения, необходимо иметь результаты обучения с использованием существующего решения, поэтому для начала был запущен алгоритм классической Байесовской оптимизации на всех датасетах 10 раз. В части 2.3 описаны результаты, которые мы получаем после завершения алгоритма. Очевидно, что для оценки результата мы будем использовать *incumbent values*, то есть улучшение значения целевой функции на каждой итерации.

В качестве примера рассмотрим результат одного конкретного запуска на одном датасете. На рисунке 5 представлен непосредственно пример результа-

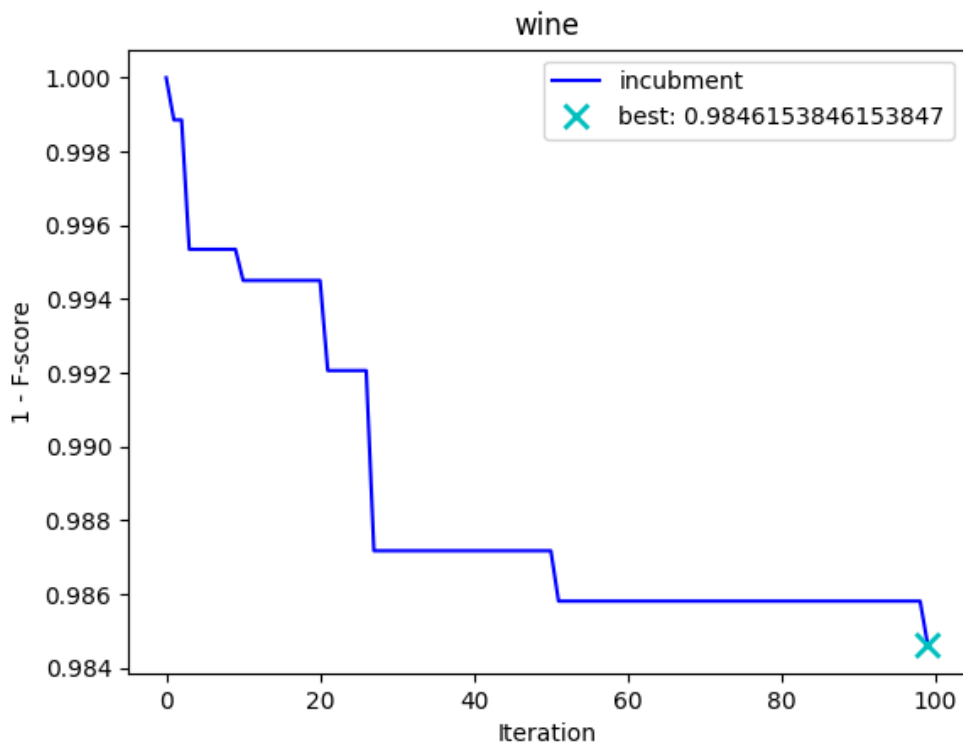


Рисунок 5 – Результат работы классической Байесовской оптимизации на наборе данных *wine*.

та классической Байесовской оптимизации после 100 итерации. По вертикали располагаются лучшие на данный момент значения целевой функции. Вспомним, что в основе нашей целевой функции лежит F -score, а так как в Байесовской оптимизации идёт процесс минимизации, то мы рассматриваем обратное

значение, то есть $(1 - F\text{-score})$, поэтому значение на графике убывает. Также вспомним, что *incumbent values* описывает **лучшее на данный момент времени** (на данной итерации) значение целевой функции, то есть, на итерации №40 нельзя утверждать, что значение целевой функции совпадает со значением на итерации №39, наоборот, вероятнее всего значение получилось хуже и просто не включилось в массив ответов (но учлось при дальнейшей оптимизации, так как каждая итерация добавляет определённости апостериорной информации). Лучшее значение целевой функции обозначено на графике как *best*, после этой отметки и до последней итерации значение не улучшалось.

На том же примере можно заметить, что разница между худшим и лучшим значениями заключена в пределах 0,02, когда, например, на рисунке 6 это

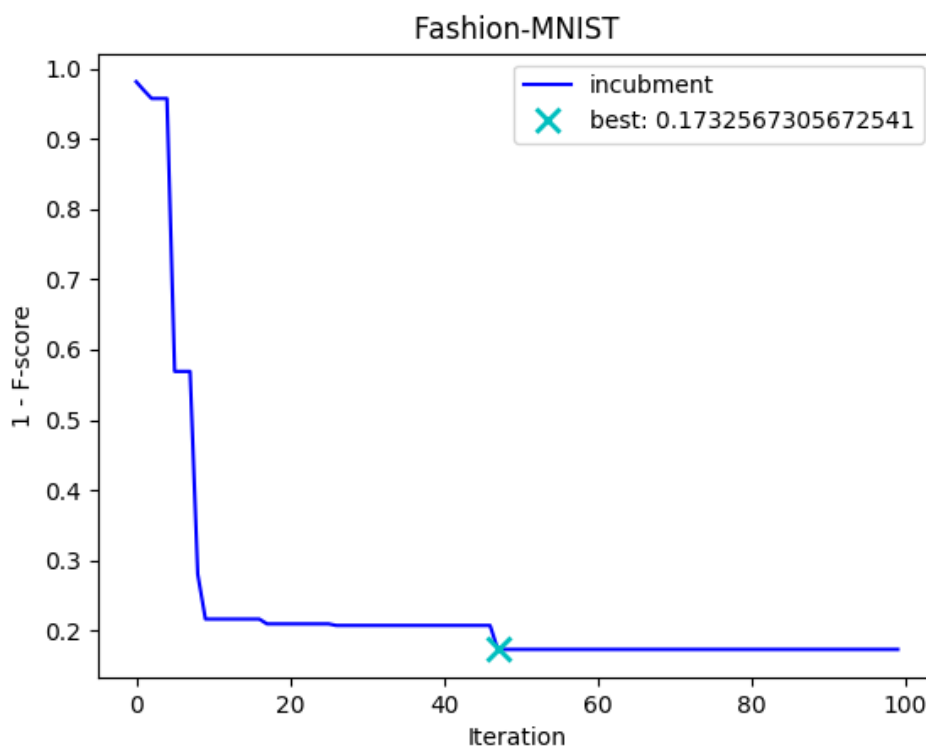


Рисунок 6 – Результат работы классической Байесовской оптимизации на наборе данных *Fashion-MNIST*.

значение заключено в диапазоне 0,8. Это наглядно показывает, что для сравнения результатов нельзя использовать фиксированное количество значащих цифр, поэтому для каждого датасета это количество нужно определять индивидуально на основе статистических результатов.

3.4. Сравнение результатов классической и расширенной Байесовской оптимизации

После того, как получены результаты классической Байесовской оптимизации, можно сравнивать их с результатами нашего модифицированного алгоритма. Но возникает вопрос: что и как сравнивать? Можно предположить, что после обучения нам интересно только лучшее полученное значение целевой функции, однако если этот оптимальный результат получен на более ранней итерации, то это тоже считается успехом. Для того, чтобы определить критерии сравнения результатов, выделим все возможные случаи по завершении последней итерации:

- а) получено лучшее оптимальное значение целевой функции на более ранней итерации – абсолютная победа предложенного решения
- б) получено лучшее оптимальное значение целевой функции на той же итерации – победа предложенного решения по абсолютному значению, ничья по номеру итерации
- в) получено лучшее оптимальное значение целевой функции, но на более поздней итерации – победа предложенного решения по абсолютному значению, проигрыш по номеру итерации
- г) оптимальные значения одинаковы (в пределах значащих цифр), но в предложенном решении это значение получено на более ранней итерации – ничья по абсолютному значению, победа предложенного решения по номеру итерации
- д) оптимальное значение хуже, но получено на более ранней итерации – проигрыш по абсолютному значению, победа по номеру итерации

Для случаев а, б, г победа предложенного решения очевидна, однако в случае в несправедливо утверждать, что мы проиграли по номеру итерации, так как возможны (и весьма вероятны) случаи, когда на момент нахождения оптимального значения классической Байесовской оптимизацией, абсолютное значение в нашем решении уже было лучше, это говорит о том, что оптимальное значение классической оптимизации нашим решением было получено раньше. В случае д же вовсе нельзя утверждать о победе нашего решения, так как всё-таки в первую очередь нас интересует абсолютное значение.

Кроме того, так как и сама оптимизация, и модель основываются на случайных процессах, кроме вышеописанного, нас также интересует статисти-

ческая значимость результатов для каждого датасета. Например, если худшее значение целевой функции было 0,8, а к концу оптимизации стало 0,101 для классического решения и 0,105 для нашего, то улучшение не так существенно, как было бы в случае оптимальных результатов 0,3 и 0,1 соответственно. Поэтому используем формальную оценку качества оптимизации.

3.4.1. Оценка качества оптимизации

Как было отмечено ранее, в первую очередь нас интересует само оптимальное значение целевой функции, поэтому из вышеперечисленных случаев можно выделить 2 группы: значения целевых функций различны и значения функций одинаковы.

Значения целевых функций различны. В этом случае применим критерий Вилкоксона, описанный в части 3.2, к абсолютному значению целевой функции по всем запускам и получим значимость результата.

Значения целевых функций одинаковы. В этом случае применим критерий Вилкоксона к номеру итерации по всем запускам и получим значимость результата.

Исходя из полученной значимости допустимо рассуждать о победе или поражении того или иного алгоритма оптимизации. Полученные результаты представлены в приложении Е. Теперь можно переходить непосредственно к оценке результатов.

3.4.2. Оценка результатов

Для каждого датасета были посчитаны средние для оптимального значения и номера итерации. Также, для каждого датасета было рассчитано количество значащих цифр (определяется средним значением дисперсии по всем запускам для датасета).

Полный список результатов классической Байесовской оптимизации представлен в приложении Б, расширенной Байесовской оптимизации – в приложении В. Посчитанная дисперсия для классической и расширенной оптимизации располагается в приложениях Г и Д соответственно.

Теперь сравним результаты классической и расширенной Байесовской оптимизации для датасета из части 3.3. На рисунке 7 розовой линией представ-

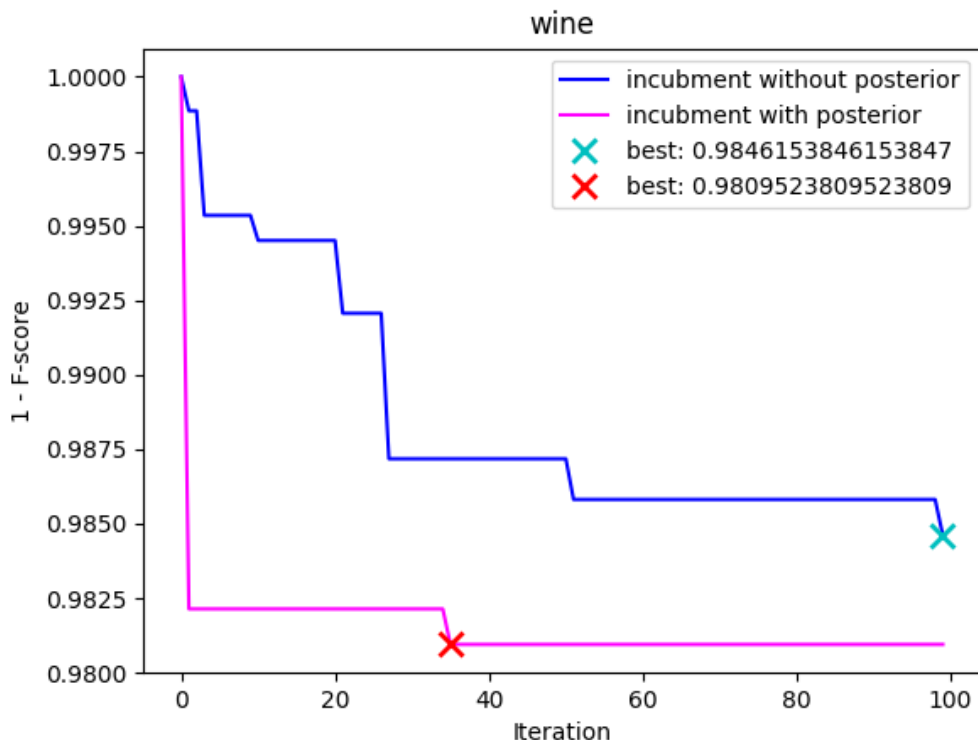


Рисунок 7 – Результат работы расширенной Байесовской оптимизации на наборе данных *wine*.

лен результат работы расширенной Байесовской оптимизации, то есть предложенного алгоритма. Можно отметить, что оптимальный результат получен значительно раньше и по абсолютному значению также побеждает классическую Байесовскую оптимизацию. На наборе данных *car* (рисунок 8) диапазон значений целевой функции гораздо шире, однако результат также побеждает как по абсолютному значению, так и по номеру итерации его достижения. На данном примере можно наглядно показать влияние начальной расстановки и последующей корректировки гиперпараметров. Мы видим, что начальное значение целевой функции для нашего алгоритма располагается значительно ниже результата классического алгоритма, это говорит о влиянии начальной расстановки из подобных задач. Дальнейшее улучшение уже является результатом корректировки на основе достоверности.

Однако не на всех датасетах получены подобные результаты, например, для датасета *desharnais* (рисунок 9) начальная расстановка сыграла отрицательную роль, а непосредственно корректировка по достоверности позволила получить оптимальный результат.

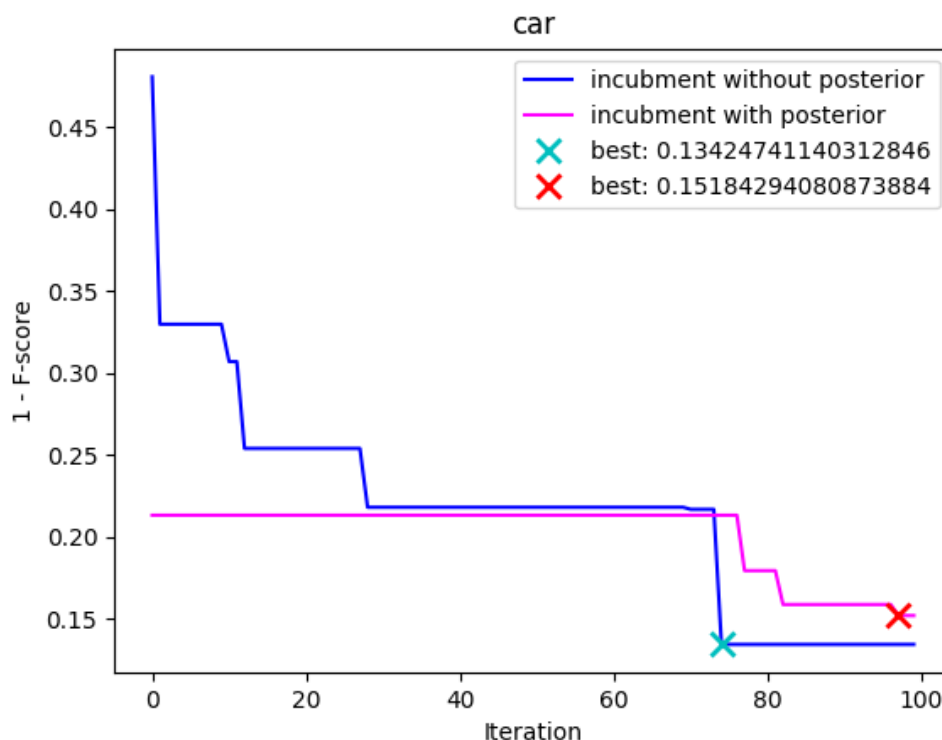


Рисунок 8 – Результат работы расширенной Байесовской оптимизации на наборе данных *car*.

Кроме того, плохие результаты могут быть вызваны тем, что подобная задача определена неверно, либо сам набор данных плохо подходит для оптимизации. Если посмотреть на приложение Е, можно заметить, что для датасетов *iris* и *synthetic-control* значимость результатов по значению не была посчитана вовсе, это вызвано тем, что и классическая Байесовская оптимизация, и наша достигают одного и того же результата на каждом запуске, поэтому критерий Вилкоксоно не может быть посчитан. В этом случае мы смотрим на тот алгоритм, который нашёл этот результат на более ранней итерации, как говорилось в параграфе 3.4.1.

Так как все наборы данных различны и имеют различные референсные значения в результатах, общий результат оценить количественно не получится, поэтому исходя из описанной в текущей главе технологии для каждого датасета определим, улучшил ли он результат классической оптимизации и потом уже посчитаем в скольких случаях предложенный алгоритм «победил» и дадим количественную оценку уже этому результату.

Абсолютными победами назовём те задачи, в которых предложенный алгоритм показал статистически значимые (часть 3.2) лучшие результаты. *Услов-*

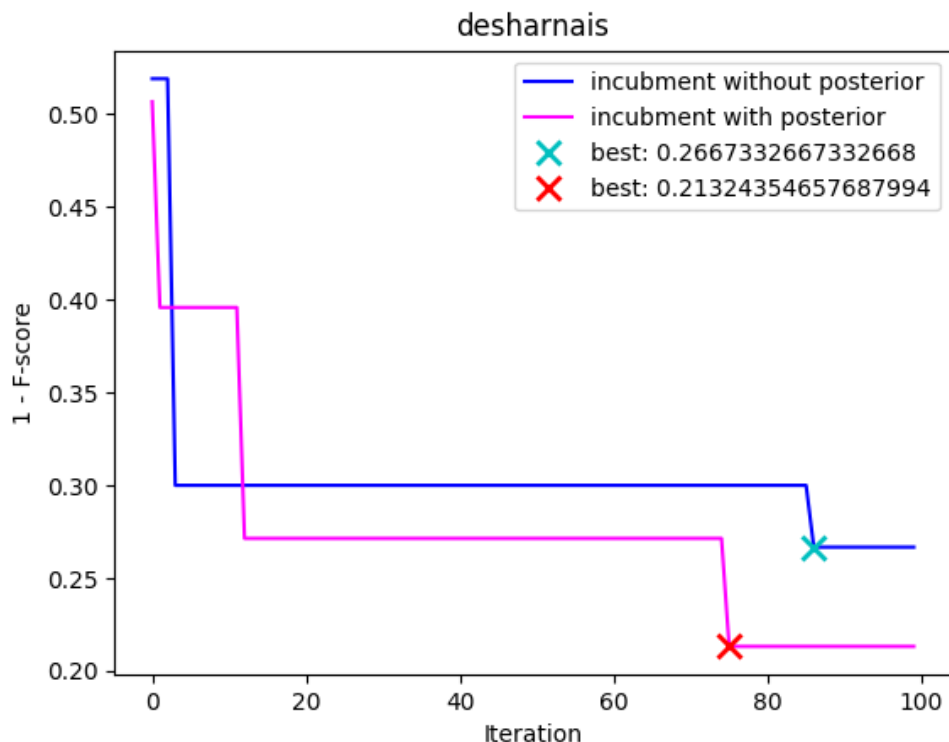


Рисунок 9 – Результат работы расширенной Байесовской оптимизации на наборе данных *desharnais*.

ными победами назовём те случаи, когда значение на обеих оптимизациях получилось одинаковое, либо оказалось статистически незначимым, однако значимым оказался номер итерации, на котором это значение было достигнуто. *Поражениями* – все оставшиеся случаи.

Общее число задач: 76

Число абсолютных побед: 36

Число условных побед: 7

Число поражений: 33

Итого, с учётом всех критериев оценки результата получено 43 победы и 33 поражения, что говорит о том, что в 57% задач предложенный алгоритм даёт значительно лучший результат либо значительно ускоряет оптимизацию. С учётом того, что на данный момент Байесовская оптимизация считается самым эффективным алгоритмом для задачи оптимизации гиперпараметров, то полученные результаты можно считать значительными и наилучшими на сегодняшний день.

Выводы по главе 3

За счёт того, что и сама Байесовская оптимизация, и модель, на основе которой она построена, от части реализованы на случайных процессах, для оценки качества оптимизации и результатов работы алгоритма не достаточно рассматривать один полученный результат. Поэтому для оценки результатов было проведено по 10 опытов, отдельно посчитана дисперсия для каждого из датасетов (так как количество значащих цифр для каждого датасета разное), кроме этого все статистические результаты оценены с точки зрения значимости с использованием критерия Вилкоксона и разработана собственная система оценки улучшения предложенного алгоритма. Полученные результаты можно интерпретировать исключительно качественно, так как для каждой отдельной задачи «улучшение» заключено в различных диапазонах. Исходя из полученных результатов с учётом всех статистических оценок и критериев можно заключить, что предложенный алгоритм однозначно работает эффективнее лучшего существующего на сегодняшний день алгоритма.

ЗАКЛЮЧЕНИЕ

В данной работе рассмотрены существующие решения задачи оптимизации гиперпараметров, такие как поиск по решётке, случайный поиск, оптимизация на основе градиентов и другие, взвешены плюсы и минусы каждого из них. Подробно рассмотрены параметры оптимизаторов, такие как модель (Gaussian Process, RandomForest, DNGO и т. д.), максимизатор (RandomSampling, SciPyOptimizer, DifferentialEvolution), функция выгоды (EI, LogEI, PI, LCB) и целевая функция. Из рассмотренных решений выбрано лучшее на сегодняшний день (Байесовская оптимизация), на основе которого строился предложенный в работе алгоритм. В качестве модели был взят гауссовский процесс, в качестве оптимизатора – RandomSampling, в качестве функции выгоды – LogEI, а за целевую функцию – F -score для классификатора многослойного парцептрона Румельхарта. В классификаторе выделены гиперпараметры, которые варьировались в ходе решения. Для проведения достоверных экспериментов из авторитетного источника было отобрано 76 максимально различных (по количеству признаков, классов и т. д.) наборов данных, подходящих для задачи классификации. Предложена метрика для определения подобия задач по сгенерированным метапризнакам. С учётом специфики модели и оптимизатора, проведено необходимое для корректной статистической оценки результатов количество опытов, посчитаны все необходимые меры и критерии, что позволило качественно оценить результаты. Для случаев, когда предложенный алгоритм по окончании заданного количества итераций оптимизации, оценены абсолютные значения и выделены случаи, когда побеждает предложенный алгоритм, а когда существующий. Сопоставлены все результаты со всеми проведёнными опытами, посчитана дисперсия и значимость для каждого отдельного датасета. В итоге получен лучший на сегодняшний день результат для задачи оптимизации гиперпараметров, что говорит о том, что данная работа является актуальной и ценной в исследуемой области и может быть использована для дальнейших исследований.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 aiDM '20: Proceedings of the Third International Workshop on Exploiting Artificial Intelligence Techniques for Data Management. — Portland, Oregon : Association for Computing Machinery, 2020. — ISBN 9781450380294.
- 2 Algorithms for Hyper-Parameter Optimization / J. S. Bergstra [et al.] // Advances in Neural Information Processing Systems 24 / ed. by J. Shawe-Taylor [et al.]. — Curran Associates, Inc., 2011. — P. 2546–2554. — URL: <http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf>.
- 3 Any-Precision Deep Neural Networks / H. Yu [et al.]. — 2019. — arXiv: 1911.07346 [cs.LG].
- 4 *Claesen M., Moor B. D.* Hyperparameter Search in Machine Learning. — 2015. — arXiv: 1502.02127 [cs.LG].
- 5 Efficient and Robust Automated Machine Learning / M. Feurer [et al.] // Advances in Neural Information Processing Systems 28 / ed. by C. Cortes [et al.]. — Curran Associates, Inc., 2015. — P. 2962–2970. — URL: <http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf>.
- 6 *Falkner S., Klein A., Hutter F.* BOHB: Robust and Efficient Hyperparameter Optimization at Scale // Proceedings of the 35th International Conference on Machine Learning (ICML 2018). — 07/2018. — P. 1436–1445.
- 7 *Feurer M., Hutter F.* Hyperparameter Optimization // AutoML: Methods, Sytems, Challenges / ed. by F. Hutter, L. Kotthoff, J. Vanschoren. — Springer, 05/2019. — Chap. 1. P. 3–33.
- 8 *Hastie T., Tibshirani R., Friedman J.* The elements of statistical learning: data mining, inference and prediction. — 2nd ed. — Springer, 2009. — URL: <http://www-stat.stanford.edu/~tibs/ElemStatLearn/>.
- 9 *Hutter F., Hoos H. H., Leyton-Brown K.* Sequential Model-Based Optimization for General Algorithm Configuration (extended version) : tech. rep. / University of British Columbia, Department of Computer Science. — 2010. — TR-2010–10. — Available online: <http://www.cs.ubc.ca/~hutter/papers/10-TR-SMAC.pdf>.

- 10 *Jomaa H. S., Schmidt-Thieme L., Grabocka J.* Dataset2Vec: Learning Dataset Meta-Features. — 2019. — arXiv: 1905.11063 [cs.LG].
- 11 *Kiss O.* Bayesian Optimization for machine learning algorithms in the context of Higgs searches at the CMS experiment. — 2019. — arXiv: 1911.02501 [physics.data-an].
- 12 *Koh P. W., Liang P.* Understanding Black-box Predictions via Influence Functions. — 2017. — arXiv: 1703.04730 [stat.ML].
- 13 *Lindauer M., Hutter F.* Warmstarting of Model-based Algorithm Configuration. — 2017. — arXiv: 1709.04636 [cs.AI].
- 14 *Malsburg C. von der.* Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms // Brain Theory. — 1986. — Jan. — P. 245–248. — DOI: 10.1007/978-3-642-70911-1_20.
- 15 *Mann H. B., Whitney D. R.* On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other // Ann. Math. Statist. — 1947. — Mar. — Vol. 18, no. 1. — P. 50–60. — DOI: 10.1214/aoms/1177730491. — URL: <https://doi.org/10.1214/aoms/1177730491>.
- 16 NBDT: Neural-Backed Decision Trees / A. Wan [et al.]. — 2020. — arXiv: 2004.00221 [cs.CV].
- 17 *Neuhäuser M.* Wilcoxon–Mann–Whitney Test // International Encyclopedia of Statistical Science / ed. by M. Lovric. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. — P. 1656–1658. — ISBN 978-3-642-04898-2. — DOI: 10.1007/978-3-642-04898-2_615. — URL: https://doi.org/10.1007/978-3-642-04898-2_615.
- 18 OpenML: Networked Science in Machine Learning / J. Vanschoren [et al.] // SIGKDD Explorations. — New York, NY, USA, 2013. — Vol. 15, no. 2. — P. 49–60. — DOI: 10.1145/2641190.2641198. — URL: <http://doi.acm.org/10.1145/2641190.2641198>.
- 19 *Powers D. M. W.* What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes. — 2015. — arXiv: 1503.06410 [cs.IR].
- 20 RoBO: A Flexible and Robust Bayesian Optimization Framework in Python / A. Klein [et al.] // NIPS 2017 Bayesian Optimization Workshop. — 12/2017.

- 21 *Yu T., Zhu H.* Hyper-Parameter Optimization: A Review of Algorithms and Applications. — 2020. — arXiv: 2003.05689 [cs.LG].

ПРИЛОЖЕНИЕ А. ИНФОРМАЦИЯ ОБ ОТОБРАННЫХ ДАТАСЕТАХ

Имя датасета	Число объектов	Число признаков	Число классов	ID на OpenML
abalone	4177	9	3	1557
artificial-characters	10218	8	10	1459
balance-scale	625	5	3	11
breast-tissue	106	10	6	1465
car	1728	7	4	40975
cardiotocography	2126	36	10	1466
cmc	1473	10	3	23
cnae-9	1080	857	9	1468
collins	1000	24	30	40971
coverttype	581012	55	7	150
desharnais	81	13	3	1117
diggle-table-a2	310	9	9	694
ecoli	336	8	8	39
energy-efficiency	768	10	37	1472
eye-movements	10936	28	3	1044
fabert	8237	801	7	41164
fars	100968	30	8	40672
Fashion-MNIST	70000	785	10	40996
gas-drift	13910	129	6	1476
gas-drift-different-concentrations	13910	130	6	1477
gina-prior2	3468	785	10	1041
glass	214	10	6	41
har	10299	562	6	1478
hayes-roth	160	5	3	329
heart-long-beach	200	14	5	1512
heart-switzerland	123	13	5	1513
helena	65196	28	100	41169
Indian-pines	9144	221	8	41972
iris	150	5	3	61

jannis	83733	55	4	41168
JapaneseVowels	9961	15	9	375
jungle-chess-2pcs- endgame-panther- elephant	4704	47	3	41000
jungle-chess-2pcs-raw- endgame-complete	44819	7	3	41027
leaf	340	16	30	1482
LED-display-domain- 7digit	500	8	10	40496
mfeat-factors	2000	217	10	12
mfeat-fourier	2000	77	10	14
mfeat-karhunen	2000	65	10	16
mfeat-morphological	2000	7	10	18
mfeat-pixel	2000	241	10	40979
microaggregation2	20000	21	5	41671
nursery	12960	9	5	26
page-blocks	5473	11	5	30
pokerhand	829201	11	10	155
PopularKids	478	11	3	1100
prnn-fglass	214	10	6	952
prnn-viruses	61	19	4	480
rmftsa-sleepdata	1024	3	4	679
robot-failures-lp1	88	91	4	1516
robot-failures-lp5	164	91	5	1520
satimage	6430	37	6	182
seeds	210	8	3	1499
segment	2310	20	7	40984
seismic-bumps	210	8	3	1500
semeion	1593	257	10	1501
shuttle	58000	10	7	40685
spectrometer	531	103	48	313
steel-plates-fault	1941	28	7	40982

synthetic-control	600	62	6	377
tae	151	6	3	48
tamilnadu-electricity	45781	4	20	40985
teachingAssistant	151	7	3	1115
thyroid-allbp	2800	27	5	40474
thyroid-allhyper	2800	27	5	40475
user-knowledge	403	6	5	1508
vehicle	846	19	4	54
vertebra-column	310	7	3	1523
volcanoes-a1	3252	4	5	1527
volcanoes-a3	1521	4	5	1529
volcanoes-a4	1515	4	5	1530
volcanoes-d1	8753	4	5	1538
wall-robot-navigation	5456	5	4	1526
waveform-5000	5000	41	3	60
wine	178	14	3	187
wine-quality-white	4898	12	7	40498
zoo	101	17	7	62

ПРИЛОЖЕНИЕ Б. РЕЗУЛЬТАТЫ КЛАССИЧЕСКОЙ БАЙЕСОВСКОЙ ОПТИМИЗАЦИИ

Имя датасета	Лучшее значение (1 - F-score)	Номер итерации
abalone	0,615400258824198	95
artificial-characters	0,3559097202747393	97
balance-scale	0,053141906070482484	96
breast-tissue	0,6818280793280793	93
car	0,16487930442466345	85
cardiotocography	0,12124778732718564	96
cmc	0,4518832129599252	92
cnae-9	0,04608401806475787	98
collins	0,8756704066611304	97
covertime	0,44623970623986625	89
desharnais	0,094592058505102	86
diggle-table-a2	0,9795551003350139	97
ecoli	0,13981443473353988	95
energy-efficiency	0,8229788731752743	92
eye-movements	0,5454590857253053	94
fabert	0,8563974922818955	13
fars	0,4919820642867753	87
Fashion-MNIST	0,1841999445039933	70
gas-drift	0,01229587283888477	99
gas-drift-different-concentrations	0,014135671941996442	92
gina-prior2	0,07109914430206611	97
glass	0,5438046693293679	85
har	0,01990235753262858	96
hayes-roth	0,1456313707054801	97
heart-long-beach	0,6881087163579751	91
heart-switzerland	0,6538168757168756	98
helena	0,9781497717820775	98
Indian-pines	0,4921257947810759	96
iris	0,0	5

jannis	0,9999873749141746	87
JapaneseVowels	0,5106516915536152	96
jungle-chess-2pcs- endgame-panther- elephant	0,0004813765790142255	88
jungle-chess-2pcs-raw- endgame-complete	0,2831540971429782	99
leaf	0,40293986047115576	95
LED-display-domain- 7digit	0,22008194071239123	79
mfeat-factors	0,06790745344168528	94
mfeat-fourier	0,16586653632657422	97
mfeat-karhunen	0,02322287779210741	99
mfeat-morphological	0,5584251769846496	98
mfeat-pixel	0,558084964753526	95
microaggregation2	0,5716839504480571	88
nursery	0,06674407462192931	81
page-blocks	0,13516414692083642	84
pokerhand	0,30047170829842273	93
PopularKids	0,0755458190128694	85
prnn-fglass	0,39034465586128364	84
prnn-viruses	0,037333333333333333	93
rmftsa-sleepdata	0,9080866235490108	95
robot-failures-lp1	0,08618397125750067	82
robot-failures-lp5	0,2881688745907933	91
satimage	0,11962597103528783	86
seeds	0,057802300267854614	96
segment	0,018440943963987422	97
seismic-bumps	0,02381313808909159	66
semeion	0,06514854990131118	87
shuttle	0,02125073667637095	98
spectrometer	0,9661251360456881	92
steel-plates-fault	0,7624399867662701	87

synthetic-control	0,0	14
tae	0,45732035806702226	86
tamilnadu-electricity	0,04787838195460763	98
teachingAssistant	0,4417724517376909	98
thyroid-allbp	0,5500200435481564	99
thyroid-allhyper	0,5189136491165705	98
user-knowledge	0,12889846264159402	95
vehicle	0,34810891893008283	96
vertebra-column	0,1430715573316498	98
volcanoes-a1	0,7917649985130863	98
volcanoes-a3	0,7748012053245856	97
volcanoes-a4	0,7954900451834821	99
volcanoes-d1	0,7979166787265425	96
wall-robot-navigation	0,02080579413576409	83
waveform-5000	0,13219632609315485	89
wine	0,974506165505316	99
wine-quality-white	0,7684229624815606	97
zoo	0,0	48

ПРИЛОЖЕНИЕ В. РЕЗУЛЬТАТЫ РАСШИРЕННОЙ БАЙЕСОВСКОЙ ОПТИМИЗАЦИИ

Имя датасета	Лучшее значение (1 - F-score)	Номер итерации
abalone	0,6207351879252487	99
artificial-characters	0,3598330107797426	77
balance-scale	0,02827846813241224	91
breast-tissue	0,6086918636918637	96
car	0,16243913101496316	97
cardiotocography	0,10327689139570051	85
cmc	0,44954138421509154	95
cnae-9	0,04282385630261483	99
collins	0,8813106037082383	94
covertime	0,4246853261465756	87
desharnais	0,20114454654309727	94
diggle-table-a2	0,9795130169154224	91
ecoli	0,3163316393963809	83
energy-efficiency	0,805047244192569	99
eye-movements	0,5558344876982456	88
fabert	0,8337679638890446	85
fars	0,5536497492518739	95
Fashion-MNIST	0,17625908674980378	97
gas-drift	0,0161137549628551	95
gas-drift-different-concentrations	0,012025636809062944	91
gina-prior2	0,08084209641898632	97
glass	0,4081753351344609	88
har	0,018236948441473853	98
hayes-roth	0,12977528107456612	84
heart-long-beach	0,6379492902067947	91
heart-switzerland	0,6953038641795858	92
helena	0,9762584929723648	87
Indian-pines	0,5222900193258982	98
iris	0,0	21

jannis	0,9999949120015931	86
JapaneseVowels	0,6079964359459699	99
jungle-chess-2pcs- endgame-panther- elephant	0,0	25
jungle-chess-2pcs-raw- endgame-complete	0,27936388546360585	97
leaf	0,40962899106002554	92
LED-display-domain- 7digit	0,24815408565298167	97
mfeat-factors	0,06053373828048423	96
mfeat-fourier	0,1654558116809039	98
mfeat-karhunen	0,028159451822638636	87
mfeat-morphological	0,5575497230838617	90
mfeat-pixel	0,5597959202824287	83
microaggregation2	0,5816731240940771	97
nursery	0,061483237305924324	80
page-blocks	0,17457611484964325	92
pokerhand	0,24886846576788696	81
PopularKids	0,06012859557176129	84
prnn-fglass	0,3577843022764862	99
prnn-viruses	0,0	21
rmftsa-sleepdata	0,8970943626058208	92
robot-failures-lp1	0,013333333333333319	63
robot-failures-lp5	0,31991536528010245	93
satimage	0,09837567310278647	71
seeds	0,06205781791685011	86
segment	0,030969414696502405	75
seismic-bumps	0,0510471348244584	74
semeion	0,042417637353081475	89
shuttle	0,003643795721823917	82
spectrometer	0,9583669060125217	93
steel-plates-fault	0,7585113444665512	99

synthetic-control	0,0	17
tae	0,36410953612768193	74
tamilnadu-electricity	0,051307682059697005	94
teachingAssistant	0,38484487698620196	88
thyroid-allbp	0,5560058002623417	97
thyroid-allhyper	0,5121464197314757	85
user-knowledge	0,140969245986393	91
vehicle	0,3800742011398082	92
vertebra-column	0,18323648228726622	77
volcanoes-a1	0,7909946678290233	82
volcanoes-a3	0,7928259101135661	96
volcanoes-a4	0,7826323129510018	99
volcanoes-d1	0,7988570800009016	80
wall-robot-navigation	0,018615250235651516	99
waveform-5000	0,13523564954603046	77
wine	0,976808965652852	89
wine-quality-white	0,7801196892884702	96
zoo	0,03478991596638655	99

ПРИЛОЖЕНИЕ Г. ДИСПЕРСИИ ДЛЯ КЛАССИЧЕСКОЙ БАЙЕСОВСКОЙ ОПТИМИЗАЦИИ

Имя датасета	Дисперсия по значению	Дисперсия по итерации
abalone	0,015720846819078437	28
artificial-characters	0,0069060337298337045	24
balance-scale	0,005114884116420693	24
breast-tissue	0,016125763125763116	18
car	0,030408332830185746	17
cardiotocography	0,02640885628698576	25
cmc	0,007568518579586481	26
cnae-9	0,003070188049180589	30
collins	0,027089081985286957	20
covertime	0,007162058848558528	21
desharnais	0,04817805704762228	26
diggle-table-a2	0,0032018488882798236	28
ecoli	0,021669426780532353	23
energy-efficiency	0,03670861719278061	29
eye-movements	0,02401922375382166	25
fabert	0,005911299687452631	2
fars	0,019321463358881564	13
Fashion-MNIST	0,00587071584740193	14
gas-drift	0,0020071276066936485	19
gas-drift-different-concentrations	0,0023336208790602055	23
gina-prior2	0,005792962456909023	26
glass	0,028936677191790404	21
har	0,0014896182396576419	24
hayes-roth	0,02005082262817782	24
heart-long-beach	0,024115587014241536	22
heart-switzerland	0,04515276427276428	35
helena	0,0036678711129311424	23
Indian-pines	0,0636900851714468	29
iris	0,0	1

jannis	1,2081682913611669e-05	16
JapaneseVowels	0,12540506424962455	26
jungle-chess-2pcs-endgame-panther-elephant	0,00048137657901422556	16
jungle-chess-2pcs-raw-endgame-complete	0,006153685014120247	29
leaf	0,00943609937504456	21
LED-display-domain-7digit	0,006403947854831693	23
mfeat-factors	0,008077540540906486	22
mfeat-fourier	0,005324730668459199	20
mfeat-karhunen	0,002654057897895279	23
mfeat-morphological	0,05379203616351239	30
mfeat-pixel	0,01917464942703887	23
microaggregation2	0,00934766748612218	19
nursery	0,009098987845676124	14
page-blocks	0,011150238105822935	15
pokerhand	0,023596955018041266	35
PopularKids	0,021028839098959595	18
prnn-fglass	0,03957761348418705	22
prnn-viruses	0,052266666666666666	27
rmftsa-sleepdata	0,015660942186313653	21
robot-failures-lp1	0,04188813963813966	27
robot-failures-lp5	0,02165421196120073	17
satimage	0,0031805747792820636	15
seeds	0,01347319057703462	24
segment	0,003130755386970039	25
seismic-bumps	0,008563257889524025	14
semeion	0,004163271740454425	22
shuttle	0,007943147956755391	22
spectrometer	0,005239020338345446	17
steel-plates-fault	0,017680627639122027	17

synthetic-control	0,0	3
tae	0,03197577418378402	17
tamilnadu-electricity	0,013371839663861967	17
teachingAssistant	0,028940494412685758	21
thyroid-allbp	0,009411102326217525	24
thyroid-allhyper	0,006046971643730392	30
user-knowledge	0,007965783294974261	23
vehicle	0,03138708775089925	18
vertebra-column	0,01944678632731688	30
volcanoes-a1	0,008333037827420153	24
volcanoes-a3	0,018693202751244455	15
volcanoes-a4	0,008007463201082421	20
volcanoes-d1	0,004737113725397235	30
wall-robot-navigation	0,0033289841828731334	18
waveform-5000	0,0024168011655843814	22
wine	0,006320011106897616	30
wine-quality-white	0,009218358807825188	20
zoo	0,0	11

ПРИЛОЖЕНИЕ Д. ДИСПЕРСИИ ДЛЯ РАСШИРЕННОЙ БАЙЕСОВСКОЙ ОПТИМИЗАЦИИ

Имя датасета	Дисперсия по значению	Дисперсия по итерации
abalone	0,012015342021105746	28
artificial-characters	0,004729817955744275	18
balance-scale	0,006429783765000435	21
breast-tissue	0,02199204499204501	24
car	0,014929221650433066	31
cardiotocography	0,028383794221548485	23
cmc	0,004417772514976182	27
cnae-9	0,001441589367774565	26
collins	0,037408028482346665	28
covertime	0,014112679110933769	29
desharnais	0,029273864621690692	33
diggle-table-a2	0,004544472030494029	32
ecoli	0,01182609059584796	21
energy-efficiency	0,03704436821037077	16
eye-movements	0,015739176884715756	22
fabert	3,334644656183272e-05	26
fars	0,01244112886703258	19
Fashion-MNIST	0,0075349440228408395	15
gas-drift	0,0014861016597044995	23
gas-drift-different-concentrations	0,0014032523085634094	26
gina-prior2	0,0035111558611975236	12
glass	0,02443007440390601	21
har	0,0014086148771750074	19
hayes-roth	0,013496673946015452	20
heart-long-beach	0,031082616818267284	21
heart-switzerland	0,024516565064816344	28
helena	0,005398182107099159	19
Indian-pines	0,04477079176038691	22
iris	0,0	6

jannis	2,586466781673291e-06	20
JapaneseVowels	0,11508245242005985	22
jungle-chess-2pcs- endgame-panther- elephant	0,0	5
jungle-chess-2pcs-raw- endgame-complete	0,006704372969595018	28
leaf	0,012217299431092565	21
LED-display-domain- 7digit	0,0076077335519361185	28
mfeat-factors	0,013087972805583395	20
mfeat-fourier	0,003133613848196237	26
mfeat-karhunen	0,002323620417466854	21
mfeat-morphological	0,05132389528232474	23
mfeat-pixel	0,011484314557708219	18
microaggregation2	0,006282391130715914	19
nursery	0,008877012776463526	17
page-blocks	0,010501001288687261	14
pokerhand	0,0419998392492623	17
PopularKids	0,011763008174606626	29
prnn-fglass	0,011365516834072054	32
prnn-viruses	0,0	7
rmftsa-sleepdata	0,008403929067370187	33
robot-failures-lp1	0,018666666666666644	17
robot-failures-lp5	0,012367401837363411	23
satimage	0,001961837422270968	20
seeds	0,008847024295621963	19
segment	0,004360536196130642	20
seismic-bumps	0,009223224152059792	26
semeion	0,002210059494674646	27
shuttle	0,0011813326411704804	21
spectrometer	0,006768922275721778	25
steel-plates-fault	0,01625969076455449	23

synthetic-control	0,0	5
tae	0,01466021702152922	15
tamilnadu-electricity	0,021838084875770512	21
teachingAssistant	0,015642474884946623	28
thyroid-allbp	0,01331566218251613	33
thyroid-allhyper	0,01133815051524456	24
user-knowledge	0,004985438383448415	24
vehicle	0,04493996057863101	26
vertebra-column	0,012102850450172897	24
volcanoes-a1	0,010619895240654709	24
volcanoes-a3	0,00993917832288611	27
volcanoes-a4	0,013223040621362647	22
volcanoes-d1	0,005733037964544274	21
wall-robot-navigation	0,0017523809556713934	28
waveform-5000	0,0030101466884798088	25
wine	0,0035429091590036045	21
wine-quality-white	0,008802420951850364	23
zoo	0,05566386554621848	30

ПРИЛОЖЕНИЕ Е. ЗНАЧИМОСТЬ РЕЗУЛЬТАТОВ

Имя датасета	Статистика (значение)	Статистика rval (значение)	Статистика (итерация)	Статистика rval (итерация)
abalone	46,0	0,3956683900503302	43,5	0,3249508432726316
artificial- characters	26,0	0,03783078607194352	47,0	0,4249697048969556
balance-scale	0,0	8,930724418684081e-05	48,0	0,45486094457277765
breast-tissue	0,0	9,133589555477501e-05	47,5	0,4398921505284963
car	49,0	0,4849249884965778	34,0	0,12030653785531809
cardiotocography	34,0	0,12066079650859002	36,0	0,15374472830934066
cmc	27,0	0,04448650585090664	36,0	0,15365376636235417
cnae-9	18,0	0,008628728041559883	39,5	0,22467534888194385
collins	45,0	0,3668649978481236	46,5	0,4102315494191477
coverttype	10,0	0,0014136360455584	47,0	0,42502551244863873
desharnais	9,0	0,000998433276441516	44,5	0,35267568603175237
digggle-table-a2	46,0	0,3956683900503302	37,5	0,18199259743545926
ecoli	0,0	9,082557304573249e-05	42,5	0,29821248072862516
energy- efficiency	34,0	0,12066079650859002	15,5	0,005054285964439421
eye-movements	36,0	0,15374472830934066	39,5	0,22467534888194385
fabert	10,0	0,000857312489973485	639,0	0,175714550579799
fars	0,0	9,133589555477501e-05	36,0	0,15356273739136644
Fashion-MNIST	25,0	0,032011050641513446	10,0	0,0014084528140968727
gas-drift	8,0	0,000853124684459798	233,5	0,11297593015667973
gas-drift- different- concentrations	29,0	0,06061225150645831	47,0	0,42502551244863873

gina-prior2	11,0	0,0018052571561648	43,0	0,31145735104705063
glass	0,0	9,082557304573249e-05	49,0	0,4849022788860054
har	20,0	0,012874040410554032	35,0	0,13651816987559418
hayes-roth	30,0	0,06978453282269872	34,0	0,12066079650859002
heart-long-beach	16,0	0,005664848342237332	45,0	0,36647812030054766
heart-switzerland	30,0	0,07023252407917747	47,5	0,43986969591458647
helen	38,0	0,19233653136775436	43,5	0,3249508432726316
Indian-pines	30,0	0,07023252407917747	41,5	0,27244787233428047
iris	-1	-1	27,0	0,041542094427088304
jannis	35,0	0,13651816987559418	43,0	0,31145735104705063
JapaneseVowels	26,0	0,03783078607194352	37,5	0,18208286732008583
jungle-chess-2pcs-endgame-panther-elephant	25,0	0,00730347106010808	1,0	0,0001217433571486643
jungle-chess-2pcs-raw-endgame-complete	34,0	0,12066079650859002	47,0	0,42491377254495055
leaf	43,0	0,3115881119410587	18,0	0,008586830498033986
LED-display-domain-7digit	2,0	0,000164919260389	36,0	0,15356273739136644
mfeat-factors	31,0	0,08098620524006306	44,5	0,3525699015067423
mfeat-fourier	35,0	0,13651816987559418	32,0	0,09269007167775994
mfeat-karhunen	14,0	0,00364227850473983	44,0	0,3386779729030972
mfeat-morphological	47,0	0,42505336956926293	35,5	0,14486853141106892
mfeat-pixel	48,0	0,45486094457277765	48,5	0,46982617293653633
microaggregation	225,0	0,032011050641513446	34,5	0,12824068326368865
nursery	34,0	0,12066079650859002	42,5	0,2981433107864735

page-blocks	0,0	9,133589555477501e-05	47,5	0,43984721611720096
pokerhand	24,0	0,026951278584693587	48,0	0,4548271126400856
PopularKids	27,0	0,044426454239524814	44,0	0,3386779729030972
prnn-fglass	29,0	0,06033231821751136	43,5	0,3244567311076058
prnn-viruses	35,0	0,038791648742565764	24,0	0,026725923112053882
rmftsa-sleepdata	33,0	0,10614691809616578	41,0	0,25978970651570954
robot-failures-lp1	9,0	0,000847099658771918	439,5	0,22467534888194385
robot-failures-lp5	12,0	0,002293196040126747	48,5	0,4698375225718726
satimage	0,0	9,133589555477501e-05	29,5	0,06514034876426138
seeds	45,0	0,365346788382636	26,0	0,03761083267406863
segment	3,0	0,000219819376313	39,5	0,22458995686364547
seismic-bumps	0,0	6,748340454864083e-05	45,0	0,36672031294839225
semeion	0,0	9,133589555477501e-05	41,5	0,27244787233428047
shuttle	17,0	0,006357167164419588	44,0	0,33879247897623777
spectrometer	23,0	0,022536893158116134	47,5	0,4397796251524991
steel-plates-fault	44,0	0,33879247897623777	38,5	0,2027511500649954
synthetic-control	-1	-1	46,0	0,39524434627911254
tae	9,0	0,0011011099712485	28,0	0,051924790898487144
tamilnadu-electricity	48,0	0,45444986940149457	49,0	0,4849193207024165
teachingAssistant	8,0	0,0008531246844598	26,0	0,03744590341511262
thyroid-allbp	37,0	0,1723521110034788	41,0	0,260104502890281
thyroid-allhyper	38,0	0,19233653136775436	30,0	0,07000863254399659
user-knowledge	13,0	0,00284277879288317	43,5	0,32470430842540254
vehicle	36,0	0,15374472830934066	44,5	0,3525699015067423
vertebra-column	12,0	0,0022855542308096	28,0	0,051989872229493375

volcanoes-a1	46,0	0,3956683900503302	49,5	0,5
volcanoes-a3	26,0	0,0377757927648433	29,0	0,06047231621499348
volcanoes-a4	29,0	0,06061225150645831	40,0	0,23608777532022296
volcanoes-d1	37,0	0,17198879328001915	44,5	0,3521978549189131
wall-robot- navigation	29,0	0,06061225150645831	49,5	0,5
waveform-5000	28,0	0,05205494483011341	46,0	0,3955147886490878
wine	41,5	0,27252360124059083	40,0	0,2357533196687387
wine-quality- white	19,0	0,01056696406458055	35,0	0,13642798424356428
zoo	40,0	0,08403915951748514	28,5	0,05559700402954738