Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных и машинного обучения**

Пояснительная записка к курсовой работе

по дисциплине “Технологии анализа данных и машинное обучение”

на тему:

«Исследование эффективности различных методов оптимизации гиперпараметров в задачах машинного обучения»

Выполнила:

студентка группы ПИ19-3 факультета информационных технологий и анализа больших данных

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Соколова А.В.

Научный руководитель:

д.э.н., доцент, профессор Коровин Д.И.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2022 г

**Оглавление**

[**Введение** 2](#_Toc104148464)

[**Актуальность** 3](#_Toc104148465)

[**Цель и задачи работы** 4](#_Toc104148466)

[**Описание алгоритма** 5](#_Toc104148467)

[**Описание набора данных** 5](#_Toc104148468)

[**Предварительный анализ** 7](#_Toc104148469)

[**Описательный анализ** 9](#_Toc104148470)

[**Деление данных на тестовую и тренировочную выборки** 14](#_Toc104148471)

[**Обучение моделей** 17](#_Toc104148472)

[**Методы оптимизации гиперпараметров** 21](#_Toc104148473)

[**Grid search** 21](#_Toc104148474)

[**Random search** 23](#_Toc104148475)

[**Bayesian optimization** 25](#_Toc104148476)

[**Эксперимент, определяющий зависимость получения разных результатов при разных запусках алгоритма от размерности данных.** 28](#_Toc104148477)

[**Заключение** 32](#_Toc104148478)

[**Список литературы** 33](#_Toc104148479)

# **Введение**

В последнее десятилетие действительно широкое распространение получил ряд алгоритмов, характерной чертой которых является не прямое решение задачи через написание алгоритма действий для достижения результата, а использование таких алгоритмов, которые позволяют давать более точные результаты в процессе обучения за счет применения решений множества сходных между собой задач.

Данный подход позволил более эффективно решать великое множество задач в различных сферах. Перевод текстов и его интерпретация, классификация изображений и компьютерное зрение, кредитный скоринг и автоматическое обнаружение аномалий в системах безопасности – это лишь малый список областей, где использование машинного обучения является базовым подходом для решения задач в связи с его высокой эффективностью.

Данная тенденция успела отразиться и на рынке труда. Так, уже сегодня специалисты в области data science получают наибольшие зарплаты, среди всех IT направлений, что говорит о востребованности таких специалистов. Помимо этого, количество таких вакансий ежегодно растет, что также указывает на всевозрастающую популярность данного направления.

Стоит отметить, что наука о данных не стоит на месте. Набор вышеописанных алгоритмов постоянно растет, с завидной частотой появляются новые, более сложные модели, или усовершенствуются старые.

## **Актуальность**

Из осознания важности методов машинного обучения для решения практических задач неизбежно вытекает мысль о важности эффективного их использования. В этом аспекте перед программистом, стремящимся решить задачу, стоит несколько основных проблем: выбор правильной модели, подготовка данных и настройка гиперпараметров модели.

Большинство моделей машинного обучения имеют 2 разновидности параметров, определяющих в конченом итоге скорость и качество их работы. Рассмотрим их подробнее.

Первый тип – это параметры модели, меняющиеся в процессе обучения. Именно за счет этого изменения происходит обучение модели, позволяющего со временем улучшать результат.

Второй тип – это те параметры, которые определяют саму структуру модели. Они устанавливаются программистом перед началом обучения и не меняются в процессе. Для обозначения данного типа параметров традиционно принято использовать термин “гиперпараметры”, как бы подчеркивая более высокий уровень их абстракции.

Как правило, процесс подбора наилучших гиперпараметров является достаточно трудоемким. Даже для простых алгоритмов поиск наилучшего их набора может быть сложен и потребует от исследователя значительных временных затрат для выполнения этой задачи. По этой причине сложно переоценить актуальность проблемы автоматического подбора гиперпараметров модели, исследованием которой и посвящена данная работа.

Важной особенностью актуальности данной работы является тот факт, что описываемые ниже способы подбора гиперпараметров актуальны не только для существующих алгоритмов, но также имеются все основания полагать, что многие будущие алгоритмы можно будет настраивать, используя описанные в данной работе методы.

## **Цель и задачи работы**

Выполняя данную работу, я ставлю перед собой целью исследовать различные методы оптимизации гиперпараметров в задачах машинного обучения. Для этого необходимо выполнить следующие шаги:

1. Выбрать набор данных для анализа.
2. Провести предобработку данных.
3. Провести описательный анализ.
4. Разделить на тестовую и тренировочную выборки.
5. Обучить несколько моделей и выбрать подходящую для последующей оптимизации.
6. Провести оптимизацию гиперпараметров несколькими методами и проанализировать результаты.

# **Описание алгоритма**

## **Описание набора данных**

Для реализации данной курсовой работы мной были выбраны данные из библиотеки [sklearn.datasets](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets). Данный модуль включает в себя утилиты для загрузки стандартных, популярных наборов данных. Как раз таким стандартным набором данных является Boston house prices, который содержит набор признаков, влияющих на цену жилья в Бостоне. Набор данных включает как числовые, так и категориальные признаки, что является безусловным плюсом для данной работы, поскольку позволяет оценить методы работы алгоритмов на различных наборах данных.

В ходе данной работы будет решаться задача предсказания цены жилья в Бостоне в зависимости от следующих признаков:

1. CRIM – уровень преступности на душу населения в разбивке по населенным пунктам.
2. ZN – доля жилой земли, зонированной на участки площадью более 25 000 кв.футов.
3. INDUS – доля промышленных площадей, не связанных с розничной торговлей, в расчете на город.
4. CHAS – категориальный признак. Переменная равняется 1 – если транспортная дорога граничит с рекой Charles River, в противном случае 0.
5. NOX – концентрация оксидов азота.
6. RM – среднее количество комнат в жилом помещении.
7. AGE – доля квартир, занятых владельцами, в домах, которые были построены до 1940 г.
8. DIS – взвешенные расстояния до центров занятости в Бостоне.
9. RAD – индекс доступности к радиальным магистралям.
10. TAX – ставка налога на недвижимость с полной стоимостью за 10 000 долларов США.
11. PTRATIO – соотношение учащихся и учителей в разбивке по городам.
12. B – доля чернокожего населения. Считается данная метрика по формуле 1000(Bk - 0,63)^2
13. LSTAT – процент более низкого статуса населения.

В качестве целевой переменной будем использовать числовые данные – MEDV (средняя стоимость домов, занятых владельцами, где стоимость указана в тысячах долларов).

Так как целевая переменная не имеет фиксированного набора значений, а напротив представляет собой бесконечное множество таких вариантов, данная задача, безусловно, является задачей регрессии.

## **Предварительный анализ**

Перед непосредственным применением машинного обучения необходимо провести два немаловажных этапа: этап чистки данных и предварительный их анализ. Под чисткой данных, обычно, понимается набор некоторых операций, которые позволяют привести данные в удобный для алгоритмов машинного обучения вид без выбросов, пустых значений и в едином формате. Устранение данных проблем необходимо так, как в ином случае мы не сможем построить случае адекватную модель.

В нашем датасете изначально данные представлены единым форматом float64 и отсутствуют пустые значения, что автоматически избавляет нас от проблем их обработки.

Вторая часть предобработки данных - предварительный анализ данных. Этот этап включает в себя вывод информации о количественных характеристиках датасета, информацию об отсутствующих значениях, характеристиках и физическом смысле каждого атрибута данных, его значимости для предсказания целевой переменной, вывод нескольких точек данных для иллюстрации структуры данных. То есть целью этого этапа является понимание, что эти данные из себя представляют, как лучше всего с ними работать, а также что они могут нам дать. Для этого производится исследование характеристик набора данных и каждого признака, в частности.

Предварительный анализ в большинстве случаев проводиться единожды для каждой конкретной задачи машинного обучения.

Итак, в данном наборе данных Boston house prices содержится 13 признаков и одна целевая переменная. И хоть число строк данных является относительно небольшим и содержит всего 506 записей, этого количества вполне достаточно для построения хорошей модели предсказания.

Как уже было упомянуто выше, все признаки в наборе имеют тип данных float64, то есть вещественные числа с плавающей запятой. В первую очередь это говорит нам о том, что датасет не содержит текстовых данных, что существенно облегчает нам задачу.

На рисунке ниже приведена некоторая сводная информация о наборе данных. Это тип данных значений датасета, количество ненулевых значений, а также общий его размер.

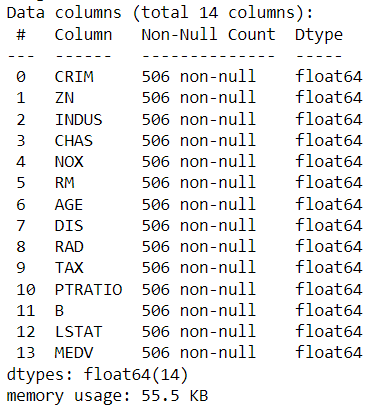


Рис. 1. Сводная информация о наборе данных.

Примеры строк данных имеют следующий вид:

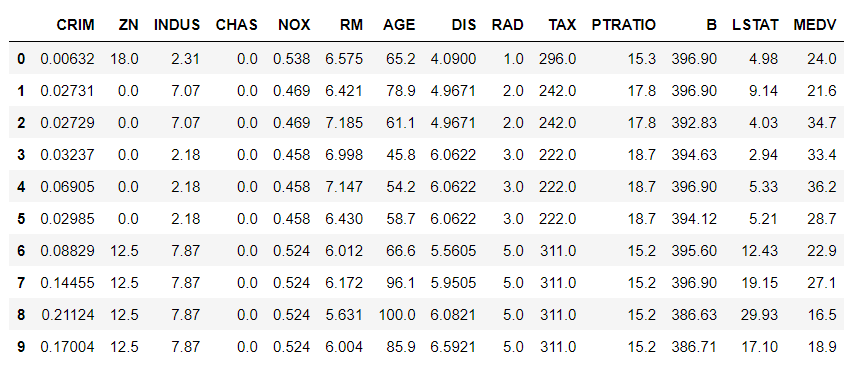


Рис. 2. Пример данных.

## **Описательный анализ**

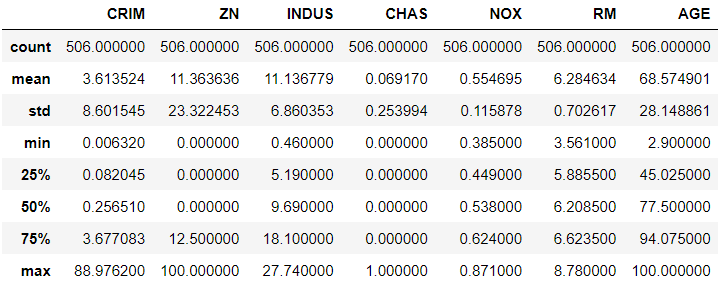
Теперь, когда мы знаем, что у нас очищенные, пригодные для работы данные, мы можем перейти к следующему этапу.

Описательный анализ — это некоторая попытка описать или обобщить данные. Хотя на этом этапе мы не делаем прогнозов на будущее, [он все же может быть чрезвычайно полезен](https://pestleanalysis.com/data-analysis-tools/).

Основной задачей описательного анализа стоит визуализация данных для создания значимых выводов.

Описательные стратегии часто включают разработку таблиц средних значений, стандартного отклонения, дисперсии, которые можно использовать для рассмотрения многочисленных разрозненных гипотез.

Описание наших данных через шкалу признаков каждого измерения выглядит следующим образом:



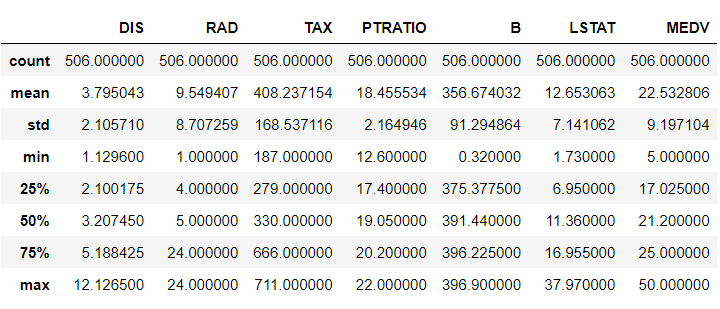


Рис. 3. Описательная статистика данных.

Преимуществом описательного анализа является то, что он может помочь отфильтровать менее значимые данные. В наших данных это было отражено в результате построения корреляционной матрицы, представленной на рисунке ниже:

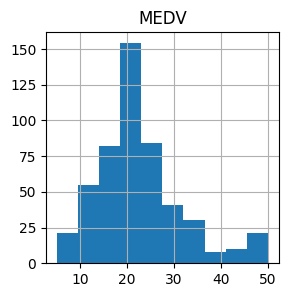
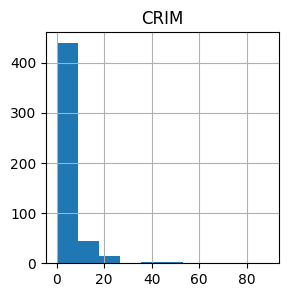
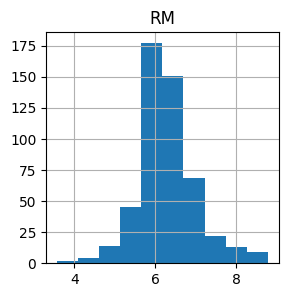
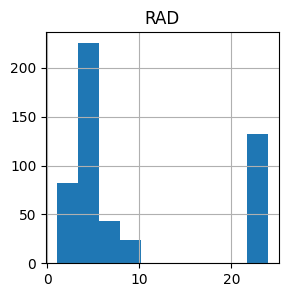
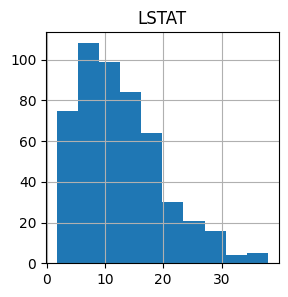
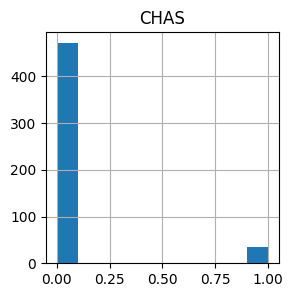
Изображение выглядит как стол

Автоматически созданное описание

Рис. 4. Корреляционная матрица данных.

Из рисунка видно, что между признаками TAX и RAD имеет место быть значимая положительная корреляция, близкая к единице. Положительный коэффициент корреляции указывает на то, что с увеличением значений одной переменной наблюдается рост другой. Из этого можно сделать вывод о том, что имеется возможность удаления одного из признаков из модели, не сильно ухудшая (а иногда и заметно улучшая) результаты машинного обучения.

Теперь давайте проведем описательный анализ для каждого признака и посмотрим его распределение:



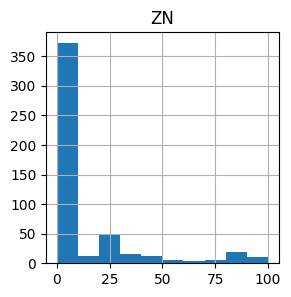
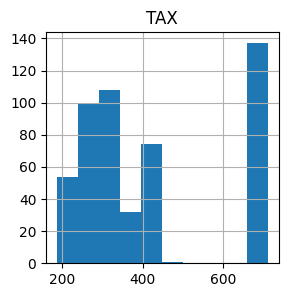
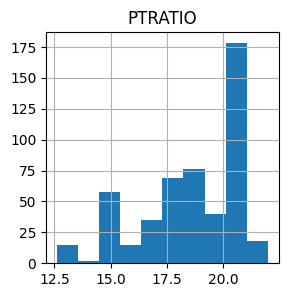
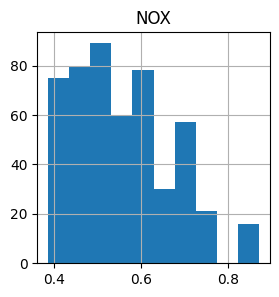
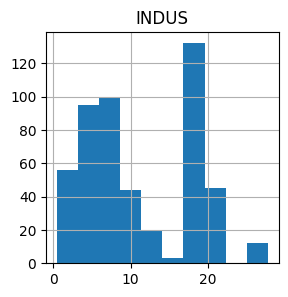
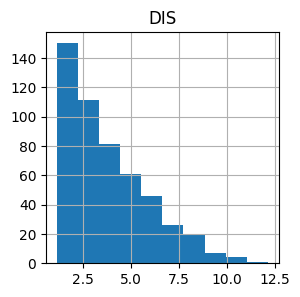
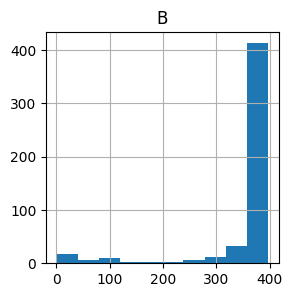
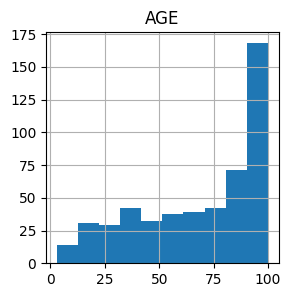


Рис. 5. Распределение признаков набора данных.

Анализируя распределения, можно прийти к следующим выводам:

1. Признак CHAS имеет ограниченное малое число вариантов (всего два), что говорит о том, что он является категориальным с двумя возможными значениями. При этом значение 0 явно доминирует над альтернативным.
2. Наибольшая часть значений LSTAT находится в диапазоне от 5 до 9. Для значений больших 9 наблюдается плавный спад их количества по мере отдаления от данной величины.
3. Признак RAD имеет 2 ярко выраженных кластера. Первый со значениями от 1 до 10 и второй, со значениями равными 24.
4. Признак RM имеет распределение, схожее с нормальным, со средним значением около 6 и стандартным отклонением около одного.
5. Подавляющая часть значений CRIM расположены от 0 до 30.
6. Признак MEDV имеет распределение, близкое к нормальному, с некоторым увеличением в правом его хвосте.
7. Значения переменной AGE находятся в диапазоне от 3 до 100, при этом их количество плавно возрастает, приближаясь к правой отметке.
8. Подавляющая часть значений переменной B расположена в диапазоне от 300 до 400. Левее наблюдается некоторое количество выбросов.
9. В случае с переменной DIS наблюдается очень плавный спад количества значений от левой границы (около одного) до правой (порядка двенадцати).
10. Переменная INDUS имеет бимодальное распределение с пиками в точках 5 и 19.
11. Для NOX наблюдается ситуация уменьшения количества наблюдений по мере роста значения.
12. Распределение PTRATIO неоднозначно.
13. Значения TAX распределены по двум кластерам: Со значениями от 200 до 450 и со значениями, приблизительно равными 700.
14. Подавляющее большинство значений переменной ZN приблизительно равняются нулю.

## **Деление данных на тестовую и тренировочную выборки**

В результате предварительного анализа данных видно, что мы не имеем дело с временными рядами, наши данные не имеют сезонности, а значит, необходимо использовать случайный метод разделения данных на тестовую и тренировочную выборки. Пару слов стоит сказать о том, зачем вообще возникла необходимость в разделении набора данных.

В процессе проверки качества работы моделей машинного обучения перед любым исследователем неизбежно встает проблема переобучения. Это такая ситуация, когда модель, реагируя на изменения в данных, вызванных случайными факторами, делает выводы на их основе. В результате получается, что данный программный алгоритм отлично работает на наборе обучающих данных, однако при встрече с незнакомыми его эффективность разительно уменьшается.

Частичным решением этой проблемы становится изначальное деление набора данных на 2 части. Первую используют для обучения самой модели, вторую для её проверки. Данный метод действительно позволяет существенно снизить эффект переобучения, поскольку теперь исследователь будет ориентироваться на результаты работы алгоритма на новых для него данных.

Однако это не решает проблему полностью. Это происходит по той причине, что со временем исследователь в попытках подобрать наилучшею модель с оптимальными параметрами сам невольно начнет переобучать свою модель, выбирая те модели и её параметры, которые дают хороший результат именно на этих тестовых данных. Это рождает абсолютно идентичный ряд проблем, в результате которых способность модели к предсказанию разительно уменьшается на данных, которых не было ни в тестовой, ни в тренировочной выборке.

Улучшить подобный порядок вещей помогает использование метода под названием Cross-Validation (перекрестная проверка в дословном переводе). Суть данного метода заключается в следующем:

1. Обучающая выборка разбивается на k наборов.
2. Для каждого из этих k наборов происходит обучение модели на k-1 наборах в качестве обучающих данных и проверка на оставшемся наборе.
3. Оценка качества моделей усредняется

После нахождения таким способом оптимальной модели и её гиперпараметров необходима окончательная проверка данный обученной модели на наборе тестовых данных, осуществляя проверку на новой для модели информации.

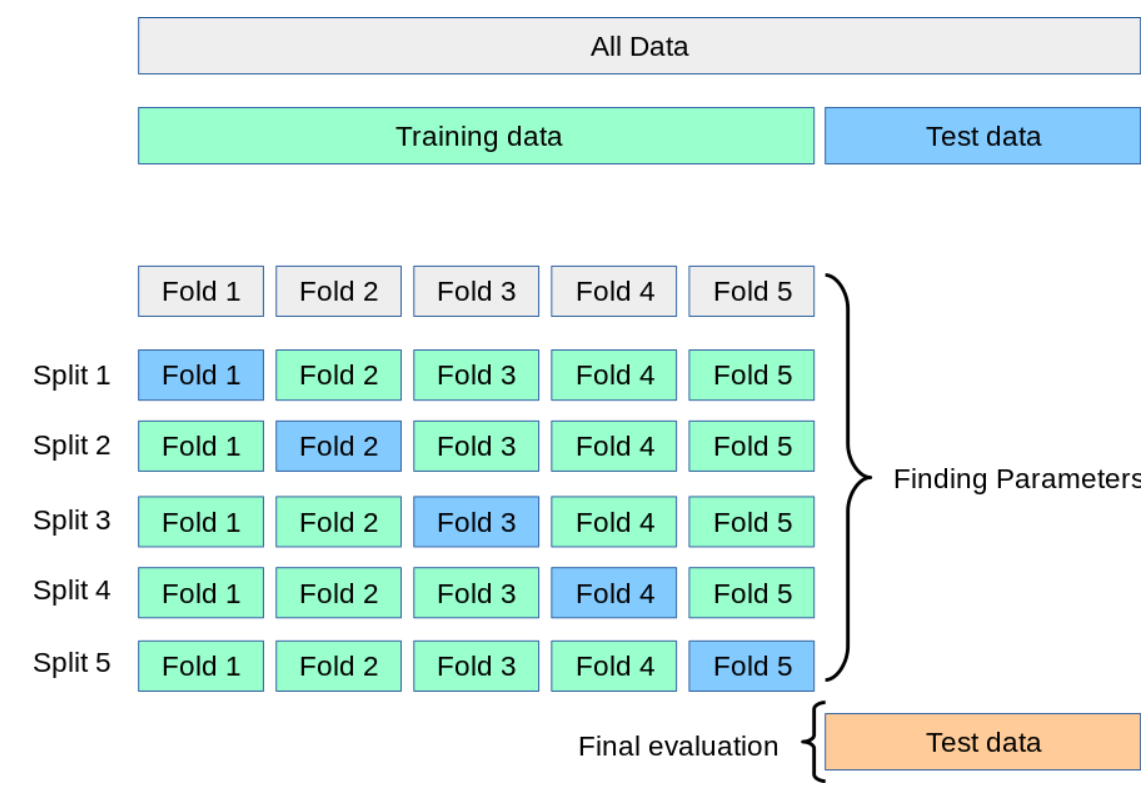


Рис. 6. Иллюстрация способа определения проверки качества модели.

Таким образом, в данной работе используется следующих подход:

1. Выбор моделей и их гиперпараметров производится, ориентируясь на результаты кросс валидации.
2. Итоговые варианты моделей с подобранными гиперпараметрами проверяем на тестовой выборке.

## **Обучение моделей**

Следующим этапом данной работы является обучение нескольких моделей со стандартным набором параметров. Для этого были выбраны следующие модели:

1. Дерево решений
2. Случайный лес
3. Градиентный бустинг

Давайте разберем алгоритм работы этих методов.

Итак, начнем с модели «Дерево Решений» (Decision Trees).

Дерево решений представляет из себя модель, похожую на блок-схему в виде дерева, в которой внутренний узел являет из себя функцию (или некоторый атрибут), ветвь представляет правило принятия решения, а каждый конечный или листовой узел является результатом.

Самый верхний узел в дереве решений принято называть корневым узлом. Алгоритм учится разделять данные на основе значений атрибута, вызывая рекурсивное разбиение.

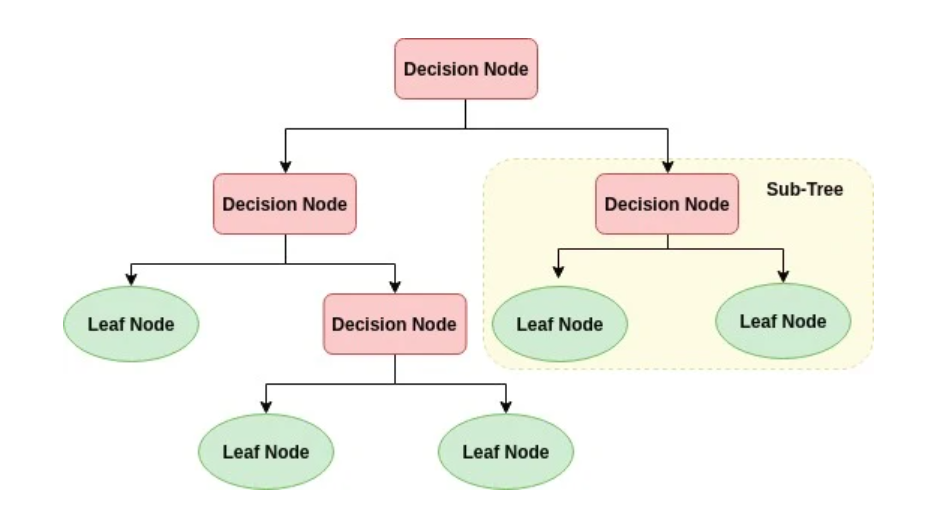


Рис. 7. Иллюстрация структуры модели Decision Trees.

Задачами, решаемыми с помощью данной модели, являются как задача классификации, так и задача регрессии.

Как и говорилась ранее перед нами стоит задача регрессии. Для ее решения мы будем использовать  [DecisionTreeRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html" \l "sklearn.tree.DecisionTreeRegressor" \o "sklearn.tree.DecisionTreeRegressor).

Данный метод обучения показал нам следующие результаты:

Значение Cross-Validation равное 0.7428, и значение score на тестовой выборке равное 0.7843. Результаты являются не совершенными и можно попробовать их улучшить.

Следующая модель, которую мы рассмотрим, будет «Случайный лес» (Random Forest).

Данная модель использует для своего алгоритма дерево решений, которое мы разобрали ранее. Она создает необходимое количество деревьев решений на случайно выбранных частях данных. В результате работы модель получает предсказание по каждому из созданных деревьев. В случае получения различных результатов деревьями встает проблема выбора итогового значения. Существуют различные способы решения этой проблемы. Мы можем принимать решение на основе медианных значений, использовать различные весовые коэффициенты или любые другие удобные исследователю способы. В большинстве же случаев и, в частности, в нашей работе в качестве результата рассматриваются среднее значение всех выходных данных дерева.

Его также можно использовать как для классификации, так и для регрессии.

Случайный лес, а именно RandomForestRegressor, показал нам более хорошие результаты работы, а именно:

* Значение Cross-Validation равное 0.8345.
* Значение score на тестовой выборке равное 0.8603.

И последней использованной в этой работе моделю является градиентный бустинг (Gradient boosting).

В данном случаем слово "Boosting" является термином для способа объединения нескольких простых моделей в единую составную. Таким способом данная модель может дать гораздо лучшие результаты в отличии от простых моделей.

Обучения модели Gradient boosting происходит в несколько этапов:

1. Выбор слабой модели обучения.

В качестве этого шага также используется дерево решений, которое мы уже не раз упомянули и разобрали.

1. Использование адаптивной модели.

Получение адаптивной модели происходит при использовании концепции остатков, как и в линейной регрессии. Модель вычисляет разницу между текущим предсказанием и известными правильными целевыми значениями. После этого повторяется обучение модели дерево решений с добавлением остатков к существующим входным данным. Повторение этого шага происходит неоднократно и неизбежно делает модель все лучше и лучше.

1. Определение функции потерь.

Функции потерь могут представлять различного рода дифференцируемые функции такие как: MAE (средняя абсолютная ошибка) и MSE (среднеквадратическая ошибка). Обычно, в модели обучения градиентного бустинга используется именно среднеквадратическая ошибка.

1. Минимизирование функции потерь.

Результаты работы этой модели показали наилучший результат, а именно:

* Значение Cross-Validation равное 0.8495.
* Значение score на тестовой выборке равное 0.8807.

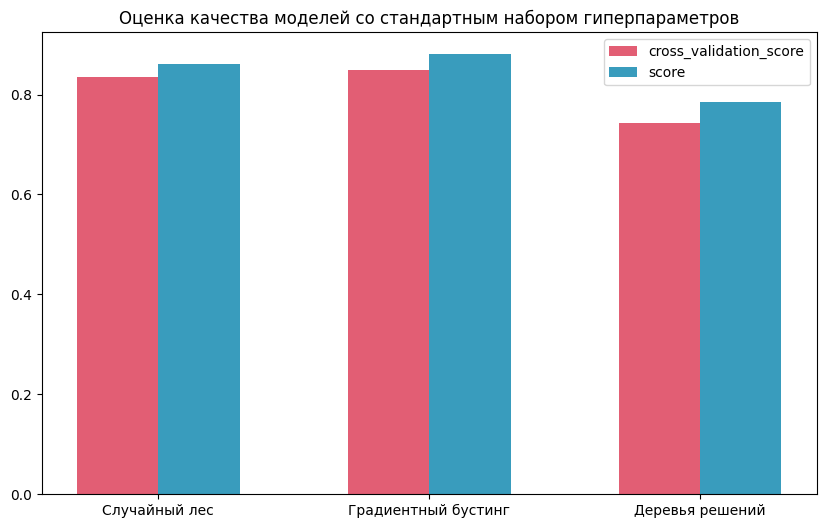


Рис. 8. Оценка качества моделей со стандартным набором гиперпараметров.

## **Методы оптимизации гиперпараметров**

В данной работе была проведена проверка скорости и эффективности по критерию качества наиболее распространенные методы автоматизированного подбора гиперпараметров моделей машинного обучения. Это Random search, Grid search и байевская оптимизация. Рассмотрим каждый из подробнее.

### **Grid search**

В переводе означает “поиск по сетке”. Данный алгоритм позволяет найти наилучшую комбинацию из различных параметров, которые мы предоставляем.

Механика работы метода довольно проста. Он перебирает все возможные комбинации параметров, обучает на каждой такой комбинации модель и считает её качество. Затем путем сравнения результатов выдает набор параметров, который дал наилучшие результаты.

Важным моментом является тот факт, что в данной работе при оценке результатов моделей использовался метод кросс валидации, описанный выше.

Пример.

Стоит задача оптимизации гиперпараметров модели Desition Tree. Для этого мы зададим на вход алгоритма словарь из списков гиперпараметров:

1. Максимальная глубина дерева: 5, 10, 20, 50, 100
2. Минимальное число объектов в листе: 1, 2, 3, 5, 10
3. Минимальное количество объектов, необходимое для разделения внутреннего узла: 2, 3, 5, 10

В этом случае при запуске Grid search будут выбраны все возможные комбинации этих признаков, начиная с [5, 1, 2] и заканчивая [100, 10, 100]; для каждого из этих случаев будет проведена оценка модели через кросс валидацию, на основании которой будет в итоге выбран наилучший набор гиперпараметров.

Не трудно догадаться, что хоть данный метод и является наиболее эффективным в плане нахождения такого набора гиперпараметров, который обеспечивает наилучший score на кросс валидации из диапазона заданных значений, он является крайне вычислительно сложным. Это особенно сказывается на сложных моделях, представляющих из себя ансамбль более простых моделей, таких как случайный лес. В этом случае внесение в список возможных гиперпараметров хотя бы одного дополнительного значения для проверки кратно увеличивает число вариантов перебора, что кардинально сказывается на временных затратах.

Эти факты подтвердились на практике. Так результат работы Cross-Validation равняется 0.7805, а качество на тестовой выборке дало 0.8145, что

намного лучше изначальных результатов.

Время выполнения этого алгоритма является относительно высоким, а

именно около 40 секунд.

### **Random search**

Данный метод родился как попытка решить вышеописанные проблемы метода grid search в виде огромной вычислительной сложности.

В первую очередь, разработчиками было обращено внимание на то обстоятельство, что исследователь, как правило, имеет лишь представление о диапазоне значений определенного признака, а не его конкретных значений (в случае с некатегориальными значениями). Это означает, что даже если мы будем использовать grid search, мы стабильно пропускаем значения, находящиеся не в вершинах нашей сетки. В этом случае мы можем поступить альтернативным способом и попробовать взять случайные точки из всего диапазона значений.

Данную ситуацию хорошо иллюстрирует схема ниже. На ней изображен случай оптимизации 2-х параметров, при котором признак по горизонтали имеет большее влияние на результат. Тогда оптимальное значение может находиться между узловыми значениями сетки.

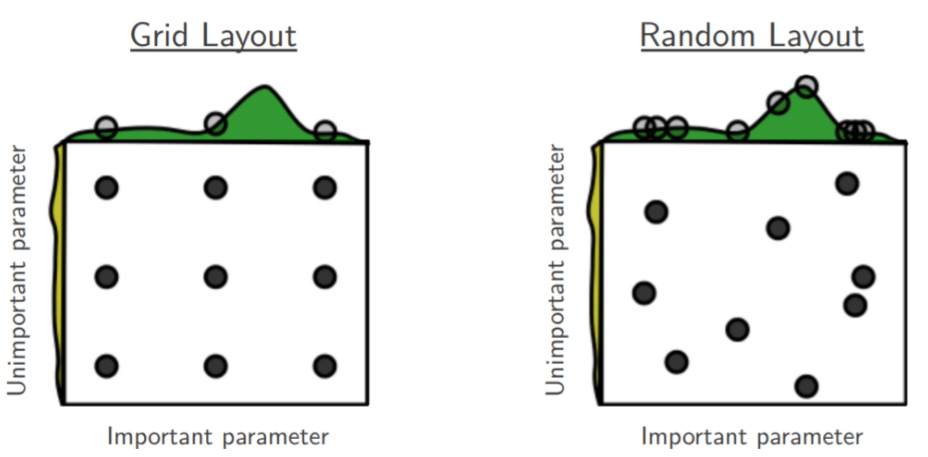


Рис. 9. Сравнение работы методов Grid Search и Random Search.

Таким образом, Random Search использует случайные значения из диапазона возможных по всем признакам, осуществляя поиск оптимального значения.

В нашей работе данный метод показал следующие результаты:

* + Значение Cross-Validation равное 0.7054.
  + Качество модели на тестовой выборке равное 0.8206.
  + Время выполнения варьируется от 0,5 до 5 секунд.

Таким образом, в сравнении с Grid Search данный метод тратит значительно меньшее время, что позволяет использовать более широкий диапазон значений, страдая при этом небольшой потерей точности или, как в нашем случае, дает даже более хорошие результаты.

### **Bayesian optimization**

Байесовский подход к оптимизации является продолжением идеи случайного поиска, являясь модификацией этого алгоритма. Рассуждая над способами улучшить оптимизацию random search, кажется абсолютно логичной мысль о том, что более привлекательными для нас являются области, находящиеся поблизости с теми точками, которые дают наибольшие показатели качества модели. Эта идея о том, что при выборе оптимальной области пространства гиперпараметров стоит использовать историю уже рассмотренных ранее точек, в которых были обучены модели и получены значения целевой функции, и легла в основу данного метода.

Данный метод включает в себя два основных момента: вероятностную суррогатную модель и функцию выбора следующей точки. На каждой следующей итерации такая модель обучается по всем уже полученным ранее выходам целевой функции. Используя такой подход, мы стремимся, используя минимальное количество ресурсов, получить аппроксимацию целевой функции. Следующим шагом является действие функции выбора, которая через использование предсказательного распределения суррогатной модели оценивает «выгодность» различных следующих точек, находя баланс между использованием доступной нам информации и «разведыванием» новой области пространства.

Исследования эффективности данного метода дали следующие результаты:

* + Значение Cross-Validation равное 0.7501.
  + Качество модели на тестовой выборке равное 0.8399.
  + Время выполнения варьируется от 2 до 5 секунд.

Таким образом, байесовская оптимизация дала лучшие результаты, при одинаковых затратах времени с random search.

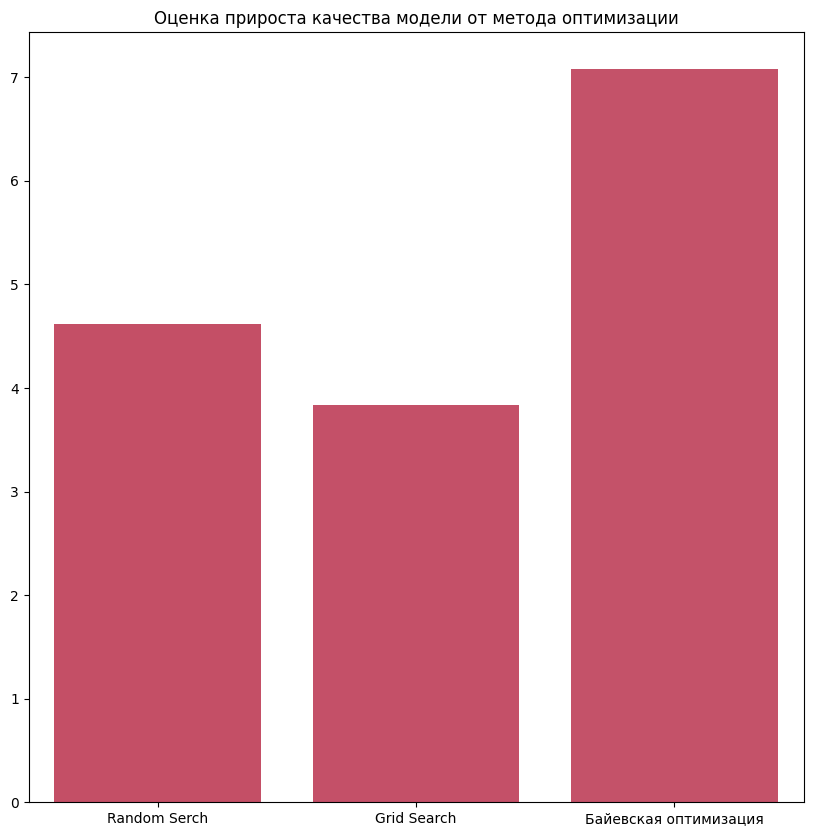


Рис. 10. Оценка прироста качества модели от метода оптимизации

На данном графике изображен прирост качества модели в % от изначальной модели со стандартными гиперпараметрами. Из графика видно, что лучшие результаты дал алгоритм работы байесовской оптимизации с относительно большим отрывом. Качество моделей с Random search и Grid search дало не значительную разницу, основанную скорее на случайности.

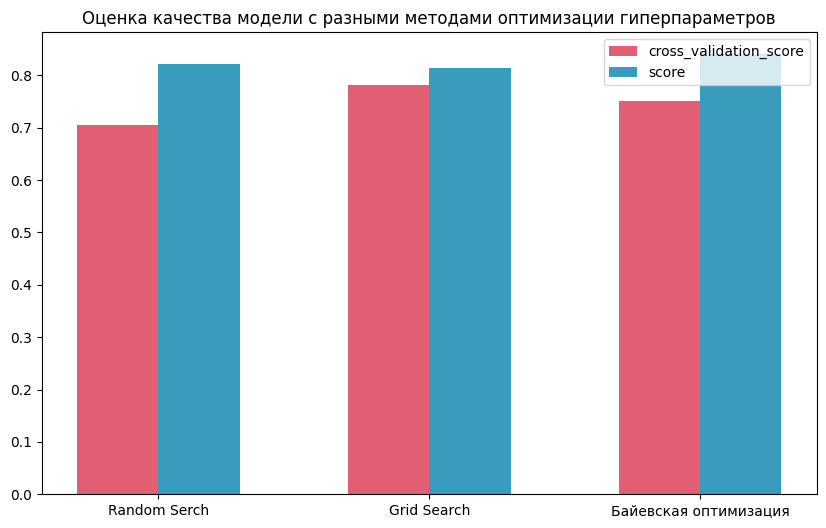


Рис. 11. График оценок качества моделей с разными методами оптимизации гиперпараметров.

На графике видно, что у всех трех алгоритмах не имеется значительной разницы между собой, как на Cross-Validation, так и на тестовой выборке. Однако имеются некоторые отличия порядка, измеряющиеся тысячными долями.

# **Эксперимент, определяющий зависимость получения разных результатов при разных запусках алгоритма от размерности данных.**

Проблемой нашего исследования является получение каждый раз различных результатов при повторном запуске алгоритмов оптимизации гиперпараметров.

В связи с этим мы выдвинем гипотезу о том, что при понижении размерности наши данные стабилизируются, что приведет к тому, что разница между полученными результатами работы одного и того же алгоритма станет существенно меньше.

Для этого мы используем следующий алгоритм:

1. Запустим наши алгоритмы 10 раз без понижения размерности и оценим, полученные результаты. В том числе разницу между самым наименьшим результатом и самым наибольшим.
2. Понизим размерность данных.
3. Повторим первый шаг уже с пониженной размерностью данных.
4. Сравним полученные результаты.

Результаты алгоритмов без понижения размерности:

| **Номер итерации** | **Random search** | **Grid search** | **Байесовская оптимизация** |
| --- | --- | --- | --- |
| **1** | 0.820594 | 0.841560 | 0.789620 |
| **2** | 0.820594 | 0.839438 | 0.780566 |
| **3** | 0.814460 | 0.839438 | 0.787042 |
| **4** | 0.820594 | 0.820594 | 0.788715 |
| **5** | 0.820594 | 0.839438 | 0.788715 |
| **6** | 0.820594 | 0.841560 | 0.788363 |
| **7** | 0.784127 | 0.841560 | 0.723753 |
| **8** | 0.820594 | 0.839438 | 0.812993 |
| **9** | 0.784127 | 0.839438 | 0.852208 |
| **10** | 0.784127 | 0.839438 | 0.788448 |

Результаты алгоритмов с понижением размерности:

| **Номер итерации** | **Random search** | **Grid search** | **Байесовская оптимизация** |
| --- | --- | --- | --- |
| **1** | 0.784449 | 0.841560 | 0.735339 |
| **2** | 0.784449 | 0.814460 | 0.809178 |
| **3** | 0.784449 | 0.841560 | 0.834948 |
| **4** | 0.784449 | 0.839438 | 0.728880 |
| **5** | 0.784127 | 0.839438 | 0.788715 |
| **6** | 0.814460 | 0.841560 | 0.714052 |
| **7** | 0.820594 | 0.814460 | 0.784499 |
| **8** | 0.814460 | 0.841560 | 0.820104 |
| **9** | 0.831254 | 0.814460 | 0.783523 |
| **10** | 0.820594 | 0.839438 | 0.787042 |

Random Search



Grid Search



Байесовская оптимизация



Проанализировав результаты, можно сказать, что гипотеза оказалась неверной. Результаты не показали значительной разницы, а небольшие отклонения вызванные, вероятно, случайностью, которую используют сами алгоритмы.

# **Заключение**

Результаты сравнения рассмотренных выше алгоритмов на модели дерева решений отражены в виде сводной таблицы ниже.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Качество модели на кросс валидации | Качество модели на тестовой выборке | Скорость работы алгоритма |
| Модель со стандартным набором гиперпараметров | 0.7428 | 0.7843 |  |
| Grid Search | 0.7805 | 0.8145 | От 38 до 42 секунд |
| Random Search | 0.7054 | 0.8206 | От 0,5 до 5 секунд |
| Bayesian optimization | 0.7501 | 0.8399 | От 2 до 5 секунд |

Таким образом, можно сказать, что все три метода серьезно улучшают качество модели на тестовой выборке и показывают менее заметные улучшения на кросс валидации (кроме низкого показателя метода Random Search, вызванного, вероятно, случайностью).

При этом методы Random Search и оптимизации Байеса работают значительно быстрее, в сравнении с Grid Search, что делает их использование более выгодным для исследователя, в случае, когда он не имеет явных намерений проверить ряд конкретных значений, а располагает лишь их возможным диапазоном значений.

# **Список литературы**

1. [М.В. Коротеев. Об основных задачах дескриптивного анализа данных.](https://drive.google.com/file/d/1JoHxg3dfc53bSZtsNfPPB5Es_C4Q9hJ0/view?usp=sharing)
2. [М.В. Коротеев. Учебное пособие по дисциплине “Анализ данных и машинное обучение” - 2018.](https://portal.fa.ru/Files/Data/85bca4b4-cd40-4dae-bb62-dbd6c04fa0a0/um_tehnanalizadannihimashobuchenie_18.pdf)
3. [Feurer, Matthias, and Frank Hutter. «Hyperparameter optimization.» In Automated Machine Learning, pp. 3-33. Springer, Cham, 2019.](https://www.automl.org/wp-content/uploads/2019/05/AutoML_Book_Chapter1.pdf)
4. [Koehrsen, Will. «A conceptual explanation of bayesian hyperparameter optimization for machine learning.» (2018).](https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f)
5. [Bergstra, James S., Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. «Algorithms for hyper-parameter optimization.» In Advances in neural information processing systems, pp. 2546-2554. 2011.](https://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf)