GeekBrains

Ансамблирование и использование деревьев решений в задачах машинного обучения.

Библиотеки Python для Data Science

Оглавление

Введение **2**

Термины, используемые в лекции **2**

Как работают деревья решений **2**

Чем отличаются деревья решений регрессии и классификации **7**

Критерий остановки алгоритма **10**

Ансамблирование **12**

Что можно почитать еще? **20**

Используемая литература **20**

Введение

В данной лекции мы рассмотрим тему "Деревья решений. Ансамблирование". Если вы когда-либо задумывались о том, как компьютерные алгоритмы могут принимать решения, подобные человеческим, то вы попали по адресу. Сегодня мы погрузимся в мир машинного обучения и исследуем, как деревья решений и их ансамблирование могут помочь нам в создании эффективных моделей прогнозирования и классификации.

Итак, давайте начнем путешествие в мир "Деревьев решений. Ансамблирование". Деревья решений представляют собой графическую модель, в которой каждый узел представляет собой признак, по которому происходит разбиение данных, а каждое ребро — возможное значение этого признака. Листья дерева представляют собой конечные классы или значения, которые нужно предсказать.

Деревья решений обладают несколькими преимуществами. Во-первых, они широко применимы в различных областях, включая медицину, финансы, маркетинг и многое другое. Во-вторых, они легко интерпретируемы, что позволяет понять, какие признаки вносят наибольший вклад в принятие решения. В-третьих, они способны обрабатывать как категориальные, так и числовые данные.

Однако деревья решений также имеют некоторые ограничения. Во-первых, они склонны к переобучению, особенно на данных с большим количеством признаков

или неточностями. Во-вторых, они не всегда дают наилучшие результаты по качеству предсказания.

Чтобы устранить эти ограничения, мы будем также изучать метод ансамблирования. Ансамблирование представляет собой комбинирование нескольких моделей в одну, с целью улучшения ее предсказательной способности. Одним из самых популярных методов ансамблирования является случайный лес, который представляет собой набор деревьев решений, каждое из которых обучается на случайной подвыборке данных. Это позволяет снизить вероятность переобучения и повысить обобщающую способность модели.

В этой лекции мы будем изучать основные принципы построения деревьев решений и случайного леса, а также различные подходы к ансамблированию.

На этой лекции вы найдете ответы на такие вопросы как / узнаете:

- Как работают деревья решений
- Чем отличаются деревья решений регрессии и классификации
- Что такое ансамблирование
- Виды ансамблей в машинном обучении

Термины, используемые в лекции

Объект — пример, шаблон, наблюдение. Объект представляет собой элемент или экземпляр данных, который подается на вход модели для принятия решений. Каждый объект обладает набором признаков или атрибутов, которые используются для прогнозирования целевой переменной. Дерево решений делает серию решений, основанных на признаках объекта, чтобы классифицировать или предсказать значение целевой переменной для данного объекта.

Атрибут — признак, независимая переменная, свойство. Атрибут представляет собой характеристику или свойство объекта, по которой принимается решение алгоритмом деревья решений. Это может быть любая измеримая характеристика, такая как возраст, пол, доход и т. д.

Узел — внутренний узел дерева, узел проверки. Узел представляет собой точку разветвления, где происходит принятие решения алгоритмом на основе значений определенного атрибута. Каждый узел представляет собой конкретное условие, а

ветви, исходящие из узла, представляют возможные результаты этого условия, ведущие к другим узлам или листьям дерева.

Корневой узел — начальный узел дерева решений. В контексте деревьев решений, корневой узел представляет собой начальный узел, откуда начинается разделение набора данных на основе значений атрибутов. Корневой узел играет ключевую роль в построении дерева решений, так как он определяет, какие атрибуты будут использоваться для классификации или предсказания целевой переменной. От корневого узла начинаются пути разделения, которые формируют структуру дерева решений, помогая принимать решения и предсказывать результаты.

Лист — конечный узел дерева, узел решения, терминальный узел. Лист в контексте деревьев решений относится к терминальным узлам, которые представляют конечные прогнозы или классификации. В деревьях решений листья содержат конечные значения целевой переменной и не разделяются на подузлы. Они представляют конечные прогнозы или классификации на основе атрибутов и условий, содержащихся в узлах дерева.

Решающее правило — условие в узле, проверка — это логическое выражение или условие, которое определяет, какое следующее действие необходимо выполнить в дереве решений. Каждое решающее правило состоит из предиката и результата. Предикат — это условие или проверка, которое анализирует значения атрибутов или признаков для принятия решения. Результат — это значение или действие, которое будет выполнено, если предикат истинен.

Энтропия Шеннона — это мера неопределенности, связанной со случайной величиной, используется в алгоритмах построения решающих деревьев, где она помогает определить, какой признак лучше всего разделит выборку на более однородные подгруппы.

Обрезание ветвей (pruning) — это процесс удаления некоторых ветвей дерева решений, чтобы улучшить его производительность и эффективность. Целью обрезания является уменьшение размера дерева, улучшение его формы и структуры, а также повышение качества принимаемых им решений. Обрезание может быть выполнено вручную или автоматически с помощью специализированных инструментов

Бутстрэп-выборка в бэггинге (от англ. "bootstrap sampling") — это процесс случайного выбора подмножества данных из исходного набора с возвращением.

Как работают деревья решений

Деревья решений используются в различных областях, включая машинное обучение, бизнес-анализ и финансы. Они могут быть использованы для прогнозирования, классификации, кластеризации и оптимизации.

Дерево решений представляет собой графическую модель, которая используется для принятия решений на основе данных. Оно состоит из узлов (решений) и ветвей (альтернативных путей), которые определяют, какие данные будут использоваться для принятия решений.

Правила деревьев решений определяются на основе данных, используемых для обучения модели. Эти правила могут быть заданы вручную или автоматически найдены алгоритмом машинного обучения.

Формально, дерево решений представляет собой способ организации решающих правил в иерархической структуре, состоящей из узлов и листьев. Узлы содержат решающие правила и выполняют проверку соответствия примеров этим правилам по определенному атрибуту обучающего набора данных.

В самом простом случае, после проверки множество примеров, попавших в узел, разделяется на два подмножества. В одно из них попадают примеры, которые удовлетворяют правилу, а в другое - которые не удовлетворяют.

Затем к каждой группе снова применяется правило, и процесс рекурсивно повторяется до достижения определенного условия остановки алгоритма. В конечном итоге в последнем узле проверка и разделение не происходят, и он объявляется листом.

Каждый пример, попадающий в лист, определяет решение. То есть, лист в контексте деревьев решений относится к терминальным узлам, которые представляют конечные прогнозы или классификации. В деревьях решений листья содержат конечные значения целевой переменной и не разделяются на подузлы. В случае классификационного дерева это класс, связанный с узлом, а для регрессионного дерева — соответствующий листу модальный интервал целевой переменной. В отличие от узла, в листе содержится не правило, а подмножество объектов, которые удовлетворяют всем правилам ветви, оканчивающейся данным листом.

Для того чтобы попасть в лист, пример должен соответствовать всем правилам на пути к этому листу. Поскольку путь до каждого листа в дереве

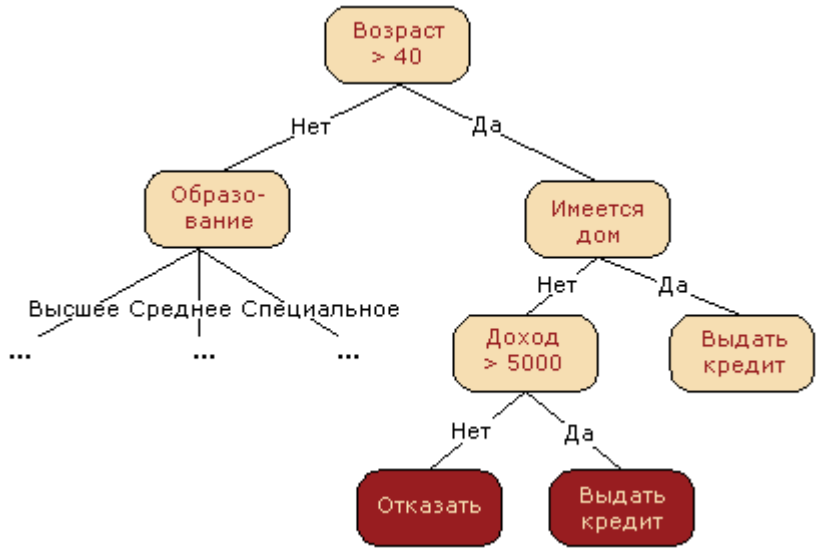
единственный, каждый пример может попасть только в один лист, что гарантирует единственность решения.

Решающее дерево предсказывает значение целевой переменной с помощью применения последовательности простых решающих правил (которые называются предикатами). Этот процесс в некотором смысле согласуется с естественным для человека процессом принятия решений. Хотя обобщающая способность решающих деревьев невысока, их предсказания вычисляются довольно просто, из-за чего решающие деревья часто используют как кирпичики для построения ансамблей — моделей, делающих предсказания на основе агрегации предсказаний других моделей.

Деревья решений находят применение в различных сферах человеческой деятельности, даже в тех, которые далеки от машинного обучения. Дерево решений — это наглядная инструкция, что делать в определенной ситуации. Например, в области консультирования научных сотрудников института Высшая Школа Экономики создает информационные схемы, которые облегчают жизнь своим сотрудникам. Вот пример фрагмента инструкции по публикации научной статьи на портале института.

С точки зрения машинного обучения, это простой классификатор, который определяет форму публикации на портале (книга, статья, глава книги, препринт, публикация в "НИУ ВШЭ и СМИ") на основе нескольких признаков: типа публикации

(монография, брошюра, статья и т.д.), типа издания, где статья опубликована (научный журнал, сборник трудов и т.д.) и других факторов.
Рассмотрим пример кредитного скоринга:



Дерево решений часто используется для обобщения опыта экспертов, передачи знаний будущим сотрудникам или моделирования бизнес-процессов компании. Например, до внедрения масштабируемых алгоритмов машинного обучения в банковской сфере, задача кредитного скоринга (Кредитный скоринг является методикой оценки платежеспособности заемщика, с целью уменьшения риска невозврата кредита) решалась экспертами. Решение о выдаче кредита заемщику принималось на основе некоторых интуитивно выведенных правил, которые можно представить в виде дерева решений.

В данном случае можно сказать, что решается задача бинарной классификации по признакам "Возраст", "Наличие дома", "Доход" и "Образование". Дерево решений как алгоритм машинного обучения является объединением логических правил вида "значение признака *a* меньше *x* и значение признака *b* меньше *y*... => Класс 1" в структуру данных "Дерево".

Одним из главных преимуществ деревьев решений является их простота в интерпретации и понимании для человека. Например, с помощью диаграммы на рисунке выше можно объяснить заемщику, почему ему было отказано в кредите. Например, из-за отсутствия у него дома и дохода, меньшего 5000. В отличие от других, более точных моделей, деревья решений не являются "черным ящиком", в который загружают данные и получают ответ. Благодаря своей "понятности" и сходству с человеческой моделью принятия решений (их легко объяснить начальнику), деревья решений стали очень популярными.

В процессе построения дерева решений выбирается первый признак, который будет использоваться для принятия решения. В случае с кредитным скорингом, возможными признаками могут быть возраст, наличие недвижимости, доход и другие. Однако, чтобы определить, какой признак выбрать первым, можно рассмотреть более простой пример, где все признаки являются бинарными.

Здесь можно вспомнить игру "20 вопросов", которая часто упоминается при изучении деревьев решений. Наверняка каждый из нас играл в нее. Один человек выбирает знаменитость, а второй пытается ее угадать, задавая только вопросы, на которые можно ответить "Да" или "Нет" (пропуская варианты "не знаю" и "не могу сказать").

Какой вопрос отгадывающий задаст в первую очередь? Естественно, тот, который сильнее всего уменьшит количество оставшихся вариантов. Например, вопрос "Это Анджелина Джоли?" при отрицательном ответе оставит более 7 миллиардов вариантов для дальнейшего перебора (конечно, поменьше, так как не каждый человек является знаменитостью, но все равно немало), а вот вопрос "Это женщина?" уже отсечет около половины знаменитостей. То есть, признак "пол" намного лучше разделяет выборку людей, чем признаки "это Анджелина Джоли", "национальность-испанец" или "любит футбол". Это интуитивно соответствует понятию прироста информации, основанного на энтропии.

Энтропия Шеннона — это мера неопределенности, связанной со случайной величиной, используется в алгоритмах построения решающих деревьев, где она помогает определить, какой признак лучше всего разделит выборку на более однородные подгруппы. (критерий основан на понятиях теории информации, а именно — информационной энтропии) определяется для системы с *N* возможными состояниями по формуле:

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i$$

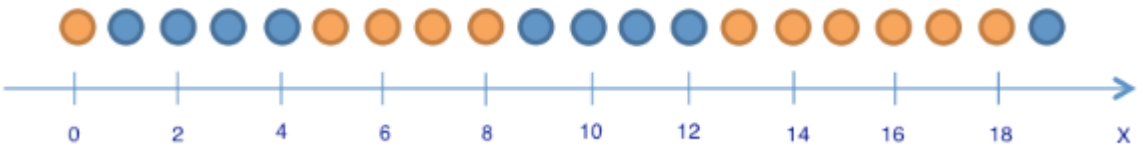
где pi — вероятности нахождения системы в i-ом состоянии. Это важное понятие широко используется в физике, теории информации и других областях. Без углубления в комбинаторные и теоретико-информационные предпосылки, отметим, что энтропия интуитивно соответствует уровню хаоса в системе. Чем выше энтропия, тем менее упорядочена система, и наоборот. Это понятие помогает нам

формализовать "эффективное разделение выборки", о котором мы говорили в контексте игры "20 вопросов".

🔥 В общем, можно сказать, что простая модель склонна к недообучению, а сложная модель - к переобучению.

Чем отличаются деревья решений регрессии и классификации

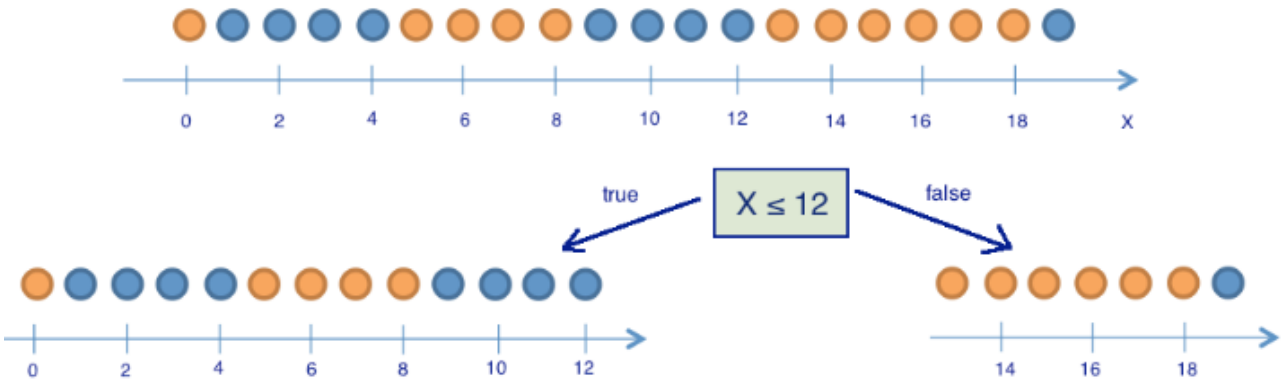
Для наглядного примера использования энтропии при выборе хороших признаков для построения дерева, приведем игрушечный пример. В данном примере мы будем предсказывать цвет шарика по его координате. Естественно, это не имеет ничего общего с реальной жизнью, однако это позволяет продемонстрировать, как энтропия используется при построении дерева решений.



Здесь 9 синих шариков и 11 желтых. Если мы наудачу вытащили шарик, то он с вероятностью $p\_1 = 9/20$ будет синим и с вероятностью $p\_2 = 11/20$ – желтым. Значит, энтропия состояния

$$S_0 = -\frac{9}{20}\log_2\frac{9}{20} - \frac{11}{20}\log_2\frac{11}{20} \approx 1.$$

Само это значение пока ни о чем нам не говорит. Теперь посмотрим, как изменится энтропия, если разбить шарики на две группы – с координатой меньше либо равной 12 и больше 12.



В левой группе было обнаружено 13 шаров, из которых 8 были синими и 5 желтыми. Энтропия этой группы составляет

$$S_1 = -\frac{5}{13}\log_2\frac{5}{13} - \frac{8}{13}\log_2\frac{8}{13} \approx 0.96.$$

В правой группе было обнаружено 7 шаров, из которых 1 был синим и 6 желтыми. Энтропия правой группы составляет

$$S_2 = -\frac{1}{7}\log_2\frac{1}{7} - \frac{6}{7}\log_2\frac{6}{7} \approx 0.6$$

Как видно, энтропия уменьшилась в обеих группах по сравнению с исходным состоянием, хотя в левой группе не сильно. Поскольку энтропия является мерой хаоса (или неопределенности) в системе, уменьшение энтропии называется приростом информации. Формально, прирост информации (information gain, IG) при разбиении выборки по признаку *Q* (в нашем примере это признак "x ≤ 12") определяется как

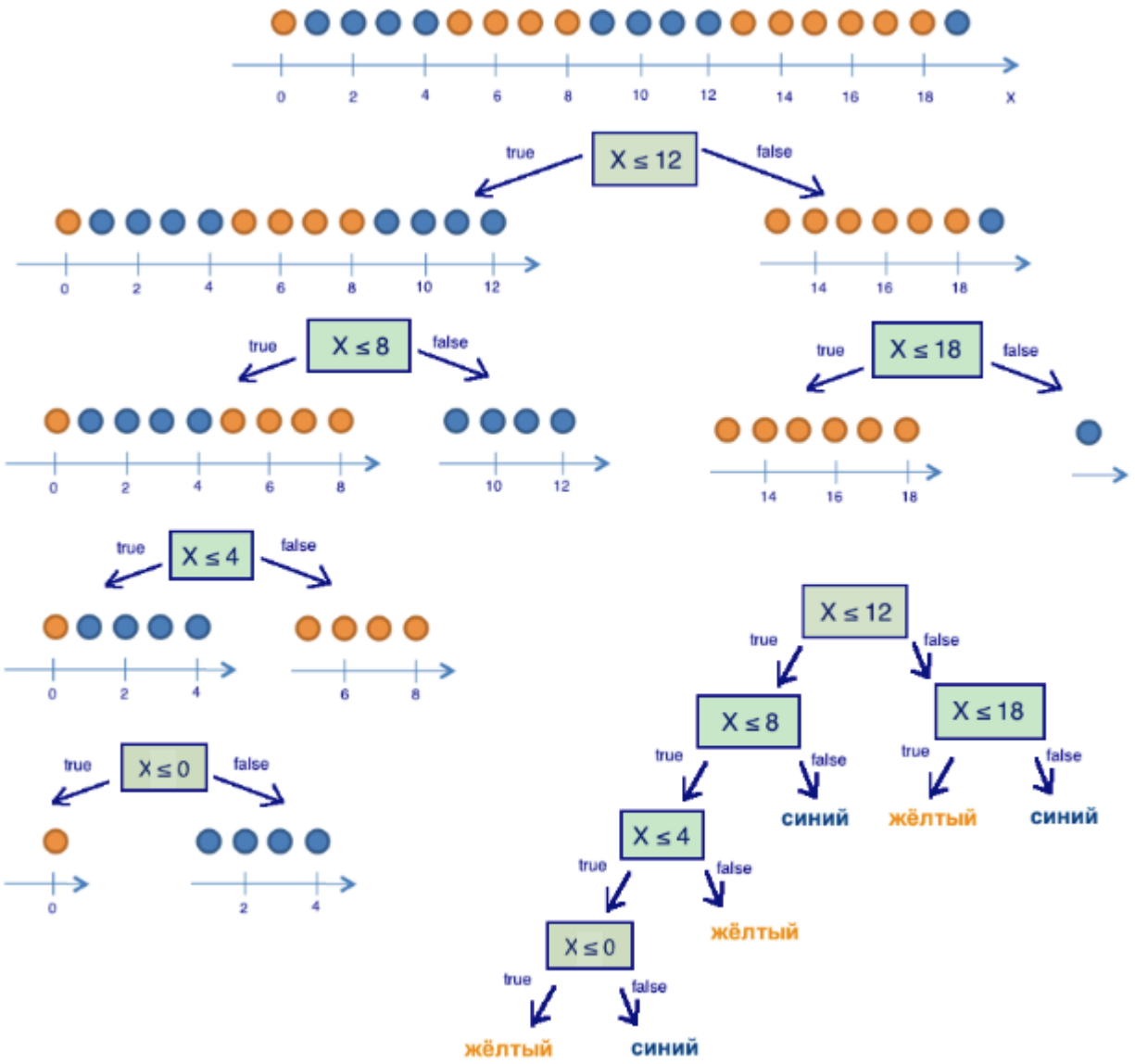
$$IG(Q) = S_O - \sum_{i=1}^{q}\frac{N_i}{N}S_i,$$

где *q* – число групп после разбиения,

Ni – число элементов выборки, у которых признак *Q* имеет *i*-ое значение.

В нашем случае после разделения получилось две группы (q = 2) – одна из 13 элементов (N_1 = 13), вторая – из 7 (N_2 = 7). Прирост информации получился

$$IG(x \le 12) = S_0 - \frac{13}{20}S_1 - \frac{7}{20}S_2 \approx 0.16.$$



Если мы разделим шарики на две группы в соответствии с условием "координата меньше или равна 12", то мы получим более упорядоченную систему, чем в начале. Мы можем продолжать делить шарики на группы до тех пор, пока в каждой группе не окажутся шарики одного цвета.

Для правой группы потребовалось всего одно дополнительное разделение по признаку "координата меньше или равна 18", а для левой группы – три разделения. Очевидно, что энтропия группы шариков одного цвета равна 0, что означает, что группа шариков одного цвета является упорядоченной.

Таким образом, мы создали дерево решений, которое предсказывает цвет шарика на основе его координаты. Следует отметить, что такое дерево решений может плохо работать для новых объектов (определения цвета новых шариков), так как оно идеально подстроилось под обучающую выборку (изначальные 20

шариков). Для классификации новых шариков лучше подойдет дерево с меньшим количеством "вопросов" или разделений, даже если оно не идеально разделяет обучающую выборку по цветам. Мы рассмотрим эту проблему, известную как переобучение, далее.

Можно убедиться в том, что дерево, построенное в предыдущем примере, является оптимальным в некотором смысле, так как для его создания потребовалось всего 5 "вопросов" о признаке *x*, чтобы сделать дерево подходящим для обучающей выборки и правильно классифицировать каждый обучающий объект. При других условиях разделения выборки дерево будет иметь большую глубину.

Алгоритмы построения дерева решений основаны на принципе жадной максимизации прироста информации. На каждом шаге выбирается признак, который обеспечивает наибольший прирост информации при разделении. Затем процедура повторяется рекурсивно до тех пор, пока энтропия не станет равной нулю или до достижения некоторого малого значения (чтобы избежать переобучения идеальной подгонки дерева к обучающей выборке). Разные алгоритмы используют различные эвристики для "ранней остановки" или "отсечения", чтобы избежать построения переобученного дерева.

Мы изучили, как понятие энтропии помогает определить качество разделения в дереве. Однако это лишь один из подходов, существуют и другие методы. Например, статистический подход, который использует Information gain (прирост информации). Information gain измеряет разницу между энтропией родительского узла и средней энтропией дочерних узлов, полученных при разбиении по данному атрибуту. Чем выше прирост информации, тем лучше атрибут разбиения.

Статистический подход

Вместо использования логарифма правдоподобия в качестве критерия классификации, можно выбрать метрику Бриера (Метрика Бриера - это метрика, используемая для оценки качества вероятностных прогнозов в задачах классификации. Она измеряет среднеквадратичное отклонение между прогнозами и фактическими значениями целевой переменной. Чем меньше значение метрики Бриера, тем лучше качество прогнозов. Она может быть особенно полезна в задачах, где важно точно оценивать вероятности принадлежности объектов к классам, например, в медицинских и финансовых приложениях. В отличие от логарифма правдоподобия, метрика Бриера не учитывает пороговое значение для

классификации объектов, что делает ее более устойчивой к несбалансированным классам.), которая представляет собой среднеквадратичную ошибку от вероятностей. Таким образом, информативность будет определяться следующим образом.

$$H(X_m) = \min_{\sum_k c_k = 1} \frac{1}{|X_m|} \sum_{(x_i, y_i) \in X_m} \sum_{k=1}^{K} (c_k - \mathbb{I}[y_i = k])^2$$

Можно показать, что оптимальное значение этой метрики достигается на векторе, состоящем из выборочных оценок частот классов. Если подставить этот вектор в выражение выше и упростить его, получим критерий Джини (не путать с индексом Джини — это способ измерить, насколько хорошо дерево решений разделяет данные на разные классы. Давай представим, что у нас есть много фруктов и мы хотим создать дерево решений, чтобы определить, является ли фрукт яблоком или апельсином.

Индекс Джини поможет нам понять, насколько хорошо дерево справляется с этой задачей. Он измеряет разнообразие фруктов в каждом классе. Если все фрукты в одном классе одинаковые, то индекс Джини будет равен 0. Если же фрукты равномерно распределены между классами, то индекс Джини будет равен 0.5. Чем ближе индекс Джини к 0, тем лучше дерево разделяет данные на классы. Таким образом, индекс Джини помогает нам понять, насколько эффективно дерево решений делает предсказания и как хорошо оно разделяет данные на разные классы.)

Давайте подведем итог, в чем разница между критерием и индексом Джинни:

Коэффициент Джинни — это статистический показатель, используемый для измерения степени неравенства в распределении какого-либо признака в обществе или регионе. Он изменяется от 0 до 1, где 0 означает полное равенство (идеальное равенство), а 1 - полное неравенство (идеальное неравенство). Чем ближе значение коэффициента Джини к 1, тем больше неравенство в распределении признака. В машинном обучении коэффициент Джини также используется для предсказания непрерывных величин, где его смысл заключается в том, чтобы погрешность была настолько равномерной, насколько возможно.

Критерий или индекс Джини, с другой стороны, используется в контексте построения решающих деревьев в машинном обучении. Он представляет собой меру неопределенности в узле дерева. Критерий Джини оценивает, насколько хорошо данный узел разделяет классы входных данных. При построении дерева решений используется критерий Джини для выбора оптимального разделения данных на каждом узле дерева.

Таким образом, коэффициент Джини и критерий Джини имеют разные применения и связаны с различными областями, хотя оба используются для измерения степени неравенства или неопределенности.

.

$$H(X_m) = \sum_{k=1}^{K} p_k(1 - p_k)$$

Критерий Джини также может быть интерпретирован как математическое ожидание числа неправильно классифицированных объектов в случае, если им будут присвоены случайные метки из дискретного распределения, заданного вероятностями.

Простыми словами, критерий Джинни — это способ измерить, насколько полезными являются определенные характеристики или признаки при решении задачи классификации. Когда мы хотим разделить объекты на группы, такие как "собаки" и "кошки", мы можем использовать разные признаки, например, цвет шерсти или размер тела. Критерий Джинни позволяет нам определить, какой из этих признаков наиболее полезен для разделения объектов на группы. Он говорит нам, насколько хорошо признак разделяет объекты, и чем он меньше, тем лучше. Таким образом, критерий Джинни помогает нам выбрать наиболее информативные признаки для решения задачи классификации.

Когда мы построим графики этих двух функций в зависимости от аргумента *p+* (вероятность объекта иметь метку +), мы заметим, что графики энтропии и удвоенной неопределенности Джини очень похожи друг на друга. Поэтому на практике эти два критерия почти равнозначны и дают схожие результаты.

Вот основные достоинства и недостатки индекса Джинни и энтропии Шеннона для измерения неравенства:

Плюсы индекса Джинни:

- Простота интерпретации — выражается как процент от максимально возможного неравенства.
- Удобство сравнения неравенства между разными распределениями.
- Чувствительность к изменениям в середине распределения.

Минусы индекса Джинни:

- Сложность вычисления по сравнению с другими мерами.
- Чувствительность к изменениям в хвостах распределения.

Плюсы энтропии Шеннона:

- Простота и удобство вычисления.
- Учитывает всё распределение целиком.
- Имеет естественную интерпретацию в теории информации.

Минусы энтропии Шеннона:

- Сложность интерпретации значений.
- Не имеет фиксированных границ изменения.
- Малая чувствительность к неравенству в середине распределения.

Таким образом, индекс Джинни и энтропия Шеннона дополняют друг друга и могут применяться совместно для более полной оценки неравенства.

При прогнозировании количественного признака идея построения дерева остается та же, но меняется критерий качества.

Шаги по поиску узла расщепления можно описать иначе:

1. Рассчитываем стандартное отклонение целевой переменной.
2. Разделяем набор данных на различные атрибуты и вычисляем стандартное отклонение для каждой ветви (для целевой переменной и предиктора).
3. Вычитаем это значение из стандартного отклонения перед разделением.
4. Результатом является уменьшение стандартного отклонения.
5. В качестве узла разделения выбираем атрибут с наибольшим уменьшением стандартного отклонения.
6. Набор данных разделяем на основе значений выбранного атрибута.
7. Этот процесс выполняется рекурсивно.

8. Чтобы избежать переобучения, используем коэффициент отклонения, который определяет, когда прекратить ветвление.

9. Наконец, каждой ветви присваивается среднее значение (в случае регрессии берется среднее значение).

Теперь хочется в целом понять, насколько данное разбиение помогает нам уменьшить ошибку, для этого нужно ввести понятие "прирост информации" (information gain). Он считается, как

$$IG = MSE_{root} - (\frac{n_{left}}{n} MSE_{left} + \frac{n_{right}}{n} MSE_{right})$$

где left - это количество объектов в левой ветке, right - это количество объектов в правой ветке, а n - количество объектов в корневом узле.

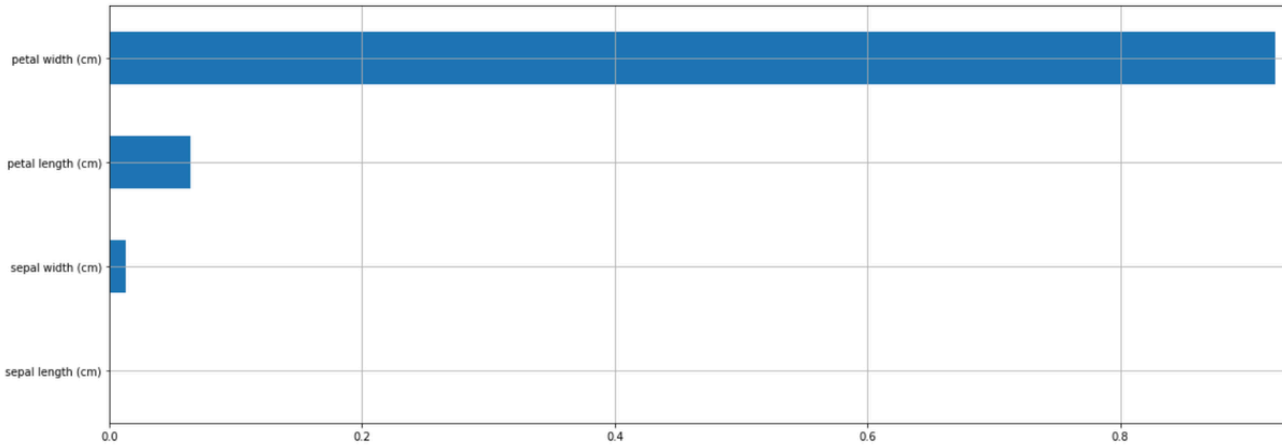
Также дерево решений позволяет выявить наиболее важные признаки:
Вот основные шаги процесса выделения наиболее важных признаков с помощью деревьев решений:

- Обучение дерева решений на имеющихся данных до максимально возможной глубины.
- Анализ полученного дерева — рассмотрение признаков, которые используются в корневых и приближенных к корневым вершинам. Эти признаки и есть наиболее важные.
- Можно посчитать значение критерия важности признаков, например, значение уменьшения энтропии или индекс Джини в узле. Чем выше значение для признака, тем он важнее.
- Визуализация важности признаков с помощью графиков. Например, с помощью диаграммы значимости признаков.
- Исключение наименее важных признаков и переобучение дерева. Оценка изменения качества модели.
- Оптимизация модели с помощью исключения и включения различных признаков, оценивая качество дерева.
- Анализ итогового набора наиболее важных признаков, выявленных с помощью дерева решений. Использование этих признаков в модели.

Этот процесс позволяет выделить действительно важные признаки и упростить модель за счёт исключения неинформативных признаков.

```
import pandas as pd
pd.Series(model.feature_importances_, index=model.feature_names_in_)\
            .sort_values(ascending=True).plot.barh(figsize=(20,7), grid=True)
```

```
<AxesSubplot:>
```



Критерий остановки алгоритма

Алгоритм обучения дерева решений будет работать до тех пор, пока не будут получены подмножества, в каждом из которых содержатся только примеры одного класса. Однако возможна ситуация, когда построится дерево, в котором для каждого примера будет создан отдельный лист. Это приведет к переобучению дерева, так как каждому примеру будет соответствовать уникальный путь и набор правил, актуальный только для этого примера.

Одним из возможных решений проблемы является принудительная остановка построения дерева до достижения переобучения. Для этого были разработаны следующие подходы.

Первый подход – ранняя остановка, при которой алгоритм будет прекращать работу, как только будет достигнуто заданное значение некоторого критерия, например, определенный процент правильно распознанных примеров. Единственным преимуществом данного подхода является сокращение времени обучения. Главным недостатком является то, что ранняя остановка всегда приводит к снижению точности дерева, поэтому многие авторы рекомендуют отдавать предпочтение отсечению ветвей.

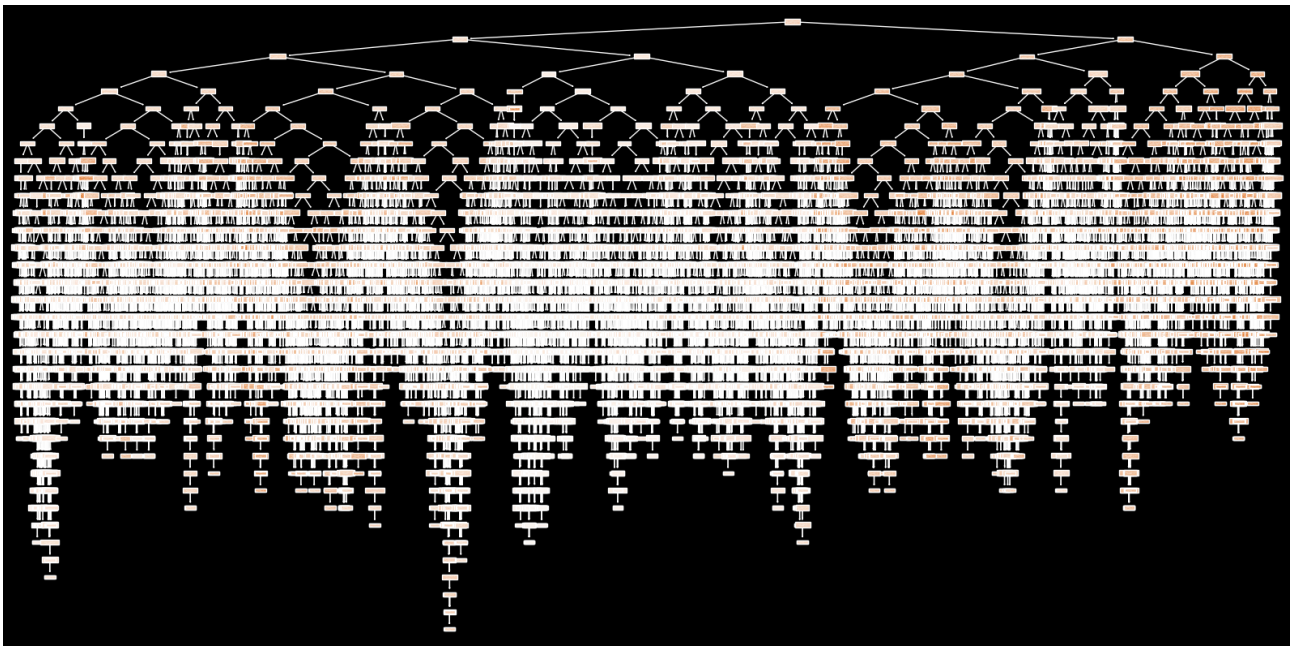
Второй подход — ограничение глубины дерева — определение максимального количества разбиений в ветвях, при достижении которого обучение прекращается. Этот метод также влияет на уменьшение точности дерева.

Третий подход — задание минимально допустимого количества примеров в узле — запрет алгоритму создавать узлы с количеством примеров меньше

заданного (например, 5). Это позволяет избежать создания тривиальных разбиений и, соответственно, незначимых правил.

Все указанные подходы являются эвристическими, что означает, что они не гарантируют оптимального результата или даже работают только в определенных случаях. Поэтому к их использованию следует подходить осторожно. В настоящее время нет обоснованных рекомендаций относительно того, какой метод работает лучше. Поэтому аналитикам приходится применять метод проб и ошибок.

Как уже было отмечено, если не ограничить рост дерева, то получится сложное дерево с большим количеством узлов и листьев, которое будет сложно интерпретировать. Также, решающие правила в таких деревьях, которые создают узлы, куда попадают всего два-три примера, оказываются малозначимыми с практической точки зрения.



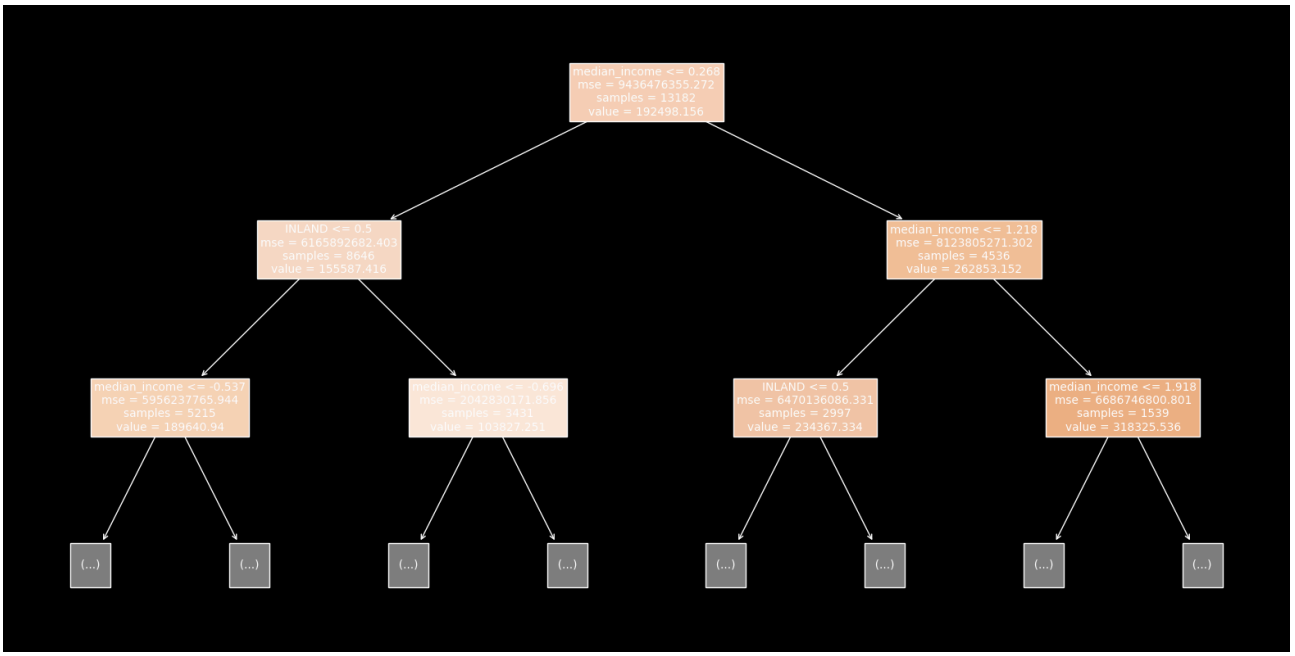
Более предпочтительным является иметь дерево с малым количеством узлов, которым соответствует большое число примеров обучающей выборки. Поэтому интересен подход, отличный от ранней остановки: построить все возможные деревья и выбрать то, которое при разумной глубине обеспечивает приемлемый уровень ошибки распознавания. То есть найти оптимальный баланс между сложностью и точностью дерева.

Другим способом является так называемое обрезание ветвей (pruning). Обрезание ветвей (pruning) — это процесс удаления некоторых ветвей дерева решений, чтобы улучшить его производительность и эффективность. Целью обрезания является уменьшение размера дерева, улучшение его формы и структуры, а также повышение качества принимаемых им решений. Обрезание может быть выполнено вручную или автоматически с помощью специализированных инструментов. Этот подход включает следующие шаги:

1. Построить полное дерево, где все листья содержат примеры одного класса.

2. Определить два показателя: относительную точность модели - отношение числа правильно распознанных примеров к общему числу примеров, и абсолютную ошибку - число неправильно классифицированных примеров.

3. Удалить из дерева листья и узлы, обрезание которых не приведет к значительному уменьшению точности модели или увеличению ошибки.



Обрезание ветвей производится в направлении, противоположном направлению роста дерева, то есть снизу вверх, путем последовательного преобразования узлов в листья. Преимущество обрезания ветвей по сравнению с ранней остановкой заключается в возможности нахождения оптимального соотношения между точностью и понятностью дерева. Однако недостатком является более длительное время обучения из-за необходимости сначала построить полное дерево.

Иногда, даже упрощенное дерево решений все еще слишком сложно для понимания и визуального восприятия. В таких случаях полезно извлечь правила принятия решений из дерева и организовать их в группы, описывающие классы.

Для извлечения правил необходимо проследить все пути от корневого узла до листьев дерева. Каждый путь дает правило, которое состоит из набора условий, проверяемых на каждом узле пути.

Представление сложных деревьев решений в виде правил принятия решений, а не иерархической структуры из узлов и листьев, может быть более удобным для визуального восприятия.

Итак вот несколько практических рекомендаций по обрезке и оптимизации деревьев решений:

Начинать с простого дерева и небольшой глубины, постепенно увеличивая глубину и оценивая качество.

Использовать кросс-валидацию для оценки - она поможет выявить оптимальные параметры и избежать переобучения.

Применять прореживание и раннее прекращение роста дерева, оценивая улучшение качества.

Удалять малоинформативные узлы постепенно, наблюдая за изменением метрик качества.

Сравнивать качество деревьев до и после обрезки на отложенной выборке.

Использовать алгоритмы вроде Reduced Error Pruning для автоматизации обрезки. Дерево обучается до максимальной глубины на обучающей выборке. Затем дерево просматривается снизу вверх. Для каждого узла оценивается, будет ли ошибка на проверочной выборке меньше, если заменить поддерево, начинающееся в этом узле, на просто лист (предсказание большинства). Если ошибка меньше - поддерево заменяется листом и обрезается. Процесс повторяется для каждого узла дерева снизу вверх. В итоге получается оптимизированное обрезанное дерево без переобучения.

Проверять влияние обрезки на баланс классов и ошибки на разных классах.

Не обрезать дерево слишком сильно, чтобы не потерять важные зависимости.

Документировать проведенную обрезку и сохранять параметры оптимальной модели.

Правильный баланс параметров и методов обрезки приходит с опытом работы с конкретными данными и задачами.

При рассмотрении основных проблем, возникающих при построении деревьев, необходимо упомянуть их преимущества:

1. Быстрый процесс обучения.

2. Генерация правил в областях, где эксперту трудно формализовать свои знания.

3. Извлечение правил на естественном языке.

4. Интуитивно понятная классификационная модель.

5. Высокая точность предсказания, сопоставимая с другими методами анализа данных (статистика, нейронные сети).

6. Построение непараметрических моделей.

Из-за этих и других причин деревья решений являются важным инструментом для работы каждого специалиста, занимающегося анализом данных.

Ансамблирование

Ансамблевые методы — это мощный инструмент для построения моделей машинного обучения.

Ансамблевый метод — это метод машинного обучения, в котором несколько моделей обучаются для решения одной и той же проблемы, а затем объединяются для получения лучших результатов. Основная идея заключается в том, что результат, полученный несколькими моделями, будет более точным, чем результат только одной модели.

При использовании ансамблей, обычно используются слабые ученики (например, линейная регрессия или дерево решений). Слабые ученики в ансамблевом машинном обучении представляют собой базовые модели, которые обладают невысокой предсказательной способностью по сравнению с более сложными моделями. Множество слабых учеников служит строительными блоками для создания более сложных моделей. Процесс объединения слабых учеников для улучшения качества модели, снижения смещения или разброса, называется созданием сильного ученика.

Виды ансамблевых методов могут быть классифицированы следующим образом: стекинг, бэггинг и бустинг.

Стекинг предполагает использование нескольких различных слабых учеников, которые обучаются и объединяются для создания прогноза, основанного на результатах каждой из слабых моделей.

Бэггинг, в свою очередь, представляет собой метод объединения однородных моделей, которые обучаются на различных наборах данных. Предсказание получается путем усреднения результатов. Если использовать в качестве слабого ученика деревья решений, то получится случайный лес RandomForestClassifier / RandomForestRegressor.

Бустинг предполагает последовательное обучение нескольких однородных моделей, каждая из которых исправляет ошибки предыдущей модели.

Стекинг — это метод работы с ансамблями, который представляет собой модификацию обычного ансамбля. В этом методе все слабые прогнозаторы получают на вход обучающий набор данных, и каждый из них создает свой собственный прогноз. Затем все эти прогнозы передаются в финальную модель, которая называется смесителем, мета-учеником или мета-моделью, и она вырабатывает окончательный прогноз. При обучении мета-модели используется метод удерживания выборки. Сначала обучающий набор данных разделяется на две части. Слабые ученики обучаются на первой части обучающего набора, а затем на второй. Затем создается новый обучающий набор данных на основе прогнозов, сделанных на первой и второй частях набора. Таким образом, на каждый образец из входного набора данных приходится столько прогнозов, сколько слабых учеников в

ансамбле (в данном примере — три). Мета-модель обучается прогнозировать значения на основе этого нового набора данных.

В отличие от использования только алгоритмов из фиксированного семейства, он может применять алгоритмы различных типов. Например, в качестве базовых алгоритмов могут быть использованы дерево решений и линейная регрессия. Результаты работы базовых алгоритмов объединяются с помощью обучаемой мета-модели, а не с помощью обычных методов агрегации, таких как суммирование или усреднение. Процесс обучения стекинга проходит через несколько этапов: общая выборка делится на тренировочную и тестовую, тренировочная выборка разбивается на *n* фолдов (подгруппы для обучения и тестирования модели). Затем эти фолды перебираются в соответствии с методом кросс-валидации: на каждом шаге *n-1* фолда используются для обучения базовых алгоритмов, а один фолд — для предсказаний (вычисления мета-факторов). Такой подход позволяет использовать всю тренировочную выборку, при этом избегая переобучения базовых алгоритмов.

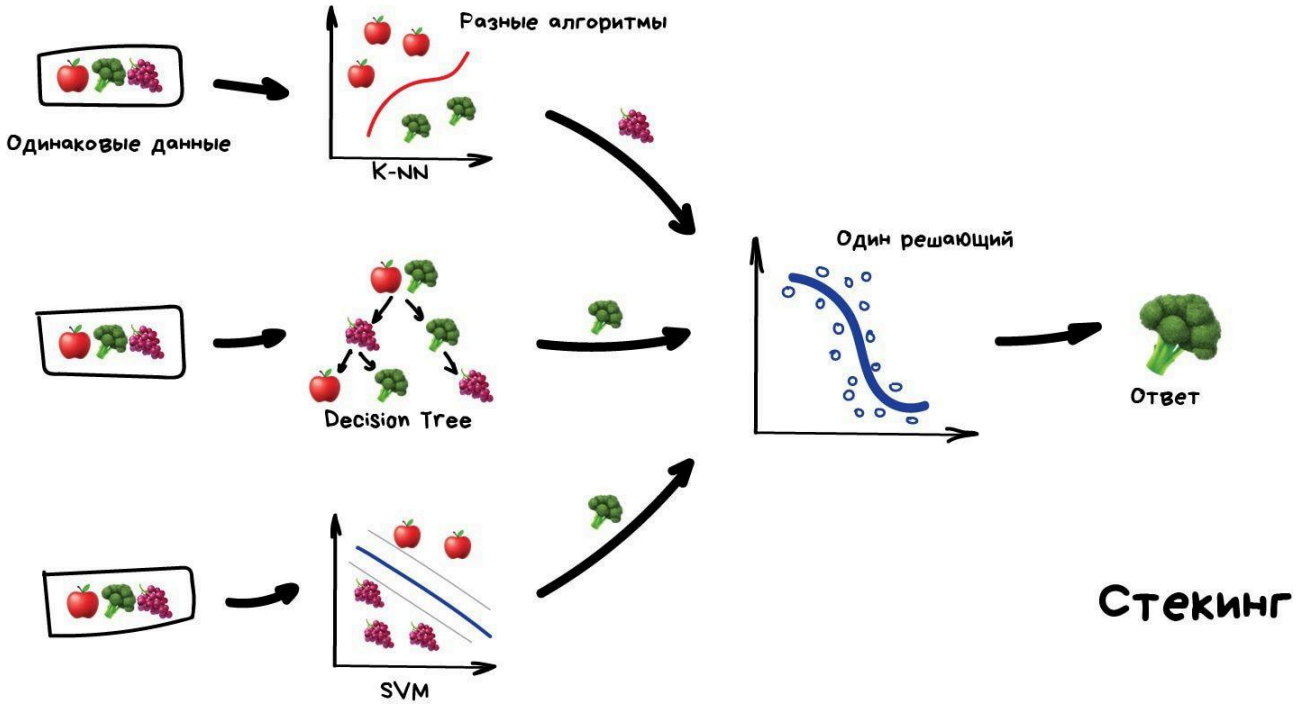
Мета-модель обучается на основе полученных мета-факторов. Она также может использовать фичи (признаки от английского “features”) из исходного датасета, в зависимости от поставленной задачи.

Для получения мета-факторов на тестовом наборе данных, базовые алгоритмы могут быть обучены на всем тренировочном наборе данных - в данном случае не должно возникнуть переобучение.

Если у нас есть большой объем данных, мы можем просто разделить их на две части: одну для обучения базовых алгоритмов и другую для работы мета-модели, которая делает предсказания и обучается. Этот подход называется "блендингом" и иногда используется вместо кросс-валидации (Кросс-валидация — это метод, используемый для оценки качества работы модели машинного обучения и ее способности обобщать данные на новых данных. В процессе кросс-валидации исходный набор данных разделяется на обучающую и валидационную выборки.

Затем модель обучается на обучающей выборке и тестируется на валидационной. Этот процесс повторяется несколько раз, при этом каждый раз выбирается новая часть данных для валидации. Оценка качества модели производится на основе среднего результата по всем итерациям. Кросс-валидация позволяет получить более достоверную оценку выходной ошибки модели и уменьшить вероятность переобучения.) на обучающих данных. Если у нас очень много данных, мы также можем разделить тестовый набор на две части: тестовую и валидационную, и использовать последнюю для подбора гиперпараметров моделей-участников.

С точки зрения смещения и разброса, стекинг не имеет прямой интерпретации, так как не минимизирует эти компоненты ошибки напрямую. Однако, успешный стекинг приводит к уменьшению ошибки в целом, и, следовательно, их компоненты также будут снижаться.



Бэггинг — это метод обучения моделей, основная идея которого заключается в том, чтобы обучить несколько одинаковых моделей на различных подвыборках данных. Поскольку распределение выборки неизвестно, модели получаются разными.

Итак, давайте введем понятие бутстрэп-выборка в бэггинге (от англ. "bootstrap sampling") - это процесс случайного выбора подмножества данных из исходного набора с возвращением. Это означает, что одни и те же данные могут быть выбраны несколько раз, а другие вообще не попасть в выборку. Такой подход позволяет создавать множество подвыборок, которые затем используются для обучения отдельных моделей в бэггинге. Благодаря этому уменьшается переобучение и повышается обобщающая способность модели.

Сначала генерируется несколько бутстрэп-выборок, то есть случайно выбираются данные из исходного датасета и представляются модели. Затем данные возвращаются в исходный датасет и процесс повторяется. После этого модели делают прогнозы на основе бутстрэп-выборок. В случае регрессии прогнозы просто усредняются, а в случае классификации применяется голосование.

Если большинство слабых моделей вносят свой прогноз, то этот класс получает больше голосов и становится результатом прогнозирования ансамбля. Это является примером жесткого голосования. В случае мягкого голосования рассматриваются вероятности прогнозирования каждого класса, затем эти вероятности усредняются, и результатом является класс с наибольшей вероятностью.

В предыдущем разделе мы предположили, что базовые алгоритмы в ансамбле некоррелированы, что позволяет значительно снизить дисперсию ансамбля по сравнению с входящими в него базовыми алгоритмами. Однако в реальной жизни достичь полной некоррелированности сложно, так как базовые алгоритмы обучаются на пересекающихся выборках, что означает, что корреляция не может быть равна нулю. Но на практике оказывается, что необходимо только некоторое отличие базовых алгоритмов друг от друга. Именно на этом принципе основано развитие идеи бэггинга для решающих деревьев - случайный лес.

Давайте построим ансамбль алгоритмов, где базовым алгоритмом будет решающее дерево. Мы будем следовать следующей схеме:

Для построения i-го дерева:

а. Сначала, как в обычном бэггинге, из обучающей выборки X выбирается с возвращением случайная подвыборка X того же размера, что и X.

б. В процессе обучения каждого дерева в каждой вершине случайно выбираются $n<N$ признаков, где N — полное число признаков (метод случайных подпространств), и среди них ищется оптимальный сплит.

Для получения прогноза ансамбля на тестовом объекте мы можем усреднить отдельные ответы деревьев (в случае регрессии) или выбрать самый популярный класс (в случае классификации).

Кстати, мы создали случайный лес — комбинацию бэггинга и метода случайных подпространств для решающих деревьев.

Возможно, заметили, что при создании случайного леса у специалиста по машинному обучению есть несколько степеней свободы. Давайте рассмотрим их подробнее.

Какую глубину должны иметь деревья в случайном лесу?

В случае ошибки модели (на которую мы можем повлиять) имеется смещение и разброс. Разброс мы уменьшаем с помощью процедуры бэггинга. Бэггинг не влияет на смещение, но хотелось бы, чтобы у леса оно было небольшим. Поэтому смещение должно быть небольшим у самих деревьев, из которых строится ансамбль. У неглубоких деревьев мало параметров, что означает, что дерево может запомнить только верхнеуровневую статистику обучающей подвыборки. Все подвыборки будут похожи, но не будут очень подробно описывать целевую зависимость.

Поэтому, когда меняется обучающая подвыборка, предсказание на тестовом объекте остается стабильным, но не точным (низкая дисперсия, высокое смещение). В отличие от этого, у глубоких деревьев нет проблемы запомнить подвыборку подробно. Поэтому предсказание на тестовом объекте будет сильно меняться в зависимости от обучающей подвыборки, но в среднем будет близко к истине (высокая дисперсия, низкое смещение). Вывод: рекомендуется использовать глубокие деревья.

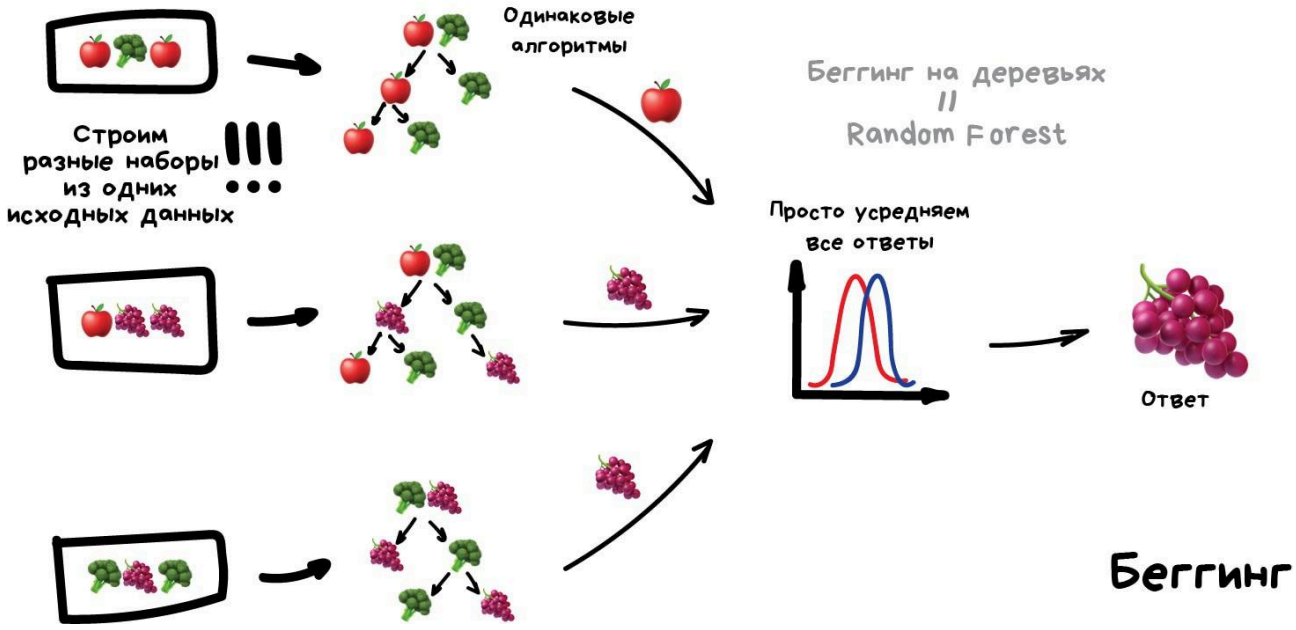
Какое количество признаков нужно использовать для обучения дерева?

Ограничивая количество признаков, используемых в обучении одного дерева, мы также контролируем качество случайного леса (RandomForestClassifier / RandomForestRegressor). Чем больше признаков, тем больше корреляция между деревьями и меньше эффект от ансамблирования. Чем меньше признаков, тем слабее сами деревья. Практическое рекомендация - брать квадратный корень из общего числа признаков для классификации и треть от общего числа признаков для регрессии.

Какое количество деревьев должно быть в случайном лесу?

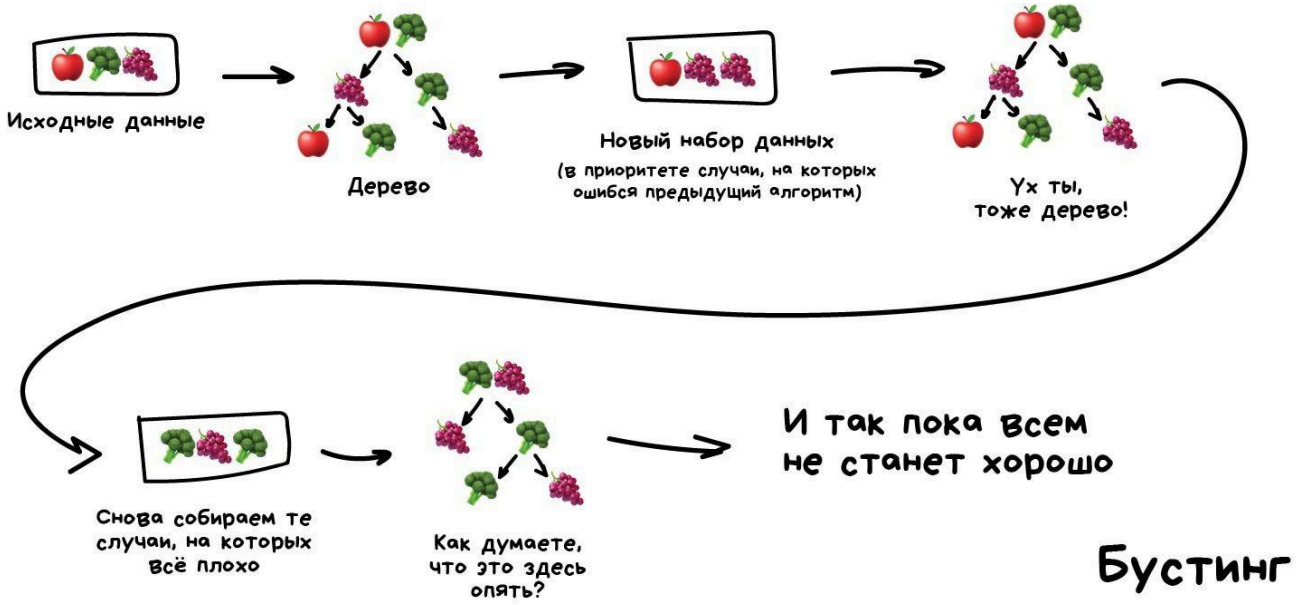
Как уже было показано, увеличение количества базовых алгоритмов в ансамбле не влияет на смещение и уменьшает разброс. Однако, так как количество признаков и вариантов подвыборок, на которых строятся деревья в случайном лесу, ограничены, невозможно бесконечно уменьшить разброс. Поэтому имеет смысл построить график ошибки от количества деревьев и ограничить размер леса в тот момент, когда ошибка перестанет существенно уменьшаться.

Вторым практическим ограничением на количество деревьев может быть время работы ансамбля. Однако у случайного леса есть положительное свойство: его можно строить и применять параллельно, что позволяет сократить время работы, если у нас есть несколько процессоров. Однако число процессоров, скорее всего, значительно меньше числа деревьев, а сами деревья обычно глубокие. Поэтому при большом числе деревьев Random Forest может работать дольше, чем мы хотели бы, и количество деревьев можно уменьшить, немного уступив в качестве.



Бустинг — метод, который отличается от метода бэггинга тем, что модели адаптируются к данным последовательно и исправляют ошибки предыдущих

моделей. В основе бустинга лежат модели с низким разбросом и высоким смещением, такие как неглубокие деревья решений. Одна из причин выбора таких моделей заключается в их меньших вычислительных затратах. Кроме того, бустинг не может быть распараллелен, в отличие от бэггинга.



Существует два наиболее распространённых алгоритма бустинга — адаптивный бустинг (Алгоритм AdaBoost) и градиентный бустинг.

Алгоритм AdaBoost начинает с обучения первой базовой модели (например, дерева решений) на тренировочном наборе данных. Веса неправильно предсказанных значений увеличиваются. Затем второй базовой модели подаются обновленные веса, и она также обучается. После этого генерируются прогнозы, и цикл повторяется.

Результат работы AdaBoost - это сумма каждой модели, взвешенная по среднему значению. Прогнозируемым значением ансамбля будет то, которое получает большинство взвешенных голосов.

Алгоритм Adaboost изменяет веса объектов на каждой итерации. Веса правильно классифицированных объектов уменьшаются, в то время как веса неправильно классифицированных объектов увеличиваются. В конечной модели ансамбля модели, которые показывают лучшие результаты, имеют больший вес.

При использовании адаптивного бустинга применяется итеративный метод. Мы добавляем слабых учеников одного за другим и на каждой итерации ищем наилучшую пару (коэффициент, слабый ученик) для добавления к текущей модели ансамбля. Веса объектов изменяются в процессе. Этот метод работает быстрее, чем аналитический метод.

Градиентный бустинг обучает слабые модели последовательно, исправляя ошибки предыдущих моделей. Результатом градиентного бустинга является сумма

результатов моделей, взвешенных по среднему. Отличие градиентного бустинга от Adaboost заключается в способе изменения весов. В градиентном бустинге используется градиентный спуск для оптимизации моделей. Таким образом, градиентный бустинг является обобщением адаптивного бустинга для дифференцируемых функций.

Ансамблевые модели представляют собой эффективный инструмент для анализа данных. Идея объединения простых моделей позволяет достигать более точных прогнозов. Однако для создания прогнозов необходимо вначале использовать более простые модели, чтобы достичь требуемой точности. Если результаты не удовлетворяют вас, то можно применять ансамбли.

Итоги:

1. Деревья решений — это графическая модель, представляющая собой древовидную структуру, используемую для принятия решений. Они основаны на разделении данных на подмножества на основе значимых атрибутов или признаков.

2. Ансамбли деревьев — это комбинирование нескольких деревьев решений в одну модель для улучшения точности и стабильности предсказаний. Примерами ансамблей являются случайный лес (Random Forest) и градиентный бустинг (Gradient Boosting).

3. Практическая польза данных алгоритмов заключается в их способности моделировать сложные отношения между признаками и целевой переменной. Они могут обработать данные различного типа (категориальные, числовые) и могут быть использованы как для задач классификации, так и для задач регрессии.

4. Плюсы деревьев решений и ансамблей:
- Интуитивно понятны и легко интерпретируемы;
- Могут обрабатывать данные с пропущенными значениями и выбросами;
- Могут работать с различными типами переменных;
- Могут автоматически выявлять важность признаков;
- Могут обобщать данные и делать предсказания.

5. Однако у этих алгоритмов также есть свои минусы:
- Могут быть склонны к переобучению, особенно в случае сложных структур данных;
- Могут иметь проблему с построением оптимальной структуры дерева, особенно в случае большого количества признаков;
- Могут обладать слабой устойчивостью к изменениям в данных.

6. При использовании данных алгоритмов следует обращать внимание на:
- Выбор оптимальных гиперпараметров модели;
- Контроль переобучения, например, с помощью кросс-валидации;
- Анализ важности признаков и выбор наиболее значимых;
- Обработку пропущенных значений и выбросов в данных;
- Объяснение и интерпретацию результатов моделирования для принятия решений.

В целом, деревья решений и ансамбли деревьев являются мощными инструментами машинного обучения, которые широко применяются в практике для решения различных задач прогнозирования и классификации. Анализ и обработка данных перед использованием их алгоритмов, а также контроль переобучения и объяснение результатов — важные шаги при работе с ними.

Что можно почитать еще?

1. Ансамбли
2. Ансамблевые методы
3. Ансамбли в машинном обучение

Используемая литература

1. https://github.com/esokolov/ml-course-hse/blob/master/2020-fall/lecture-notes/lecture08-ensembles.pdf
2. https://alexanderdyakonov.wordpress.com/2019/04/19/ансамбли-в-машинном-обучении/
3. https://nagornyy.me/it/postroenie-ansamblei-modelei/