

Лекция 5. Деревья решений. Композиции деревьев.
Случайный лес.

Глинский А.В.

Содержание

0.1	Деревья решений	3
0.1.1	Описание алгоритма решающего дерева	3
0.1.2	Критерии информативности	4
0.1.3	Процесс построения дерева для классификации и регрессии	7
0.1.4	Регуляризация деревьев решений	9
0.1.5	Вопросы для самопроверки	10
0.1.6	Резюме по разделу	11
0.2	Композиции деревьев	12
0.2.1	Подходы к построению композиций	12
0.2.2	Бэггинг	13
0.2.3	Смещение и разброс	14
0.2.4	Вопросы для самопроверки	16
0.2.5	Резюме по разделу	17
0.3	Случайный лес	18
0.3.1	Визуальная демонстрация алгоритма	18
0.3.2	Подготовка данных для алгоритма	18
0.3.3	Процесс обучения	19
0.3.4	Оценка качества алгоритма	20
0.3.5	Интерпретация признаков с помощью алгоритма	20
0.3.6	Процесс применения	20
0.3.7	Область применения алгоритма	21
0.3.8	Плюсы и минусы алгоритма	21
0.3.9	Модификации алгоритма	22
0.3.10	Реализация алгоритма в Python	22
0.3.11	Вопросы для самопроверки	23
0.3.12	Резюме по разделу	24
0.4	Резюме по модулю	24

0.1 Деревья решений

Цель занятия: В этом занятии мы познакомимся с деревьями решений. После прохождения занятия ученики смогут понимать основные принципы работы моделей на основе решающих деревьев.

Ранее мы рассматривали метрические (Метод ближайших соседей) и линейные (линейная регрессия, логистическая регрессия, SVM) модели. Теперь давайте поговорим о новом классе моделей, которые хорошо справляются с нелинейными зависимостями — это решающие деревья.

Метрические и линейные модели предполагают линейную независимость признаков, что далеко не всегда хорошо. Например, в задаче предсказания цены на недвижимость увеличение расстояния от метро далеко не всегда будет означать уменьшение стоимости квадратного метра: если мы рассмотрим дополнительный признак, площадь объекта недвижимости, то для некоторых объектов увеличение расстояния до метро и увеличение площади объекта будет означать, что этот объект относится к загородной недвижимости, соответственно, цена на квадратный метр в такой недвижимости может и повыситься. В метрических и линейных моделях можно попытаться строить новые признаки на основе комбинаций двух или нескольких признаков, однако это влечет снижение интерпретируемости модели и повышение ее вычислительной сложности. Поэтому используется другой подход — алгоритм решающего дерева, который во многом похож на процедуру принятия решения человеком-экспертом. Рассмотрим этот подход подробнее.

0.1.1 Описание алгоритма решающего дерева

Решающее дерево - это древовидная модель машинного обучения, которая представляет собой дерево, где каждый узел представляет тест на значение признака, а ветви представляют возможный результат этого теста. Для теста, какой путь (ребро) в дереве нужно выбрать, используются логические правила. Логические правила (также их называют предикатами или индикаторами) в решающих деревьях - это условия, которые определяют, какие действия должны быть выполнены на каждом узле дерева. Каждое правило состоит из предиката, который представляет собой условие на признаки и соответствующей операции принятия решения. Например, для классификации объектов на основе признаков, правило может быть записано как «Если значение признака X_1 меньше 5, а значение признака X_2 больше 10, то идем в левую ветвь/объект принадлежит классу А, иначе идем в правую ветвь/объект принадлежит классу В». В данном случае предикатом является «значение признака X_1 меньше 5, а значение признака X_2 больше 10», а операция принятия решения - «объект принадлежит классу А». Логические правила в решающих деревьях могут с одним условием или содержать несколько условий. Они помогают определить, какой путь в дереве следует выбрать для получения правильного результата. Условие на предикат можно записать так:

$$[x_i \leq t],$$

где t - порог по определенному признаку.

В целом, ничего не мешает использовать в предикатах сложные модели и условия, но в этом обычно нет необходимости, и в качестве предиката берут сравнение значения по одному определенному признаку с порогом.

Давайте опишем устройство решающего дерева на примере наглядного набора данных в виде двух подков (Рисунок 1). Самая верхняя вершина называется корневой. Другие вер-

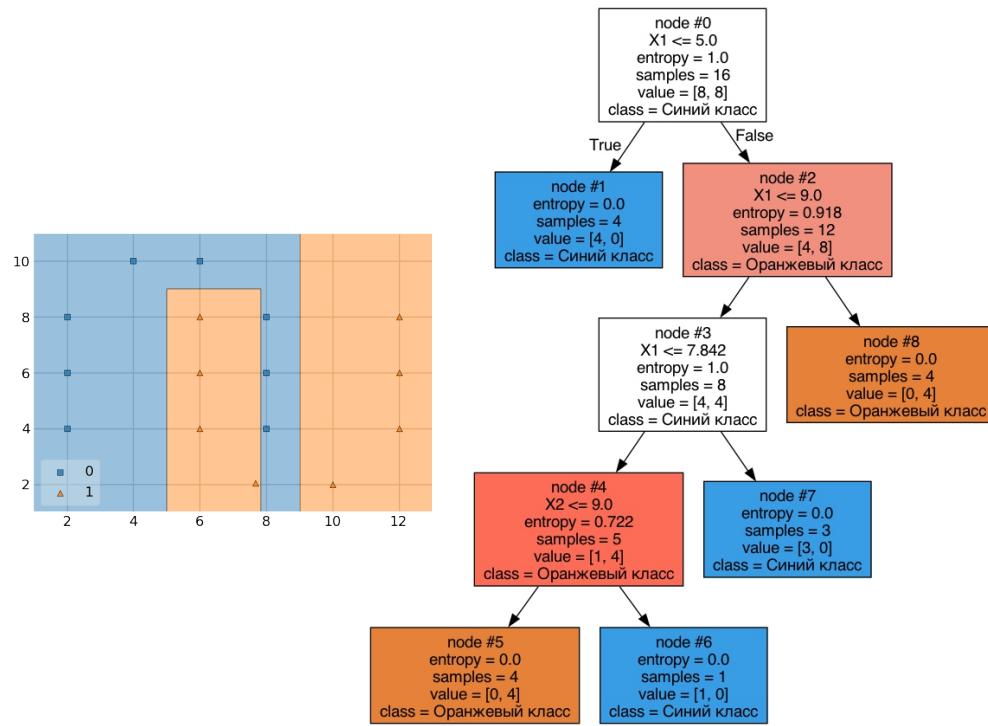


Рисунок 1: Построение решающего дерева на примере набора данных в виде двух подков

шины с предикатами называются внутренними. Логическое правило для корневой вершины формулируется так: « $X1 \leq 5.0$ ». Всего в этой вершине рассматривается 16 объектов ($\text{samples} = 16$), поделенных на 2 класса ($[8, 8]$). Если предиктор « $X1 \leq 5.0$ » для объекта равен True, то объект отправляется в левую вершину, иначе — в правую. После обработки объектов в левой вершине станет 4 объекта класса «Синий класс». Левая вершина называется листовой (для неё уже нет предикторов, только итоговый прогноз для части выборки). В дереве решений это будет тупиковая, конечная ветка, которая не ведёт к новым условиям — она выглядит как лист на дереве и поэтому зовётся листовой. В листовой вершине присваивается метка самого популярного класса в случае классификации или вероятность класса (равная доле объектов класса в вершине), либо выдается константный прогноз (усредненный по объектам из обучающей выборки, попавшим в данную вершину) в случае регрессии.

Для правого поддерева процесс деления выборки с помощью предиктов продолжится, пока все вершины не станут листовыми. Построение разделяющей поверхности для данного примера показано на рисунке 2. По сути, решающее дерево разбило все пространство признаков на прямоугольные области, в каждой определенной области объекту присваивается определенный класс.

0.1.2 Критерии информативности

Очень важным является то, как выбираются предикаты. Для этого используются критерии информативности. Критерии информативности — это метрики, которые используются при построении моделей машинного обучения для оценки качества разделения данных на

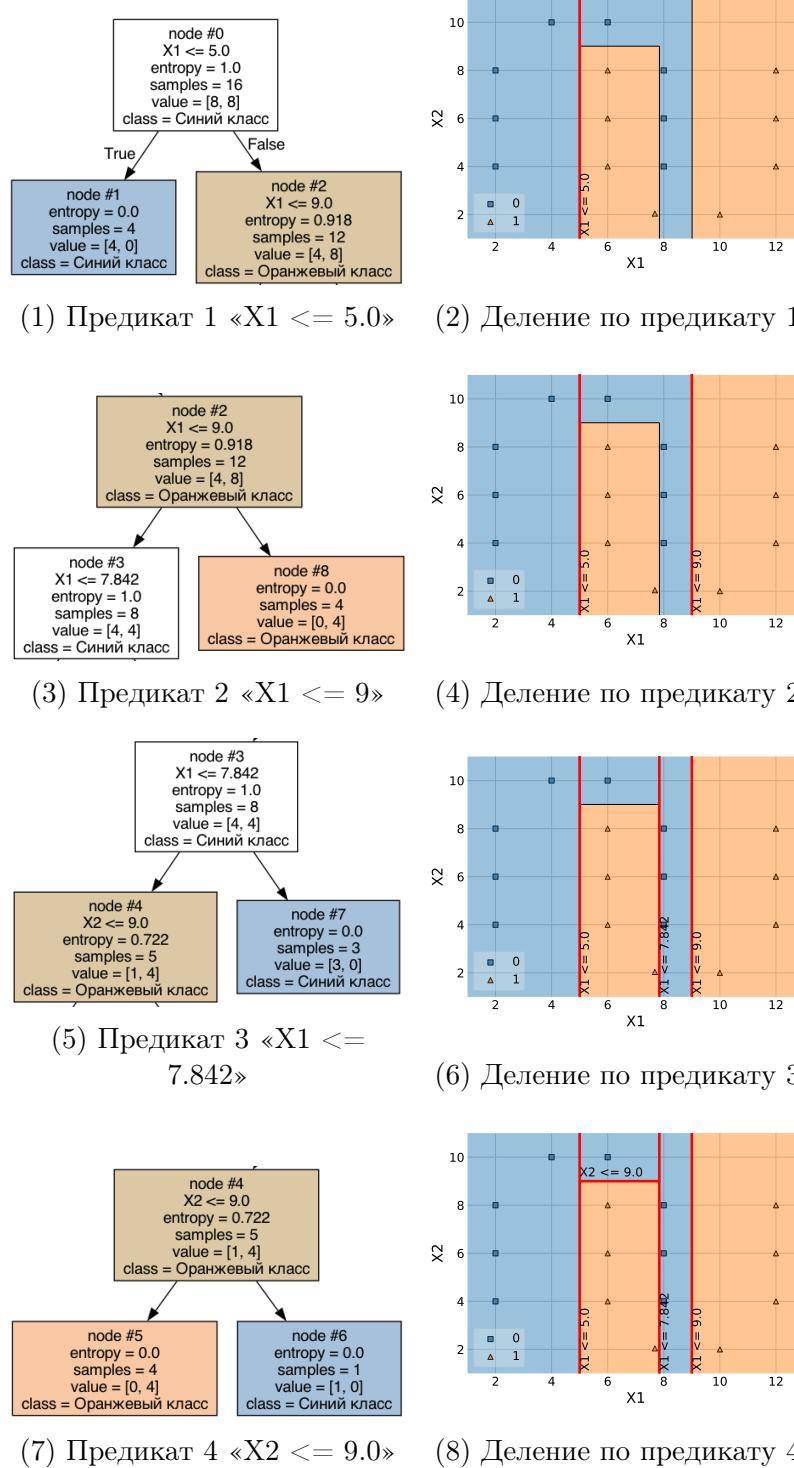


Рисунок 2: Построение разделяющей поверхности для набора данных в виде двух подков

разные классы или значения (в случае регрессии). В контексте деревьев решений, критерии информативности используются для выбора наилучшего разбиения признаков на узлах дерева.

Основными критериями информативности являются:

- для классификации — энтропия и критерий Джини
- для регрессии — дисперсия

Энтропия является мерой неопределенности в системе и может быть выражена следующим образом (Формула 1):

$$H(X) = - \sum_{i=1}^C p_i \log_2(p_i), \quad (1)$$

где X - это признак, C - множество классов,

p_i - вероятность появления класса i .

Чем больше энтропия, тем больше неопределенности в системе. Если все значения переменной X равновероятны, то энтропия будет максимальной, а если все значения имеют одно и ту же величину, то энтропия будет минимальной.

Критерий Джини (Gini impurity) - это еще один критерий информативности, используемый для построения деревьев решений. Он также измеряет неопределенность выборки и может использоваться вместо энтропийного критерия.

Формула для расчета критерия Джини выглядит следующим образом (Формула 2):

$$Gini(X) = 1 - \sum_{i=1}^C p_i^2, \quad (2)$$

где X - это признак, C - множество классов,

p_i - вероятность появления класса i .

Чем меньше значение критерия Джини, тем более однородными являются объекты в выборке. И наоборот, чем больше значение критерия Джини, тем менее однородными являются объекты в выборке. При использовании критерия Джини в деревьях решений выбирается тот признак, который позволяет наиболее эффективно уменьшить значение критерия Джини при разбиении выборки на подгруппы.

При построении дерева решений мы пытаемся разбить данные на подгруппы, где каждая группа будет иметь меньшую энтропию или меру Джини, чем исходная группа. Таким образом, критерий информативности в деревьях может быть определен как разность между энтропией/критерием Джини исходного узла и суммой энтропий/критериев Джини полученных поддеревьев.

Можно продемонстрировать смысл критериев информативности диаграммой (Рисунок 3).

Дисперсия является критерием информативности для задач регрессии в деревьях решений. Критерий дисперсии позволяет оценить насколько хорошо определенный признак разделяет целевую переменную.

Для расчета критерия дисперсии используется следующая формула (Формула 3):

$$Var(X) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2, \quad (3)$$

где n - количество объектов в выборке,

x_i - значение целевой переменной для i -го объекта,

\bar{x} - среднее значение целевой переменной в выборке.

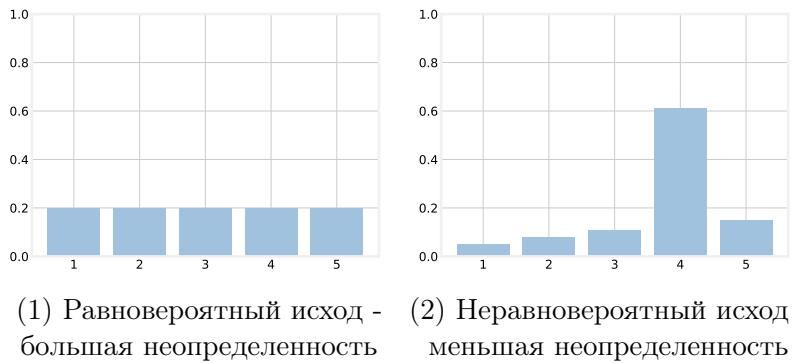


Рисунок 3: Смысл энтропии и критерия Джини

При разбиении выборки на подгруппы по значению определенного признака, происходит уменьшение значения дисперсии в каждой из подгрупп. Информативность признака оценивается как разница между начальным значением дисперсии и суммой дисперсий в подгруппах, взвешенной на количество объектов в каждой подгруппе.

Выбирается тот признак, который позволяет максимально уменьшить значение дисперсии при разбиении выборки на подгруппы. Этот признак становится корневым узлом дерева, а процесс разбиения продолжается для каждой подгруппы до тех пор, пока не будет достигнут критерий остановки.

0.1.3 Процесс построения дерева для классификации и регрессии

Давайте посмотрим, как происходит процесс построения дерева на примере энтропии. Дерево обычно строится с использованием жадного алгоритма (выбираются оптимальные для уровня признак и порог на этом признаке только на этом узле, не пытаясь просчитать несколько шагов вперед). Процесс построения дерева происходит с помощью прироста информации. **Прирост информации (Information Gain)** в деревьях решений является мерой, которая используется для определения того, какой признак лучше всего разделяет данные на различные классы или категории.

В деревьях решений признаки используются для разделения данных на более чистые группы, каждая из которых относится к определенному классу или категории. Цель состоит в том, чтобы создать дерево решений, которое максимизирует число правильно классифицированных данных.

Information Gain измеряет изменение энтропии (меры неопределенности) или критерия Джини в данных, которое происходит при разделении данных на основе конкретного признака. Таким образом, прирост информации является мерой, которая показывает, какой признак наиболее полезен для разделения данных на более чистые группы, которые максимально отличаются друг от друга по классам или категориям. Чем выше прирост информации, тем более полезен признак для разделения данных на более чистые группы.

Соответственно, процесс построения дерева включает в себя следующие шаги:

1. Начните с корневого узла, содержащего все обучающие данные.
2. Для каждого признака, рассчитайте энтропию (количество неопределенности) набора данных, используя формулу энтропии (Формула 1).

3. Для каждого признака, рассчитайте прирост информации (Information Gain), используя формулу 4.

$$IG(S, A) = H(S) - \sum \frac{|S_v|}{|S|} * H(S_v),$$

где $IG(S, A)$ - прирост информации для признака A,

$|S_v|$ - количество элементов в поднаборе данных, где значение признака A равно v,

$|S|$ - общее количество элементов в наборе данных,

$H(S_v)$ - энтропия поднабора данных, где значение признака A равно v.

(4)

4. Выберите признак с наибольшим приростом информации или минимальной энтропией. Разделите данные на основе выбранного признака, создавая два дочерних узла.
5. Рекурсивно повторяйте шаги 2-4 для каждого дочернего узла до выполнения некоторого условия остановки, например, максимальной глубины дерева или минимального числа объектов в узле.

Процесс построения дерева с помощью прироста информации для регрессии аналогичен процессу для классификации, но использует другую меру неопределенности. Вместо энтропии, используется дисперсия.

Вот общий процесс построения дерева с помощью прироста информации для регрессии:

1. Начните с корневого узла, содержащего все обучающие данные.
2. Для каждого признака вычислите прирост информации или изменение дисперсии, которое будет получено при разбиении данных на основе этого признака, используя формулу 5.

$$IG(S, A) = Var(S) - \sum \frac{|S_v|}{|S|} * Var(S_v),$$

где $IG(S, A)$ - прирост информации для признака A,

$|S_v|$ - количество элементов в поднаборе данных, где значение признака A равно v,

$|S|$ - общее количество элементов в наборе данных,

$Var(S_v)$ - дисперсия поднабора данных, где значение признака A равно v.

(5)

3. Выберите признак с наибольшим приростом информации или минимальной дисперсией. Разделите данные на основе выбранного признака, создавая два дочерних узла.
4. Рекурсивно повторяйте шаги 2-4 для каждого дочернего узла до выполнения некоторого условия остановки, например, максимальной глубины дерева или минимального числа объектов в узле.

Решающее дерево — хороший инструмент для прогнозирования на данных, но этот инструмент обладает одним ярко выраженным недостатком — нестабильностью. Процесс построения дерева с помощью прироста информации может приводить к появлению неоптимальных разбиений, особенно в случае, когда имеется большое количество признаков с высокой корреляцией.

При применении модели мы обычно ожидаем, что на приблизительно одинаковых данных мы получим схожую модель. С решающими деревьями это не так. Даже при небольшом изменении выборки решающее дерево сильно изменяет разделяющую поверхность (Рисунок 4), что, по сути, является признаком переобучения.

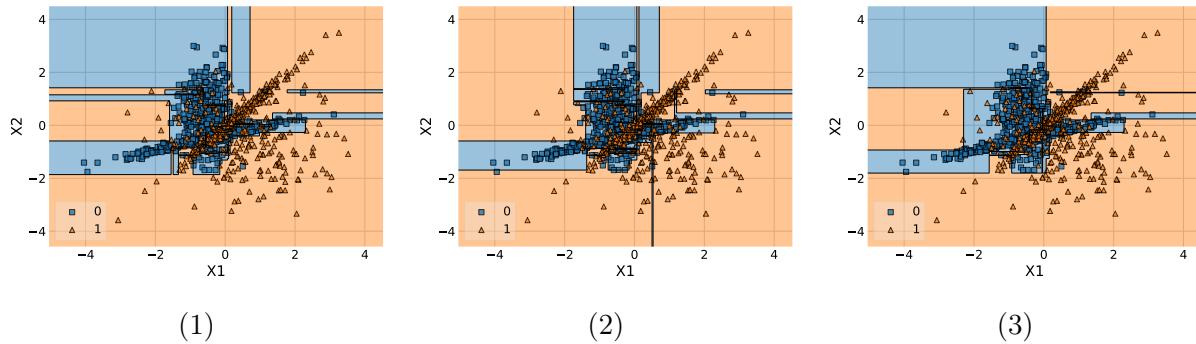


Рисунок 4: Изменение разделяющей поверхности при изменении 10% выборки

На рисунке 4 видно, что одиночное решающее дерево достаточно сильно изменяется при небольшом изменении данных.

Для уменьшения риска переобучения, можно использовать регуляризацию деревьев и композиции деревьев, такие как случайный лес (Random Forest) или градиентный бустинг деревьев (Gradient Boosted Trees).

0.1.4 Регуляризация деревьев решений

Регуляризация деревьев решений - это процесс добавления дополнительных ограничений на дерево решений, чтобы уменьшить переобучение модели. Деревья решений, как правило, склонны к переобучению, если они имеют слишком много листьев и слишком глубоко ветвятся на тренировочных данных, что может привести к неспособности модели обобщать на новые данные.

Существуют несколько способов регуляризации деревьев решений:

- **Ограничение глубины дерева (max_depth)** - это ограничение на максимальную глубину дерева. Если дерево достигает максимальной глубины, оно перестает ветвиться и превращается в лист, что предотвращает переобучение.
- **Минимальное количество элементов в листе (min_samples_leaf)** - это ограничение на минимальное количество обучающих элементов, которые должны быть в каждом листе дерева. Если количество элементов в листе меньше заданного значения, то он не будет разветвляться, что уменьшает переобучение.
- **Максимальное количество листьев (max_leaf_nodes)** - это ограничение на максимальное количество листьев в дереве. Если дерево достигает максимального количества листьев, оно не будет больше разветвляться, что предотвращает переобучение.
- **Регуляризация посредством снижения веса (weight regularization)** - в этом методе добавляется штраф к функции потерь модели за большие веса, что приводит к более простым моделям и уменьшает переобучение.
- **Подрезка деревьев (англ. Pruning)** - это процесс уменьшения размера дерева решений путем удаления частей дерева, которые не приносят значимого улучшения качества предсказаний (Рисунок 5). Основная идея прунинга заключается в том, что большие деревья решений могут слишком сильно подстроиться под обучающие данные и, следовательно, переобучаться. Удаление части дерева может привести к уменьшению переобучения и улучшению обобщающей способности модели.

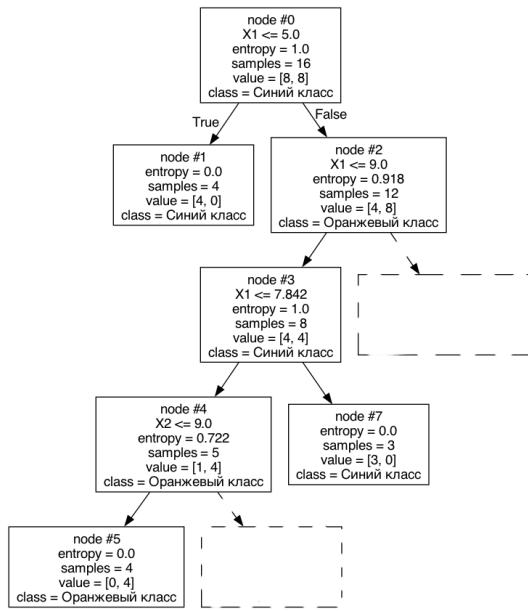


Рисунок 5: Подрезка деревьев

Существуют различные методы прунинга деревьев решений, включая pre-pruning, post-pruning и reduced error pruning. Pre-pruning означает остановку роста дерева решений до достижения определенной глубины или количества листьев. Post-pruning, с другой стороны, является процессом подрезки дерева после того, как оно было полностью построено. Reduced error pruning - это метод, который удаляет узлы из дерева решений, если это приводит к уменьшению ошибок на отложенной выборке.

Применение прунинга деревьев решений позволяет улучшить обобщающую способность модели, снизить вероятность переобучения и уменьшить сложность модели. Однако важно найти правильный баланс между уменьшением размера дерева и сохранением информации, которую дерево содержит.

Бороться с переобучением также помогают композиции деревьев, они рассмотрены в следующем разделе.

0.1.5 Вопросы для самопроверки

В каких частях деревьев аккумулируются итоговые прогнозы модели решающего дерева?

1. Корень
2. Ребра
3. Внутренние вершины
4. Листья

Правильные ответы: 4

Какой метод используется для вычисления значения регрессии в листе у решающего дерева для нескольких объектов?

1. Среднее значение целевой переменной у всех объектов в листе

2. Медианное значение целевой переменной у всех объектов в листе
3. Среднее квадратичное отклонение целевой переменной у всех объектов в листе
4. Наиболее часто встречающееся значение целевой переменной у всех объектов в листе

Правильные ответы: 1

Какие методы используется для выбора оптимального разделения в решающем дереве в задачах классификации?

1. Дисперсия
2. Критерий Джини
3. Энтропия
4. Метод опорных векторов

Правильные ответы: 2, 3

0.1.6 Резюме по разделу

В этом занятии мы познакомились с деревьями решений, узнали, что такое критерии информативности, и то, как строится дерево решения для задач классификации и регрессии.

0.2 Композиции деревьев

Цель занятия: ученик может применять смещение и разброс в моделях машинного обучения, композиции деревьев для повышения устойчивости моделей на основе деревьев решений.

0.2.1 Подходы к построению композиций

В машинном обучении агрегирование нескольких моделей для получения более точного и устойчивого прогноза носит название «композиция моделей» или «ансамблирование». В целом, композиция моделей на основе деревьев решений может значительно повысить качество предсказаний в задачах классификации и регрессии (Рисунок 6).

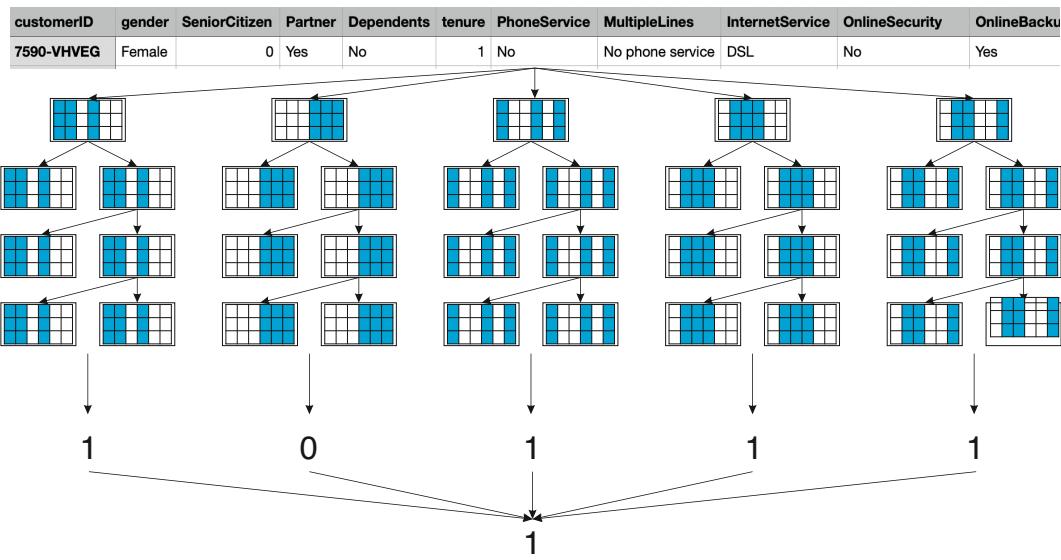


Рисунок 6: Композиция деревьев

Наиболее простым способом определения класса для композиции деревьев является голосование по большинству (англ. Majority vote), когда метка класса присваивается по относительному большинству отдельных моделей, предсказавших её. При одинаковом количестве голосов существуют разные стратегии: случайный выбор, приоритет определенных классов и пр. Для регрессии наиболее очевидный способ агрегирования — это усреднение предсказаний моделей из композиции.

На рисунке 7 можно увидеть, что применение композиции деревьев повышает устойчивость модели.

Бэггинг (Bootstrap Aggregating) и бустинг (Boosting) - это два популярных подхода к построению композиций моделей машинного обучения.

Бэггинг - это метод построения композиции моделей, в котором **обучается несколько независимых моделей** на случайных подмножествах обучающих данных с повторениями, а затем усредняются их предсказания для получения окончательного предсказания. Бэггинг позволяет снизить дисперсию модели, т.к. усреднение предсказаний моделей позволяет уменьшить эффект переобучения. Примерами алгоритмов бэггинга являются случайный лес и бэггинг деревьев решений.

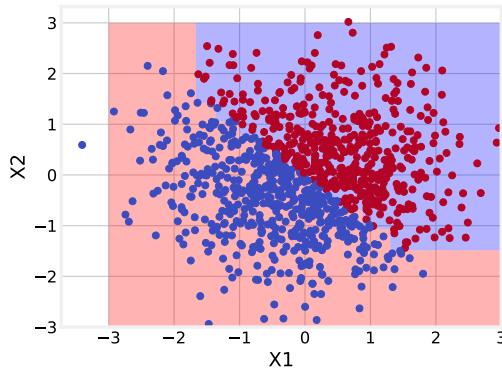


Рисунок 7: Повышение устойчивости предсказаний при использовании композиции деревьев

Бустинг - это метод построения композиции моделей, в котором **модели обучаются последовательно**, каждая из моделей корректирует ошибки предыдущей на обучающих данных. Поэтому каждая следующая модель фокусируется на тех объектах, на которых предыдущие модели ошибались. Бустинг позволяет снизить смещение модели, т.к. последующие модели корректируют ошибки предыдущих моделей. Примерами алгоритмов бустинга являются градиентный бустинг и AdaBoost.

Общий подход при использовании бэггинга и бустинга заключается в создании ансамбля моделей, который является более точным, чем каждая из отдельных моделей. Однако бэггинг и бустинг имеют различные преимущества и недостатки, и выбор подхода зависит от специфики данных и задачи.

В этом занятии мы обсудим такой подход, как бэггинг.

0.2.2 Бэггинг

В бэггинге (Bootstrap Aggregating) выбор случайных подмножеств объектов и признаков происходит с целью уменьшения корреляции между отдельными моделями в композиции. Это позволяет снизить дисперсию композиции и сделать ее более устойчивой к шуму (Рисунок 8).

Выбор случайного подмножества объектов происходит путем генерации новых выборок из исходного набора данных методом бутстрэпа (с повторениями). Каждый раз случайно выбирается подмножество объектов, которые используются для обучения отдельной модели. Количество объектов в каждом подмножестве также может быть настроено как гиперпараметр.

Выбор случайных подмножеств признаков (факторов) также может применяться в бэггинге для уменьшения корреляции между отдельными моделями. Однако, в отличие от выбора случайных подмножеств объектов, выбор случайных подмножеств признаков не производится на каждом шаге генерации новых выборок, а применяется на уровне отдельной модели. Выбор случайных подмножеств признаков может осуществляться различными способами, например, случайным выбором признаков на каждой вершине дерева при построении случайного леса или случайным выбором признаков при обучении каждой модели в композиции. Однако стоит помнить, что выбор подмножества признаков может негативно отразиться на качестве моделей, так как важность признаков неравнозначна, и следует с осторожностью применять такое

age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	target
0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034823	0.043400	-0.002592	0.019907	-0.017646	151.0
-0.01882	-0.044642	-0.031474	-0.026328	-0.030849	-0.019163	0.074412	-0.034983	0.068332	-0.092204	75.0
0.085299	0.050680	0.044443	-0.005670	-0.045999	-0.034194	0.072358	-0.02592	0.002861	-0.029390	141.0
-0.089063	-0.044642	-0.011595	0.036656	0.012194	0.024991	0.036038	0.013409	0.022688	0.009362	206.0
0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.019988	-0.046441	135.0
-0.092695	-0.044642	-0.040696	-0.019442	-0.068991	-0.079288	0.041277	-0.076394	0.041176	-0.096346	97.0
0.045472	0.050680	0.0347163	-0.015999	-0.040996	-0.024809	0.009779	-0.034981	-0.062917	-0.038357	138.0
0.063504	0.050680	-0.001895	0.066429	0.009620	0.108914	0.022869	0.017703	0.035816	0.030664	63.0
0.041708	0.050680	0.061696	0.040699	0.013973	0.006202	0.023679	0.002592	0.014990	0.011149	110.0
0.010980	-0.044642	0.039682	-0.033213	-0.01257	0.014580	0.023991	0.002592	0.067737	-0.015848	310.0
-0.096328	-0.044642	-0.083508	0.080101	-0.03389	-0.090561	0.013948	0.076394	0.066397	-0.034215	101.0
0.027178	0.050680	0.001756	-0.032213	-0.007073	0.045972	0.065491	0.072101	0.096435	-0.059667	69.0
0.016281	-0.044642	-0.028840	-0.009113	-0.004321	-0.009769	0.044958	0.039493	0.037478	-0.042499	179.0
0.005383	0.050680	-0.001895	0.080101	-0.004321	-0.015719	0.002903	0.002592	0.03394	-0.015304	185.0
0.045341	-0.044642	-0.025607	-0.012556	0.017694	-0.000603	0.081775	-0.034981	0.01988	-0.075636	118.0
0.052738	0.050680	-0.018062	0.009401	0.089244	-0.017662	0.039719	0.108111	0.036060	-0.042499	171.0
0.005315	-0.044642	0.047296	0.008415	0.074574	-0.023861	0.074412	0.036040	0.052777	-0.079117	166.0
0.070769	0.050680	0.021217	0.056201	0.034206	0.049416	-0.029716	0.034309	0.027364	0.001078	144.0
0.038207	-0.044642	-0.010517	-0.03656	-0.037344	-0.019476	0.023674	0.002592	-0.018114	-0.017646	97.0
0.027310	-0.044642	-0.018062	-0.040099	-0.02945	-0.011355	0.037594	0.034891	0.009443	-0.054925	168.0
0.049105	-0.044642	-0.058683	-0.043542	-0.045999	-0.043276	0.009779	0.038981	0.011897	0.013491	68.0
0.085430	0.050680	-0.022373	0.061216	-0.037344	0.026366	0.015005	0.035983	0.072113	-0.017646	49.0
0.085430	-0.044642	-0.004050	-0.009113	-0.029495	0.007767	0.022869	0.035983	0.061176	-0.015064	68.0
0.045341	0.050680	0.06608	0.01106	0.023702	0.047327	0.005446	0.072120	0.133397	0.133812	245.0
0.063535	-0.044642	0.038529	0.022885	-0.030664	-0.018559	0.006858	0.002592	0.029931	-0.054923	184.0
-0.067268	0.050680	-0.012673	-0.040099	-0.015328	0.004636	0.038123	0.013409	0.019196	-0.034215	202.0
-0.107226	-0.044642	-0.077342	-0.026228	-0.089630	-0.096109	0.026550	-0.076391	0.042571	-0.005220	137.0
-0.023677	-0.044642	0.059541	-0.040099	0.042843	0.043389	0.011824	0.034983	0.015999	0.040433	85.0
0.052606	-0.044642	-0.021295	-0.074527	-0.040096	-0.017639	0.006858	0.034981	0.006612	-0.054925	131.0

Рисунок 8: Выбор случайных подмножеств объектов и признаков в бэггинге

В целом, выбор случайных подмножеств объектов или признаков является важной составляющей бэггинга и позволяет увеличить устойчивость композиции к шуму и повысить ее точность. В целом, в машинном обучении устойчивость и качество работы моделей формализуются через понятия смещения и разброса. Давайте рассмотрим их.

0.2.3 Смещение и разброс

Смещение и разброс - это два ключевых понятия в машинном обучении, связанные с проблемой переобучения (overfitting) и недообучения (underfitting). Во многом, это обобщение оценки недообучения и переобучения, в том числе для композиций моделей.

Смещение (bias) - это ошибка модели, связанная с её недостаточной способностью улавливать реальную зависимость между признаками и целевой переменной. Модель с большим смещением может быть недообучена, т.е. её способность к обобщению данных будет низкой.

Разброс (variance) - это ошибка модели, связанная с её чрезмерной чувствительностью к случайному шумам в данных. Модель с большим разбросом может быть переобучена, т.е. она может слишком точно подогнаться под тренировочные данные и показывать плохие результаты на новых данных.

Ошибка модели может быть разложена на составляющие (Формула 6).

$$\text{Ошибка} = \text{смещение}^2 + \text{разброс} + \text{шум},$$

где смещение^2 обозначает квадрат ошибки модели, связанной с её

недостаточной способностью улавливать реальную зависимость между признаками и целевой переменной, разброс - ошибку модели, связанную

с её чрезмерной чувствительностью к случайному шумам в данных,

шум - случайную ошибку, не поддающуюся объяснению моделью.

Смысл смещения и разброса модели показан на рисунке (Рисунок 9). Trade-off между смещением и разбросом - это баланс между уменьшением ошибки модели, связанной с смещением, и ошибки модели, связанной с разбросом. Цель заключается в том, чтобы достичь

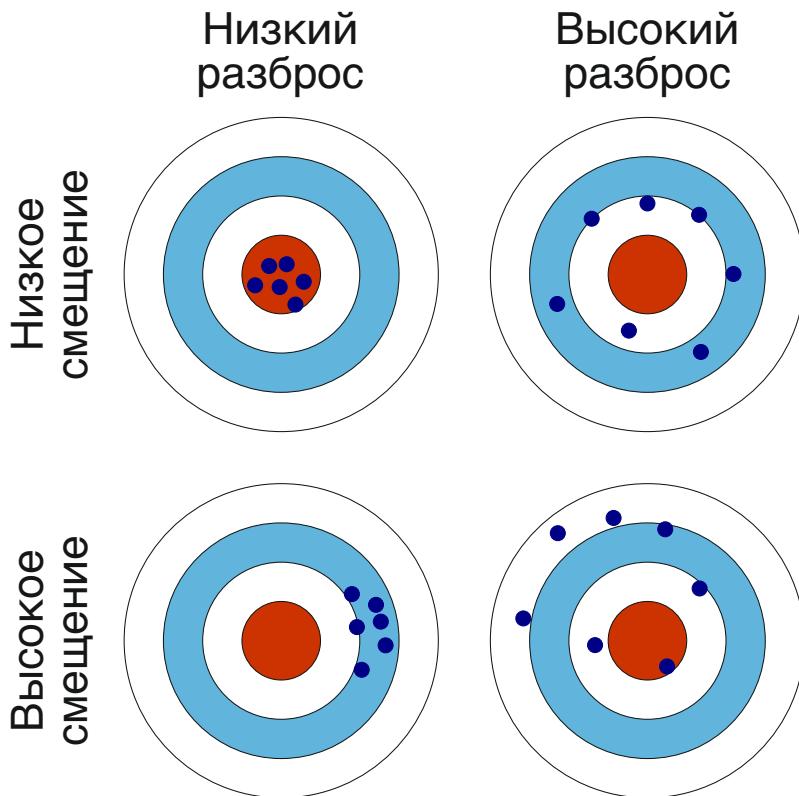


Рисунок 9: Отображение смысла смещения и разброса

оптимального уровня ошибки модели, который обеспечивает наилучшую способность к обобщению данных. Увеличение сложности модели может уменьшить ошибку смещения, но увеличить ошибку разброса, что может привести к переобучению. Уменьшение сложности модели может уменьшить ошибку разброса, но увеличить ошибку смещения, что может привести к недообучению (Рисунок 10).

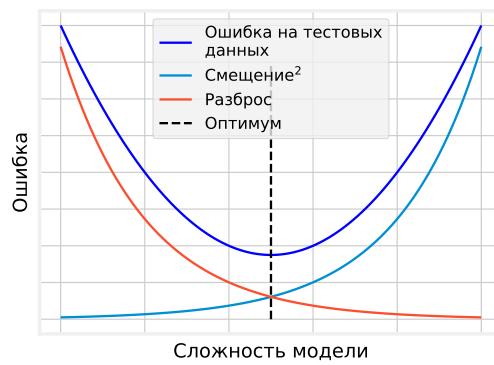


Рисунок 10: Баланс смещения и разброса

Бэггинг помогает уменьшить разброс, за счет использования нескольких случайных подмножеств обучающих данных и обучения независимых моделей на каждом из этих подмножеств. Каждая модель будет иметь свои сильные и слабые стороны, что позволит уменьшить разброс в итоговой модели.

С другой стороны, бэггинг не влияет на смещение, так как он все еще использует ту же функциональную форму модели на каждом подмножестве данных. Однако, в случае использования решающих деревьев в качестве базовых моделей для бэггинга, можно ожидать, что смещение также уменьшится. Это связано с тем, что решающие деревья могут описывать сложные зависимости в данных, а их комбинация позволяет учесть большее количество зависимостей.

Подбор оптимальных параметров модели (например, глубины дерева) позволяет найти баланс между смещением и разбросом, когда модель не переобучена и способна обобщать данные (Рисунок 11).

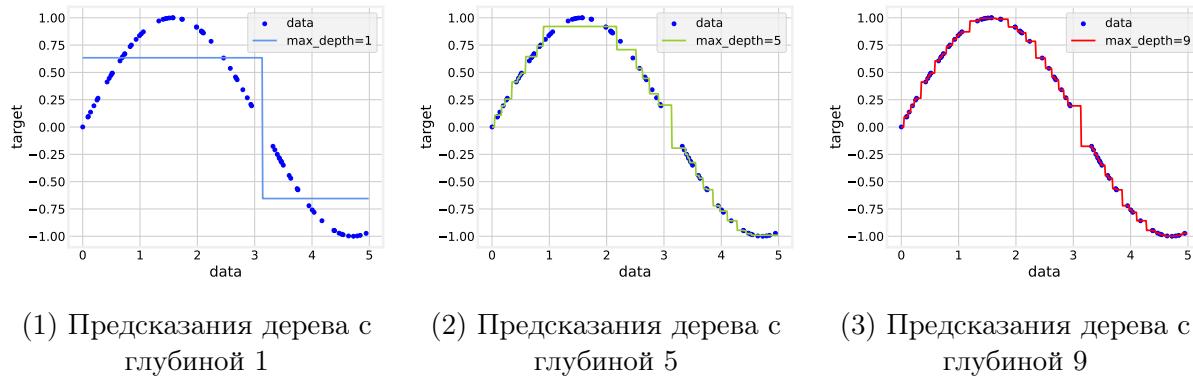


Рисунок 11: Подбор оптимальных параметров дерева для нахождения баланса bias-variance

При этом важно помнить, что если модели в композиции взаимосвязаны и ошибки одной модели могут быть связаны с ошибками другой модели, то использование бэггинга для уменьшения разброса может не дать значительного эффекта. Чем более связаны и похожи модели, тем меньший эффект оказывает бэггинг, и тем больше разброса остается в композиции. Следовательно, для построения композиции моделей необходимо, чтобы базовые модели были максимально независимыми и различными друг от друга. Кроме того, базовые модели также должны быть сложными, чтобы обеспечить достаточную гибкость для моделирования сложных зависимостей в данных. Эти требования к бэггингу реализованы в алгоритме случайного леса, о нем мы поговорим далее.

0.2.4 Вопросы для самопроверки

Какова разница между бэггингом (Bagging) и бустингом (Boosting)?

1. Бэггинг и бустинг - это одно и то же, просто разные названия одного метода обучения.
2. Бэггинг и бустинг используют одинаковые алгоритмы, но разные функции потерь.
3. Бэггинг строит ансамбль из нескольких независимых моделей, бустинг строит ансамбль из нескольких последовательных моделей.
4. Бэггинг и бустинг - это разные методы, которые используют один и тот же алгоритм обучения, но разные наборы данных.

Правильные ответы: 3

Какой из нижеперечисленных пунктов является главной характеристикой бэггинга для решающих деревьев (один ответ)?

1. Использует множество разнородных моделей для получения одной композиции
2. Стремится уменьшить смещение модели
3. Уменьшает разброс модели
4. Использует градиентный спуск для настройки моделей

Правильные ответы: 3

Каково определение смещения (bias) в контексте концепции «bias-variance trade-off»?

1. Мера различий между предсказаниями модели и реальными значениями в тестовом наборе данных.
2. Степень изменчивости предсказаний модели при разных значениях входных данных.
3. Мера ошибки модели, вызванная недостаточной ее сложностью.
4. Сумма квадратов отклонений предсказаний модели от среднего значения на обучающем наборе данных.

Правильные ответы: 3

0.2.5 Резюме по разделу

Одиночное решающее дерево имеет склонность к нестабильности, когда даже при небольшом изменении выборки предсказания модели значительно меняются. Для решения этой проблемы применяют композиции моделей, такие как бэггинг и бустинг.

Бэггинг на деревьях (Bagging of Trees) - это метод ансамблирования, в котором используется множество независимых решающих деревьев для получения одной композиционной модели. Каждое дерево строится на основе выборки из обучающего набора данных, которая формируется случайным образом с повторениями. При формировании композиции каждое дерево получает равный вес в соответствии с принципом голосования. Бэггинг на деревьях помогает уменьшить разброс (variance) модели и повысить ее устойчивость к выбросам.

Смещение и разброс - это два ключевых понятия в статистике и машинном обучении, которые связаны с ошибками предсказаний модели.

Смещение (bias) - это мера того, насколько среднее значение предсказаний модели отличается от правильных значений целевой переменной. Разброс (variance) - это мера того, насколько различаются предсказания модели для разных наблюдений из обучающей выборки. Идеальной моделью является та, у которой низкое смещение и низкий разброс. Однако, уменьшение смещения может привести к увеличению разброса, а уменьшение разброса может привести к увеличению смещения. Поэтому важно найти баланс между смещением и разбросом при разработке модели.

0.3 Случайный лес

Цель занятия: ученик может применить алгоритм случайного леса для решения задач классификации на подготовленных и неподготовленных данных.

План занятия:

- Визуальная демонстрация алгоритма
- Подготовка данных для алгоритма
- Процесс обучения
- Оценка качества алгоритма
- Интерпретация признаков с помощью алгоритма
- Процесс применения
- Область применения алгоритма
- Плюсы и минусы алгоритма
- Модификации алгоритма
- Реализация алгоритма в Python

0.3.1 Визуальная демонстрация алгоритма

Случайный лес (Random Forest) - это алгоритм машинного обучения, который используется для задач классификации и регрессии. Он является композицией решающих деревьев, где каждое дерево обучается независимо друг от друга на разных случайных подвыборках данных и с разными случайными подмножествами признаков в каждой вершине.

В процессе построения случайного леса сначала случайным образом выбирается подмножество обучающих данных (bootstrap-выборка) и подмножество признаков в каждой вершине. Затем на этом подмножестве данных строится решающее дерево, используя выбранные признаки. Этот процесс повторяется множество раз, что приводит к созданию ансамбля решающих деревьев. При предсказании каждое дерево выдает свой ответ, а итоговый ответ случайного леса определяется путем голосования или усреднения ответов всех деревьев. Визуальная демонстрация алгоритма представлена на рисунке 12.

Давайте разберемся с тем, что изображено на иллюстрации 12. Мы строим композицию из 5 деревьев. Далее **для каждого нового узла с предикатами выбирается свое подмножество признаков**. Как мы упоминали, проблемой классического бэггинга может быть то, что подмножество признаков выбирается для каждого дерева — соответственно, в это подмножество могут не попасть какие-нибудь важные признаки. Поэтому в random forest подмножества выбираются для каждой вершины с предикатами. Для нового объекта строится предсказание по каждому дереву, затем происходит голосование по большинству для классификации или усреднение для регрессии.

0.3.2 Подготовка данных для алгоритма

Подготовка данных является важным шагом для построения эффективной модели случайного леса. Подготовка данных для алгоритма может включать в себя следующие шаги:

- Работа с выбросами. Случайный лес не чувствителен к выбросам, но они могут негативно повлиять на качество модели, особенно если они присутствуют в большом количестве. Соответственно, можно применить различные методы обработки выбросов.

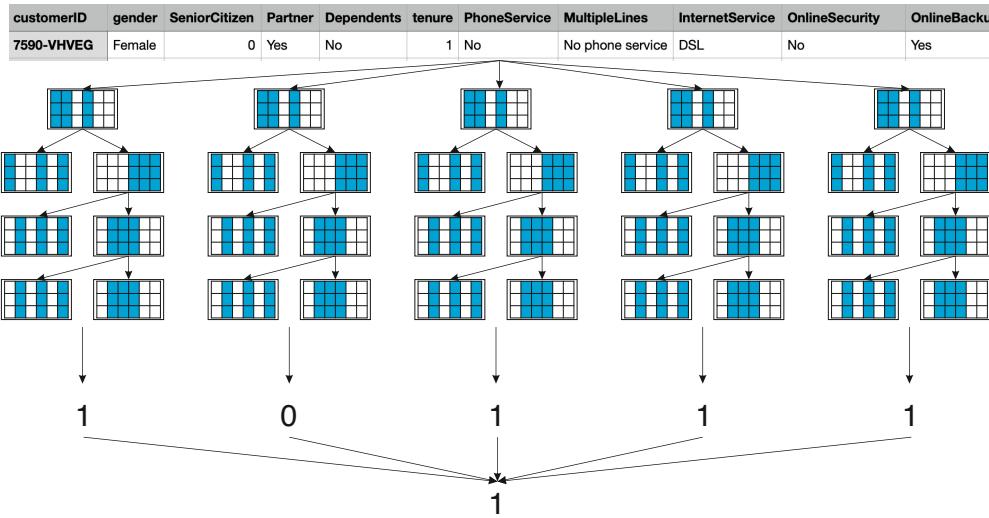


Рисунок 12: Визуальная демонстрация алгоритма случайный лес

- Преобразование категориальных переменных в числовые. Случайный лес не может работать с категориальными переменными напрямую, поэтому их необходимо преобразовать в числовые значения.
- Отбор признаков. Если в наборе данных есть большое количество признаков, можно использовать методы отбора признаков, такие как метод главных компонент или методы выбора признаков на основе значимости, чтобы уменьшить количество признаков и улучшить производительность модели.
- Разделение данных на обучающую и тестовую выборки.

0.3.3 Процесс обучения

Процесс обучения случайного леса включает в себя следующие шаги:

1. Для каждого из деревьев в ансамбле производится случайный выбор подмножества данных из обучающей выборки. Этот шаг называется подвыборкой с заменой (bootstrapping).
2. Случайный выбор подмножества признаков для каждого нелистового узла в дереве. Обычно для каждого узла выбирается \sqrt{d} признаков, где d - общее количество признаков.
3. Построение деревьев. Для каждой выборки данных и подмножества признаков строится дерево решений. Деревья строятся до определенной глубины или до тех пор, пока каждый лист не содержит минимальное количество примеров.

4. Прогнозирование. Для каждого нового примера производится прогноз с помощью каждого дерева. Класс, выбранный большинством деревьев, становится итоговым прогнозом.
5. Тюнинг гиперпараметров. Для улучшения качества модели можно провести тюнинг гиперпараметров, таких как количество деревьев, глубина деревьев, размер подвыборки и т.д.
6. Повторение. Шаги 1-5 повторяются несколько раз для получения стабильного прогноза.

В результате обучения случайноголеса получается ансамбль деревьев, каждое из которых принимает решение на основе случайной подвыборки данных и признаков. Это помогает уменьшить переобучение и увеличить стабильность модели. Кроме того, случайный лес может обрабатывать данные с большим количеством признаков и работать с несбалансированными данными.

0.3.4 Оценка качества алгоритма

Для оценки качества алгоритма random forest часто используют следующие метрики:
Классификация:

- **accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$
- **precision:** $\frac{TP}{TP+FP}$
- **recall:** $\frac{TP}{TP+FN}$
- **F1:** $2 * \frac{precision*recall}{precision+recall}$

Регрессия:

- **MSE**
- **MAE**

0.3.5 Интерпретация признаков с помощью алгоритма

Случайный лес может рассчитывать важность признаков на основе их вклада в уменьшение критерия ошибки, например, величиной критерия информативности Gini или приростом информации (Information Gain). Важность признака может быть использована для оценки вклада признака в предсказание целевой переменной. Признаки с более высокой важностью считаются более значимыми для модели.

0.3.6 Процесс применения

Random Forest создает ансамбль решающих деревьев, который демонстрирует лучшую точность, чем каждое отдельное дерево. Он также способен автоматически обрабатывать пропущенные значения, шум и выбросы в данных. Этот алгоритм легко распараллеливается, что делает его эффективным в использовании на больших датасетах.

Предсказание для нового объекта делается на основе агрегирования предсказаний всех деревьев в случайному лесу (например, путем голосования).

0.3.7 Область применения алгоритма

Некоторые из основных областей применения алгоритма случайного леса включают в себя:

- **Финансы:** анализ кредитных рисков, определение мошенничества, прогнозирование финансовых показателей.
- **Медицина:** диагностика заболеваний, прогнозирование риска заболеваний, анализ медицинских данных.
- **Реклама:** прогнозирование эффективности рекламы, анализ поведения потребителей.
- **Интернет-магазины:** рекомендательные системы, прогнозирование продаж, сегментация аудитории.
- **Обработка естественного языка:** классификация текстов, анализ тональности.
- **Изображения и видео:** распознавание объектов, классификация изображений, анализ видео.
- **Промышленность:** мониторинг и управление качеством продукции, прогнозирование отказов оборудования.
- **Транспорт и логистика:** прогнозирование спроса на перевозки, оптимизация маршрутов, анализ логистических данных.

В целом, алгоритм случайного леса может быть полезным в любой области, где требуется классификация или регрессия на основе большого количества признаков и где есть достаточно данных для обучения модели.

0.3.8 Плюсы и минусы алгоритма

Алгоритм случайного леса имеет ряд преимуществ и недостатков.

Плюсы:

- Хорошая точность предсказаний: случайный лес обладает хорошей точностью при классификации и регрессии, особенно при использовании большого количества деревьев в лесу.
- Устойчивость к переобучению: случайный лес имеет способность устранять переобучение благодаря случайности выборки и выбора признаков, что делает его устойчивым к выбросам и шуму в данных.
- Высокая скорость обучения: обучение случайного леса можно распараллелить, что позволяет быстро обрабатывать большие объемы данных.
- Возможность оценки важности признаков: случайный лес может определить наиболее важные признаки для классификации или регрессии.
- Универсальность: алгоритм случайного леса применим для решения широкого круга задач машинного обучения.

Минусы:

- Неинтерпретируемость: каждое дерево в случайном лесу можно интерпретировать, но в целом, сам алгоритм не обеспечивает понимания, как именно происходит принятие решения.
- Зависимость от выбора гиперпараметров: необходимо подбирать гиперпараметры, такие как количество деревьев и максимальная глубина дерева, для достижения наилучшей производительности.

- Неэффективность для работы с большим количеством категориальных признаков: алгоритм случайного леса не всегда хорошо работает с большим количеством категориальных признаков, поскольку в таком случае деревья могут становиться слишком глубокими.

В целом, алгоритм случайного леса является мощным инструментом машинного обучения с высокой точностью предсказаний и способностью устранять переобучение, но требует тщательной настройки гиперпараметров и может быть неэффективным при работе с большим количеством категориальных признаков.

0.3.9 Модификации алгоритма

Существует несколько модификаций алгоритма случайного леса, которые могут быть применены в зависимости от задачи и данных. Некоторые из них:

- **Random Decision Forest (RDF):** RDF - это расширение стандартного случайного леса, которое использует случайное количество деревьев для каждого класса. Это может улучшить предсказания в ситуациях, когда классы имеют различные размеры.
- **Balanced Random Forest (BRF):** BRF - это модификация случайного леса, в которой используется взвешенное голосование деревьев, чтобы уравновесить несбалансированные данные. Каждое дерево в BRF обучается на случайном подмножестве данных, сбалансированном по классам.

0.3.10 Реализация алгоритма в Python

В библиотеке Scikit-learn реализован класс **RandomForestClassifier** для алгоритма случайного леса. Основные параметры класса RandomForestClassifier:

- **n_estimators:** количество деревьев в лесу. По умолчанию n_estimators=100.
- **criterion:** функция для измерения качества разделения. Поддерживаются два критерия: «gini» для критерия Джини и «entropy» для энтропийного критерия. По умолчанию criterion=«gini».
- **max_depth:** максимальная глубина каждого дерева в лесу. По умолчанию max_depth=None, что означает, что деревья разрастаются до тех пор, пока все листья не будут чистыми (т.е. содержат только элементы одного класса), или пока не будет достигнуто минимальное количество элементов для разделения.
- **min_samples_split:** минимальное количество элементов, необходимое для того, чтобы узел мог быть разделен на два подузла. По умолчанию min_samples_split=2.
- **min_samples_leaf:** минимальное количество элементов, которые должны быть в листьях дерева. По умолчанию min_samples_leaf=1.
- **min_weight_fraction_leaf:** минимальная доля суммы весов (всех элементов), которая должна быть в листьях дерева. По умолчанию min_weight_fraction_leaf=0.0.
- **max_features:** количество признаков, которые должны быть рассмотрены при каждом разделении. Поддерживаются следующие значения: «auto» (выбор $\sqrt{n_features}$) признаков), «sqrt» (то же, что «auto»), «log2» (выбор $\log_2(n_features)$ признаков), None (выбор всех признаков), целое число (выбор конкретного количества признаков) или доля (выбор доли от общего количества признаков). По умолчанию max_features=«auto».

- **max_leaf_nodes:** максимальное количество листьев в дереве. По умолчанию max_leaf_nodes=None, что означает, что нет ограничения на количество листьев.

Класс **RandomForestRegressor** для решения задач регрессии имеет практически те же параметры, за исключением:

- **criterion:** функция для измерения качества разделения. Поддерживаются критерии: «squared_error», «absolute_error», «friedman_mse», «poisson». По умолчанию criterion=«squared_error».

Класс **RandomForestClassifier/RandomForestRegressor** имеют методы **fit(X, y)** для обучения модели на данных X и y, а также метод **predict(X)** для предсказания целевых значений для новых данных X. Кроме того, классы **RandomForestClassifier/RandomForestRegressor** имеют методы **score(X, y)** и **get_params()** для получения оценки точности модели и параметров модели соответственно.

0.3.11 Вопросы для самопроверки

Как происходит выбор подмножества признаков в случайном лесе?

1. Подмножество признаков выбирается для всей композиции деревьев
2. Подмножество признаков выбирается для каждого дерева
3. Подмножество признаков выбирается для каждой нелистовой вершины в дереве
4. Используются все признаки из набора данных

Правильные ответы: 3

Как можно использовать алгоритм случайного леса для задач регрессии?

1. Использовать критерий Джини для разбиения по предикатам и голосование по большинству для прогнозирования
2. Использовать энтропию для разбиения по предикатам и голосование по большинству для прогнозирования
3. Использовать MSE для разбиения по предикатам и усреднение для прогнозирования

Правильные ответы: 3

Выберите плюсы алгоритма случайного леса:

1. Возможность оценки важности признаков: случайный лес может определить наиболее важные признаки для классификации или регрессии.
2. Высокая вычислительная сложность: поскольку в случайном лесе много деревьев, обработка данных может занять длительное время.
3. Высокая скорость обучения: обучение случайного леса можно распараллелить, что позволяет быстро обрабатывать большие объемы данных.
4. Зависимость от выбора гиперпараметров: необходимо подбирать гиперпараметры, такие как количество деревьев и максимальная глубина дерева, для достижения наилучшей производительности.
5. Неинтерпретируемость: каждое дерево в случайном лесу можно интерпретировать, но в целом, сам алгоритм не обеспечивает понимания, как именно происходит принятие решения.

6. Неэффективность для работы с большим количеством категориальных признаков: алгоритм случайного леса не всегда хорошо работает с большим количеством категориальных признаков, поскольку в таком случае деревья могут становиться слишком глубокими.
7. Универсальность: алгоритм случайного леса применим для решения широкого круга задач машинного обучения.
8. Устойчивость к переобучению: случайный лес имеет способность устранять переобучение благодаря случайности выборки и выбора признаков, что делает его устойчивым к выбросам и шуму в данных.
9. Хорошая точность предсказаний: случайный лес обладает хорошей точностью при классификации и регрессии, особенно при использовании большого количества деревьев в лесу.

Правильные ответы: 1, 3, 7, 8, 9

0.3.12 Резюме по разделу

Алгоритм случайного леса основан на идее комбинирования нескольких деревьев решений, обученных на разных подмножествах признаков в каждой нелистовой вершине и подмножествах объектов. В результате получается ансамбль деревьев, который позволяет уменьшить влияние переобучения и повысить точность классификации или регрессии.

Одним из ключевых преимуществ случайного леса является его способность работать с большим количеством признаков и наблюдений, что делает его подходящим для обработки сложных и многоуровневых данных. Кроме того, случайный лес может быть использован для интерпретации важности признаков, что может быть полезно для понимания вклада каждого признака в процесс принятия решения.

0.4 Резюме по модулю

В этом модуле мы рассмотрели такие темы, как деревья решений, композиции деревьев и случайный лес.

Дерево решений - это алгоритм машинного обучения, который используется для решения задач классификации и регрессии. Он представляет собой древовидную структуру, в которой каждый узел представляет тест на определенном признаке, а каждая ветвь - возможный результат этого теста.

Деревья решений могут быть очень эффективными в решении задач, так как они просты в интерпретации и могут использоваться как для категориальных, так и для количественных данных. Кроме того, они могут быть применены для любого количества классов или значений целевой переменной.

Однако деревья решений могут также страдать от переобучения, особенно если они слишком глубокие или содержат много признаков. Этот недостаток может быть устранен с помощью различных методов, включая подрезку дерева, ограничение глубины и использование ансамблевых методов, таких как случайный лес.