Обучение

ИЦ «Гевисста»

Артем Груздев

Прогнозное моделирование в R и Python

Модуль 4. Введение в метод случайного леса

Москва — 2017

СОДЕРЖАНИЕ

Модуль 4 Введение в метод случайного леса

Лекция 4.1. Описание метода случайного леса

Как уже отмечалось, основным недостатком деревьев решений является их склонность к переобучению и нестабильность результатов, когда небольшие изменения в наборе данных могут приводить к построению совершенно другого дерева (особенно это актуально для метода CART). Случайный лес стал одним из способов решения этой проблемы. По сути случайный лес – это набор деревьев решений, где каждое дерево немного отличается от остальных. Идея случайного леса заключается в том, что каждое дерево может довольно хорошо прогнозировать, но скорее всего переобучается на определенной части данных. Если мы построим много деревьев, которые хорошо работают и переобучаются с разной степенью, мы можем уменьшить переобучение путем усреднения их результатов.

Для реализации вышеизложенной стратегии нам нужно построить большое количество деревьев решений, то есть ансамбль деревьев. Каждое дерево должно на приемлемом уровне прогнозировать зависимую переменную и должно отличаться от других деревьев (условие декоррелированности деревьев). Для этого в процесс построения деревьев вносим случайность, которая призвана обеспечить уникальность каждого дерева (отсюда случайный лес и получил свое название). Для получения рандомизированных деревьев в случайном лесе последовательно применяются две техники: сначала случайным образом отбираем наблюдения, которые будут использоваться для построения дерева, а затем для каждого узла дерева осуществляем случайный отбор фиксированного количества предикторов для поиска наилучшего расщепления.

Чтобы построить случайный лес, сначала необходимо определиться с количеством деревьев. Допустим, мы хотим построить ансамбль из 5 деревьев. Для построения каждого дерева мы сначала сформируем *бутстреп-выборку* (*bootstrap sample*) наших данных. То есть из набора данных объемом n наблюдений мы случайным образом выбираем наблюдение с возвращением n раз (поскольку отбор с возвращением, то одно и то же наблюдение может быть выбрано несколько раз). Мы получаем выборку, которая имеет такой же размер, что и исходный набор данных, однако некоторые наблюдения будут отсутствовать в нем (примерно 37% наблюдений исходного набора), а некоторые попадут в него несколько раз.

Чтобы проиллюстрировать это, предположим, что мы хотим создать бутстреп-выборку для списка из 10 наблюдений ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']. Возможная бутстреп-выборка может выглядеть как ['10', '9', '7', '8', '1', '3', '9', '10', '10', '7']. Другой возможной бутстреп-выборкой

может быть ['4', '8', '5', '8', '3', '9', '2', '6', '1', '6']. В нашем случае нам нужно построить 5 деревьев, поэтому будет сформировано 5 бутстреп-выборок. Наглядно механизм бутстрепа показан на рисунке 4.1.

Рис. 4.1 Механизм работы бутстрепа

На основе каждой сформированной бутстреп-выборки строится полное бинарное дерево решений[1], то есть разбиения узлов будут продолжаться до тех пор, пока не будет достигнуто минимальное количество наблюдений в терминальных узлах (изначально в качестве минимального количества наблюдений в терминальном узле Лео Брейман, один из авторов случайного леса, предложил для дерева классификации брать значение 1, а для дерева регрессии – значение 5). Однако алгоритм, который был описан для полного бинарного дерева решений (смотрите главу 3), теперь немного изменен. Вместо поиска наилучшей точки расщепления по каждому предиктору, алгоритм для разбиения узла случайным образом отбирает подмножество предикторов и затем находит наилучшую точку расщепления среди наилучших точек, найденных по каждому из случайно отобранных предикторов. Для выбора наилучшей точки разбиения используется уже знакомый вам критерий уменьшения неоднородности (для количественной зависимой переменной используется сумма квадратов остатков или

[1] В оригинальном подходе Лео Бреймана и в большинстве пакетов используется дерево решений CART, однако существуют реализации, где в качестве деревьев ансамбля используются деревья QUEST, деревья C4.5.

среднеквадратичная ошибка, а для категориальной зависимой переменной – мера Джини). Отбор подмножества предикторов повторяется отдельно для каждого узла, поэтому в каждом узле дерева может быть принято решение с использованием «своего» подмножества предикторов.

Необходимо отметить, что идея случайного отбора определенного количества предикторов в каждом узле дерева появилась не сразу. Сначала Лео Брейман в 1996 году предложил метод бэггинга или бутстреп-агрегирования, когда на основе исходного набора данных мы генерируем бутстреп-выборки, по ним строим полные бинарные деревья и затем агрегирум их результаты путем голосования или простого усреднения. Бэггинг стал предшественником случайного леса. Примерно в это же время Тин Кам Хо предложил идею ансамбля деревьев, построенных с помощью случайных подпространств признаков. Томас Диттерих в 2000 году предложил улучшить бэггинг дополнительной рандомизацией. Его подход заключался в том, чтобы ранжировать 20 наилучших вариантов разбиений по каждому узлу и случайным образом выбирать один вариант. Используя моделирование на синтетических и реальных наборах данных, он доказал, что дополнительная рандомизация улучшает качество работы бэггинга. Однако именно Лео Брейман и Адель Катлер предложили случайный отбор предикторов для поиска наилучшего расщепления каждого узла.

Использование бутстрепа приводит к тому, что деревья решений в случайном лесе строятся на немного отличающихся между собой бутстреп-выборках. Из-за случайного отбора переменных в каждом узле все расщепления в деревьях будут основаны на отличающихся подмножествах предикторов. Вместе эти два механизма приводят к тому, что все деревья в случайном лесе будут отличаться друг от друга. Кроме того, для достижения лучшего качества Лео Брейман предлагал при построении ансамбля варьировать размер терминального узла. Идея была развита Хемантом Ишвараном и Джеймсом Мэлли в рамках метода synthetic random forests или синтетических случайных лесов, а сам метод реализован в пакете R `rfsrcSyn`. Также для улучшения обобщающей способности Брейман предлагал вместо одномерных расщеплений использовать многомерные расщепления, когда наилучшее разбиение узла определяется не отдельным признаком, а линейными комбинациями признаков. Такую комбинацию можно найти с помощью обычной модели машинного обучения (гребневой регрессии, метода частичных наименьших квадратов, метода опорных векторов и др.). Идея получила развитие в работах Бьерна Менце и Микаэля Кельма, предложивших метод oblique random forest или косоугольный случайный лес, сам метод был реализован в пакете R `obliqueRF`.

Решая задачу классификации, каждое дерево сначала вычисляет для наблюдения листовые вероятности классов зависимой переменной. Листовая вероятность класса – это доля объектов класса в листе (терминальном узле) дерева, в который попало классифицируемое наблюдение. Каждое дерево голосует за класс с наибольшей вероятностью в листе. В итоге побеждает класс, за который проголосовало большинство деревьев. Итоговыми вероятностями классов для наблюдения будут доли голосов деревьев, поданных за данный класс.

Данный подход реализован в оригинальном программном коде Лео Бреймана и Адель Катлер, на базе которого написан пакет R `randomForest`. Для каждого дерева фиксируется только «победивший класс», листовые вероятности классов, полученные для отдельного дерева, отбрасываются и в дальнейших расчетах не участвуют. Поэтому в пакете `randomForest` итоговые вероятности классов для наблюдения – это доли голосов деревьев, поданных за данный класс.

№	фактический класс	спрогнозированный класс (итог голосования деревьев, построенных по всем бутстреп-выборкам)	итоговые вероятности классов (доли голосов деревьев, поданных за соответствующие классы)
1	Остается	Уходит	0.3 0.7
2	Уходит	Уходит	0.2 0.8
3	Уходит	Уходит	0.4 0.6

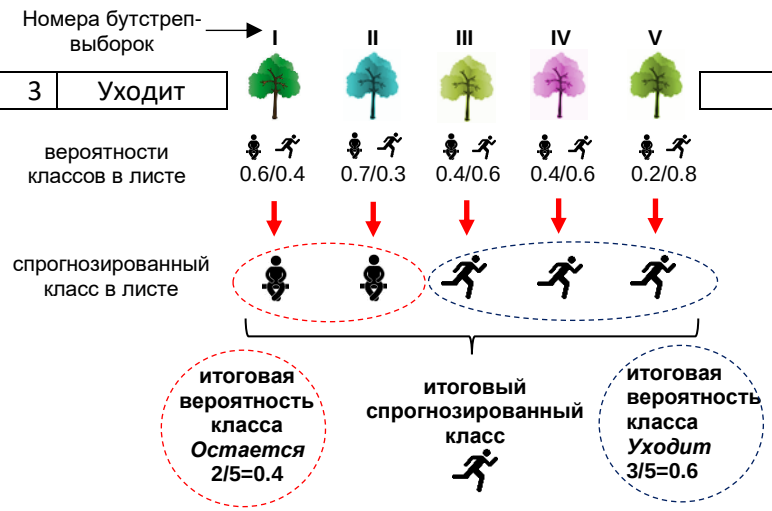


Рис. 4.2 Получение прогнозов для задачи классификации (оригинальный подход «усреднение прогнозов», реализован в пакете R randomForest)

В питоновской библиотеке `scikit-learn` для моделей `RandomForestClassifier` и `RandomForestRegressor` используется другой подход. Каждое дерево вычисляет для наблюдения листовые вероятности классов. Эти листовые вероятности усредняются по всем деревьям и прогнозируется класс с наибольшей усредненной листовой вероятностью. Поэтому для классов `RandomForestClassifier` и `RandomForestRegressor` итоговыми вероятностями классов будут листовые вероятности классов, усредненные по всем деревьям.

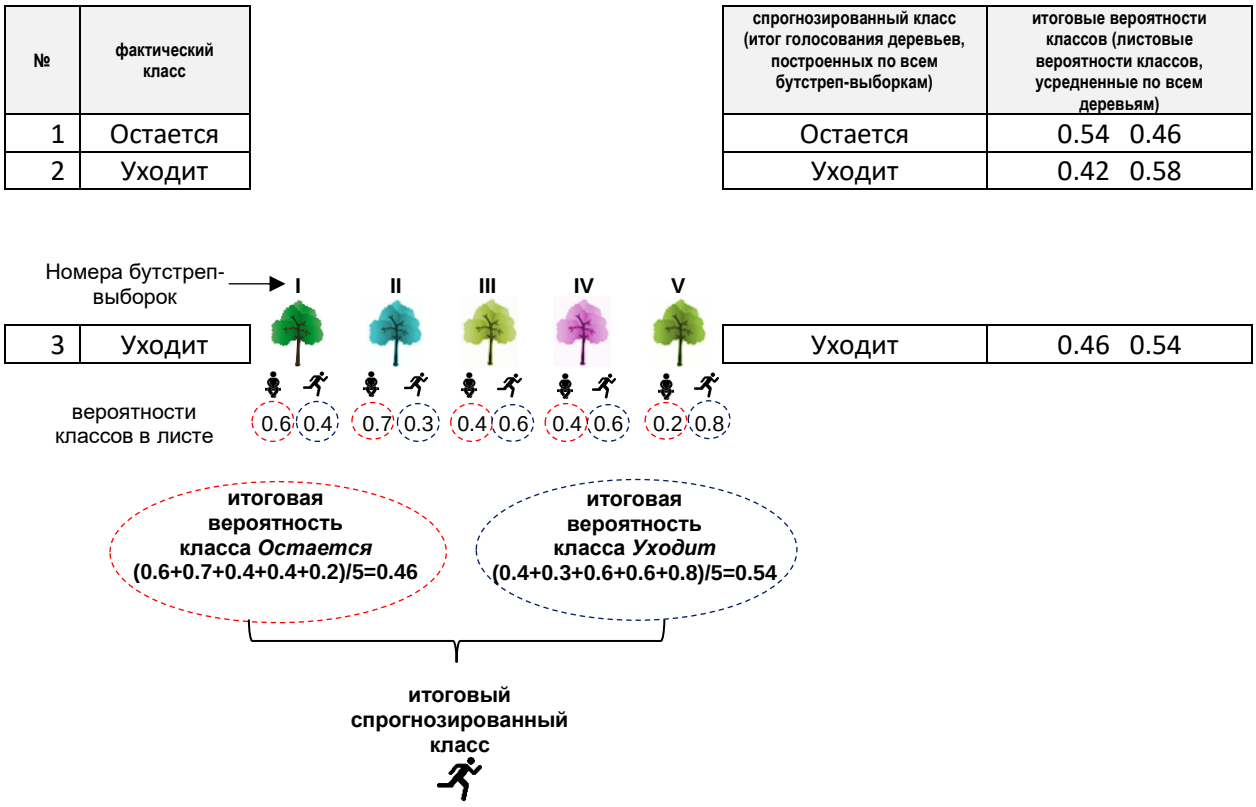


Рис. 4.3 Получение прогнозов для задачи классификации (подход «усреднение листовых вероятностей», реализован в классе RandomForestClassifier питоновской библиотеки scikit-learn)

Решая задачу регрессии, каждое дерево прогнозирует для наблюдения среднее значение зависимой переменной в листе (терминальном узле), в который это наблюдение попало, и затем происходит усреднение полученных средних значений по всем деревьям.

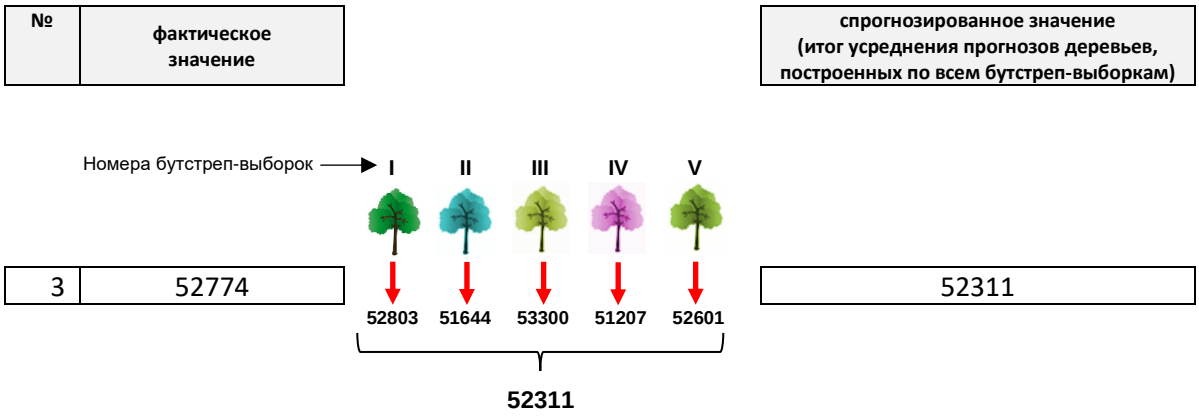


Рис. 4.4 Получение прогнозов для задачи регрессии

В итоге математически алгоритм случайного леса можно описать следующим образом:

1. Для $b = 1, 2... B$ (где B – количество деревьев в ансамбле):
   (a) извлечь бутстреп-выборку S размера N из обучающих данных;
   (b) по бутстреп-выборке S построить полное дерево T_b, рекурсивно повторяя следующие шаги для каждого терминального узла, пока не будет достигнуто минимальное количество наблюдений в нем (для классификации – одно наблюдение, для регрессии – 5 наблюдений):
      i. из первоначального набора M предикторов случайно выбрать m предикторов;
      ii. из m предикторов выбрать предиктор, который обеспечивает наилучшее расщепление;
      iii. расщепить узел на два узла-потомка.
2. В результате получаем ансамбль деревьев решений $\{T_b\}_{b=1}^{B}$
3. Предсказание новых наблюдений осуществлять следующим образом:

для регрессии: $\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$;

для классификации: пусть $\hat{C}_b(x)$ - класс, спрогнозированный деревом решений T_b, то есть $T_b(x) = \hat{C}_b(x)$; тогда $\hat{C}_{rf}^{B}(x)$ - это класс, наиболее часто встречающийся в множестве $\{\hat{C}_b(x)\}_{b=1}^{B}$

Лекция 4.2. Оценка качества модели

Качество модели случайного леса может быть оценено обычным способом и с помощью метода OOB. В рамках обычного способа мы берем каждое наблюдение и для прогноза используем все бутстреп-выборки, затем для задачи классификации подсчитываем количество неправильно классифицированных наблюдений, взятое от общего количества наблюдений, а для задачи регрессии подсчитываем сумму квадратов остатков, усредненную по всем наблюдениям. Как уже говорилось, каждая бутстреп-выборка не содержит примерно 37% наблюдений исходной обучающей выборки. Поэтому в рамках метода OOB мы берем каждое наблюдение и для прогноза используем бутстреп-выборки, которые не содержат данное наблюдение (т.е. наблюдение «выпало» из выборки и данную выборку для этого наблюдения можно назвать out-of-bag выборкой, отсюда и название метода), а затем подсчитываем количество неправильно классифицированных наблюдений, взятое от общего количества наблюдений, или сумму квадратов остатков, усредненную по всем наблюдениям. Метод OOB используется только для оценки качества модели на обучающей выборке. К каждому наблюдению контрольной выборки мы просто применяем правила, сформулированные каждым деревом леса в ходе обучения. Затем осуществляем голосование или усреднение и на основе полученных прогнозов обычным способом вычисляем метрику качества. Рассмотрим процесс оценки качества модели с помощью метода OOB наглядно (рис. 4.5). Допустим, у нас есть исходный набор из 10 наблюдений, на его основе мы сгенерировали 5 бутстреп-выборок. Для каждого наблюдения мы должны зафиксировать бутстреп-выборки, в которых оно отсутствует. Например, на рис. 4.5 видно, что наблюдение 4 отсутствует в бутстреп-выборках I, III, IV и V. Эти выборки будут для наблюдения 4 out-of-bag выборками. Для классификации или вынесения прогноза по наблюдению 4 нас как раз будут интересовать голоса или прогнозы деревьев, построенных по этим четырем out-of-bag выборкам.

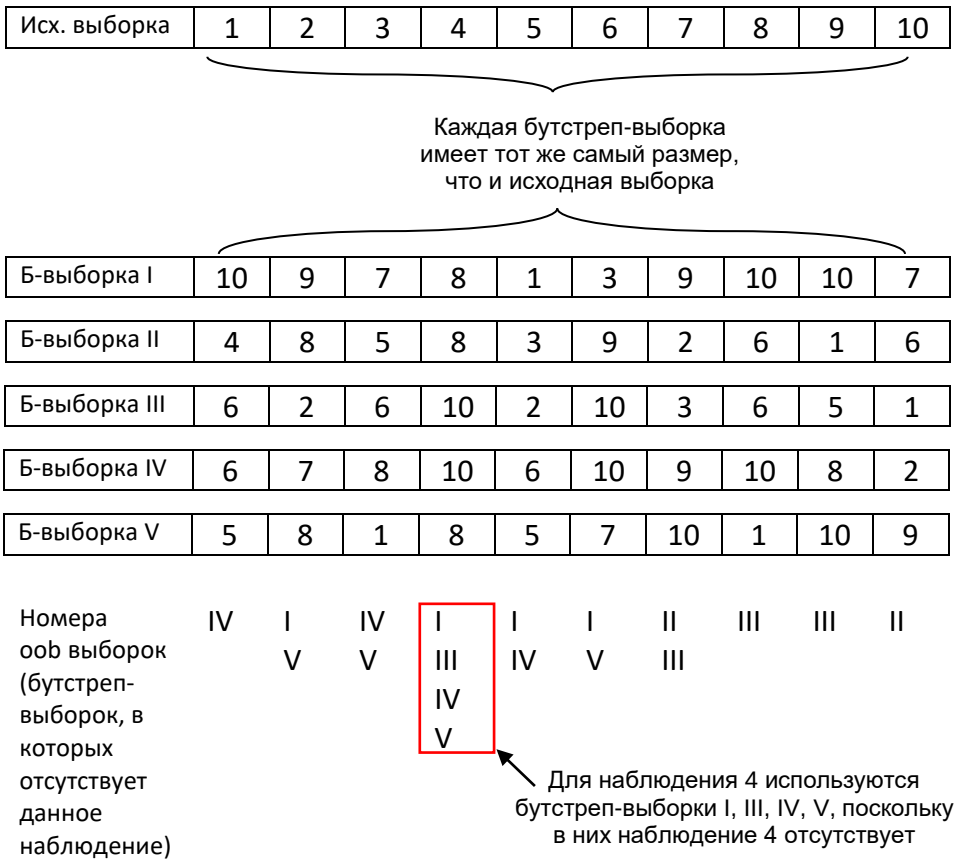


Рис. 4.5 Out-of-bag выборки для оценки качества модели

Возьмем задачу классификации (рис. 4.6). Допустим, необходимо отнести наблюдение 4 к тому или иному классу зависимой переменной *Статус клиента [status]*. Фактически оно принадлежит классу *Уходит*. Для оценки качества модели используются только те деревья решений, которые строились по бутстреп-выборкам, не содержащим наблюдение 4, и затем проводится голосование деревьев. Наблюдение 4, фактически принадлежащее классу *Уходит*, отсутствует в 4 бутстреп-выборках: I, III, IV, V. В голосовании участвуют 4 дерева, построенных по этим 4 out-of-bag выборкам. Мы предъявляем наше наблюдение каждому дереву, оно проверяет наблюдение на соответствие своим правилам классификации, вычисляет листовые вероятности классов и соответствующий класс. Например, деревья классифицировали наблюдения так: *Остается*, *Остается*, *Остается* и *Уходит*. В результате побеждает класс *Остается*. Случайный лес ошибочно относит наблюдение 4 к классу *Остается*. В итоге мы подсчитываем количество таких неверно классифицированных наблюдений, делим на общее количество наблюдений и получаем оценку качества случайного леса для классификации. Ее еще называют ошибкой классификации по методу OOB или OOB ошибкой для классификации.

$$ER^{OOB} = \frac{1}{n}\sum_{i=1}^{n} 1\left(\hat{Y}^{OOB}(X_i) \neq Y_i\right)$$

где:

Y_i – фактическое значение зависимой переменной;

$\hat{Y}^{OOB}(X_i)$ – расчетное значение зависимой переменной (результат голосования деревьев, построенных по выборкам, не содержащим X_i)

№	номера out-of-bag выборок, участвующих в голосовании	фактическое значение	спрогнозированное значение (итог голосования деревьев, построенных по out-bag выборкам)	результат классификации
1	IV	Остается	Остается	ВЕРНО
2	I, V	Уходит	Уходит	ВЕРНО
3	IV, V	Уходит	Уходит	ВЕРНО
4	I, III, IV, V	Уходит	Остается	НЕВЕРНО
5	I, IV	Остается	Остается	ВЕРНО
6	I, V	Уходит	Уходит	ВЕРНО
7	II, III	Уходит	Уходит	ВЕРНО
8	III	Остается	Уходит	НЕВЕРНО
9	III	Остается	Остается	ВЕРНО
10	II	Уходит	Уходит	ВЕРНО

количество неверных ответов=2

Ошибка классификации = количество неверно классифицированных наблюдений/общее количество наблюдений = 2/10=0,2

Рис. 4.6 Вычисление OOB ошибки для задачи классификации

Теперь возьмем задачу регрессии (рис. 4.7). Допустим, необходимо спрогнозировать значение зависимой переменной *Оценка дохода [income]* для наблюдения 4. Для оценки качества модели используются только те деревья решений, которые строились по бутстреп-выборкам, не содержащим наблюдение 4, и затем проводится усреднение прогнозов, выданных деревьями.

Наблюдение 4, имеющее фактическое значение *45304*, отсутствует в 4 бутстреп-выборках: I, III, IV, V. Таким образом, в усреднении участвуют 4 дерева, построенных по этим 4 бутстреп-выборкам. Мы предъявляем наше наблюдение каждому дереву, оно проверяет наблюдение на соответствие своим правилам прогнозирования, вычисляет среднее значение. Допустим, деревья прогнозируют следующие значения: *44470*, *45112*, *46790* и *47230*. На этот раз нас интересует квадрат остатка – разницы между фактическим значением зависимой переменной *45304* и ее спрогнозированным значением *45901* (результатом усреднения прогнозов, вычисленных деревьями). В итоге по каждому наблюдению вычисляем квадрат остатка, суммируем и полученную сумму квадратов

остатков делим на общее количество наблюдений. Сумма квадратов остатков, поделенная на общее количество наблюдений, становится оценкой качества случайного леса для регрессии. Ее еще называют среднеквадратичной ошибкой по методу OOB или OOB ошибкой для регрессии.

$$MSE^{OOB} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}^{OOB}(X_i) - Y_i\right)^2$$

где:

Y_i – фактическое значение зависимой переменной;

$\hat{Y}^{OOB}(X_i)$ – расчетное значение зависимой переменной (результат усреднения прогнозов деревьев, построенных по выборкам, не содержащим X_i)

Наглядно процесс вычисления среднеквадратичной ошибки по методу OOB показан на рисунке 4.7.

№	номера out-of-bag выборок, участвующих в прогнозе	фактическое значение	спрогнозированное значение (результат усреднения прогнозов деревьев, построенных по out-bag выборкам)	квадрат остатка (факт. знач. – спрогн. знач.)²
1	IV	50451	50037	171396
2	I, V	52700	52127	328329
3	IV, V	32704	32028	456976
4	I, III, IV, V	45304	45901	356409
5	I, IV	29518	29067	203401
6	I, V	29508	29029	229441
7	II, III	24018	23501	267289
8	III	26369	26938	323761
9	III	26109	26905	633616
10	II	19369	19857	238144

сумма квадратов остатков=3208762
Среднеквадратичная ошибка = сумма квадратов остатков/общее количество наблюдений = 3208762/10=320876,2

Рис. 4.7 Вычисление OOB ошибки для задачи регрессии

На практике оценка OOB ошибки достоверна, когда количество деревьев в ансамбле достаточно велико. В ситуации, когда вы используете мало деревьев, то есть мало бутстреп-выборок, высока вероятность того, что наблюдение встретится во всех бутстреп-выборках (иными словами, ни разу не выпадет из бутстреп-выборок) и таким образом для него не будет получена OOB оценка. Необходимо настраивать качество модели, увеличивая количество деревьев до достаточно большого числа, пока OOB ошибка не перестанет уменьшаться. Обратите внимание, что вычисление OOB ошибки не заменяет собой проверку на контрольной выборке. Эксперименты показывают, что ошибка классификации и

среднеквадратичная ошибка, вычисленные по методу OOB, являются более оптимистичными, чем ошибка классификации и среднеквадратичная ошибка, вычисленные на контрольной выборке или усредненные по контрольным блокам перекрестной проверки.
В различных статистических пакетах оценка качества случайного леса может быть вычислена либо с помощью обычного способа, либо с помощью OOB метода.
В пакете R `randomForest` функция `print` для ансамбля деревьев классификации выводит ошибку классификации по методу OOB, а для ансамбля деревьев регрессии – среднеквадратичную ошибку и процент объясненной дисперсии, вычисленные по методу OOB. Функция `predict` позволяет вычислить вероятности классов и значения зависимой переменной двумя способами. При наличии аргумента, задающего обучающую выборку, вероятности классов и значения зависимой переменной прогнозируются обычным способом, при пропуске этого аргумента для наблюдений обучающей выборки будут возвращены вероятности классов и значения зависимой переменной, спрогнозированные по методу OOB.
В библиотеке `scikit-learn` для экземпляра класса `RandomForestClassifier` вычисляется обычная правильность, а для экземпляра класса `RandomForestRegressor` обычный R-квадрат (по умолчанию значение параметра `oob_score` равно `False`). Вероятности классов и значения зависимой переменной вычисляются также обычным способом. Однако если использовать значение параметра `oob_score=True`, то в атрибут `oob_score_` для экземпляра класса `RandomForestClassifier` будет записана правильность, вычисленная по методу OOB, а для экземпляра класса `RandomForestRegressor` – R^2, вычисленный по методу OOB.

Лекция 4.3. Настройка параметров случайного леса

Главный параметр случайного леса – это количество деревьев в ансамбле. Как правило, большее количество деревьев практически всегда дает лучший результат. Это обусловлено тем, что усреднение результатов по большему количеству деревьев позволит получить более устойчивый ансамбль за счет снижения переобучения. Следует отметить, что если при увеличении числа деревьев улучшения качества не происходит или даже наблюдается уменьшение качества прогноза, то это может говорить о плохом качестве выборки, о присутствии значительного шума в данных. Подобное явление также часто наблюдается на небольших выборках. Количество деревьев в ансамбле, необходимое для хорошего качества модели, возрастает с числом предикторов и ростом объема данных.

Помните, что с ростом количества деревьев требуется больше памяти и больше времени для обучения.

Наилучший способ определить, сколько деревьев построить, это сравнить метрики качества моделей с разным количеством (последовательно увеличиваем значение, строим 100, 200, 300, 400, 500 деревьев) на контрольной выборке. Как вариант, можно воспользоваться перекрестной проверкой, здесь мы смотрим метрики качества, усредненные по контрольным блокам перекрестной проверки. Обычно по мере увеличения числа деревьев качество модели на обучающей выборке увеличивается (можно добиться даже 100%-ной правильности или значения AUC, равного 1), а на контрольной выборке качество увеличивается и затем стабилизируется на определенном значении.

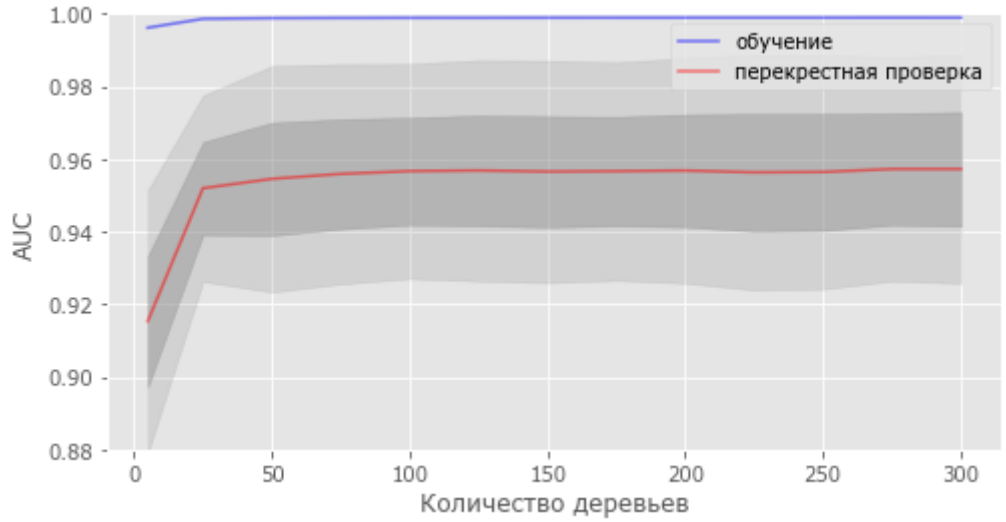


Рис. 4.8 График зависимости AUC от количества деревьев, используется перекрестная проверка

Взглянем на рис. 4.8. Анализируя значения AUC, усредненные по обучающим блокам перекрестной проверки, мы видим, что даже совсем небольшое количество деревьев дает идеальную дискриминирующую способность (значение AUC равно 1). Однако рассматривая метрики, усредненные на контрольных блоках перекрестной проверки, мы видим, что качество увеличивается медленнее и стабилизируется примерно на 150 деревьях при значении AUC, равном 0,957.

Как мы уже говорили выше, большее количество деревьев всегда дает лучшее качество, однако вы должны регулировать стоимость улучшения качества с вычислительной точки зрения. Например, мы можем использовать ансамбль из 300 деревьев и получить на контрольной выборке AUC 0.811, затем дополнительно натренировать еще 100 деревьев и получить AUC 0.83, в этом случае идея увеличить количество деревьев имеет смысл, если же дополнительная тренировка 100 деревьев

дает в итоге AUC 0.812, то скорее всего стоимость такого улучшения будет сомнительна.

В пакете R `randomForest`, представляющим собой реализацию классического алгоритма случайного леса, параметр, задающий количество деревьев в ансамбля, называется `ntree`, в пакете R `ranger` – быстрой реализации случайного леса, использующей параллельные вычисления, он называется `num.trees`, в классах `RandomForestClassifier` и `RandomForestRegressor` питоновской библиотеки `scikit-learn` он называется `n_estimators`, в версиях библиотеки `h2o` для R (функция `h2o.randomForest`) и Python (класс `H2ORandomForestEstimator`) он называется `ntrees`.

Определив такое количество деревьев в ансамбле, которое дает приемлемое качество модели на контрольной выборке и обучается за приемлемое время, мы задаем количество случайно отбираемых предикторов. Лео Брейман называл этот параметр mtry. Поясним, как работает mtry. Количество случайно отбираемых предикторов, равное 1, означает, что при разбиении отбор признаков не будет осуществляться вообще, будет выполнен поиск точек расщепления для одного случайно выбранного признака. Если количество случайно отбираемых предикторов задать равным общему количеству предикторов в наборе данных, это будет обозначать, что в каждом разбиении смогут участвовать все предикторы набора данных, в отбор признаков не будет привнесена случайность (останется лишь случайность, обусловленная бутстрепом). При таком варианте все деревья в случайном лесе будут в большей степени схожи между собой, нежели при более низких значениях mtry, и качество модели может ухудшится. На практике подбирают значения mtry, которые составляют примерно 20–40% от общего числа предикторов. Эти значения были сформулированы на основе дополненных правил Лео Бреймана, приведенных на рис. 4.9.

Количество случайно отбираемых предикторов (mtry)	Для классификации	Для регрессии
корень/треть от общего количества предикторов, деленная пополам	$m = 0{,}5 \times \sqrt{M}$	$m = 0{,}5 \times (M/3)$
корень/треть от общего количества предикторов	$m = \sqrt{M}$	$m = M/3$
удвоенный корень/треть от общего количества предикторов	$m = 2 \times \sqrt{M}$	$m = 2 \times (M/3)$
общее количество предикторов		$m = M*$
	m – случайно отбираемое число предикторов, *M* – общее число предикторов в наборе	
Примечания: Полученные значения mtry округляем в меньшую сторону *Правило было сформулировано по итогам экспериментов уже после смерти Лео Бреймана		

Рис. 4.9 Правила для определения оптимального количества случайно отбираемых предикторов

Если есть несколько переменных с сильной прогнозной силой (прогнозную силу обычно определяют с помощью пермутированной важности, см. раздел 4.1.4 *Важность переменных*), меньшие значения mtry могут дать лучшее качество. Если данные содержат много переменных со слабой прогнозной силой, нужно попробовать б*о*льшие значения mtry. При построении случайного леса с помощью классов `RandomForestClassifier` и `RandomForestRegressor` библиотеки `scikit-learn` помимо правил Бреймана необходимо попробовать б*о*льшие значения mtry. По мнению ряда специалистов[2], на практике варьирование значений mtry не оказывает существенного влияния на качество модели.

[2] D. Richard Cutler, Thomas C. Edwards, Jr., Karen H. Beard, Adele Cutler, Kyle T. Hess, Jacob Gibson, and Joshua J. Lawler. 2007. Random forests for classification in ecology. Ecology 88:2783–2792.

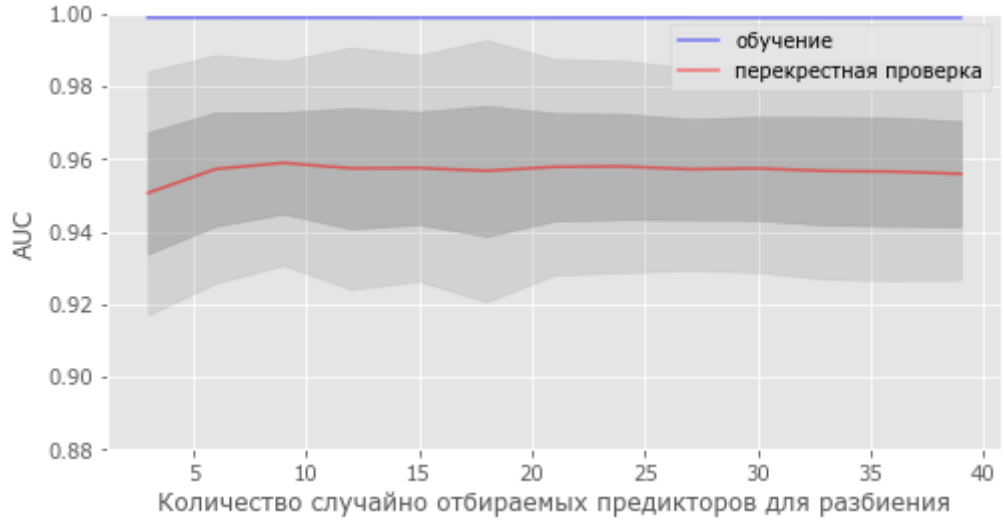


Рис. 4.10 График зависимости AUC модели из 300 полных деревьев от количества случайно отбираемых предикторов, используется перекрестная проверка

Взглянем на рис. 4.10. Здесь мы строим модели на основе полных деревьев. Набор содержит 40 предикторов. Видно, что варьирование количества случайно отбираемых предикторов не очень сильно влияет на качество модели.

Обрезка деревьев, как правило, требует корректировки mtry. Посмотрите на рис. 4.11. Здесь мы уже строим модели, используя обрезку деревьев (оставлено 10 уровней ниже корневого узла). Рассматривая значения AUC, усредненные на контрольных блоках перекрестной проверки, мы видим, что при увеличении количества случайно отбираемых предикторов качество модели постепенно снижается. Оптимальное качество мы получим, используя от 9 до 15 признаков.

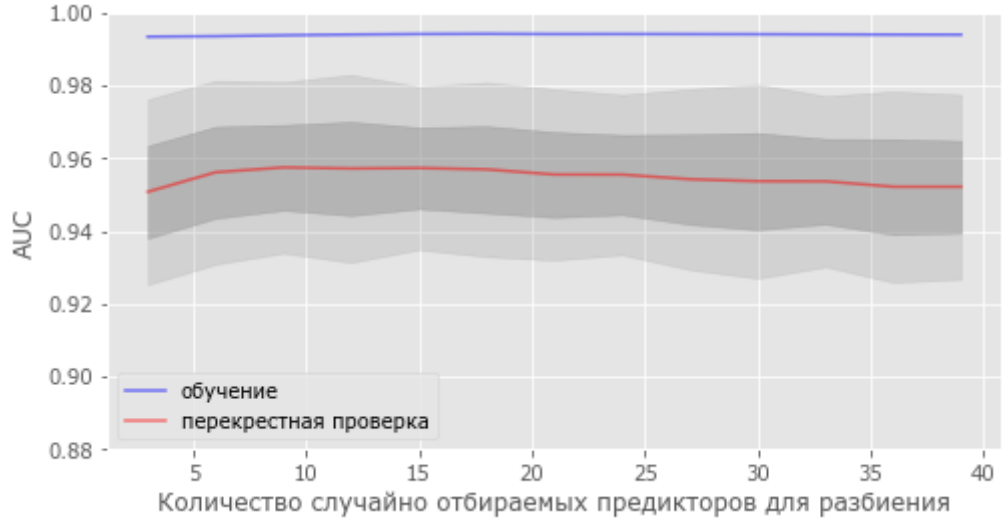


Рис. 4.11 График зависимости AUC модели из 300 обрезанных деревьев от количества случайно отбираемых предикторов, используется перекрестная проверка

В итоге при фиксированном количестве деревьев необходимо выбрать такое количество случайно отбираемых предикторов, которое дает максимальное качество модели на контрольной выборке.
В пакетах R `randomForest` и `ranger` параметр, задающий количество случайно отбираемых предикторов, называется `mtry` (оригинальное название, предложенное Брейманом), в классах `RandomForestClassifier` и `RandomForestRegressor` питоновской библиотеки `scikit-learn` он называется `max_features`, в версиях библиотеки `h2o` для R (функция `h2o.randomForest`) и Python (класс `H2ORandomForestEstimator`) он называется `mtries`.
Еще один параметр, который стоит учитывать при построении модели случайного леса – это глубина деревьев. Как правило, при увеличении глубины возрастает время обучения, но при этом увеличивается качество модели на обучающей и контрольной выборках.

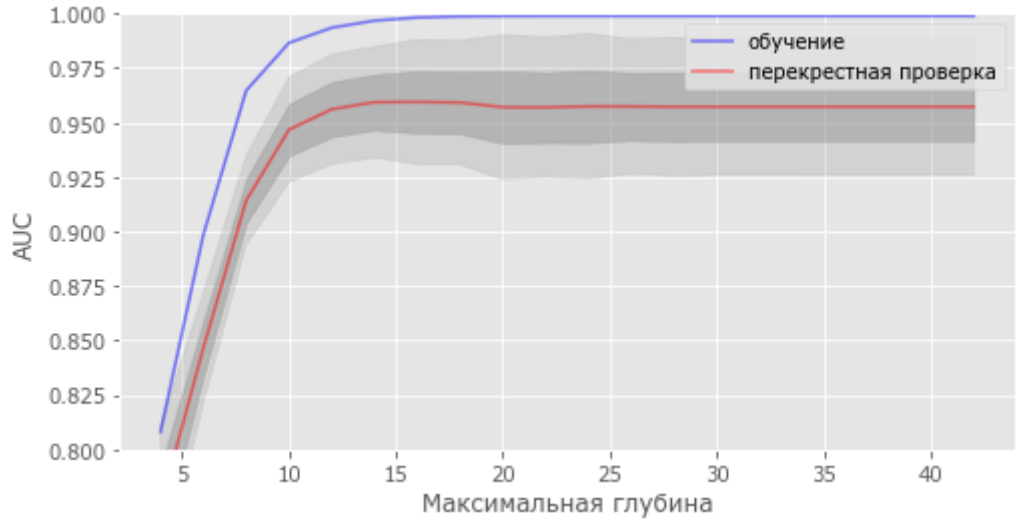


Рис. 4.12 График зависимости AUC модели из 300 деревьев от глубины, используется перекрестная проверка

В ряде случаев, например, при работе с зашумленными данными, построение деревьев с максимальной глубиной не дает хорошего качества модели. Это обусловлено тем, что при очень высоких значениях глубины деревья становятся излишне сложными и чувствительными к случайным возмущениям данных и в случае большого количества шумовых объектов рандомизация и усреднение по ансамблю не позволяют скомпенсировать возникшее переобучение. В таких случаях нужно выбрать меньшее значение глубины. Вместе с тем нужно избегать и слишком низкого значения глубины, при котором деревья не смогут в достаточной степени обучиться и возникнет эффект недообучения.
В пакетах R `randomForest` и `ranger` данный параметр отсутствует, поэтому максимальную глубину в них регулируют косвенно с помощью варьирования минимального количества наблюдений в терминальном

узле (увеличивая количество наблюдений в терминальном узле, снижаем глубину). В пакете R `randomForest` минимальное количество наблюдений в терминальном узле регулируется параметром `nodesize`, а в пакете R ranger – параметром `min.node.size`. В классах `RandomForestClassifier` и `RandomForestRegressor` питоновской библиотеки `scikit-learn`, в версиях библиотеки `h2o` для R (функция `h2o.randomForest`) и Python (класс `H2ORandomForestEstimator`) параметр, задающий максимальную глубину, называется `max_depth`. Кроме того, в этих инструментах максимальную глубину можно настроить косвенно, изменив количество наблюдений в терминальном узле. Соответствующий параметр в классах `RandomForestClassifier` и `RandomForestRegressor` питоновской библиотеки `scikit-learn` называется `min_samples_leaf`, в версиях библиотеки `h2o` для R (функция `h2o.randomForest`) и Python (класс `H2ORandomForestEstimator`) он получил название `min_rows`.

	Количество деревьев в ансамбле	**Количество случайно отбираемых предикторов**	**Максимальная глубина**	**Минимальное количество наблюдений в листе**
пакет R randomForest	ntree	mtry	-	nodesize
пакет R ranger	num.trees	mtry	-	min.node.size
классы RandomForestClassifier и RandomForestRegressor, библиотека scikit-learn	n_estimators	max_features	max_depth	min_samples_leaf
библиотека h2o	ntress	mtries	max_depth	min_rows

Рис. 4.13 Названия параметров случайного леса в пакетах R и Python

Дополнительного улучшения качества случайного леса можно добиться за счет использования различных стратегий поиска точек расщепления. В классической реализации случайного леса для количественного предиктора с k категориями может быть рассмотрено k-1 вариантов разбиения. Значения будут отсортированы в порядке возрастания и в качестве точек расщепления могут быть рассмотрены средние по каждой паре упорядоченных смежных значений. Например, у нас есть значения количественной переменной *Возраст* 74, 70, 64 66, 65, 68, 69. Точки расщепления будут выглядеть следующим образом:

Предиктор *Возраст*

Значения	64		65		68		69		70		74
Точки расщепления		64,5		66,5		68,5		69,5		72	

Рис. 4.14 Создание точек расщепления классическим способом

В реализации случайного леса, предлагаемой библиотекой H2O, по каждому количественному предиктору будет создана гистограмма,

состоящая из интервалов или бинов. Количество бинов для количественных предикторов регулируется параметрами `nbins` и `nbins_top_level`. Тип гистограммы для количественных предикторов регулируется параметром `histogram_type`. По умолчанию для количественного предиктора создается не менее 20 бинов одинаковой ширины (для параметра `nbins` по умолчанию используется значение `20`, а для параметра `histogram_type` по умолчанию задано значение `UniformAdaptive`, которое назначает интервалы одинаковой ширины). При этом на первом уровне дерева гистограмма не может иметь количество бинов, превышающее значение `nbins_top_level` (по умолчанию используется значение `1024`), затем на каждом последующем уровне происходит уменьшение этого максимального значения вдвое. Параметр `nbins_top_level` работает в паре с параметром `nbins`, который регулирует, когда нужно прекратить уменьшение количества бинов вдвое. Например, если на определенном уровне дерева `nbins_top_level` становится равным 32, а `nbins` задан равным `20`, то на последующем уровне разбиение количества бинов на 2 не происходит. Ширина бина будет определена по формуле (max-min)/N, где max – максимальное значение, min – минимальное значение, N – количество интервалов. В нашем случае будет создано 20 бинов с шагом (74-64)/20=0,5: от 64 до 64,5, от 64,5 до 70, ..., от 73,5 до 74. По границам этих интервалов будут найдены точки расщепления.

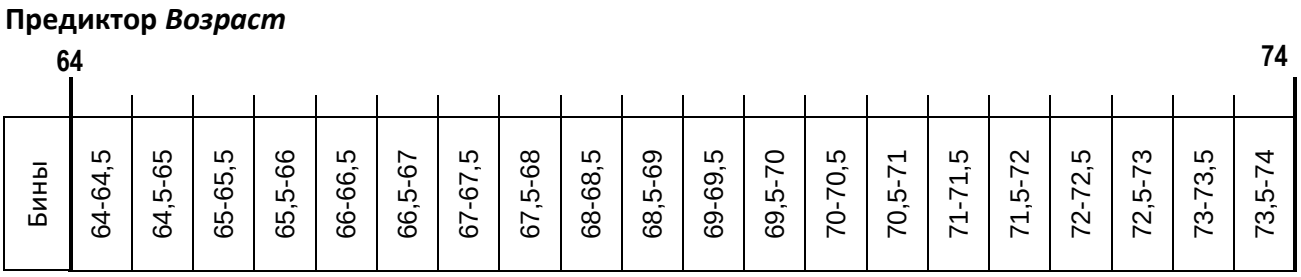


Рис. 4.15 Создание точек расщепления с помощью гистограммирования

Перебор значений `nbins` и `nbins_top_level` может улучшить качество модели.

Обратите внимание, при обработке количественных переменных с асимметричными распределениями и выбросами значение `UniformAdaptive`, установленное по умолчанию для параметра `histogram_type`, может привести к выбору менее оптимальных вариантов расщепления. Например, у нас есть количественная переменная, которая меняется в диапазоне от 0 до 10. Кроме того, у нас есть экстремальные значения 9999 (также вспомните, что часто пропуски кодируют большим отрицательным или положительным значением, например, -9999 или 9999). В этом случае биннинг, подразумеваюший создание интервалов

одинаковой ширины, может дать неоптимальное решение, потому что при создании интервалов будет опираться на значения минимума и максимума. Допустим, мы зададим `nbins` равным 10. Тогда ширина интервала будет вычислена как (9999-0)/10=999,9. У нас будут всего два бина: 0-999,9 и 999,9-9999. Все значения переменной, кроме значений 9999, окажутся в одном бине 0-999,9, в итоге будет рассмотрен только один вариант разбиения. Поэтому при обработке переменных с асимметричными распределениями и выбросами для параметра `histogram_type` лучше использовать значение `QuantilesGlobal`, которое создает интервалы, содержащие одинаковое количество наблюдений (квантили). В данном случае `nbins` уже будет задавать количество квантилей.

Кроме того, вы можете получить более оптимальные расщепления по итогам гистограммировании переменной с асимметричным распределением и выбросами, предварительно выполнив преобразования, максимизирующие нормальность распределения. Для распределения, скошенного вправо (положительный коэффициент асимметрии), обычно применяются следующие преобразования: квадратный корень sgn(x)*(abs(x)^1/2), кубический корень sgn(x)*(abs(x)^1/3), свернутый корень sgn(x)*sqrt(sqrt(abs(x))) и логарифм.

Для распределения, скошенного влево (отрицательный коэффициент асимметрии), обычно применяются следующие преобразования: квадратный корень (константа – x), кубический корень (константа – x) и логарифм (константа – x). Поскольку логарифм нуля, а равно и любого отрицательного числа, неопределен, перед использованием логарифмического преобразования ко всем значениям нужно добавить константу, чтобы сделать их положительными. При использовании корней обычно корень берут от модуля числа (чтобы не вычислять корни отрицательных чисел) и затем учитывают знак числа.

В ряде случаев улучшения можно добиться, задав для параметра `histogram_type` значение `Random`. В этом случае алгоритм случайным образом отбирает N−1 точек расщепления из диапазона значений и затем использует упорядоченный список этих точек разбиения для поиска наилучшей точки расщепления. N определяется параметрами `nbins` и `nbins_top_level`.

Перспективным является использование циклического перебора типов гистограмм расщепляющих значений. Его можно применить, задав для параметра `histogram_type` значение `RoundRobin`. Для первого дерева расщепляющие значения будут определены по границам интервалов одинаковой ширины, для второго дерева расщепляющие значения будут определены по границам интервалов одинакого размера, для третьего дерева расщепляющие значения будут получены случайным образом, для

четвертого дерева расщепляющие значения вновь будут определены по границам интервалов одинаковой ширины и так по кругу.

В классической реализации случайного леса для категориального предиктора с k категориями может быть рассмотрено $2^{k-1}-1$ вариантов разбиения. Категории делятся всеми возможными способами на две группы.

В реализации случайного леса, предлагаемой библиотекой H2O, по каждому категориальному предиктору будет создана гистограмма, состоящая из интервалов или бинов. При этом в H2O в отличие от библиотеки `scikit-learn` не используется one-hot-кодирование и бины будут сформированы из исходных категорий. Количество бинов для категориальных предикторов регулируется параметрами `nbins_cats` и `nbins_top_level`. По умолчанию для категориального предиктора создается не менее 1024 бинов (для параметра `nbins_cats` по умолчанию используется значение `1024`). При этом на первом уровне дерева гистограмма не может иметь количество бинов, превышающее значение `nbins_top_level` (по умолчанию используется значение `1024`), затем на каждом последующем уровне происходит уменьшение этого максимального значения вдвое. Параметр `nbins_top_level` работает в паре с параметром `nbins_cats`, который регулирует, когда нужно прекратить уменьшение количества бинов вдвое. Например, если на определенном уровне дерева `nbins_top_level` становится равным 32, а `nbins_cats` задан равным `20`, то на последующем уровне разбиение количества бинов на 2 не происходит.

Давайте посмотрим, как происходит разбиение на бины. Если количество категорий меньше значения параметра `nbins_cats`, каждая категория получает свой бин. Допустим, у нас есть переменная *Class*. Если у нее есть уровни A, B, C, D, E, F, G и мы зададим `nbins_cats=8`, то будут сформировано 7 бинов: {A}, {B}, {C}, {D}, {E}, {F} и {G}. Каждая категория получает свой бин. Будет рассмотрено $2^6-1=63$ точки расщепления. Если мы зададим `nbins_cats=10`, то все равно будут получены те же самые бины, потому что у нас всего 7 категорий. Если количество категорий больше значения параметра `nbins_cats`, категории будут сгруппированы в бины в лексикографическом порядке. Например, если мы зададим `nbins_cat=2`, то будет сформировано 2 бина: {A, B, C, D} и {E, F, G}. У нас будет одна точка расщепления. A, B, C и D попадут в один и тот же узел и будут разбиты только на последующем, более нижнем уровне или вообще не будут разбиты.

Значение параметра `nbins_cats` для категориальных предикторов оказывает гораздо большее влияние на обобщающую способность модели, чем значение параметра `nbins` для количественных предикторов (обычно более высокие значения параметра `nbins` приводят к выбору более оптимальных точек расщепления). Для предикторов с большим

количеством категорий небольшое значение параметра `nbins_cats` может внести в процесс создания точек расщепления дополнительную случайность (поскольку категории группируется в определенном смысле произвольным образом), в то время как большие значения параметра `nbins_cats` (например, значение параметра `nbins_cats`, совпадающее с количеством категорий), наоборот, снижают эту случайность, каждая отдельная категория может быть рассмотрена при формировании точки разбиения, что приводит к переобучению на обучающем наборе (AUC= 1). Таким образом, этот параметр является очень важным параметром настройки. Как мы уже говорили, значение по умолчанию для `nbins_cats` равно `1024`. Значение `nbins_cats` может достигать 65 тыс. и этого должно хватить при работе с большими наборами данных. Если вы хотите получить более простую модель, необходимо уменьшить значения параметров `nbins_top_level` и `nbins_cats`. Если вы хотите получить более сложную, гибкую модель, необходимо увеличить значения `nbins_top_level` и `nbins_cats`. Имейте в виду, что увеличение числа `nbins_cats` может существенно повлиять на переобучение.

Качество случайного леса также зависит от способа обработки категориальных предикторов. Одним из преимуществ деревьев решений и случайных лесов является их способность работать как с количественными, так и с категориальными переменными напрямую, без необходимости one-hot-кодирования, которое обычно требуется, например, обобщенным линейным моделям и нейронным сетям. Реализации случайного леса в пакетах R `randomForest`, `ranger` и `h2o` обрабатывают категориальные переменные по принципу «как есть», то есть каждое значение категориальной переменной представляет собой уровень (категорию) переменной. Однако в питоновской библиотеке `scikit-learn` каждый уровень категориальной переменной должен быть представлен дамми-переменной. Для этого нужно выполнить one-hot-кодирование. Помимо того, что one-hot-кодирование может привести к огромному увеличению размерности пространства данных, применительно к деревьям решений и случайному лесу оно стирает важную информацию о структуре категориального признака, по сути разбив один цельный признак на множество отдельных бинарных признаков. Бинарный признак может быть разбит только одним способом, а категориальный признак с k уровнями может быть разбит $2^{k-1}-1$ способами. Таким образом, в полученном пространстве признаков количественные переменные получат большую важность, чем категориальные переменные, представленные бинарными признаками. Все это может привести к ухудшению качества модели. Поэтому при построении случайного леса на данных, содержащих большое количество категориальных переменных, бывает полезно сравнить результаты,

полученные с помощью традиционных классов `RandomForestClassifier`/`RandomForestRegressor` и класса `H2ORandomForestEstimator`.

Помимо того, что библиотека H2O обрабатывает категории по принципу «как есть», она предлагает и другие способы кодировки категориальных предикторов, что позволяет в ряде случаев вовысить качество модели. Кодировку категориального предиктора можно изменить с помощью параметра `categorical_encoding`. По умолчанию используется значение `auto`, которое использует кодировку `Enum`. Кодировка `Enum` обрабатывает категории напрямую, то есть для каждого категориального предиктора создается по одному столбцу. При этом под капотом категориям в лексикографическом порядке будут присвоены целочисленные значения. Допустим, у нас есть переменная *Class*. Если у нее есть уровни A, B, C, D, E, F, G, то внутренне им будут присвоены целочисленные значения: A – 0, B – 1, C – 2, D – 3, E – 4, F – 5, G – 6. Кодировка `OneHotExplicit` задает N+1 столбцов для каждого категориального признака с N уровнями. Дополнительный столбец создается для пропущенных значений. Кодировка `Binary` задает не более 32 столбцов для категориального признака (используется хеширование). Кодировка `Eigen` выполняет one-hot-кодирование и оставляет *k* первых главных компонент для категориального признака.

Кодировка `EnumLimited` использует для каждого категориального при знака только *k* самых часто встречающихся категорий, *k* определяется параметром `max_categorical_levels`. Допустим, у нас есть переменная с 4 категориями A, B, C и D. `EnumLimited` посчитает долю наблюдений в каждой категории: A – 0,12, B – 0,51, C – 0,23, D – 0,14. Затем `EnumLimited` отсортирует категории: B – 0,51, C – 0,23, D – 0,14, A – 0,12. Если *k* = 2, `EnumLimited` будет использовать категории B и C.

В случае использования кодировки `LabelEncoder` категории в лексикографическом порядке будут преобразованы в целочисленные значения (начиная с 0) и в итоге теряют свою категориальную природу. Таким образом, с помощью `LabelEncoder` мы получаем количественную переменную. Допустим, у нас есть переменная с 4 категориями A, B, C и D. Тогда `LabelEncoder` присвоит A – 0, B – 1, C – 2, D – 3.

Кодировка `SortByResponse` каждой категории, отсортированной по среднему значению зависимой переменной в уровне предиктора, сопоставляет целое число, начиная с 0. Допустим, у нас есть переменная с 4 категориями A, B, C и D. Сначала `SortByResponse` посчитает среднее значение зависимой переменной для каждой категории: A – 0.5, B – 0.25, C – 0.43, D – 0.1. Потом `SortByResponse` отсортирует категории по возрастанию: D – 0.1, B – 0.25, C – 0.43, A – 0.5. Затем `SortByResponse` каждой категории присвоит порядковый номер, начиная с 0: D – 0, B – 1, C – 2, A – 3.

Поскольку случайный лес использует рандомизацию, установка различных стартовых значений генератора случайных чисел (или вообще отказ от использования стартового значения) может кардинально изменить построение модели. Существует даже шутка, которую опытные моделеры часто отпускают в адрес новичков: «если не удалось повысить качество модели за счет увеличения количества деревьев и количества отбираемых признаков, попробуй стартовое значение». Чем больше деревьев в лесу, тем более устойчивым он будет к изменению стартового значения. Однако если вы хотите получить результаты, которые потом нужно будет воспроизвести, то важно перед построением модели зафиксировать стартовое значение.

Лекция 4.4. Важность предикторов

Случайный лес обладает возможностью оценивать важность отдельного предиктора с точки зрения улучшения классификации и прогнозирования. Первая мера важности – это усредненное уменьшение неоднородности, вторая – усредненное уменьшение правильности.

4.4.1. Важность предиктора на основе усредненного уменьшения неоднородности

В основе расчета важности переменных лежит критерий уменьшения неоднородности в узлах-потомках дерева.

В деревьях классификации оценивается уменьшение неоднородности распределения категорий зависимой переменной при разбиении родительского узла на узлы-потомки. Как уже говорилось в модуле 3, однородным узлом является тот, в котором все наблюдения относятся к одной и той же категории зависимой переменной, в то время как узел с максимальной неоднородностью содержит равное количество наблюдений во всех категориях зависимой переменной. Для расчета неоднородности в деревьях классификации используется уже знакомая мера Джини.

В деревьях регрессии под уменьшением неоднородности понимается уменьшение разброса значений зависимой переменной относительно среднего значения при разбиении родительского узла на узлы-потомки. Здесь уже вместо меры Джини используется среднеквадратичная ошибка или девианс – сумма квадратов остатков. Если эти метрики трактовать с точки зрения неоднородности, то абсолютно однородным узлом является узел, в котором все наблюдения имеют одинаковые значения зависимой переменной, в то время как узлом с высоким значением неоднородности (в случае количественной зависимой переменной ограничения

максимально возможного значения неоднородности не существует) является узел, включающий наблюдения с сильно различающимися значениями зависимой переменной. Уменьшение неоднородности еще называют улучшением.
Алгоритм вычисления важности предиктора на основе усредненного уменьшения неоднородности выглядит так:
1. Для каждого дерева случайного леса вычисляем сумму уменьшений неоднородности (улучшений) на всех ветвлениях, связанных с данным предиктором.
2. Итоговую сумму уменьшений неоднородности, полученную по ансамблю, усредняем путем деления на общее количество деревьев.
3. Вышеописанные шаги повторяем для всех остальных предикторов.
Наиболее важный предиктор – тот, который дает наибольшее усредненное уменьшение неоднородности (для деревьев классификации – уменьшение меры Джини, для деревьев регрессии – уменьшение суммы квадратов остатков).
Вновь вспомним про недостаток важности на основе уменьшения неоднородности. По сути важность складывается из частоты использования переменной в качестве предиктора разбиения, то есть наиболее важными будут переменные, по которым можем быть рассмотрено больше вариантов разбиения и у них больше шансов стать предиктором разбиения. Поэтому наиболее важными переменными чаще будут переменные с большим количеством уникальных значений.

4.4.2. Важность предиктора на основе усредненного уменьшения качества прогнозирования

Кроме уменьшения важности на основе усредненного уменьшения неоднородности Лео Брейман предложил алгоритм вычисления важности предиктора на основе усредненного уменьшения качества прогнозирования. Для задачи классификации вычисляем усредненное уменьшении правильности – количестве правильно классифицированных наблюдений от общего количества наблюдений. Для задачи регрессии вычисляется усредненное увеличение среднеквадратичной ошибки. Рассмотрим подробнее вычисление важности на основе усредненного уменьшения правильности.
1. Для каждого дерева классификации случайного леса берем out-of-bag выборку (наблюдения, не попавшие в бутстреп-выборку, по которой строилось данное дерево).
Допустим, у нас есть набор данных из 10 наблюдений. Наблюдение может принадлежать либо классу N, либо классу P. Мы построили ансамбль из 5 деревьев. По каждому из 5 деревьев получаем out-of-bag

выборку. Например, для дерева I out-of-bag выборкой будут наблюдения 2, 4, 5, 6.

Исх. выборка	1	2	3	4	5	6	7	8	9	10
Фактический класс	**N**	**P**	**P**	**N**	**N**	**P**	**N**	**N**	**N**	**P**

											Out-of-bag выборки
Дерево I											
Б-выборка I	10	9	7	8	1	3	9	10	10	7	2 4 5 6
Дерево II											
Б-выборка II	4	8	5	8	3	9	2	6	1	6	7 10
Дерево III											
Б-выборка III	6	2	6	10	2	10	3	6	5	1	4 7 8 9
Дерево IV											
Б-выборка IV	6	7	8	10	6	10	9	10	8	2	1 3 4 5
Дерево V											
Б-выборка V	5	8	1	8	5	7	10	1	10	9	2 3 4 6

Рис. 4.16 Определение out-of-bag выборок

2. Для каждой out-of-bag выборки вычисляем правильность. Считаем количество раз, когда спрогнозированный класс для наблюдения out-of-bag выборки совпал с фактическим, и делим на размер out-of-bag выборки.
Например, для out-of-bag выборки 1 мы получаем 3 верных ответа и делим на 4 наблюдения, получаем правильность 3/4=0,75.

Out-of-bag выборка 1	2	4	5	6	
Фактический класс	**P**	**N**	**N**	**P**	
Спрогнозированный класс до пермутации	**P**	**P**	**N**	**P**	
Верные ответы до пермутации (отмечены X)	**X**		**X**	**X**	**правильность 3/4=0,75**

. . . . . .

Рис. 4.17 Вычисление правильности для каждой out-of-bag выборки

2. В каждой out-of-bag выборке осуществляем случайную перестановку значений предиктора и вычисляем правильность в каждой out-of-bag выборке с перестановленными значениями предиктора.
Например, для out-of-bag выборки 1 после пермутации мы получаем 2 верных ответа и делим на 4 наблюдения, получаем правильность 2/4=0,5.

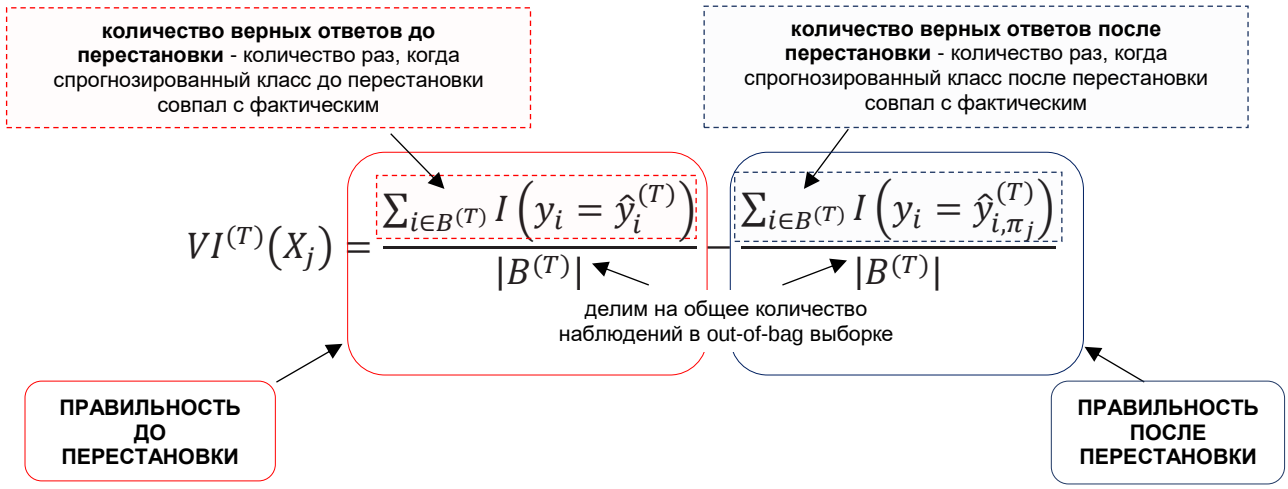
Out-of-bag выборка 1	2	4	5	6	
Фактический класс	**P**	**N**	**N**	**P**	
Спрогнозированный класс после пермутации	**N**	**P**	**N**	**P**	
Верные ответы после пермутации (отмечены X)			**X**	**X**	**правильность 2/4=0.5**

. . . . . .

Рис. 4.18 Вычисление правильности для каждой out-of-bag выборки после пермутации

4. Вычисляем разность между правильностью с исходными значениями предиктора и правильностью с перестановленными значениями предиктора в каждой out-of-bag выборке.
5. Суммируем разности по out-of-bag выборкам и делим на количество деревьев. Получаем сырое значение важности переменной.
6. Сырое значение важности переменной нормализуем путем деления на стандартную ошибку.
7. Повторяем шаги 2-5 для всех остальных предикторов.
На рис. 4.19 приводится математический аппарат вычисления важности на основе усредненного уменьшения правильности.

1. Вычисляем важность переменной как уменьшение правильности после перестановки в каждой out-of-bag выборке



где:
$VI^{(T)}(X_j)$ – это важность переменной X_j для дерева T ;
$|B^{(T)}|$ – это out-of-bag выборка для дерева T ;
y_i – фактический класс зависимой переменной;
$\hat{y}_i^{(T)}$ – спрогнозированный класс зависимой переменной перед перестановкой значений предиктора;
$\hat{y}_{i,\pi_j}^{(T)}$– спрогнозированный класс зависимой переменной после перестановки значений предиктора.

2. Вычисляем сырую важность переменной по ансамблю, просуммировав важности по всем out-of-bag выборкам и усреднив по всем деревьям

$$VI(X_j) = \frac{\sum_{T=1}^{N} VI^{(T)}(X_j)}{N}$$

3. Вычисляем нормализованную важность переменной по ансамблю

$$z_j = \frac{VI(X_j)}{\frac{\sigma}{\sqrt{N}}}$$

Рис. 4.19 Математический аппарат вычисления сырой и нормализованной важности предиктора на основе усредненного уменьшения правильности

Логическое объяснение алгоритма перестановки состоит в следующем: случайно переставляя значения предиктора, мы разрушаем взаимосвязь между ним и зависимой переменной. При использовании перестановленных значений предиктора (вместе с неперестановленными значениями остальных предикторов) для прогнозирования/ классификации по out-of-bag данным правильность существенно уменьшается (а среднеквадратичная ошибка существенно увеличивается), если между исходным предиктором и зависимой переменной была взаимосвязь. Поэтому чем больше уменьшение

правильности (увеличение среднеквадратичной ошибки), тем важнее предиктор. Важность на основе усредненного уменьшения правильности/увеличения среднеквадратичной ошибки в результате перестановок еще называют пермутированной важностью.
Обратите внимание, что при наличии высоко коррелированных предикторов обе метрики важности не способны определить релевантные переменные. Кроме того, обнаружив предикторы с небольшими значениями важностей, не спешите их удалять, посмотрите, как они будут работать, если включить дополнительные переменные. Необходимо понимать, что вычисляемая важность показывает, как данный предиктор работает не по отдельности, а в сочетании с другими переменными. Может оказаться, что предиктор, который является маловажным при совместном использовании с одними переменными, в сочетании с другими переменными станет важным.

Лекция 4.5. Графики частной зависимости

Несмотря на то что важность переменных несет ценную информацию, не меньший интерес представляет взаимосвязь предиктора с зависимой переменной. Метод частной зависимости довольно прост, основывается на прогнозах, получаемых с помощью случайного леса, и позволяет визуализировать взаимосвязь между зависимой переменной и предикторами. Основная идея заключается в том, чтобы изучить взаимосвязь между конкретным предиктором и зависимой переменной при условии, что все остальные предикторы остаются неизменными. Поскольку случайный лес может аппроксимировать практически любую функциональную зависимость между откликом и предиктором, с помощью графика можно обнаружить нелинейные зависимости, не выдвигая предварительных гипотез о характере взаимосвязей. Это особенно ценно, когда у нас отсутствует какая-либо априорная информация о виде распределения данных.
График частной зависимости строится следующим образом.
1. Для каждого значения интересующего предиктора создается специальный набор данных, в котором всем наблюдениям присваивается одно и то же значение интересующего предиктора, а все остальные предикторы фиксируются в своих текущих значениях. Например, если предиктор – это возраст в годах и есть 40 уникальных значений этой переменной, будет создано 40 специальных наборов данных, по одному для каждого уникального значения возраста. В каждом наборе во всех наблюдениях возраст принимает одно из 40 уникальных значений (например, все наблюдения получают значение возраста 21), независимо от того, так ли это на самом деле или нет. Остальные предикторы фиксируются в своих текущих значениях. Важно понять, что значения

остальных предикторов в специальных наборах не меняются и в этом смысле остаются неизменными.

2. Затем этот специальный набор, соответствующий конкретному значению интересующего предиктора (например, набор для значения возраста 26), прогоняется через случайный лес и получаем прогноз для каждого наблюдения набора.

3. Усредняем эти прогнозы по всем наблюдениям и получаем единый прогноз для набора в целом. Для количественной зависимой переменной таким прогнозом будет усредненное значение зависимой переменной. Для категориальной зависимой переменной прогнозом становится разность между логарифмом доли голосов, поданных деревьями за интересующий класс зависимой переменной, и усредненным логарифмом голосов, поданных деревьями за каждый класс. Он вычисляется по формуле:

$$f(x) = \log[p_k(x)] - \frac{1}{K}\sum_{j=1}^{K}\log[p_j(x)]$$

где x – предиктор, для которого строится график частной зависимости, K – количество классов, k – интересующий класс и p_j – доля голосов, поданных за класс j.

4. Повторяем шаги 2-3 для остальных значений интересующего предиктора.

5. Строим график, по оси абсцисс откладываем значения интересующего предиктора, по оси ординат – прогнозы.

6. Повторяем вышеописанные шаги для каждого предиктора.

Таким образом, график показывает, как значение интересующей переменной влияет на прогнозы модели при том, что все остальные переменные фиксируются и рассматриваются как константы.

Итак, графики частной зависимости можно построить как для количественной, так и для категориальной зависимой переменной. Однако если в случае с количественной зависимой переменной значения, отложенные по оси y – это естественные значения зависимой переменной, то в случае с категориальной зависимой переменной значения отклика, отложенные по вертикальной оси, выглядят необычно и могут легко сбить с толку.

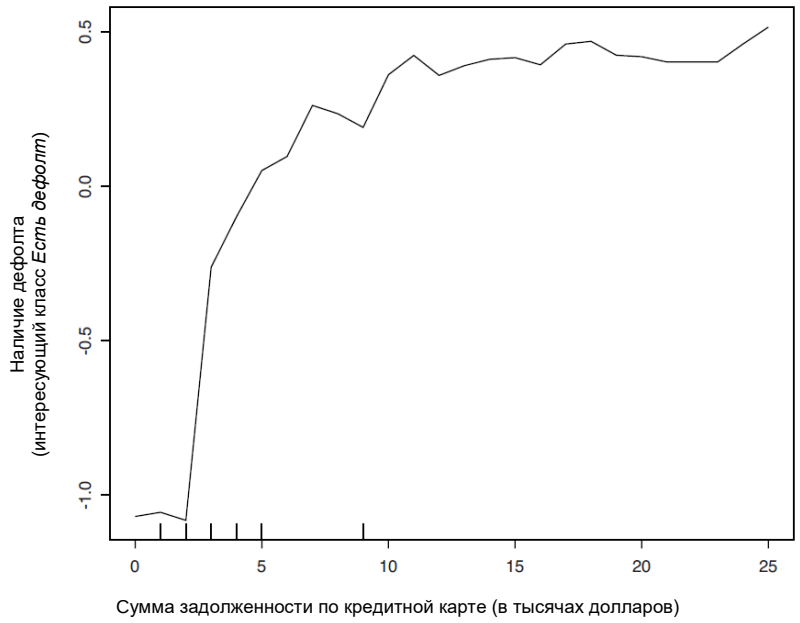


Рисунок 4.20 График частной зависимости для категориальной зависимой переменной

Допустим, у нас есть предиктор *Сумма задолженности по кредитной карте (в тысячах долларов)* и бинарная зависимая переменная *Наличие дефолта*, которая принимает два значения *Нет дефолта* и *Есть дефолт*. Интересующим классом является класс *Есть дефолт*. Для значения суммы задолженности, равного 1, доля голосов, отданных за класс *Есть дефолт*, оказалась равна 0,11. Если доля голосов, поданных за класс *Есть дефолт*, равна 0,11, то значение, откладываемое по оси y, будет равно $\log(0{,}11)-[\log(0{,}11)+\log(0{,}89)]/2 = -1{,}04537$ (используются натуральные логарифмы). Точно такой же подход можно применить, чтобы понять какое значение по оси y будет отложено, если интересующим классом станет класс *Нет дефолта*. Доля голосов, поданных за класс *Нет дефолта*, равна 0,89, таким образом, значение, откладываемое по оси y, равно $\log(0{,}89)-[\log(0{,}89)+\log(0{,}11)]/2 = 1{,}04537$. В случае бинарной классификации график частной зависимости для одного класса зависимой переменной является зеркальным отражением графика частной зависимости для другого класса.

В бинарной классификации вычисления логитов не представляют трудностей. Значение, которое мы вычисляем с помощью вышеприведенного уравнения – это половина обычного логарифма отношения шансов. Поэтому легко получить привычные вероятности. Например, умножаем -1,04537 на 2, экспоненцирование дает нам отношение шансов, равное 0,1236. Теперь умножаем 0,1236 на 0,89 и получаем значение 0,11. Мы вернулись к тому, с чего начали.

Лекция 4.6. Матрица близостей

Близости – один и наиболее полезных инструментов случайного леса. Близость между двумя наблюдениями – это частота, с которой они оба попадают в один и тот же терминальный узел.

Метод вычисления близостей включает четыре этапа:

1. Сначала все близости приравниваются к нулю.
2. После того, как дерево построено, оно применяется ко всем наблюдениям (включая обучающую и out-of-bag выборку) и для каждой пары наблюдений вычисляются близости (и так по каждому дереву).
3. Если два наблюдения попадают в один и тот же терминальный узел, их близость увеличивается на единицу.
4. Оценки близости агрегируются по всем деревьям и нормализуются путем деления на общее количество деревьев.

В итоге получаем квадратную матрицу близостей N x N. Ниже приведена матрица близостей для первых 9 наблюдений (рисунок 4.21). Значения по главной диагонали (единицы) – это «идеальные» близости наблюдений по отношению к самим себе (выделены серым фоном). Наблюдения, которые «схожи», имеют значения, близкие к 1 (выделены светло-зеленым фоном). Наблюдения, которые «непохожи», имеют значения, близкие к 0 (выделены светло-коричневым фоном).

	A	B	C	D	E	F	G	H	I	J
1	RECORD	X0000001	X0000002	X0000003	X0000004	X0000005	X0000006	X0000007	X0000008	X0000009
2	1	1	0.488	0.198	0.13	0.092	0.12	0.076	0.106	0.026
3	2	0.488	1	0.146	0.11	0.086	0.104	0.074	0.086	0.01
4	3	0.198	0.146	1	0.252	0.1	0.208	0.032	0.062	0.032
5	4	0.13	0.11	0.252	1	0.046	0.194	0.028	0.068	0.038
6	5	0.092	0.086	0.1	0.046	1	0.332	0.076	0.094	0.058
7	6	0.12	0.104	0.208	0.194	0.332	1	0.052	0.08	0.064
8	7	0.076	0.074	0.032	0.028	0.076	0.052	1	0.514	0.04
9	8	0.106	0.086	0.062	0.068	0.094	0.08	0.514	1	0.048
10	9	0.026	0.01	0.032	0.038	0.058	0.064	0.04	0.048	1
11										

Рисунок 4.21 Матрица близостей для первых 9 наблюдений

Необходимо помнить, что с матрицей близостей удобно работать, когда наборы данных невелики. Например, если набор данных превышает более 5000 наблюдений, понадобится более 25 миллионов ячеек данных. Поэтому при обработке больших массивов обычно выводится «сжатая» форма матрицы близостей – для каждого наблюдения записывается лишь M ближайших наблюдений. M обычно меньше 100.

Близости между наблюдениями образуют матрицу сходств, которая симметрична, положительно определена, каждый элемент матрицы принимает значение в интервале от 0 до 1. Матрица сходств может быть использована для выполнения многомерного шкалирования.

Классическое многомерное шкалирование – это метод обучения без учителя, задача которого – поиск и интерпретация латентных (ненаблюдаемых) переменных, которые позволяют объяснить сходства между объектами, представленными в виде точек в пространстве низкой размерности. При этом объекты размещаются в пространстве таким образом, чтобы расстояния между точками были максимально близки изначально найденным мерам сходства этих объектов. На выходе мы получаем числовые значения координат по каждому объекту в некоторой новой системе координат, в которой оси соответствуют латентным переменным.
Кроме того, матрица близостей может использоваться для импутации пропущенных значений, обнаружения выбросов.

Лекция 4.7. Обработка пропущенных значений

В случайном лесе используются два метода импутации пропущенных значений.
В экономичном методе импутации пропущенные значения непрерывных переменных заменяются медианой, а пропущенные значения категориальных переменных – модой.
Оптимальный метод импутации, более затратный по времени, включает четыре этапа:
1. Сначала используется экономичный метод для первоначальной импутации пропущенных значений.
2. Затем по этому импутированному набору данных выращивается лес деревьев.
3. Затем матрица близостей, полученная с помощью леса деревьев, используется для итеративного изменения импутированных значений.
Если предиктор является количественным, импутированное значение для наблюдения – это взвешенное среднее всех наблюдений с непропущенными значениями, в качестве весов используются близости между данным наблюдением и наблюдением с непропущенным значением. Для категориальных предикторов импутированное значение – это наиболее часто встречающееся значение (категория), где частота взвешивается по близости.
4. Шаги 2 и 3 повторяются несколько раз до достижения сходимости. Обычно 4-6 итераций достаточно.
Если задана контрольная выборка, алгоритм запускает различные варианты импутации и определяется лучший кандидат-замена для пропущенного значения.

Лекция 4.8. Обнаружение выбросов

Выбросы можно определить как наблюдения, сильно отличающиеся от основной массы элементов выборки. В случайном лесе выброс – это наблюдение, которое имеет маленькие значения близостей по отношению ко всем остальным наблюдениям. Применительно к категориальным зависимым переменным выбросы определяются внутри категорий зависимой переменной. Таким образом, выброс в конкретной категории зависимой переменной – это наблюдение, у которого маленькие значения близостей по отношению ко всем остальным наблюдениям, принадлежащим к данной категории зависимой переменной. Вычисление показателя выброса происходит следующим образом.

1. Для конкретного наблюдения n_c, принадлежащей категории c, вычисляем сумму квадратов близостей до всех остальных наблюдений, относящихся к этой же категории. Берем обратную величину суммы квадратов близостей.

$$out_{raw}(n_c) = \frac{1}{\sum_c [prox(n_c, k)]^2}$$

Показатель выброса будет большим при малом значении знаменателя. Повторяем ту же самую процедуру для всех остальных наблюдений в этой категории. Будем считать полученные значения нестандартизированными.

2. Вычисляем медиану нестандартизированных значений для категории c и среднее абсолютное отклонение относительно медианы нестандартизированных значений для категории c.

3. Вычитаем медиану из каждого нестандартизированного значения и делим на среднее абсолютное отклонение. Таким образом, стандартизируем нестандартизированные значения.

$$out(n_c) = \frac{out_{raw}(n_c) - median(c)}{deviation(c)}$$

4. Значения меньше нуля приравниваются к 0.

Вышеперечисленные шаги повторяем для всех остальных категорий зависимой переменной. Наблюдения со значениями больше 10 рассматриваются в качестве выбросов.

Лекция 4.9. Преимущества и недостатки случайного леса

Как уже говорилось в самом начале этой главы, использование ансамбля по сравнению с одиночным деревом решений дает более лучшее качество модели за счет усреднения результатов по-разному переобучающихся деревьев. Случайный лес легко настраивается, для получения модели хорошего качества требуется лишь настроить количество деревьев и количество случайно отбираемых переменных для разбиения. Другим безусловным преимуществом метода является тот факт, что случайный лес способен более точно оценить вклад и поведение каждого предиктора, даже когда эффект одного предиктора ослаблен более значимыми предикторами, что характерно для регрессионных моделей. Как и деревья решений, случайный лес не требует процедуры предварительной подготовки данных (в частности, не нужно масштабировать данные). Случайный лес можно легко распараллелить между несколькими ядрами процессора в компьютере. С помощью случайного леса можно решать задачи подготовки данных и отбора переменных, например, определить наиболее важные переменные для включения в модель логистической регрессии или импутировать пропущенные значения.

Вместе с тем случайный лес является моделью «черного ящика», мы не можем сразу взглянуть на несколько сотен деревьев и интерпретировать их. Он не дает непосредственной информации о том, как меняется значение отклика в зависимости от значения того или иного предиктора. Судить о взаимосвязях между предиктором и зависимой переменной можно лишь по графикам частной зависимости, приняв во внимание ряд ограничений. Случайный лес плохо работает на данных очень высокой размерности, разреженных данных, примером которых являются текстовые данные. Для подобного рода данных линейные модели подходят больше. Как и дерево решений, случайный лес не умеет экстраполировать данные. Случайный лес склонен к переобучению при работе с сильно зашумленными данными. Кроме того, нельзя не отметить большой размер получаемых моделей случайного леса.