Тема 5. Композиции: бэггинг, случайный лес

* [Машинное обучение](https://habrahabr.ru/hub/machine_learning/),
* [Математика](https://habrahabr.ru/hub/maths/),
* [Алгоритмы](https://habrahabr.ru/hub/algorithms/),
* [Python](https://habrahabr.ru/hub/python/),
* [Блог компании Open Data Science](https://habrahabr.ru/company/ods/)

Привет всем, кто дожил до пятой темы нашего курса!

Эту статью мы посвятим простым методам композиции: бэггингу и случайному лесу. Вы узнаете, как можно получить распределение среднего по генеральной совокупности, если у нас есть информация только о небольшой ее части; посмотрим, как с помощью композиции алгоритмов уменьшить дисперсию и таким образом улучшить точность модели; разберём, что такое случайный лес, какие его параметры нужно «подкручивать» и как найти самый важный признак. Сконцентрируемся на практике, добавив «щепотку» математики.

[Видеозапись](https://youtu.be/G0DmuuFeC30) лекции по мотивам этой статьи в рамках второго запуска открытого курса (сентябрь-ноябрь 2017).

Список статей серии

План этой статьи

1. [Бэггинг](https://habrahabr.ru/company/ods/blog/324402/#1-begging)  
   * [Ансамбли](https://habrahabr.ru/company/ods/blog/324402/#ansambli)
   * [Бутстрэп](https://habrahabr.ru/company/ods/blog/324402/#butstrep)
   * [Бэггинг](https://habrahabr.ru/company/ods/blog/324402/#begging)
   * [Out-of-bag ошибка](https://habrahabr.ru/company/ods/blog/324402/#out-of-bag-error)
2. [Случайный лес](https://habrahabr.ru/company/ods/blog/324402/#2-sluchaynyy-les)  
   * [Алгоритм](https://habrahabr.ru/company/ods/blog/324402/#algoritm)
   * [Сравнение с деревом решений и бэггингом](https://habrahabr.ru/company/ods/blog/324402/#sravnenie-s-derevom-resheniy-i-beggingom)
   * [Параметры](https://habrahabr.ru/company/ods/blog/324402/#parametry)
   * [Вариация и декорреляционный эффект](https://habrahabr.ru/company/ods/blog/324402/#variaciya-i-dekorrelyacionnyy-effekt)
   * [Смещение](https://habrahabr.ru/company/ods/blog/324402/#smeschenie)
   * [Сверхслучайные деревья](https://habrahabr.ru/company/ods/blog/324402/#sverhsluchaynye-derevya)
   * [Схожесть с алгоритмом k-ближайших соседей](https://habrahabr.ru/company/ods/blog/324402/#shozhest-sluchaynogo-lesa-s-algoritmom-k-blizhayshih-sosedey)
   * [Преобразование признаков в многомерное пространство](https://habrahabr.ru/company/ods/blog/324402/#preobrazovanie-priznakov-v-mnogomernoe-prostranstvo)
3. [Оценка важности признаков](https://habrahabr.ru/company/ods/blog/324402/#3-ocenka-vazhnosti-priznakov)
4. [Плюсы и минусы случайного леса](https://habrahabr.ru/company/ods/blog/324402/#4-plyusy-i-minusy-sluchaynogo-lesa)
5. [Домашнее задание №5](https://habrahabr.ru/company/ods/blog/324402/#5-domashnee-zadanie)
6. [Полезные источники](https://habrahabr.ru/company/ods/blog/324402/#6-poleznye-istochniki)

1. Бэггинг

Из прошлых лекций вы уже узнали про разные алгоритмы классификации, а также научились правильно валидироваться и оценивать качество модели. Но что делать, если вы уже нашли лучшую модель и повысить точность модели больше не можете? В таком случае нужно применить более продвинутые техники машинного обучения, которые можно объединить словом «ансамбли». Ансамбль — это некая совокупность, части которой образуют единое целое. Из повседневной жизни вы знаете музыкальные ансамбли, где объединены несколько музыкальных инструментов, архитектурные ансамбли с разными зданиями и т.д.

Ансамбли

Хорошим примером ансамблей считается теорема Кондорсе «о жюри присяжных» (1784). Если каждый член жюри присяжных имеет независимое мнение, и если вероятность правильного решения члена жюри больше 0.5, то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри и стремится к единице. Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.  
N — количество присяжных  
p — вероятность правильного решения присяжного  
μ — вероятность правильного решения всего жюри

μ=∑i=mNCNipi(1−p)N−i

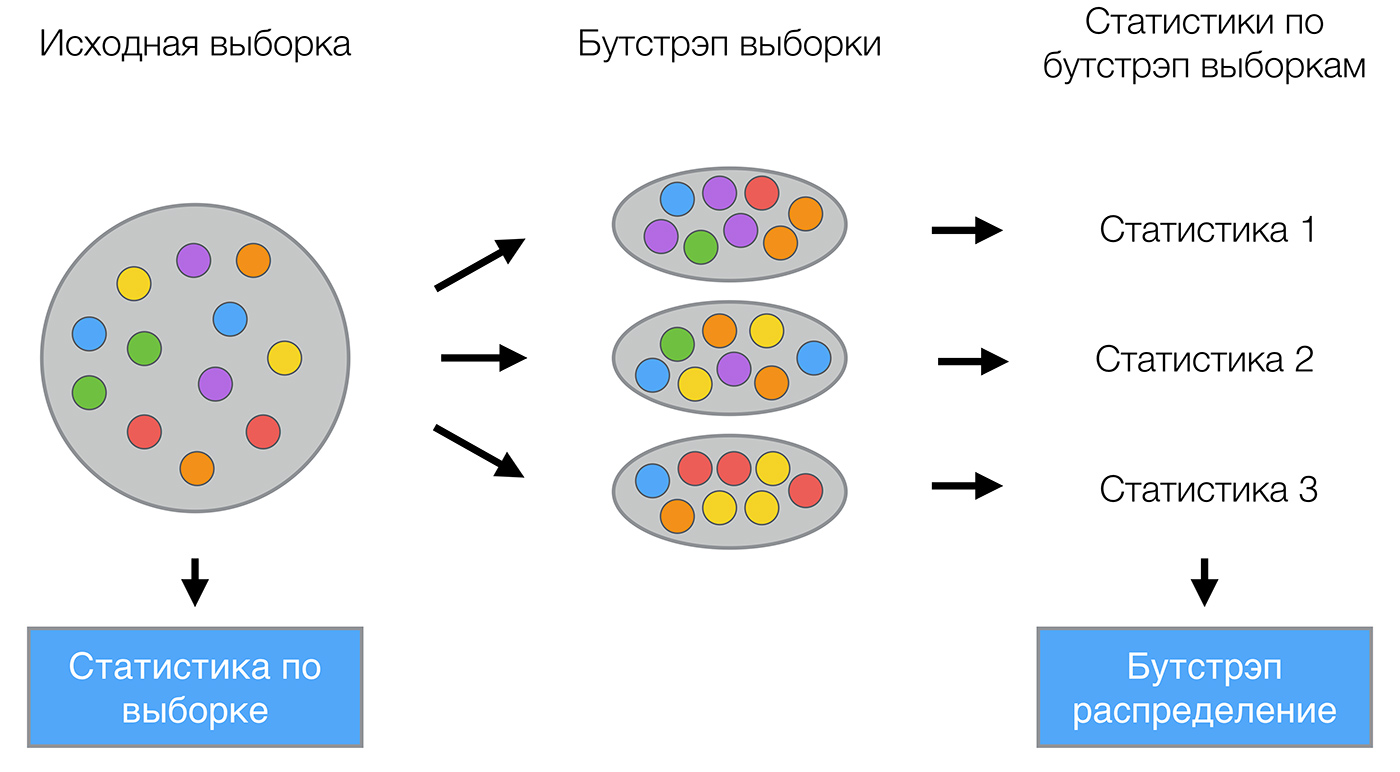
Если p>0.5, то μ>p  
Если N→∞, то μ→1

Давайте рассмотрим ещё один пример ансамблей — "Мудрость толпы". Фрэнсис Гальтон в 1906 году посетил рынок, где проводилась некая лотерея для крестьян.   
Их собралось около 800 человек, и они пытались угадать вес быка, который стоял перед ними. Бык весил 1198 фунтов. Ни один крестьянин не угадал точный вес быка, но если посчитать среднее от их предсказаний, то получим 1197 фунтов.  
Эту идею уменьшения ошибки применили и в машинном обучении.

Бутстрэп

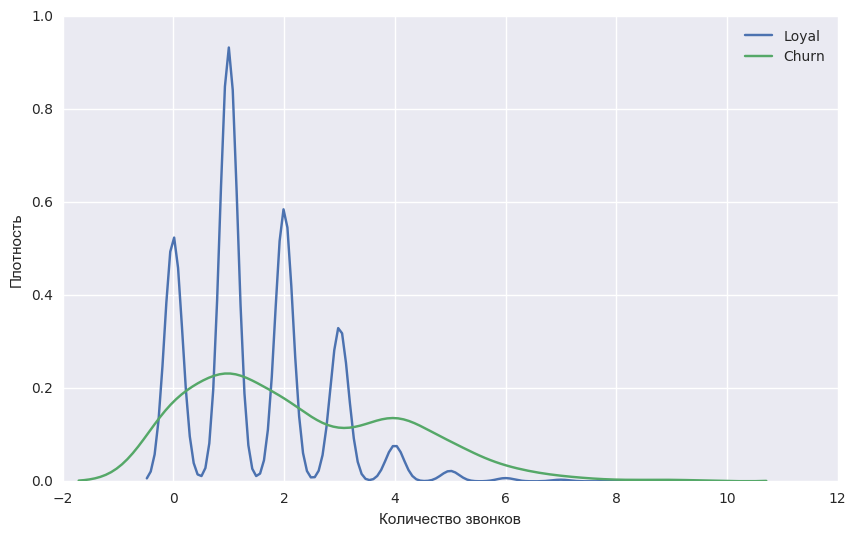
Bagging (от Bootstrap aggregation) — это один из первых и самых простых видов ансамблей. Он был придуман [Ле́о Бре́йманом](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B5%D0%B9%D0%BC%D0%B0%D0%BD,_%D0%9B%D0%B5%D0%BE) в 1994 году. Бэггинг основан на статистическом методе бутстрэпа, который позволяет оценивать многие статистики сложных распределений.

Метод бутстрэпа заключается в следующем. Пусть имеется выборка X размера N. Равномерно возьмем из выборки N объектов с возвращением. Это означает, что мы будем N раз выбирать произвольный объект выборки (считаем, что каждый объект «достается» с одинаковой вероятностью 1N), причем каждый раз мы выбираем из всех исходных N объектов. Можно представить себе мешок, из которого достают шарики: выбранный на каком-то шаге шарик возвращается обратно в мешок, и следующий выбор опять делается равновероятно из того же числа шариков. Отметим, что из-за возвращения среди них окажутся повторы. Обозначим новую выборку через X1. Повторяя процедуру M раз, сгенерируем M подвыборок X1,…,XM. Теперь мы имеем достаточно большое число выборок и можем оценивать различные статистики исходного распределения.



Давайте для примера возьмем уже известный вам набор данных telecom\_churn из прошлых уроков нашего курса. Напомним, что это задача бинарной классификации оттока клиентов. Одним из самых важных признаков в этом датасете является количество звонков в сервисный центр, которые были сделаны клиентом. Давайте попробуем визулизировать данные и посмотреть на распределение данного признака.

Код для загрузки данных и построения графика



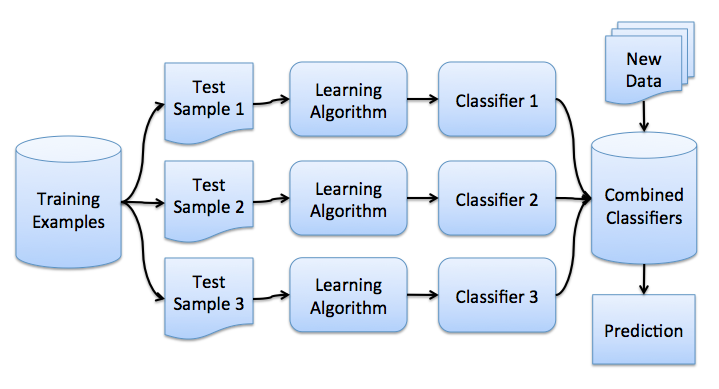
Как вы уже могли заметить, количество звонков в сервисный центр у лояльных клиентов меньше, чем у наших бывших клиентов. Теперь было бы хорошо оценить, сколько в среднем делает звонков каждая из групп. Так как данных в нашем датасете мало, то искать среднее не совсем правильно, лучше применить наши новые знания бутстрэпа. Давайте сгенерируем 1000 новых подвыборок из нашей генеральной совокупности и сделаем интервальную оценку среднего.

Код для построения доверительного интервала с помощью бутстрэпа

В итоге мы получили, что с 95% вероятностью среднее число звонков от лояльных клиентов будет лежать в промежутке между 1.40 и 1.50, в то время как наши бывшие клиенты звонили в среднем от 2.06 до 2.40 раз. Также ещё можно обратить внимание, что интервал для лояльных клиентов уже, что довольно логично, так как они звонят редко (в основном 0, 1 или 2 раза), а недовольные клиенты будут звонить намного чаще, но со временем их терпение закончится, и они поменяют оператора.

Бэггинг

Теперь вы имеете представление о бустрэпе, и мы можем перейти непосредственно к бэггингу. Пусть имеется обучающая выборка X. С помощью бутстрэпа сгенерируем из неё выборки X1,…,XM. Теперь на каждой выборке обучим свой классификатор ai(x). Итоговый классификатор будет усреднять ответы всех этих алгоритмов (в случае классификации это соответствует голосованию): a(x)=1M∑i=1Mai(x). Эту схему можно представить картинкой ниже.



Рассмотрим задачу регрессии с базовыми алгоритмами b1(x),…,bn(x). Предположим, что существует истинная функция ответа для всех объектов y(x), а также задано распределение на объектах p(x). В этом случае мы можем записать ошибку каждой функции регрессии

εi(x)=bi(x)−y(x),i=1,…,n

и записать матожидание среднеквадратичной ошибки

Ex(bi(x)−y(x))2=Exε2(x).

Средняя ошибка построенных функций регрессии имеет вид

E1=1nExεi2(x)

Предположим, что ошибки несмещены и некоррелированы:

Exεi(x)=0,Exεi(x)εj(x)=0,i≠j.

Построим теперь новую функцию регрессии, которая будет усреднять ответы построенных нами функций:

a(x)=1n∑i=1nbi(x)

Найдем ее среднеквадратичную ошибку:

En=Ex(1n∑i=1nbi(x)−y(x))2=Ex(1n∑i=1nεi)2=1n2Ex(∑i=1nεi2(x)+∑i≠jεi(x)εj(x))=1nE1

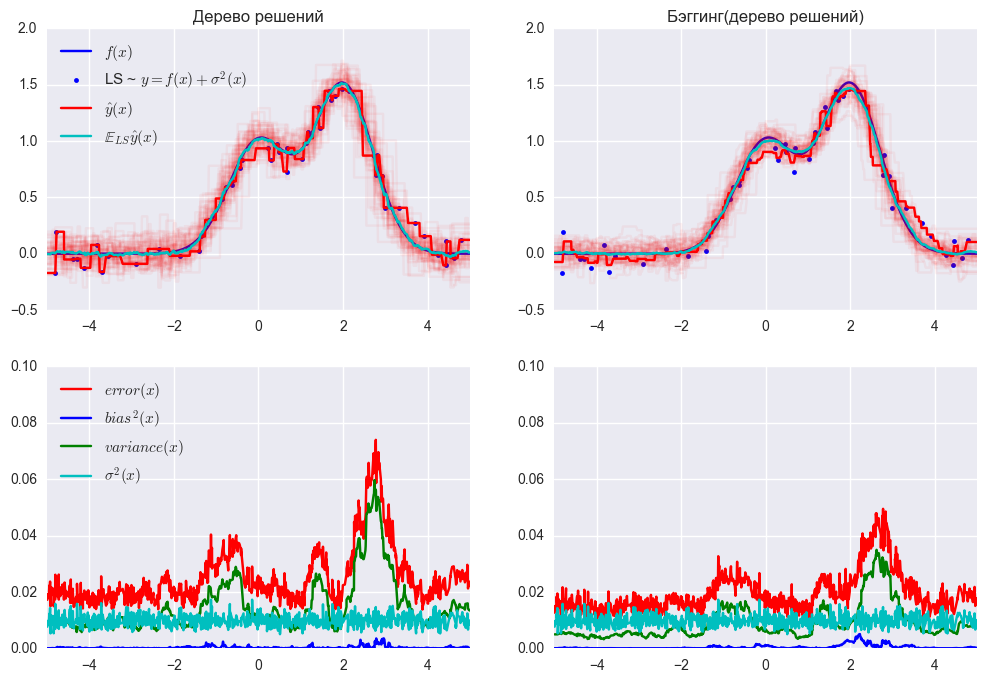
Таким образом, усреднение ответов позволило уменьшить средний квадрат ошибки в n раз!

Напомним вам из нашего предыдущего [урока](https://habrahabr.ru/company/ods/blog/323890/#razlozhenie-oshibki-na-smeschenie-i-razbros-bias-variance-decomposition), как раскладывается общая ошибка:

Err(x→)=E[(y−f^(x→))2]=σ2+f2+Var(f^)+E[f^]2−2fE[f^]=(f−E[f^])2+Var(f^)+σ2=Bias(f^)2+Var(f^)+σ2

Бэггинг позволяет снизить дисперсию (variance) обучаемого классификатора, уменьшая величину, на сколько ошибка будет отличаться, если обучать модель на разных наборах данных, или другими словами, предотвращает переобучение. Эффективность бэггинга достигается благодаря тому, что базовые алгоритмы, обученные по различным подвыборкам, получаются достаточно различными, и их ошибки взаимно компенсируются при голосовании, а также за счёт того, что объекты-выбросы могут не попадать в некоторые обучающие подвыборки.

В библиотеке scikit-learn есть реализации BaggingRegressor и BaggingClassifier, которые позволяют использовать большинство других алгоритмов "внутри". Рассмотрим на практике, как работает бэггинг, и сравним его с деревом решений, пользуясь примером из [документации](http://scikit-learn.org/stable/auto_examples/ensemble/plot_bias_variance.html#sphx-glr-auto-examples-ensemble-plot-bias-variance-py).



Ошибка дерева решений

0.0255(Err)=0.0003(Bias2)+0.0152(Var)+0.0098(σ2)

Ошибка бэггинга

0.0196(Err)=0.0004(Bias2)+0.0092(Var)+0.0098(σ2)

По графику и результатам выше видно, что ошибка дисперсии намного меньше при бэггинге, как мы и доказали теоретически выше.

Бэггинг эффективен на малых выборках, когда исключение даже малой части обучающих объектов приводит к построению существенно различных базовых классификаторов. В случае больших выборок обычно генерируют подвыборки существенно меньшей длины.

Следует отметить, что рассмотренный нами пример не очень применим на практике, поскольку мы сделали предположение о некоррелированности ошибок, что редко выполняется. Если это предположение неверно, то уменьшение ошибки оказывается не таким значительным. В следующих лекциях мы рассмотрим более сложные методы объединения алгоритмов в композицию, которые позволяют добиться высокого качества в реальных задачах.

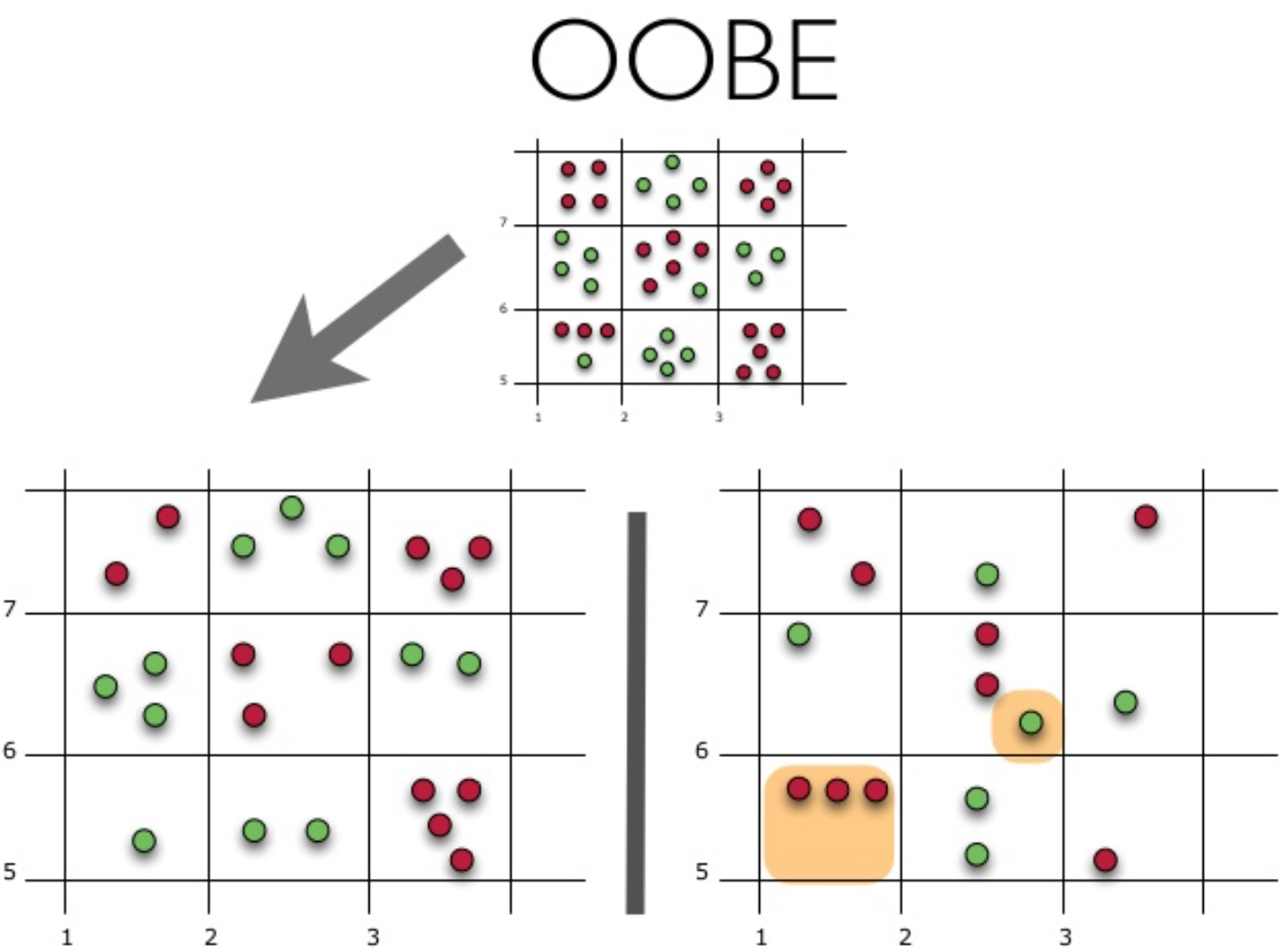
Out-of-bag error

Забегая вперед, отметим, что при использовании случайных лесов нет необходимости в кросс-валидации или в отдельном тестовом наборе, чтобы получить несмещенную оценку ошибки набора тестов. Посмотрим, как получается "внутренняя" оценка модели во время ее обучения.

Каждое дерево строится с использованием разных образцов бутстрэпа из исходных данных. Примерно 37% примеров остаются вне выборки бутстрэпа и не используются при построении k-го дерева.

Это можно легко доказать: пусть в выборке ℓ объектов. На каждом шаге все объекты попадают в подвыборку с возвращением равновероятно, т.е отдельный объект — с вероятностью 1ℓ. Вероятность того, что объект НЕ попадет в подвыборку (т.е. его не взяли ℓ раз): (1−1ℓ)ℓ. При ℓ→+∞получаем один из "замечательных" пределов 1e. Тогда вероятность попадания конкретного объекта в подвыборку ≈1−1e≈63%.

Давайте рассмотрим, как это работает на практике:

  
На рисунке изображена оценка oob-ошибки. Верхний рисунок – это наша исходная выборка, ее мы делим на обучающую(слева) и тестовую(справа). На рисунке слева у нас есть сетка из квадратиков, которая идеально разбивает нашу выборку. Теперь нужно оценить долю верных ответов на нашей тестовой выборке. На рисунке видно, что наш классификатор ошибся в 4 наблюдениях, которые мы не использовали для обучения. Значит, доля верных ответов нашего классификатора: 1115∗100%=73.33%

Получается, что каждый базовый алгоритм обучается на ~63% исходных объектов. Значит, на оставшихся ~37% его можно сразу проверять. Out-of-Bag оценка — это усредненная оценка базовых алгоритмов на тех ~37% данных, на которых они не обучались.

2. Случайный лес

Лео Брейман нашел применение бутстрэпу не только в статистике, но и в машинном обучении. Он вместе с Адель Катлер усовершенстовал алгоритм случайного леса, предложенный [Хо](http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf), добавив к первоначальному варианту построение некоррелируемых деревьев на основе CART, в сочетании с методом случайных подпространств и бэггинга.

Решающие деревья являются хорошим семейством базовых классификаторов для бэггинга, поскольку они достаточно сложны и могут достигать нулевой ошибки на любой выборке. Метод случайных подпространств позволяет снизить коррелированность между деревьями и избежать переобучения. Базовые алгоритмы обучаются на различных подмножествах признакового описания, которые также выделяются случайным образом.  
Ансамбль моделей, использующих метод случайного подпространства, можно построить, используя следующий алгоритм:

1. Пусть количество объектов для обучения равно N, а количество признаков D.
2. Выберите L как число отдельных моделей в ансамбле.
3. Для каждой отдельной модели l выберите dl(dl<D) как число признаков для l. Обычно для всех моделей используется только одно значение dl.
4. Для каждой отдельной модели l создайте обучающую выборку, выбрав dl признаков из D, и обучите модель.
5. Теперь, чтобы применить модель ансамбля к новому объекту, объедините результаты отдельных Lмоделей мажоритарным голосованием или путем комбинирования апостериорных вероятностей.

Алгоритм

Алгоритм построения случайного леса, состоящего из N деревьев, выглядит следующим образом:

* Для каждого n=1,…,N:  
  + Сгенерировать выборку Xn с помощью бутстрэпа;
  + Построить решающее дерево bn по выборке Xn:  
    — по заданному критерию мы выбираем лучший признак, делаем разбиение в дереве по нему и так до исчерпания выборки  
    — дерево строится, пока в каждом листе не более nmin объектов или пока не достигнем определенной высоты дерева  
    — при каждом разбиении сначала выбирается m случайных признаков из n исходных,  
    и оптимальное разделение выборки ищется только среди них.

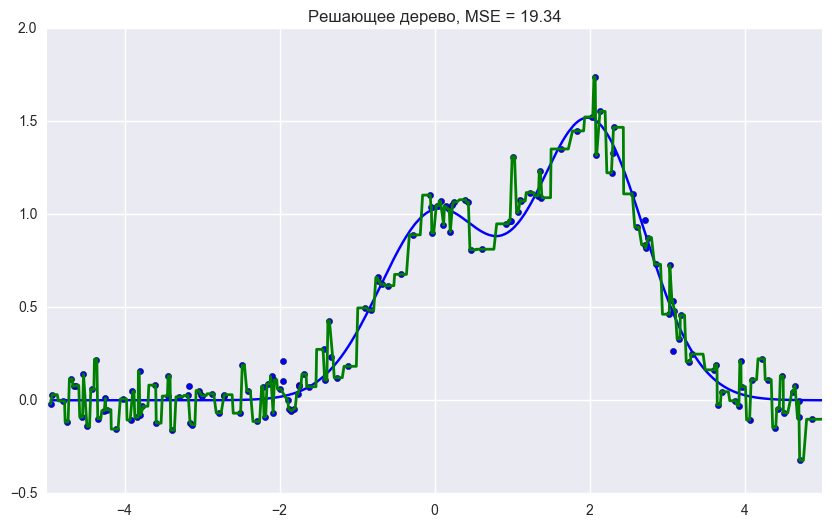
Итоговый классификатор a(x)=1N∑i=1Nbi(x), простыми словами — для задачи кассификации мы выбираем решение голосованием по большинству, а в задаче регрессии — средним.

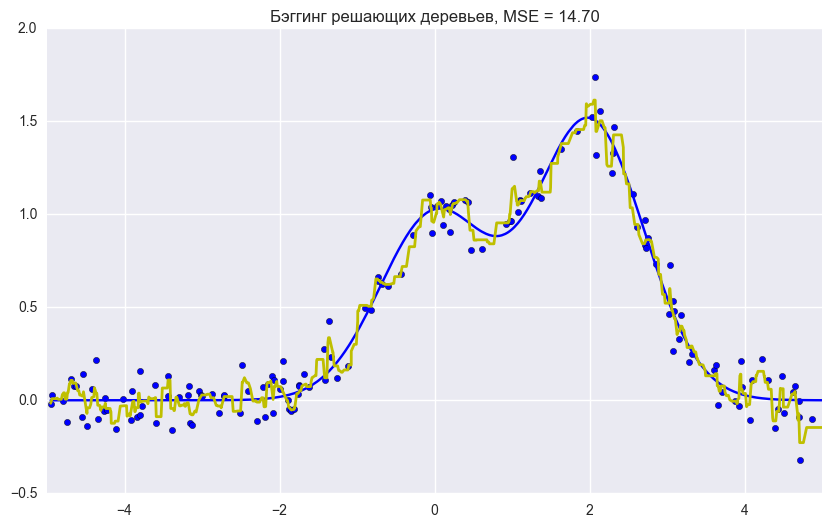
Рекомендуется в задачах классификации брать m=n, а в задачах регрессии — m=n3, где n— число признаков. Также рекомендуется в задачах классификации строить каждое дерево до тех пор, пока в каждом листе не окажется по одному объекту, а в задачах регрессии — пока в каждом листе не окажется по пять объектов.

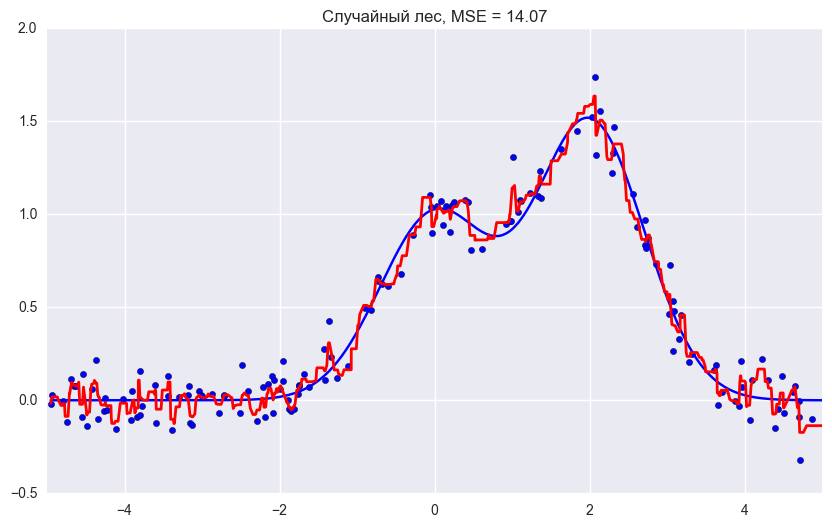
Таким образом, случайный лес — это бэггинг над решающими деревьями, при обучении которых для каждого разбиения признаки выбираются из некоторого случайного подмножества признаков.

Сравнение с деревом решений и бэггингом

Код для сравнения решающего дерева, бэггинга и случайного леса для задачи регрессии



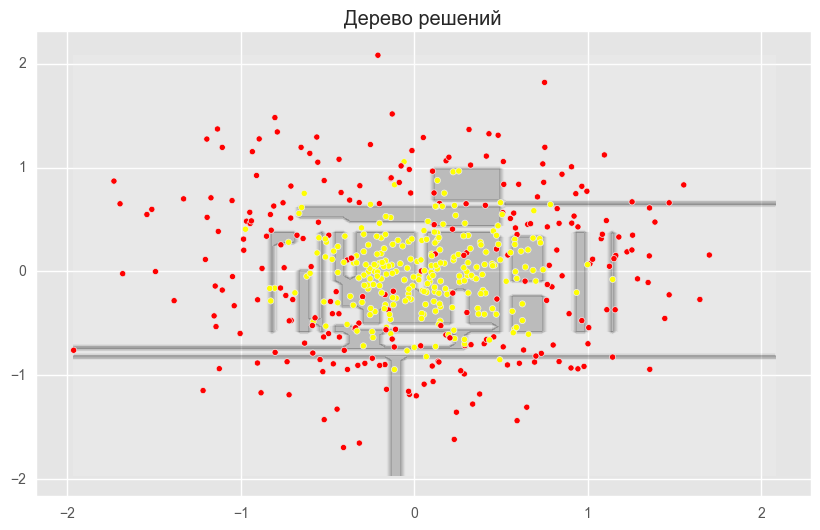


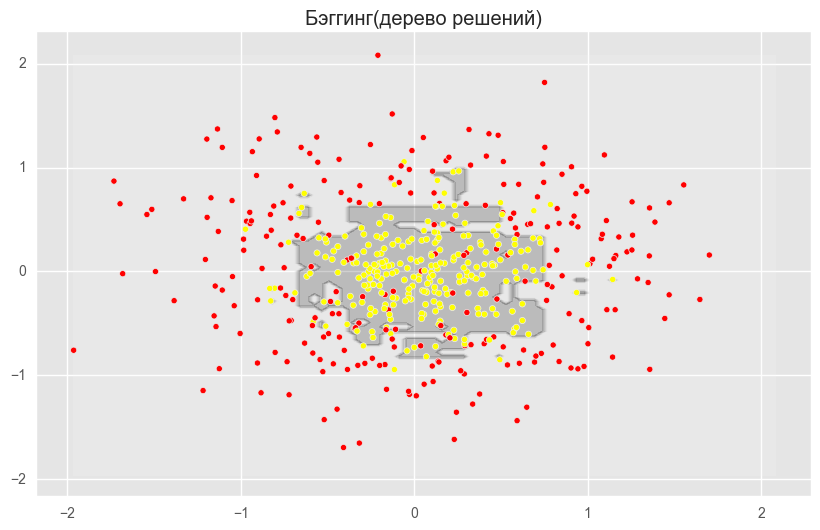


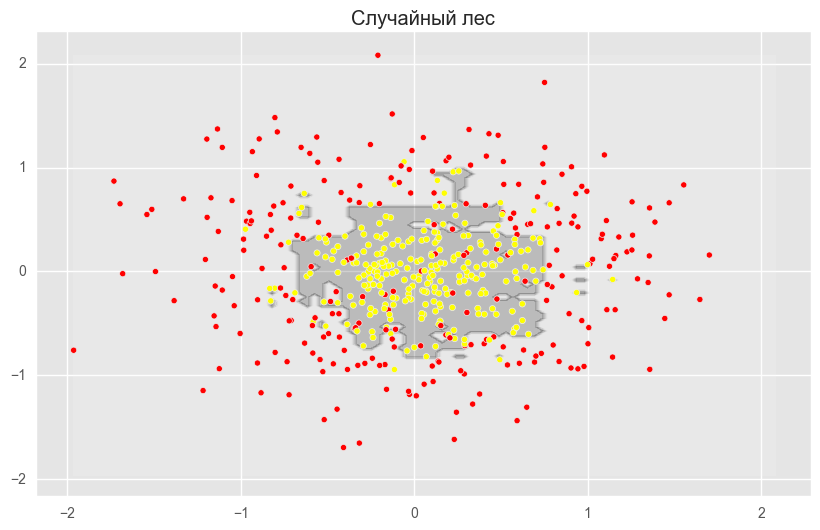
Как мы видим из графиков и значений ошибки MSE, случайный лес из 10 деревьев дает лучший результат, чем одно дерево или бэггинг из 10 деревьев решений. Основное различие случайного леса и бэггинга на деревьях решений заключается в том, что в случайном лесе выбирается случайное подмножество признаков, и лучший признак для разделения узла определяется из подвыборки признаков, в отличие от бэггинга, где все функции рассматриваются для разделения в узле.

Также можно увидеть преимущество случайного леса и бэггинга в задачах классификации.

Код для сравнения решающего дерева, бэггинга и случайного леса для задачи классификации







На рисунках выше видно, что разделяющая граница дерева решений очень «рваная» и на ней много острых углов, что говорит о переобучении и слабой обобщающей способности. В то время как у бэггинга и случайного леса граница достаточно сглаженная и практически нет признаков переобучения.

Давайте теперь попробуем разобраться с параметрами, с помощью подбора которых мы cможем увеличить долю правильных ответов.

Параметры

Метод случайного леса реализован в библиотеке машинного обучения [scikit-learn](http://scikit-learn.org/stable/) двумя классами RandomForestClassifier и RandomForestRegressor.

Полный список параметров случайного леса для задачи регрессии:

class sklearn.ensemble.RandomForestRegressor(

n\_estimators — число деревьев в "лесу" (по дефолту – 10)

criterion — функция, которая измеряет качество разбиения ветки дерева (по дефолту — "mse" , так же можно выбрать "mae")

max\_features — число признаков, по которым ищется разбиение. Вы можете указать конкретное число или процент признаков, либо выбрать из доступных значений: "auto" (все признаки), "sqrt", "log2". По дефолту стоит "auto".

max\_depth — максимальная глубина дерева (по дефолту глубина не ограничена)

min\_samples\_split — минимальное количество объектов, необходимое для разделения внутреннего узла. Можно задать числом или процентом от общего числа объектов (по дефолту — 2)

min\_samples\_leaf — минимальное число объектов в листе. Можно задать числом или процентом от общего числа объектов (по дефолту — 1)

min\_weight\_fraction\_leaf — минимальная взвешенная доля от общей суммы весов (всех входных объектов) должна быть в листе (по дефолту имеют одинаковый вес)

max\_leaf\_nodes — максимальное количество листьев (по дефолту нет ограничения)

min\_impurity\_split — порог для остановки наращивания дерева (по дефолту 1е-7)

bootstrap — применять ли бустрэп для построения дерева (по дефолту True)

oob\_score — использовать ли out-of-bag объекты для оценки R^2 (по дефолту False)

n\_jobs — количество ядер для построения модели и предсказаний (по дефолту 1, если поставить -1, то будут использоваться все ядра)

random\_state — начальное значение для генерации случайных чисел (по дефолту его нет, если хотите воспроизводимые результаты, то нужно указать любое число типа int

verbose — вывод логов по построению деревьев (по дефолту 0)

warm\_start — использует уже натренированую модель и добавляет деревьев в ансамбль (по дефолту False)

)

Для задачи классификации все почти то же самое, мы приведем только те параметры, которыми RandomForestClassifier отличается от RandomForestRegressor

class sklearn.ensemble.RandomForestClassifier(

criterion — поскольку у нас теперь задача классификации, то по дефолту выбран критерий "gini" (можно выбрать "entropy")

class\_weight — вес каждого класса (по дефолту все веса равны 1, но можно передать словарь с весами, либо явно указать "balanced", тогда веса классов будут равны их исходным частям в генеральной совокупности; также можно указать "balanced\_subsample", тогда веса на каждой подвыборке будут меняться в зависимости от распределения классов на этой подвыборке.

)

Далее рассмотрим несколько параметров, на которые в первую очередь стоит обратить внимание при построении модели:

* n\_estimators — число деревьев в "лесу"
* criterion — критерий для разбиения выборки в вершине
* max\_features — число признаков, по которым ищется разбиение
* min\_samples\_leaf — минимальное число объектов в листе
* max\_depth — максимальная глубина дерева

**Рассмотрим применение случайного леса в реальной задаче**

Для этого будем использовать пример с задачей оттока клиентов. Это задача классификации, поэтому будем использовать метрику accuracy для оценки качества модели. Для начала построим самый простой классификатор, который будет нашим бейслайном. Возьмем только числовые признаки для упрощения.

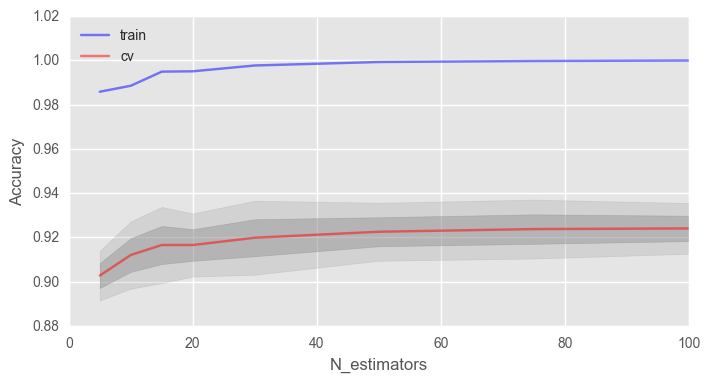
Код для построения бейслайна для случайного леса

Получили долю верных ответов 91.21%, теперь попробуем улучшить этот результат и посмотреть, как ведут себя кривые валидации при изменении основных параметров.

Начнем с количества деревьев:

Код для построения кривых валидации по подбору количества деревьев

Код для построения графика кривых валидации

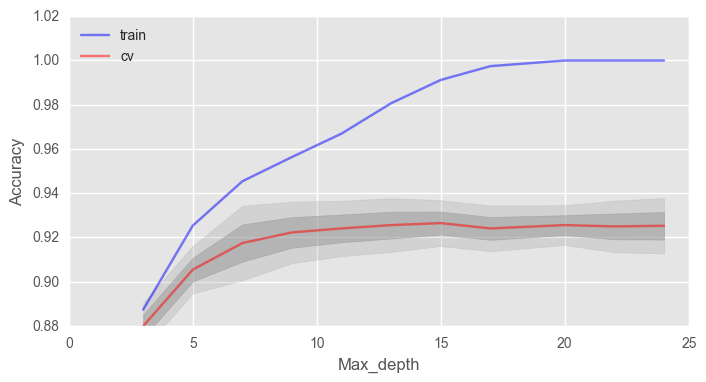


Как видно, при достижении определенного числа деревьев наша долья верных ответов на тесте выходит на асимптоту, и вы можете сами решить, сколько деревьев оптимально для вашей задачи.  
На рисунке также видно, что на тренировочной выборке мы смогли достичь 100% точности, это говорит нам о переобучении нашей модели. Чтобы избежать переобучения, мы должны добавить параметры регуляризации в модель.

Начнем с параметра максимальной глубины – max\_depth. (зафиксируем к-во деревьев 100)

Код для построения кривых обучения по подбору максимальной глубины дерева

Код для построения графика кривых обучения

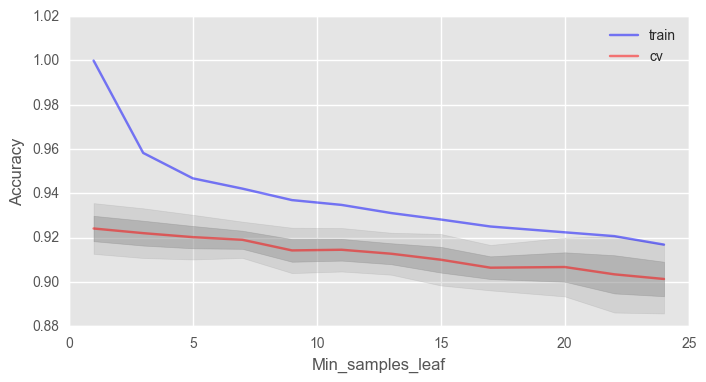


Параметр max\_depth хорошо справляется с регуляризацией модели, и мы уже не так сильно переобучаемся. Долья верных ответов нашей модели немного возросла.

Еще важный параметр min\_samples\_leaf, он также выполняет функцию регуляризатора.

Код для построения кривых валидации по подбору минимального числа объектов в одном листе дерева

Код для построения графика кривых валидации

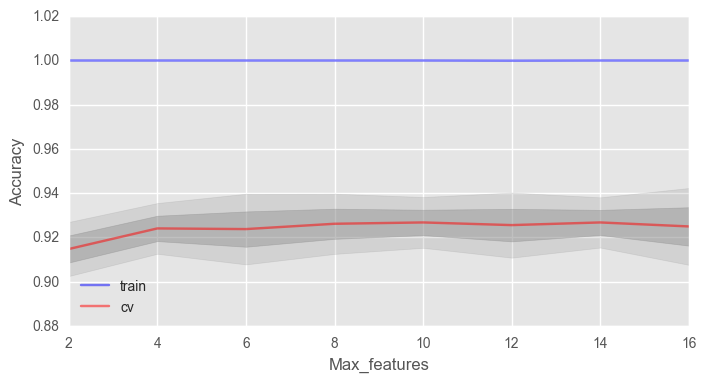


В данном случае мы не выигрываем в точности на валидации, но зато можем сильно уменьшить переобучение до 2% при сохранении точности около 92%.

Рассмотрим такой параметр как max\_features. Для задач классификации по умолчанию используется n, где n — число признаков. Давайте проверим, оптимально ли в нашем случае использовать 4 признака или нет.

Код для построения кривых валидации по подбору максимального количества признаков для одного дерева

Код для построения графика кривых валидации



В нашем случае оптимальное число признаков — 10, именно с таким значением достигается наилучший результат.

Мы рассмотрели, как ведут себя кривые валидации в зависимости от изменения основных параметров. Давайте теперь с помощью GridSearchCV найдем оптимальные параметры для нашего примера.

Код для подбора оптимальных параметров модели

Лучшая доля верных ответов, который мы смогли достичь с помощью перебора параметров — 92.83% при 'max\_depth': 15, 'max\_features': 7, 'min\_samples\_leaf': 3.

Вариация и декорреляционный эффект

Давайте запишем дисперсию для случайного леса как

Varf(x)=ρ(x)σ2(x)

Тут 

* ρ(x) – корреляция выборки между любыми двумя деревьями, используемыми при усреднении

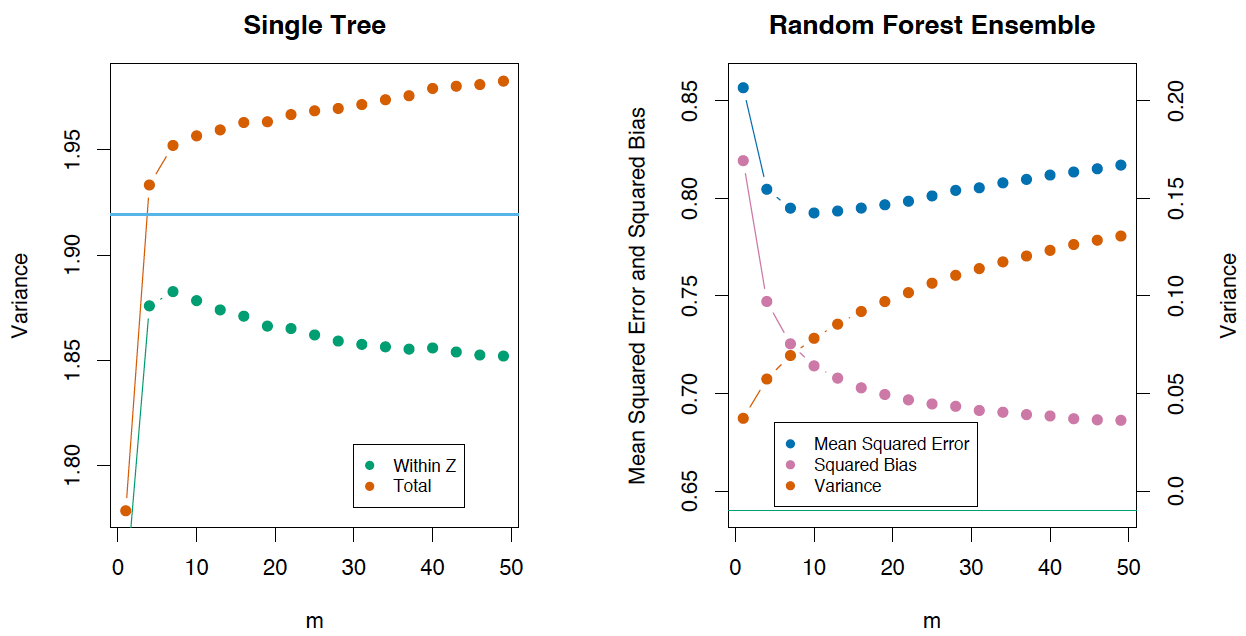
ρ(x)=corr[T(x;Θ1(Z)),T(x2,Θ2(Z))],

где Θ1(Z) и Θ2(Z) – случайно выбранная пара деревьев на случайно выбранных объектах выборки Z  
σ2(x) — это выборочная дисперсия любого произвольно выбранного дерева:

σ2(x)=VarT(x;Θ(X)

Легко спутать ρ(X) со средней корреляцией между обученными деревьями в данном случайном лесе, рассматривая деревья как N-векторы и вычисляя среднюю парную корреляцию между ними. Это не тот случай. Эта условная корреляция не имеет прямого отношения к процессу усреднения, а зависимость от x в ρ(x) предупреждает нас об этом различии. Скорее ρ(x) является теоретической корреляцией между парой случайных деревьев, оцененных в объекте x, которая была вызвана многократным сэмплированием обучающей выборки из генеральной совокупности Z, и после этого выбрана данная пара случайных деревьев. На статистическом жаргоне это корреляция, вызванная выборочным распределением Z и Θ.

По факту, условная ковариация пары деревьев равна 0, потому что бустрэп и отбор признаков — независимы и одинаково распределены.

Если рассмотреть дисперсию по одному дереву, то она практически не меняется от переменных для разделения (m), а вот для ансамбля это играет большую роль, и дисперсия для дерева намного выше, чем для ансамбля.  
В книге *The Elements of Statistical Learning (Trevor Hastie, Robert Tibshirani и Jerome Friedman)* есть отличный пример, который это демонстрирует.  


Смещение

Как и в бэггинге, смещение в случайном лесе такое же, как и смещение в отдельно взятом дереве T(x,Θ(Z)):

Bias=μ(x)−EZfrf(x)=μ(x)−EZEΘ|ZT(x,Θ(Z))

Это также обычно больше (в абсолютных величинах), чем смещение «неусеченного» (unprunned) дерева, поскольку рандомизация и сокращение пространства выборки налагают ограничения. Следовательно, улучшения в прогнозировании, полученные с помощью бэггинга или случайных лесов, являются исключительно результатом уменьшения дисперсии.

Сверхслучайные деревья

В сверхслучайных деревьях (Extremely Randomized Trees) больше случайности в том, как вычисляются разделения в узлах. Как и в случайных лесах, используется случайное подмножество возможных признаков, но вместо поиска наиболее оптимальных порогов, пороговые значения произвольно выбираются для каждого возможного признака, и наилучший из этих случайно генерируемых порогов выбирается как лучшее правило для разделения узла. Это обычно позволяет немного уменьшить дисперсию модели за счет несколько большего увеличения смещения.

В библиотеке scikit-learn есть реализация [ExtraTreesClassifier](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html" \l "sklearn.ensemble.ExtraTreesClassifier) и [ExtraTreesRegressor](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html" \l "sklearn.ensemble.ExtraTreesRegressor). Данный метод стоит использовать, когда вы сильно переобучаетесь на случайном лесе или градиентном бустинге.

Схожесть случайного леса с алгоритмом k-ближайших соседей

Метод случайного леса схож с методом ближайших соседей. Случайные леса, по сути, осуществляют предсказания для объектов на основе меток похожих объектов из обучения. Схожесть объектов при этом тем выше, чем чаще эти объекты оказываются в одном и том же листе дерева. Покажем это формально.

Рассмотрим задачу регрессии с квадратичной функцией потерь. Пусть Tn(x) — номер листа n(x)-го дерева из случайного леса, в который попадает объект x. Ответ объекта x равен среднему ответу по всем объектам обучающей выборки, которые попали в этот лист Tn(x). Это можно записать так

bn(x)=∑i=1lwn(x,xi)yi,

где

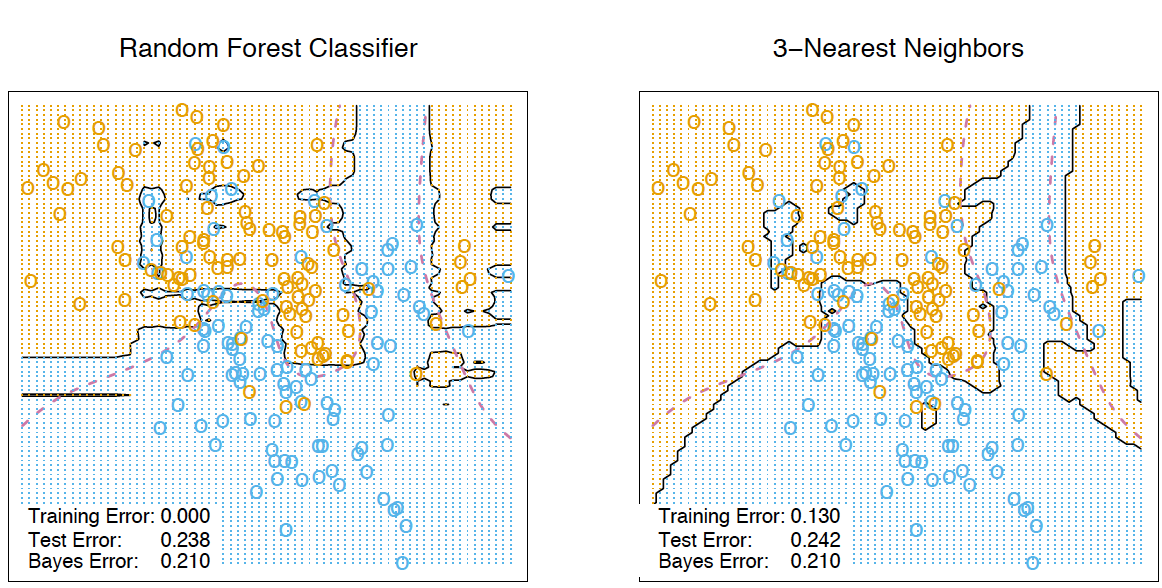
wn(x,xi)=[Tn(x)=Tn(xi)]∑j=1l[Tn(x)=Tn(xj)]

Тогда ответ композиции равен

an(x)=1N∑n=1N∑i=1lwn(x,xi)yi=∑i=1l(1N∑j=1Nwn(x,xj))yi

Видно, что ответ случайного леса представляет собой сумму ответов всех объектов обучения с некоторыми весами. Отметим, что номер листа Tn(x), в который попал объект, сам по себе является ценным признаком. Достаточно неплохо работает подход, в котором по выборке обучается композиция из небольшого числа деревьев с помощью случайного леса или градиентного бустинга, а потом к ней добавляются категориальные признаки T1(x),…,Tn(x). Новые признаки являются результатом нелинейного разбиения пространства и несут в себе информацию о сходстве объектов.

Все в той же книге *The Elements of Statistical Learning* есть хороший наглядный пример сходства случайного леса и k-ближайших соседей.



Преобразование признаков в многомерное пространство

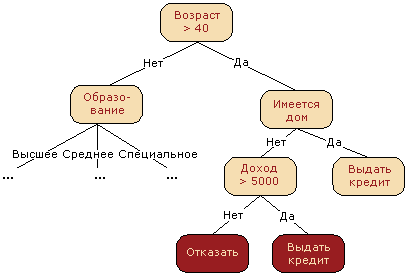
Все привыкли использовать случайный лес для задач обучения с учителем, но также есть возможность проводить обучение и без учителя. С помощью метода [RandomTreesEmbedding](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomTreesEmbedding.html" \l "sklearn.ensemble.RandomTreesEmbedding) мы можем сделать трансформацию нашего датасета в многомерное разреженное его представление. Его суть в том, что мы строим абсолютно случайные деревья, и индекс листа, в котором оказалось наблюдение, мы считаем за новый признак. Если в первый лист попал объект, то мы ставим 1, а если не попал, то 0. Так называемое бинарное кодирование. Контролировать количество переменных и также степень разреженности нашего нового представления датасета мы можем увеличивая/уменьшая количество деревьев и их глубины. Поскольку соседние точки данных скорее всего лежат в одном и том же листе дерева, преобразование выполняет неявную, непараметрическую оценку плотности.

3. Оценка важности признаков

Очень часто вы хотите понять свой алгоритм, почему он именно так, а не иначе дал определенный ответ. Или если не понять его полностью, то хотя бы какие переменные больше всего влияют на результат. Из случайного леса можно довольно просто получить данную информацию.

Суть метода

По данной картинке интуитивно понятно, что важность признака «Возраст» в задаче кредитного скоринга выше, чем важность признака «Доход». Формализуется это с помощью понятия прироста информации.



Если построить много деревьев решений (случайный лес), то чем выше в среднем признак в дереве решений, тем он важнее в данной задаче классификации/регрессии. При каждом разбиении в каждом дереве улучшение критерия разделения (в нашем случае неопределенность Джини(Gini impurity)) — это показатель важности, связанный с переменной разделения, и накапливается он по всем деревьям леса отдельно для каждой переменной.

Давайте немного углубимся в детали. Среднее снижение точности, вызываемое переменной, определяется во время фазы вычисления out-of-bag ошибки. Чем больше уменьшается точность предсказаний из-за исключения (или перестановки) одной переменной, тем важнее эта переменная, и поэтому переменные с бо́льшим средним уменьшением точности более важны для классификации данных. Среднее уменьшение неопределенности Джини (или ошибки mse в задачах регрессии) является мерой того, как каждая переменная способствует однородности узлов и листьев в окончательной модели случайного леса. Каждый раз, когда отдельная переменная используется для разбиения узла, неопределенность Джини для дочерних узлов рассчитывается и сравнивается с коэффициентом исходного узла. Неопределенность Джини является мерой однородности от 0 (однородной) до 1 (гетерогенной). Изменения в значении критерия разделения суммируются для каждой переменной и нормируются в конце вычисления. Переменные, которые приводят к узлам с более высокой чистотой, имеют более высокое снижение коэффициента Джини.

А теперь представим все вышеописанное в виде формул.

VIT=∑i∈BTI(yi=y^iT)|BT|−∑i∈BTI(yi=y^i,πjT)|BT|

y^i(T)=fT(xi) — предсказание класса перед перестановкой/удалением признака  
y^i,πj(T)=fT(xi,πj) — предсказание класса после перестановки/удаления признака  
xi,πj=(xi,1,…,xi,j−1,xπj(i),j,xi,j+1,…,xi,p)  
Заметим, что VI(T)(xj)=0, если Xj не находится в дереве T

Расчет важности признаков в ансамбле:  
— ненормированные

VI(xj)=∑T=1NVIT(xj)N

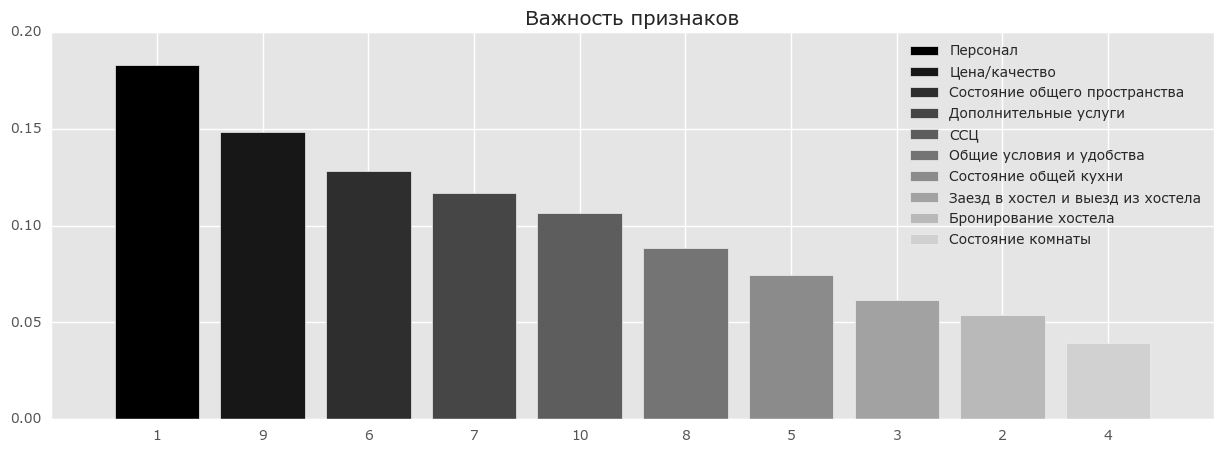
— нормированные

zj=VI(xj)σ^N

**Пример**

Рассмотрим результаты анкетирования посетителей хостелов с сайтов Booking.com и TripAdvisor.com. Признаки — средние оценки по разным факторам (перечислены ниже) — персонал, состояние комнат и т.д. Целевой признак — рейтинг хостела на сайте.

Код для оценки важности признаков

  
На рисунке выше видно, что люди больше всего обращают внимание на персонал и соотношение цена/качество и на основе впечатления от данных вещей пишут свои отзывы. Но разница между этими признаками и менее влиятельными признаками не очень значительная, и выкидывание какого-то признака приведет к уменьшению точности нашей модели. Но даже на основе нашего анализа мы можем дать рекомендации отелям в первую очередь лучше готовить персонал и/или улучшить качество до заявленной цены.

4. Плюсы и минусы случайного леса

**Плюсы**:  
— имеет высокую точность предсказания, на большинстве задач будет лучше линейных алгоритмов; точность сравнима с точностью бустинга  
— практически не чувствителен к выбросам в данных из-за случайного сэмлирования  
— не чувствителен к масштабированию (и вообще к любым монотонным преобразованиям) значений признаков, связано с выбором случайных подпространств  
— не требует тщательной настройки параметров, хорошо работает «из коробки». С помощью «тюнинга» параметров можно достичь прироста от 0.5 до 3% точности в зависимости от задачи и данных  
— способен эффективно обрабатывать данные с большим числом признаков и классов  
— одинаково хорошо обрабатывет как непрерывные, так и дискретные признаки  
— редко переобучается, на практике добавление деревьев почти всегда только улучшает композицию, но на валидации, после достижения определенного количества деревьев, кривая обучения выходит на асимптоту  
— для случайного леса существуют методы оценивания значимости отдельных признаков в модели  
— хорошо работает с пропущенными данными; сохраняет хорошую точность, если большая часть данных пропущенна  
— предполагает возможность сбалансировать вес каждого класса на всей выборке, либо на подвыборке каждого дерева  
— вычисляет близость между парами объектов, которые могут использоваться при кластеризации, обнаружении выбросов или (путем масштабирования) дают интересные представления данных  
— возможности, описанные выше, могут быть расширены до неразмеченных данных, что приводит к возможности делать кластеризацию и визуализацию данных, обнаруживать выбросы  
— высокая параллелизуемость и масштабируемость.

**Минусы**:  
— в отличие от одного дерева, результаты случайного леса сложнее интерпретировать  
— нет формальных выводов (p-values), доступных для оценки важности переменных  
— алгоритм работает хуже многих линейных методов, когда в выборке очень много разреженных признаков (тексты, Bag of words)  
— случайный лес не умеет экстраполировать, в отличие от той же линейной регрессии (но это можно считать и плюсом, так как не будет экстремальных значений в случае попадания выброса)  
— алгоритм склонен к переобучению на некоторых задачах, особенно на зашумленных данных  
— для данных, включающих категориальные переменные с различным количеством уровней, случайные леса предвзяты в пользу признаков с большим количеством уровней: когда у признака много уровней, дерево будет сильнее подстраиваться именно под эти признаки, так как на них можно получить более высокое значение оптимизируемого функционала (типа прироста информации)  
— если данные содержат группы коррелированных признаков, имеющих схожую значимость для меток, то предпочтение отдается небольшим группам перед большими  
— больший размер получающихся моделей. Требуется O(NK) памяти для хранения модели, где K — число деревьев.