Оглавление

[Random Forest и Extra Trees 2](#_Toc88908925)

[Логическая регрессия 5](#_Toc88908926)

[Naive Bayes 7](#_Toc88908927)

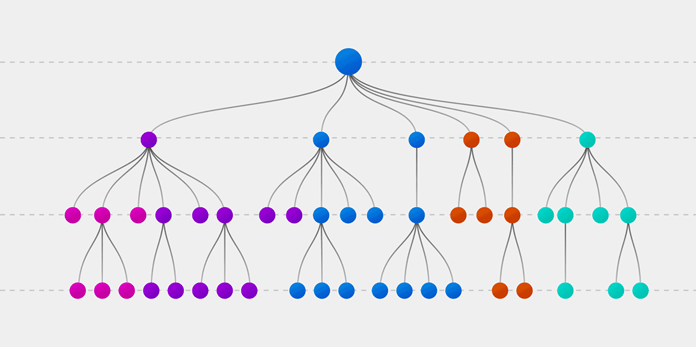
[VotingClassifier 8](#_Toc88908928)

# **Random Forest и Extra Trees**

Случайный лес — один из самых потрясающих алгоритмов машинного обучения, придуманные Лео Брейманом и Адель Катлер ещё в прошлом веке. Он дошёл до нас в «первозданном виде» (никакие эвристики не смогли его существенно улучшить) и является одним из немногих универсальных алгоритмов. Универсальность заключается, во-первых, в том, что он хорош во многих задачах (по моим оценкам, 70% из встречающихся на практике, если не учитывать задачи с изображениями), во-вторых, в том, что есть случайные леса для решения задач классификации, регрессии, кластеризации, поиска аномалий, селекции признаков и т. д.

Алгоритм состоит из четырех этапов:

1. Создание случайной выборки из заданного набора данных.
2. Для каждой выборки необходимо построить дерево решений и получить результат предсказания, используя данное дерево.
3. Провести голосование за каждый полученный прогноз.
4. Выбрать предсказание с наибольшим количеством голосов в качестве окончательного результата.



Фундаментальная концепция в основе случайного леса проста, но сильна — это мудрость толпы. Причина, по которой модель случайного леса работает так хорошо, заключается в том, что:

* Большое число относительно некоррелированных деревьев, работающих совместно, будет превосходить любую из их отдельных составляющих.

Преимущества

* Способность эффективно обрабатывать данные с большим числом признаков и классов.
* Нечувствительность к масштабированию (и вообще к любым монотонным преобразованиям) значений признаков.
* Одинаково хорошо обрабатываются как непрерывные, так и дискретные признаки. Существуют методы построения деревьев по данным с пропущенными значениями признаков.
* Существуют методы оценивания значимости отдельных признаков в модели.
* Внутренняя оценка способности модели к обобщению (тест по неотобранным образцам).
* Высокая параллелизуемость и масштабируемость.

Недостатки

* Интерпретируемость модели: модели случайных лесов не так уж и интерпретируемы они, как черные ящики.
* Для очень больших наборов данных размер деревьев может занимать много памяти.
* Он может иметь тенденцию к переоснащению, поэтому вам следует настроить гиперпараметры.

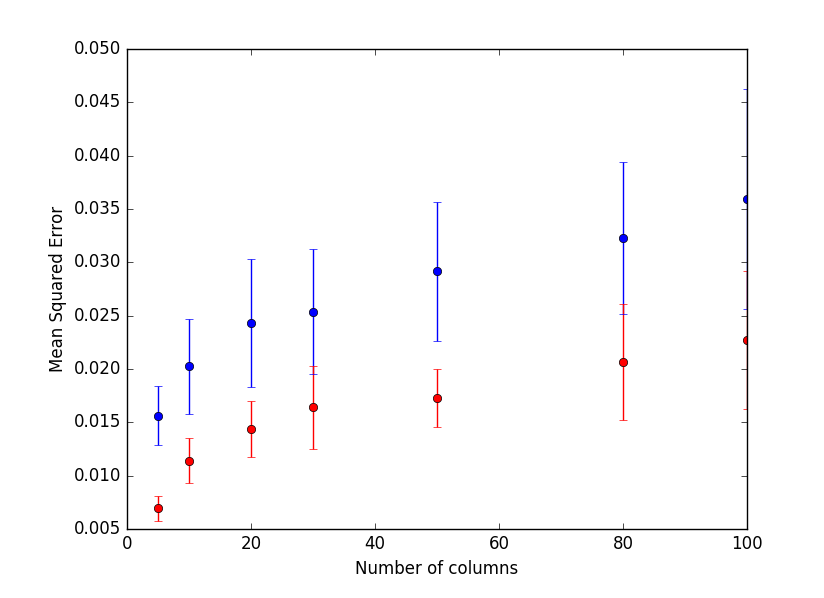
Дополнительные деревья похожи на случайный лес в том смысле, что он строит несколько деревьев и разделяет узлы, используя случайные подмножества объектов, но с двумя ключевыми отличиями: он не запускает наблюдения (то есть выборки без замены), а узлы разбиваются на случайные разбиения, не лучшие расколы. Итак, в заключение, ExtraTrees:

* строит несколько деревьев с Bootstrap = False по умолчанию, что означает, что выборки без замены
* узлы разделены на основе случайного разделения среди случайного подмножества объектов, выбранных на каждом узле

В Extra Trees случайность не возникает из-за начальной загрузки данных, а скорее из случайного разделения всех наблюдений. ExtraTrees назван в честь (чрезвычайно рандомизированных деревьев).

Изображение выглядит как стол

Автоматически созданное описание

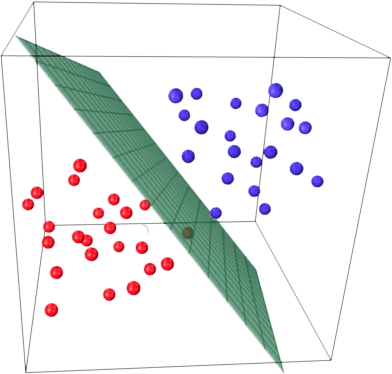


Синим цветом представлены результаты случайного леса, а красным - дополнительные деревья.

# **Логическая регрессия**

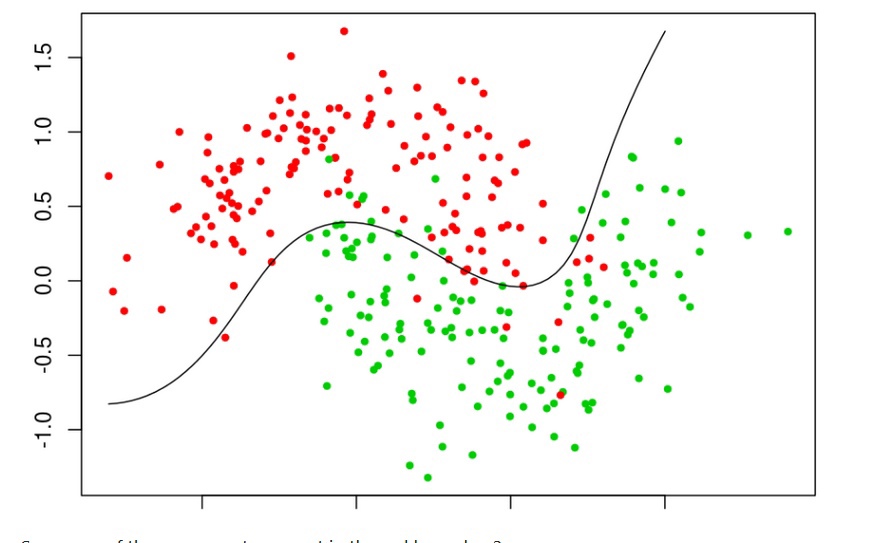
Логистическая регрессия является одним из статистических методов классификации с использованием линейного дискриминанта Фишера. Также она входит в топ часто используемых алгоритмов в науке о данных. В этой статье суть логистической регрессии описана так, что она станет понятна даже людям не очень близким к статистике.

Основная идея логистической регрессии заключается в том, что пространство исходных значений может быть разделено линейной границей (т. е. прямой) на две соответствующих классам области. Итак, что же имеется в виду под линейной границей? В случае двух измерений — это просто прямая линия без изгибов. В случае трех — плоскость, и так далее. Эта граница задается в зависимости от имеющихся исходных данных и обучающего алгоритма. Чтобы все работало, точки исходных данных должны разделяться линейной границей на две вышеупомянутых области. Если точки исходных данных удовлетворяют этому требованию, то их можно назвать линейно разделяемыми. Посмотрите на изображение.



Остался не отвеченным вопрос: «Каким образом обучается граничная функция image?» Математическая основа этого выходит за рамки статьи, но общая идея заключается в следующем:  
Рассмотрим функцию image, где image — точка данных обучающей выборки. В простой форме image можно описать так:

если image является частью класса "+", image (здесь image — выходное значение, полученное из модели логистической регрессии). Если image является частью класса "-", image.  
Функция image проводит количественную оценку вероятности того, что точка обучающей выборки классифицируется моделью правильным образом. Поэтому, среднее значение для всей обучающей выборки показывает вероятность того, что случайная точка данных будет корректно классифицирована системой, независимо от возможного класса.  
Скажем проще — механизм обучения логистической регрессии старается максимизировать среднее значение image. А название этого метода — метод максимального правдоподобия. Если вы не математик, то вы сможете понять каким образом происходит оптимизация, только если у вас есть хорошее представление о том, что именно оптимизируется.



# **Naive Bayes**

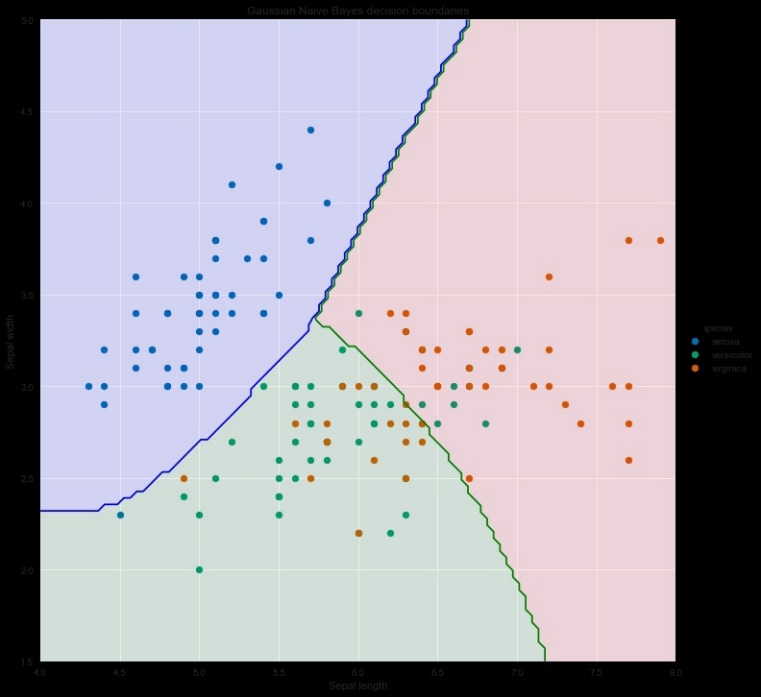
Наивный байесовский классификатор (Naive Bayes classifier) – это очень популярный в машинном обучении алгоритм, который в основном используется для получения базовой точности набора данных. Изучим его преимущества и недостатки, а также реализацию на языке Python.

**Плюсы**

* Алгоритм легко и быстро предсказывает класс тестового набора данных. Он также хорошо справляется с многоклассовым прогнозированием.
* Производительность наивного байесовского классификатора лучше, чем у других простых алгоритмов, таких как логистическая регрессия. Более того, вам требуется меньше обучающих данных.
* Он хорошо работает с категориальными признаками (по сравнению с числовыми). Для числовых признаков предполагается нормальное распределение, что может быть серьезным допущением в точности нашего алгоритма.

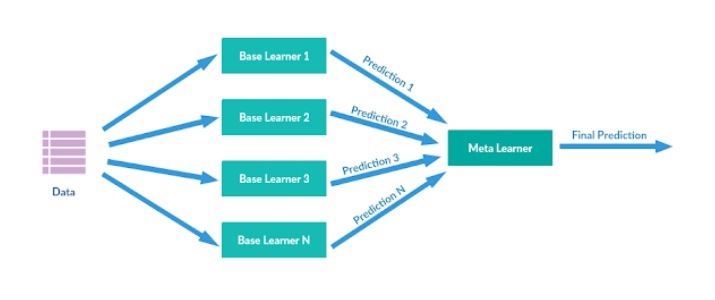
**Минусы**

* Если переменная имеет категорию (в тестовом наборе данных), которая не наблюдалась в обучающем наборе данных, то модель присвоит 0 (нулевую) вероятность и не сможет сделать предсказание. Это часто называют нулевой частотой. Чтобы решить эту проблему, мы можем использовать технику сглаживания. Один из самых простых методов сглаживания называется оценкой Лапласа.
* Значения спрогнозированных вероятностей, возвращенные методом predict\_proba, не всегда являются достаточно точными.
* Ограничением данного алгоритма является предположение о независимости признаков. Однако в реальных задачах полностью независимые признаки встречаются крайне редко.



# **VotingClassifier**

VotingClassifier является интегрированным алгоритмом. Этот алгоритм использует несколько моделей машинного обучения и выбирает результаты путем голосования. Мы используем вспомогательные векторные машины, knn, random forest. Существует два метода мягкого и жесткого голосования: мягкое голосование - взвешенное голосование, которое может дать разным алгоритмам разные весовые коэффициенты.



Вывод:

Проанализировав все алгоритмы, для поиска dga, мы пришли к выводу, что наилучшим алгоритмом по следующим пунктам:

* Эффективность
* Скорость обучения
* Надёжность

является алгоритм Random Forest.