Ну и давайте вспомни что мы проходили на прошлом занятии, обучили простенькую модель линейной регресии.

Сейчас поговорим о классификации и попытаемся уже с ней поработать.

Ну давайте подробнее поговорим о классификации.

Классификация вещей — самая популярная задача во всём машинном обучении. Машина в ней как ребёнок, который учится раскладывать игрушки: роботов в один ящик, танки в другой. Опа, а если это робот-танк? Штош, время расплакаться и выпасть в ошибку.

Для классификации всегда нужен учитель — размеченные данные с признаками и категориями, которые машина будет учиться определять по этим признакам. Дальше классифицировать можно что угодно: пользователей по интересам — так делают алгоритмические ленты, статьи по языкам и тематикам — важно для поисковиков, музыку по жанрам — вспомните плейлисты Спотифая и Яндекс.Музыки, даже письма в вашем почтовом ящике.

Раньше все спам-фильтры работали на алгоритме [Наивного Байеса](http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/). Машина считала сколько раз слово «кредит» встречается в спаме, а сколько раз в нормальных письмах. Перемножала эти две вероятности по формуле Байеса, складывала результаты всех слов и бац, всем лежать, у нас машинное обучение!

Позже спамеры научились обходить фильтр Байеса, просто вставляя в конец письма много слов с «хорошими» рейтингами. Метод получил ироничное название [Отравление Байеса](https://en.wikipedia.org/wiki/Bayesian_poisoning), а фильтровать спам стали другими алгоритмами. Но метод навсегда остался в учебниках как самый простой, красивый и один из первых практически полезных.

Возьмем другой пример полезной классификации. Вот берёте вы кредит в банке. Как банку удостовериться, вернёте вы его или нет? Точно никак, но у банка есть тысячи профилей других людей, которые уже брали кредит до вас. Там указан их возраст, образование, должность, уровень зарплаты и главное — кто из них вернул кредит, а с кем возникли проблемы.

Я надеюсю, что все догадались, где здесь данные и какой надо предсказать результат. Обучим машину, найдём закономерности, получим ответ — вопрос не в этом. Проблема в том, что банк не может слепо доверять ответу машины, без объяснений. Вдруг сбой, злые хакеры или плохой сотрудник решили испортить скрипт.

Для этой задачи придумали [Деревья Решений](https://www.youtube.com/watch?v=eKD5gxPPeY0). Машина автоматически разделяет все данные по вопросам, ответы на которые «да» или «нет». Вопросы могут быть не совсем адекватными с точки зрения человека, например «зарплата заёмщика больше, чем 25934 рубля?», но машина придумывает их так, чтобы на каждом шаге разбиение было самым точным.

Так получается дерево вопросов. Чем выше уровень, тем более общий вопрос. Потом даже можно загнать их аналитикам, и они навыдумывают почему так.

Деревья нашли свою нишу в областях с высокой ответственностью: диагностике, медицине, финансах.

Два самых популярных алгоритма построения деревьев — [CART](https://en.wikipedia.org/wiki/Predictive_analytics#Classification_and_regression_trees_.28CART.29) и [C4.5](https://en.wikipedia.org/wiki/C4.5_algorithm).

В чистом виде деревья сегодня используют редко, но вот их ансамбли лежат в основе крупных систем и зачастую уделывают даже нейросети. Например раньше, не знаю как сейчас, когда вы задаете вопрос Яндексу, именно толпа деревьев бежит ранжировать вам результаты.

Так что же такое ансамбли, давайте разберемся.

Сегодня используют для:

* Всего, где подходят классические алгоритмы (но работают точнее)
* Поисковые системы (★)
* Компьютерное зрение
* Распознавание объектов

Ансамбли и нейросети — наши главные бойцы. Сегодня они дают самые точные результаты и используются всеми крупными компаниями в продакшене. Только о нейросетях трещат на каждом углу, а слова «бустинг» и «бэггинг», наверное, пугают многих.

При всей их эффективности, идея до издевательства проста. Оказывается, если взять несколько не очень эффективных методов обучения и обучить исправлять ошибки друг друга, качество такой системы будет аж сильно выше, чем каждого из методов по отдельности.

Причём даже лучше, когда взятые алгоритмы максимально нестабильны и сильно плавают от входных данных. Поэтому чаще берут Регрессию и Деревья Решений, которым достаточно одной сильной аномалии в данных, чтобы поехала вся модель. А вот Байеса и K-NN не берут никогда — они хоть и глупые, но очень стабильные.

Ансамбль можно собрать как угодно, хоть случайно нарезать в тазик классификаторы и залить регрессией. За точность, правда, тогда никто не ручается. Потому есть три проверенных способа делать ансамбли.

Стекинг Обучаем несколько разных алгоритмов и передаём их результаты на вход последнему, который принимает итоговое решение. Что-то вроде спросить у всех их мнение, чтобы принять решение в вопросе.

Ключевое слово — разных алгоритмов, ведь один и тот же алгоритм, обученный на одних и тех же данных не имеет смысла. Каких алгоритмов выбрать — ваше дело, разве что в качестве решающего алгоритма чаще берут регрессию.

Ну и насколько мне известно — стекинг на практике применяется редко, потому что два других метода обычно точнее.

Беггинг Он же [Bootstrap AGGregatING](https://en.wikipedia.org/wiki/Bootstrap_aggregating). Обучаем один алгоритм много раз на случайных выборках из исходных данных. В самом конце усредняем ответы.

Данные в случайных выборках могут повторяться. То есть из набора 1-2-3 мы можем делать выборки 2-2-3, 1-2-2, 3-1-2 и так пока не надоест. На них мы обучаем один и тот же алгоритм несколько раз, а в конце вычисляем ответ простым голосованием.

Самый популярный пример беггинга — алгоритм [Random Forest](https://www.youtube.com/watch?v=J4Wdy0Wc_xQ" \t "_blank), беггинг на деревьях, который и нарисован на картинке. Когда вы открываете камеру на телефоне и видите как она очертила лица людей в кадре желтыми прямоугольниками — скорее всего это их работа. Нейросеть будет слишком медлительна в реальном времени, а беггинг идеален, ведь он может считать свои деревья параллельно на всех шейдерах видеокарты.

Дикая способность параллелиться даёт беггингу преимущество даже над следующим методом, который работает точнее, но только в один поток. Хотя можно разбить на сегменты, запустить несколько.. вообщем сами потом можете погуглить.

Бустинг Обучаем алгоритмы последовательно, каждый следующий уделяет особое внимание тем случаям, на которых ошибся предыдущий.

Как в беггинге, мы делаем выборки из исходных данных, но теперь не совсем случайно. В каждую новую выборку мы берём часть тех данных, на которых предыдущий алгоритм отработал неправильно. То есть как бы доучиваем новый алгоритм на ошибках предыдущего.

Плюсы — неистовая, даже нелегальная в некоторых странах, точность классификации, которой позавидуют все. Минусы уже названы — не параллелится. Хотя всё равно работает быстрее нейросетей, которые как гружёные камазы с песком по сравнению с шустрым бустингом.