KARPOV.COURSES >>>
КОНСПЕКТ

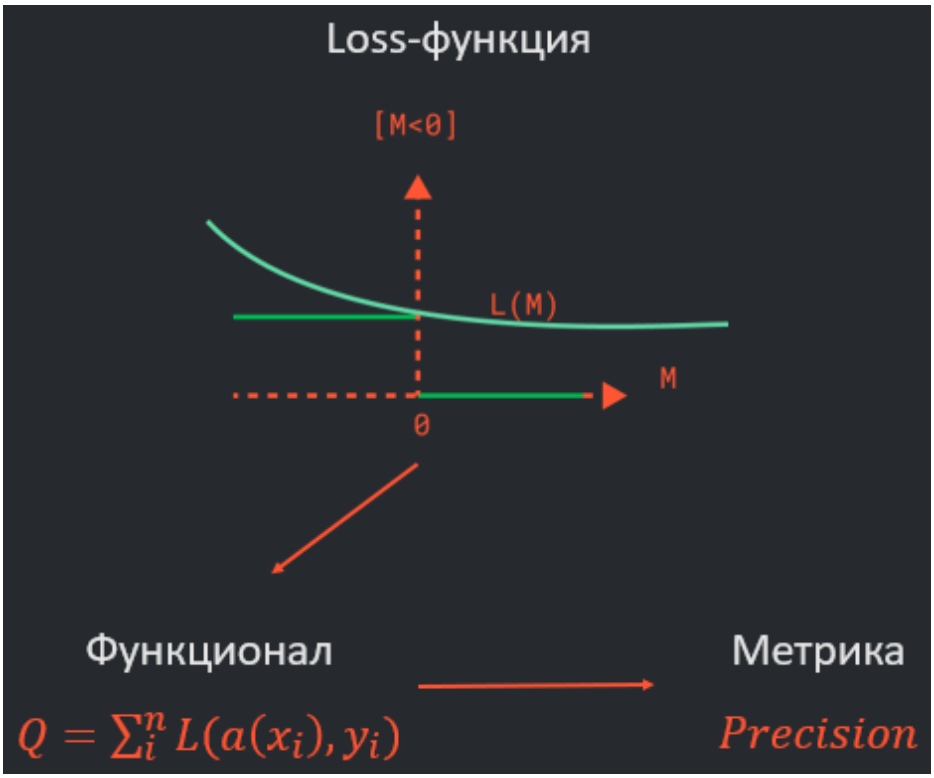
# Конспект > 23 урок > Машинное обучение: классические задачи и алгоритмы I

# >Чем отличается метрика от Loss?

Loss-функция это функция позволяющая замерить ошибку алгоритма на одном наблюдении.

Обычно сумму loss’ов по всем объектам оптимизируют, обучая модели.

Метрика – финальный замер качества алгоритмов, по ней производятся презентации и отбор модели.




. При оценке работы алгоритма в среднем, loss’ы усредняются и получается функционал качества.

Метрика это тоже функция, которая “пробегается по всем объектам” и замеряет среднее качество работы модели. Разница в том, что функционал используется именно во время обучения модели, метрика же используется для финального замера и в самой задаче оптимизации не используется. Она нужна, чтобы после обучения сравнить между собой ряд моделей. Метрику также можно использовать в презентации и интерпретации модели.

# >Что такое несимметричные метрики и зачем они нужны?

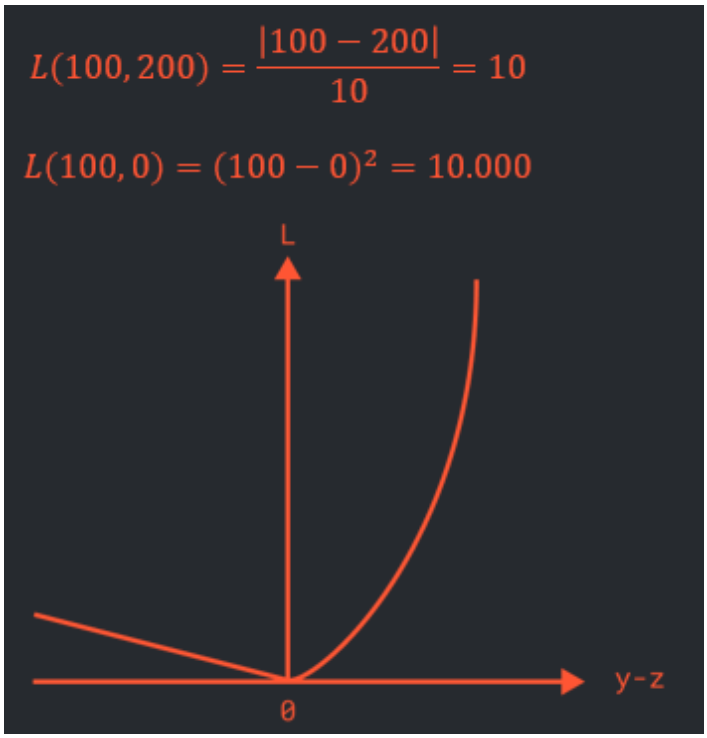
Представим loss-функцию, несимметричную относительно ошибок алгоритма $y_i - a(x_i)$

Например, используют, когда хотят по-разному штрафовать модель за пере- и недооценку: медицина, финансы

Под несимметрией метрики подразумевают её несимметрию относительно ошибки алгоритма, т.е если мы возьмем таргет, отнимем предсказание модели, то получим ошибку у модели на конкретном объект. Метрика называется несимметричной, когда её функциональная форма относительно полученных ошибок выглядит по разному.

$$L(y, z) = \begin{cases} (\frac{|y-z|}{10}, \text{если} z \geq y \\ (y-z)^2, \text{если} z < y \end{cases}$$

Например, метрика основанная на примере выше будет несимметричной. Если у модели перепрогноз, т.е $z \geq y$, тогда оценивать ошибку модели как $\frac{|y-z|}{10}$. Если же у модели выход алгоритма меньше чем истинный таргет, тогда оценка ошибки квадратичная $(y-z)^2$.




Обычно, несимметричные метрики используются когда необходимо построить модель, которая скорее будет склонна перепредсказывать, чем недопредсказывать (или наоборот). Среди популярных несимметричных метрик можно выделить MSLE

# >Разница между precision и recall

Precision и Recall это две самые популярные метрики.

$$Precision = \frac{TP}{TP+FP}$$

Precision (точность) позволяет понять, какая доля объектов среди тех, которые мы назвали положительным классом, действительно к нему относится.

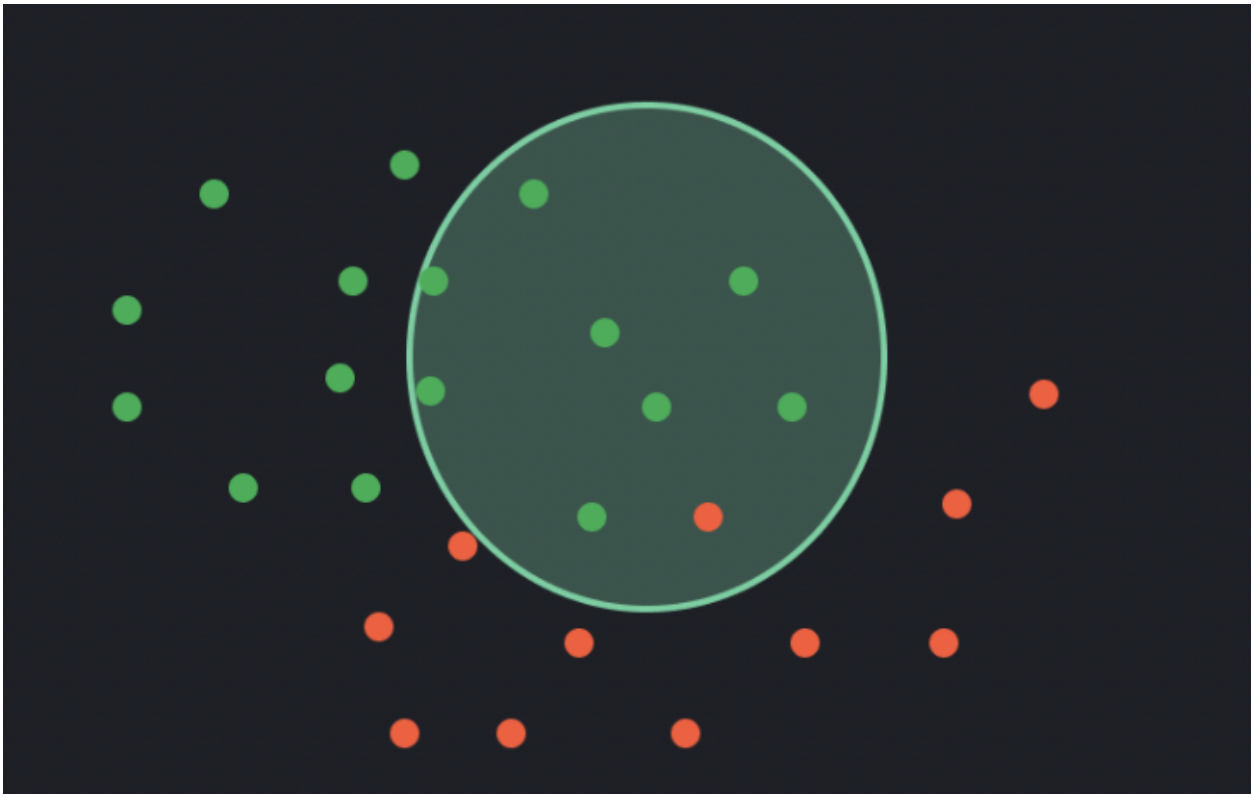
$$Recall = \frac{TP}{TP+FN}$$

Recall (полнота) же показывает, сколько объектов среди тех, которые в общем были положительные, наша модель смогла выявить.

| | $y = +1$ | $y = -1$ |
|---|---|---|
| $a(x) = +1$ | True Positive | False Positive |
| $a(x) = -1$ | False Negative | True Negative |

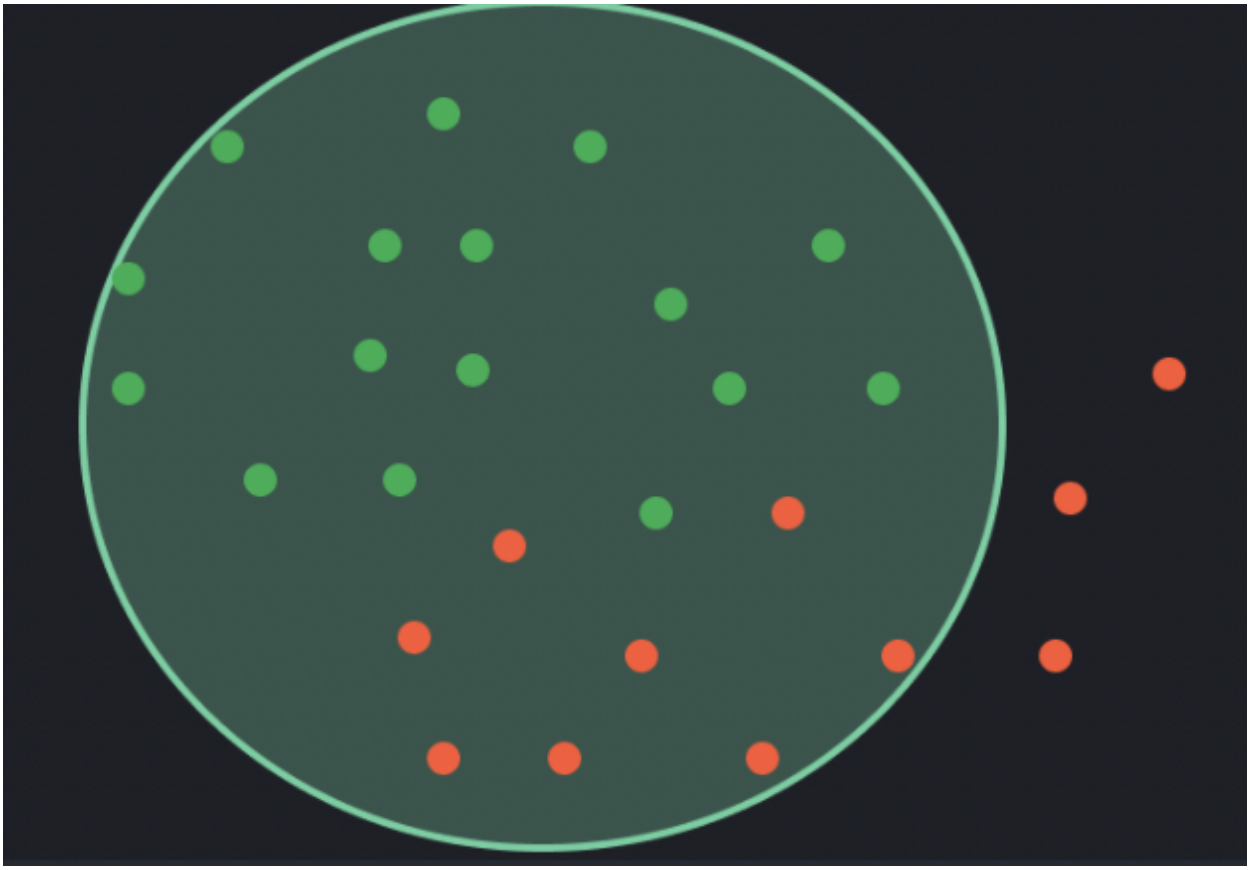
## >Хороший Precisio

Precision будет высоким, если количество неправильно предсказанных объектов будет минимальным, но при этом будет высокая доля верно размеченных



### >Хороший Recall

Recall будет высоким, в случае, когда модель покрывает как можно большее количество положительных классов изначально которые были в выборке.



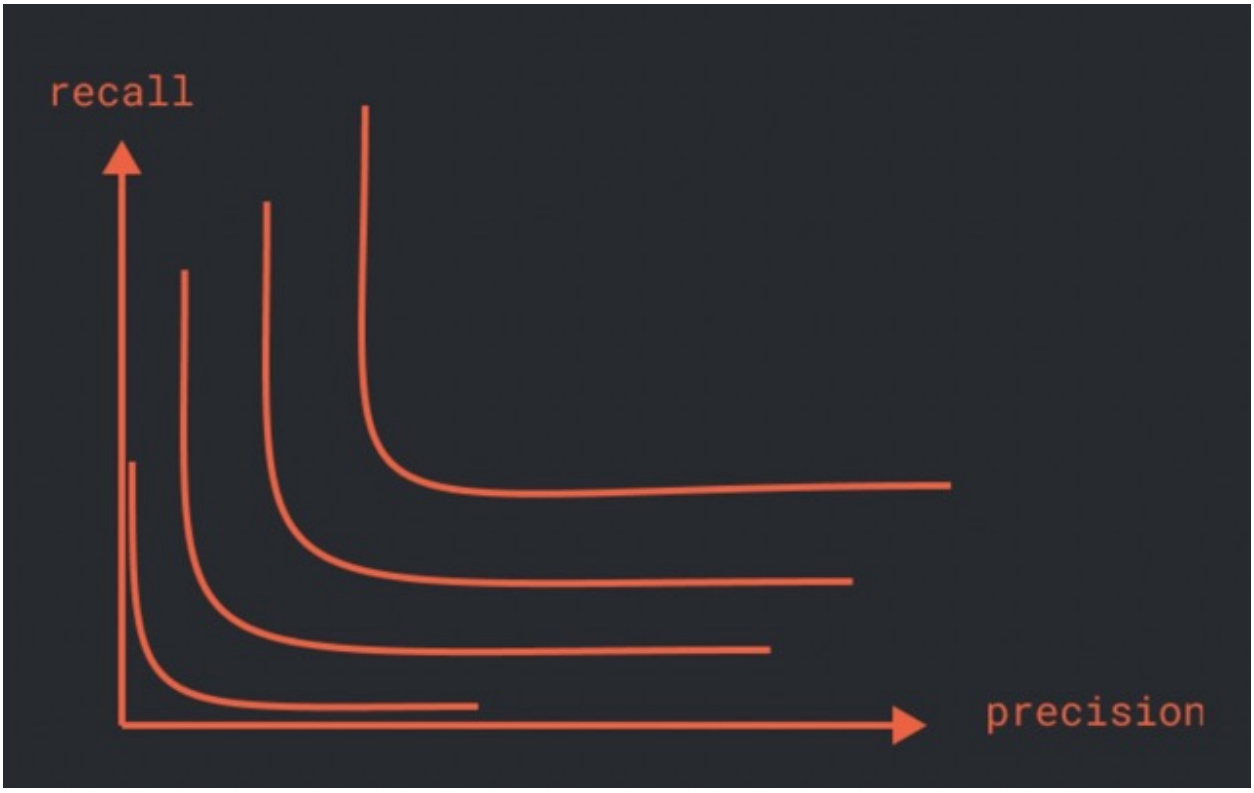
Но модель достаточно редко обладает как хорошим значением Precision, так и хорошим значением Recall одновременно. И, как правило, приходится балансировать между двумя этими метриками.

## >Какие ещё есть метрики классификации?

Выше мы обсудили, что, как правило, приходится балансировать между двумя этими метриками. Выбирая между большим количеством моделей классификации хочется опираться на какую-то метрику, которая будет учитывать как точность так и полноту. Можно посчитать F-меру:

$$F = \frac{2 * precision * recall}{precision + recall}$$

По-другому это значение еще называют средним гармоническим. F-мера достаточно близка к функции минимума: к той функции, у которой и precision, и recall должны быть достаточно большими, ведь она возвращает всего лишь минимальное значение среди двух параметров.




Линии уровня оказались еще более пологими в зависимости от того, к какой из осей мы подходим. Это говорит о том, что F-мера еще более склонна к тому, чтобы метрики оказывались где-то посередине.

Преимущество F-меры еще в том, что можно немного изменить формулу и добавить еще учет предпочтения между precision и recall.

Модифицированная формула выглядит следующим образом:

$$F = \frac{(1+\beta^2)*2*precision*recall}{\beta^2*(precision+recall)}$$

Чем больше $\beta^2$, тем больше уклон в recall. И наоборот: при $0 < \beta^2 < 1$ уклон будет больше в precision.

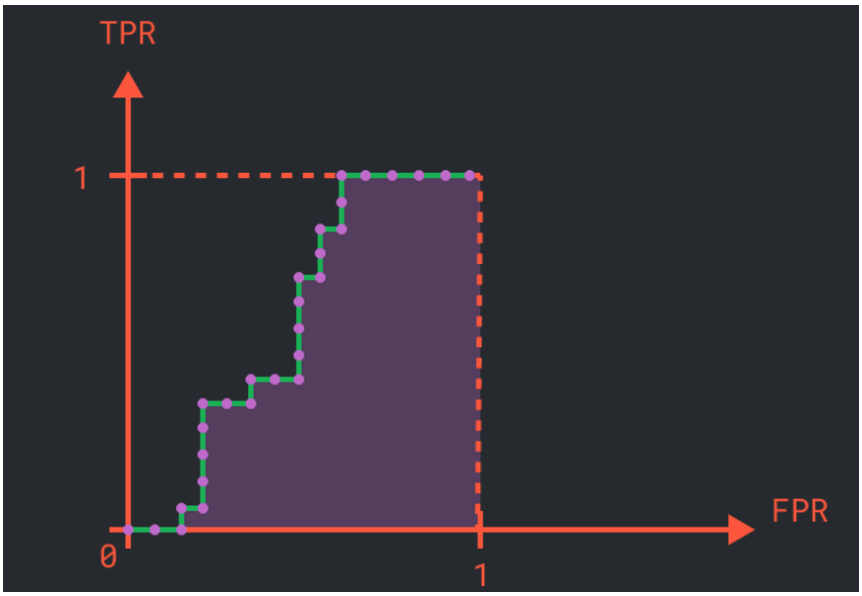
Если вместо $\beta$ положить 1, то мы получим среднее гармоническое. А еще, линии уровня такой функции напоминают линии уровня минимума, и поэтому стоит

выбрать ту модель, у которой значение между precision и recall максимальное.

## >ROC-кривая

Помимо F-меры ещё есть ROC (receiver operating characteristic)-кривая.

Модели классификации способны возвращать вероятности того, принадлежит ли объект к положительному классу. И итоговый ответ модели может отсекаться по некоторому порогу. Скажем, мы финально относим какой-то объект к положительному классу только если уверенность модели больше 70%. Те можно замерить качество работы алгоритма в среднем по всем трешхолдам:




Можно взять площадь под кривой — AUC (Area under the ROC Curve).

Чем больше ROC-AUC модели, тем лучше модель в среднем. У идеальной модели $ROC - AUC = 1$, у плохой она близка к 0, а если $ROC - AUC = 0$, значит модель классифицирует положительный класс как отрицательный и наоборот. Если же $ROC - AUC = 0.5$, значит модель случайным образом размечает классы.

## >PR-кривая

Метрика ROC-AUC является чувствительной к несбалансированным классам. Поэтому, для работы с несбалансированными классами используется другая кривая — PR (Precision-Recall) -кривая. Она считается аналогично ROC-кривой, только на метриках precision и recall. Аналогично находится и PR-AUC.

Резюме:

- ROC-кривая используется тогда, когда главная задача – как можно больше объектов правильно отранжировать с точки зрения вероятностей.
- PR-кривая используется тогда, когда присутствует ситуация дисбаланса классов. Важнее всего для данной кривой является положительный класс.

# >Что такое взрыв и затухание градиента?

Градиентный спуск — метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента.

Градиент — это вектор, своим направлением указывающий направление наибольшего
возрастания некоторой скалярной величины, а по модулю равный скорости роста этой величины в этом направлении.

Градиент функции — это вектор, координатами которого являются частные производные этой функции по всем её переменным. Направление градиента есть направление
наибыстрейшего возрастания функции, то есть производная функции (а точнее её знак) показывает, в каком направлении функции возрастает.

**Этапы градиентного спуска для функции нескольких переменных:**

1. Инициализируемся в случайной точке $x_{start}$.
2. Спускаемся до сходимости модели.

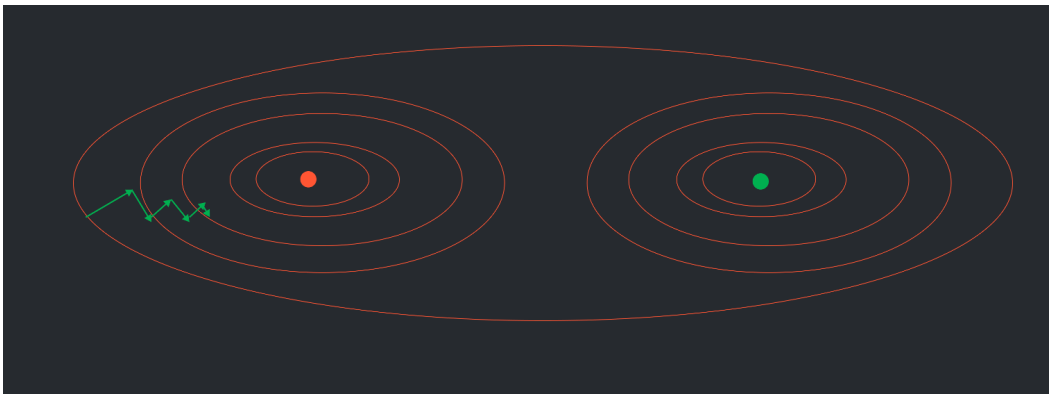
$step = \nabla z\prime(x_{start})$

$x_{next} = x_{start} - \eta_i * step$

$xstart = xnext$

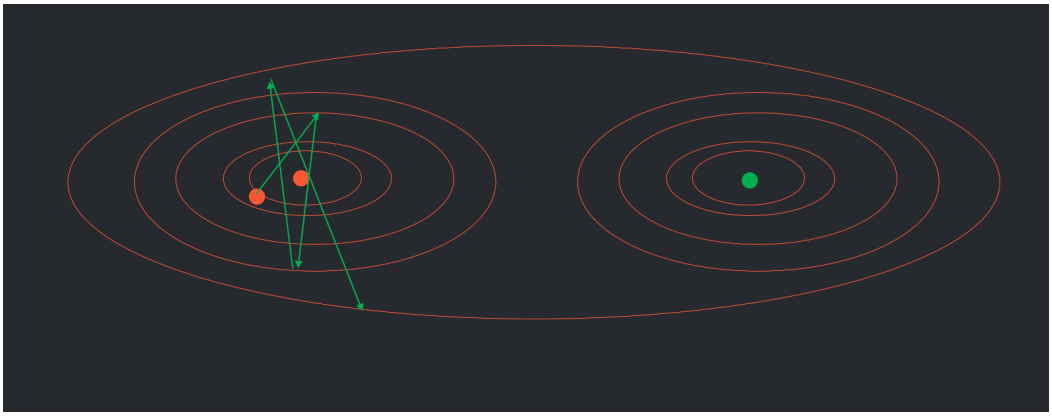
3. Останавливаемся тогда, когда значение производной становится очень маленьким (те же три варианта порога).

Затухание градиента — слишком маленькая длина шага приводит к раннему срабатыванию критерия останова.



«Взрыв» градиента — бесконечно блуждание по функции без схождения градиента.

*Взрыв градиента при минимизации функции одной переменной.*



# >Чем отличается Lasso от Ridge регуляризации?

Регуляризировать модель это значит в минимизируемый функционал добавить норму от вектора коэффициентов. Теперь посмотрим как можно настраивать регуляризатор$(R(\beta))$

Выделяют два типа:

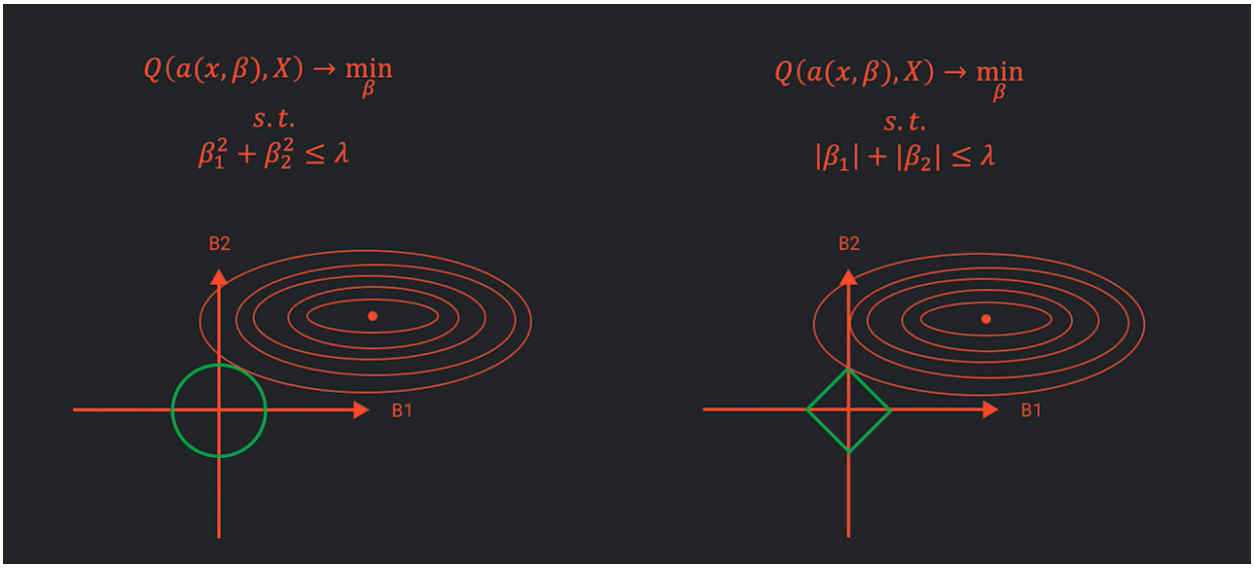
- **L-2(Ridge)** - когда мы в качестве регуляризатора берем сумму квадратов всех параметров модели

$$R(\beta) = \sum(\beta_i^2) = \beta_1^2 + \beta_2^2 ... \beta_n^2$$

- **L-1(Lasso)** - когда мы в качестве регуляризатора берем сумму модулей всех параметров модели

$$R(\beta) = \sum(|\beta_i|) = |\beta_1| + |\beta_2| + ... + |\beta_n|$$

Оптимизируемую задачу можно свести к задаче поиска условного экстремума. После добавления в задачу регуляризатора появилось условие, теперь нельзя взять любой $\beta_1$ и $\beta_2$, теперь есть ограничение:




Условный экстремум для функций регуляризации Ridge (слева) и Lasso (справа). Условные экстремумы возникают когда линии функции касаются области из ограничения.

Отличие Lasso-регуляризации от Ridge-регуляризации, заключается в том, что она склонна занулять некоторые коэффициенты.

# >Что такое мультиколлинеарность и как бороться?

Мультиколлинеарность - наличие линейной завивимости между признаками, т.е. когда можно выразить один признак через другой. Это приводит к целому комплексу проблем:

- Неустойчивости модели и очень часто к переобучению модели.
- В случает OLS регрессии - отсутствие одного минимума.

- Невозможно воспользоваться формулой $\beta^* = (X^T - X)^{-1} X^T \cdot Y$ (невозможно вычислить обратную матрицу).

Так как мы работаем с признаками как с векторами, когда среди них присутствуют такие, которые можно выразить через другие признаки, появляется проблема мультиколлинеарности.

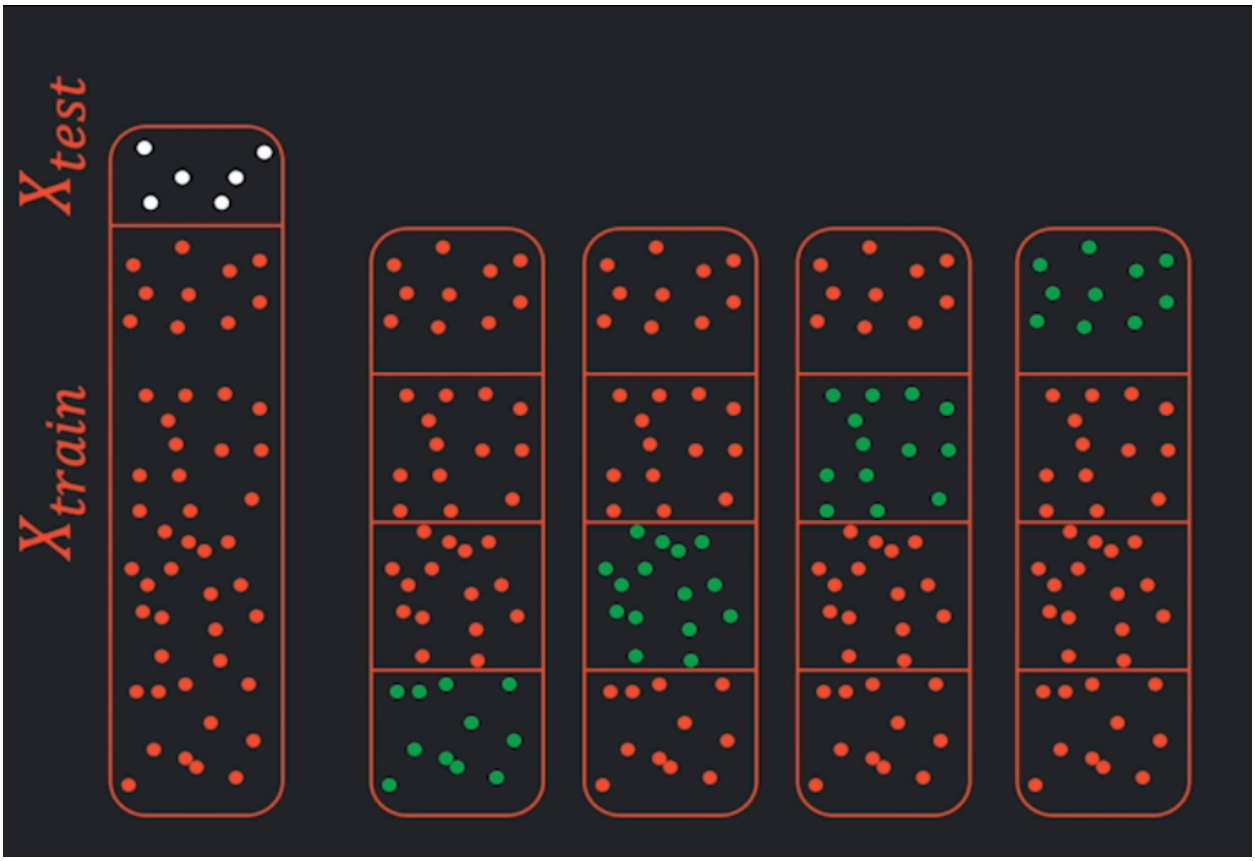
Мультиколлинеарность можно также обозначить как избыточность признаков, то есть убрав один из таких признаков мы не лишимся какой-либо важной информации. Т.е, можно бороться с мультиколлинеарностью с помощью отбора признаков и регуляризации

Различают слабую мультиколлинеарность и сильную. Сильная мультиколлинеарность  возникает когда есть сильная линейная зависимость между колонками. Слабая возникает когда линейная связь между колонками присутствует, но она зашумлена.

# >Что такое кросс-валидация

**Кросс-валидация** (перекрестная проверка, скользящий контроль) – это метод оценки обобщающей способности модели. Кросс-Валидация делит выборку на несколько блоков, обучает модель на всех кроме одного, и на нем же происходит замер качества. Самый популярный способ – Kfold:

- Представим, что у нас есть некий набор данных. Делим исходные данные на **k** фолдов(частей) примерно одинаково размера.
- Затем на (**k** - 1) блоке обучаем модели, а оставшуюся часть используем для тестирования.
- Повторяем процедуру **k** раз, при этом на каждой итерации для проверки выбирается новый блок, а обучение производится на оставшихся.
- Замеряем **k** средних ошибок на валидационной выборке (ее мы отложили ранее) и если их среднее/их распределение нас устраивает, то строим финальную модель




По умолчанию **k** берут равным 5 или 10, но можно принимать и любое другое значение

Но есть и другие технологии: например, TimeSeriesSplit Кросс-Валидация. Если данные имеют временную структуру, то валидировать модель можно по следующей технологии:

- Возьмем самый ранний блок данных.
- Обучаем по этому кусочку модель и предсказываем более поздний блок.
- На второй итерации добавляем к первому блоку второй, строим модель и на ней предсказываем более поздний блок и т.д.
- Затем, по полученным моделям усредняемся и получаем оценку предсказаний моделью таргетов из будущего.

# >Какие есть техники кодирования текста?

Модели машинного обучения предпочитают числа. Поэтому различные признаки, которые не являются числами нужно кодировать. Для текста в классическом ML

обычно используют:

- Bag of Words
- TF-IDF

**Зачем нужно выделять признаки из текста?**

Наименования объектов обычно уникальны. Они воспринимаются как одно целое — id. С
другой стороны, можно попробовать погрузиться в смысл описания объекта и перевести текстовое описание объекта в мир чисел и матриц. Там может крыться что-то очень важное, что поможет предсказывать таргет.

Например, в таблице внизу есть явная зависимость бюджета проекта от его описания. Как же понять, какие слова влияют на успех?

| Наименование проекта | Признаки | Сколько $$ соберет? |
|---|---|---|
| Запись скучного ROCK альбома | ... | 200 |
| Много плюшек инвесторам! Приходите!! | ... | 2000 |
| Всем подарки! Без СМС и регистрации! | ... | 3500 |
| Фэнтези-роман про бедного фермера | ... | 300 |
| Кофе в переходах метро | ... | 300 |
| Первым инвесторам - большие скидки! | ... | 1500 |

Для этого существует принцип "bag of words" — для каждого слова подсчитываем его встречаемость в тексте. Представим, у нас есть много объектов и у каждого объекта есть текстовое описание. Замерим какие уникальные слова встречаются в датасете. А потом, для каждого такого уникального слова создадим колонку,

которая для каждого объекта посчитает количество вхождений уникальных слов в его описании.

| Наименование проекта | Инвестору | - | ! | Дарим | ... |
|---|---|---|---|---|---|
| Инвестору - скидка | 1 | 1 | 0 | 0 | ... |
| Дарим каждому инвестору подарок! | 1 | 0 | 1 | 1 | ... |
| Дарим каждому инвестору бонус! | 1 | 0 | 1 | 1 | ... |

**Преимущества**

- Простой в применении;
- Не требует обработки текста.

**Недостатки**

- Может порождать огромные матрицы;
- Не учитываем общий контекст задачи и среднюю частоту всех слов;
- Например, если мы прогнозируем цены на овощи, то очевидно, что в описании к объекту почти всегда будет слово «овощ», то есть необходимо погружаться в контекст.

Для избежания упомянутых проблем нужен метод для более умной векторизации описания объектов.

С этим справляется трансформация текста методом TF-IDF - *term frequency-inverse document frequency*. Он позволяет выделить самые важные слова в каждом конкретном описании, учитывая общий контекст задачи и то, что написано в других объектах.

Данный метод состоит из двух компонент: TF и IDF.

**Как подсчитывается TF?**

Здесь нужно подсчитать, сколько раз слово входит в описание, затем поделить это число на общее количество слов.

$tf(t, d) = \frac{n_t}{(\sum_i n_i)}$, где $n_t$ — это количество вхождений слова $t$ в описание $d$.

Таким образом можно узнать, насколько данное слово важно для конкретного описания. Чем чаще слово встречается в описании, тем больше у этого слова в данном описании TF.

**Как подсчитывается IDF?**

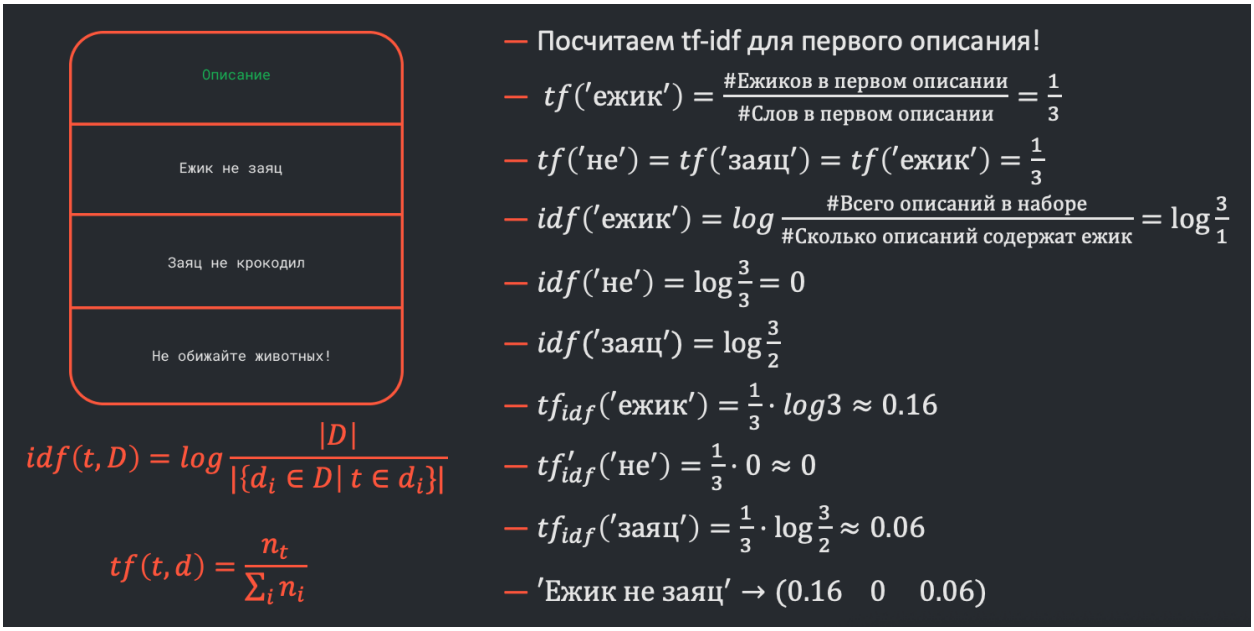
Здесь нужно взять логарифм частного количества всех описаний и количества описаний, которые содержат данное слово.

$idf(t, D) = log(\frac{|D|}{|\{di \in D | t \in di\}|})$, где $|\ D\ |$ – это количество описаний, а знаменатель – это количество описаний с словом $t$.

Таким образом можно узнать, насколько данное слово является общим для всех описаний. Чем чаще слово встречается в описании других объектов, тем меньше у него IDF.

Финальной метрикой является произведение TF и IDF. Большие значения TF-IDF будут иметь слова, которые часто встречаются в конкретном документе, но при этом оказываются редкими в других.

Ниже представлен пример вычисления TF-IDF.




**Как использовать данную метрику?**

В итоге мы получаем некоторый множественный признак, поверх которого можно сделать агрегацию, например, посчитать максимум или среднее.

**Преимущества TF-IDF**

- Более компактный, чем bag of words;
- Учитывает важность слов.

**Недостатки TF-IDF**

- Получаем оценку важности каждого слова через частности, но не погружаемся в контекст;
- Как и bag of words, у данного метода есть проблемы с однокоренными словами, падежами, склонениями и так далее.

# >Какие способы отбора признаков вы знаете?

Методы отбора признаков (feature selection) можно поделить на три большие группы:

- Встроенные алгоритмы: Lasso, Решающие деревья (feature importance)

- Методы фильтрации: корреляционный анализ, VarianceThreshold
- Методы обертки: прямой/обратный отбор

## Встроенные методы

Встроенные методы — это методы, которые одновременно строят модель и фильтруют признаки.

Встроенные методы обладают рядом преимуществ:

- Способ достаточно быстрый, т.к работает под капотом.
- По вычислительным мощностям не требуем доп. нагрузку. Используем только те, что зашиты в самой модели.

И недостатков:

- Ограничены заданной функциональной формой. То есть, например, для задачи линейной регрессии мы ограничены только линейной зависимостью между признаком и таргетом.
- В случае регуляризации – всегда есть шанс удалить признак, который давал бы значимую прибавку на Кросс-Валидации

## Метод фильрации

Основная идея заключается в том, чтобы внимательно посмотреть на данные и посчитать какую-то характеристику (статистические метрики, например).

Как и в EDA, в этом методе мы не строим модели, здесь мы делаем акцент на инструменты статистики и оцениваем, например:

- Как вещественные показатели связаны с таргетом.
- Как вещественные показатели связаны друг с другом.
- Какое влияние категории имеют на тергетную переменную.

Методы фильтрации обладают рядом преимуществ:

- Дешевые с точки зрения времени и вычислительных мощностей. Мы не строим модели, а просто смотрим на данные и считаем какие-то характеристики.

И недостатков:

- Требуют работы с данными.
- Зачастую используемые “критерии” не улавливают сложные взаимосвязи между данными, а ограничиваются только линейными

### Метод обертки

Основная идея метода заключается в поиске всевозможных признаков и оценки их качества путем прогона через модель

Если мы не знаем на каких признаках лучше обучать модель, а на каких нет, то давайте просто переберём все варианты и сравним модели между собой на кросс-валидации. Можно использовать любой алгоритм обучения.

Методы обёртки обладают рядом преимуществ:

- Получаем более полную картину относительно различных подмножеств наших признаков
- Не требуем обработки данных (например, масштабирование)

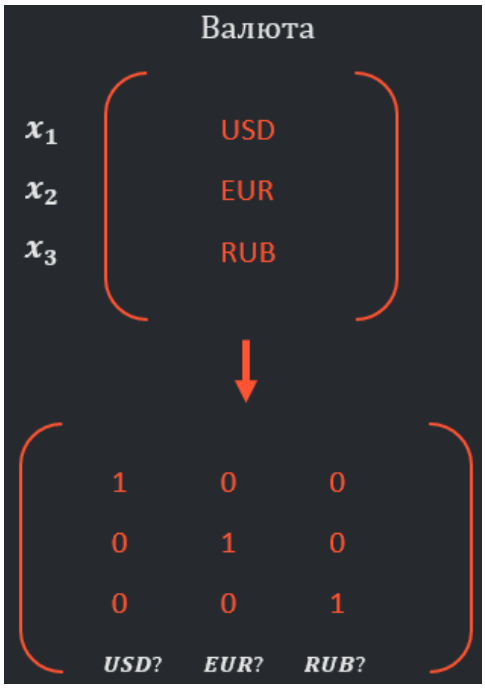
И недостатков:

- Долго и очень дорого
- А жадные алгоритмы всегда оставляют чувство недосказанности(”вдруг, стоило выкинуть другой признак...”)
- Склонны к переобучению

# >Что такое Dummy Variable Trap?

Возникает при OneHotEncoding трансформации категориальных фичей в линейных моделях

One Hot Encoding - метод позволяющий преобразовать каждое значение признака в отдельный признак, где каждому объекту будет соответствовать 0 или 1, в зависимости от того, обладает объект этим признаком или нет.




!В конце преобразования признаков не забываем выкинуть одну колонку, чтобы избежать избыточной информации и переобучения (если было 12 значений признака, оставляем 11 колонок). Ведет к мультиколлинеарности: каждую новую бинарную колонку можно выразить линейно через остальные.

Подробнее о OneHotEncoding

# >Как устроена логистическая регрессия?

Логистическая регрессия это линейная модель для решения задачи бинарной классификации. В задачи классификации можно свести максимизацию accuracy. Для этого нужно минимизировать количество несовпадений ответа алгоритма с истинными таргетами.

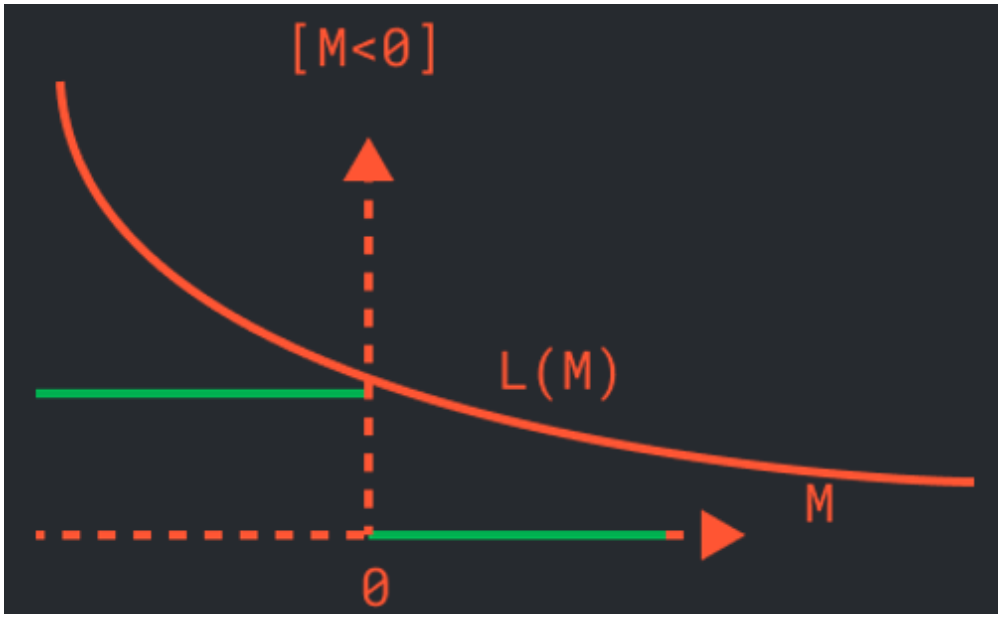
$$Q(\beta, \chi) = \frac{1}{n} \sum_n^i [\alpha(x_i) \neq y_i] \rightarrow min$$

Выбирается вектор фичей, вектор коэффициентов $\beta$, скалярно перемножаются и выдают знак выражения

$$Q(\beta, \chi) = \frac{1}{n} \sum_n^i [\alpha(x_i) \neq y_i] = \sum_n^i [sgn(\langle \beta, \chi \rangle) \neq y_i]$$

Модель будет ошибаться, т.е её знак не будет равен $y_i$, когда произведение $[y_i * \langle \beta, \chi \rangle]$ будет отрицательным. Данное произведение $[y_i * \langle \beta, \chi \rangle]$ называют отступом и обозначают в виде $M_i$. Задача сводится к тому чтобы сумму

индикаторов $M_i < 0$ проминимизировать. Классическими методами (градиентным спуском, например) данную задачу не решить. Поэтому loss можно сгладить какой-то верхней оценкой:




И в качестве функции, с помощью которой $M_i$ сглаживаем сверху, можно взять

$L(M) = \log(1 + e^{-M})$

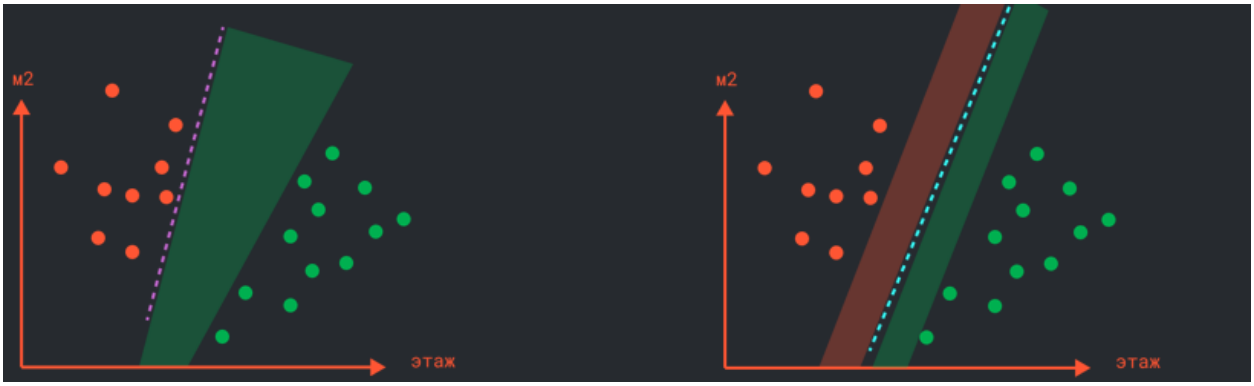
Минимизируя сумму $L(M)$ можно надеяться, что в итоге получится точка, которая будет минимизировать изначальную функцию, хотя бы приближенно.

# >Отличия логистической регрессии и SVM

Метод опорных векторов(SVM - Support Vector Machine) - способ построения модели классификации, суть которого заключается в построении разделяющей гиперплоскости.

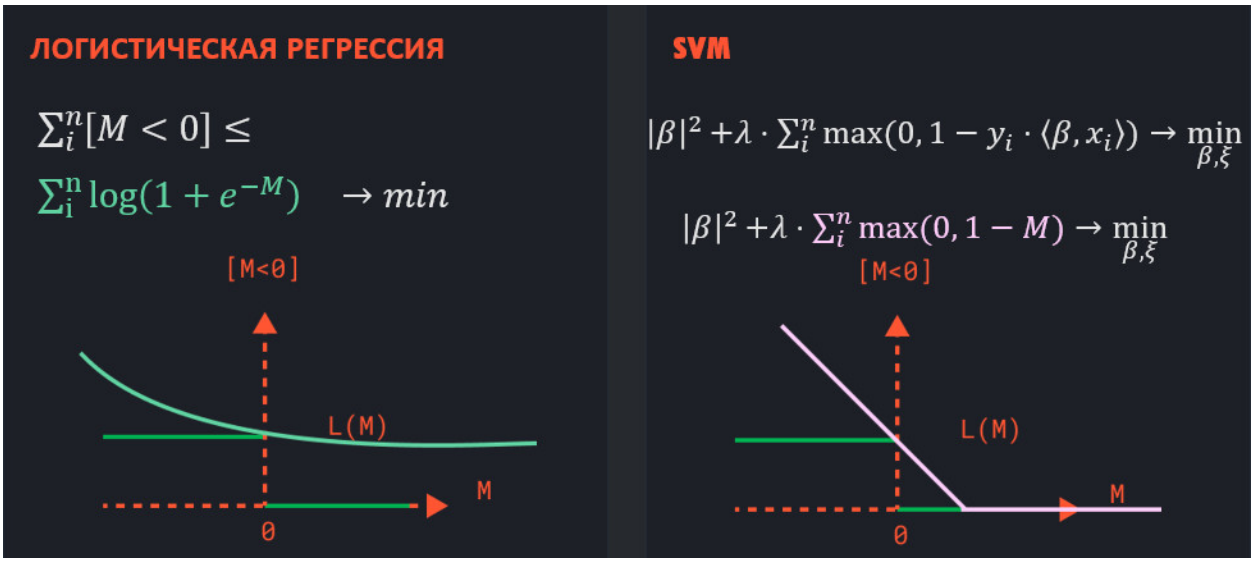
В отличии от логистической регрессии, данный алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора.

При этом гиперплоскость должна пройти так, чтобы расстояние до ближайшего объекта было максимальным.




LR старается как можно корректнее оценить вероятности, когда SVM максимизирует все зазоры между гиперплоскостью и границами классов.

Формально оба подхода классификации решают похожие задачи




Сравним SVM с методом логистической регресии:

- SVM  является частным случаем минимизации индикаторов отступов со специфической верхней оценкой и модификацией в виде регуляризации.
- В отличии от логистической регрессии, он не учитывает вероятности и не стремится к максимизации уверенности в прогнозе.
- Его основная задача - построение разделяющей гиперплоскости с достаточно широкой полосой.
- Функция потерь для SVM носит название hinge loss

- При этом мы все еще можем оценить вероятность с помощью метода калибровки.

# >Стоит ли стандартизировать таргет?

На самом деле, с точки зрения качества работы модели смысла в стандартизации нет совершенно никакого, потому что это два последовательных константных преобразования Если научились предсказывать $\left\{ \frac{y_i - y^{\text{ср}}}{a_i} \right\}_i$, то и с $\{y_i\}_i$ проблем не будет.

Пусть мы стандартизировали данные и построили модель, которая идеально восстанавливает стандартизированный таргет, тогда можно построить модель с точно таким же качеством, но на изначальных данных до стандартизации.

С точки зрения эффективности и качества работы модели стандартизировать смысла нет, но это может повлиять на скорость работы алгоритма.