# Вопросы по теоретической части:

# Понятие машинного обучения. Отличие машинного обучения от других областей программирования.

**Машинное обучение (MО)** – это форма искусственного интеллекта (ИИ), позволяющая компьютеру обучаться без необходимости непосредственного программирования. Вместо конкретных команд компьютеру для выполнения задачи, МО дает возможность компьютеру самостоятельно разработать шаги по решению, используя данные для самообучения. Чем больше данных, к которым компьютер имеет доступ, тем эффективнее он обучается и тем «умнее» становится, улучшая собственную точность и работоспособность с течением времени.

разница в том, что в отличие от программирования, с МО вам не нужно самостоятельно строить модель. Эта задача возлагается на алгоритмы МО, правки в которые может внести дата-инженер.

Также отличием является объем данных, который влияет на точность модели и подвергается значительным ограничениям в программировании, однако может быть использован во всей своей полноте в МО. Прибегая к МО, вы можете использовать столько параметров, сколько вам необходимо.

# Классификация задач машинного обучения. Примеры задач из различных классов.

**Все задачи, решаемые с помощью ML, относятся к одной из следующих категорий.**

1)**Задача регрессии** – прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число (2, 35, 76.454 и др.), к примеру цена квартиры, стоимость ценной бумаги по прошествии полугода, ожидаемый доход магазина на следующий месяц, качество вина при слепом тестировании.

2)**Задача классификации** – получение категориального ответа на основе набора признаков. Имеет конечное количество ответов (как правило, в формате «да» или «нет»): есть ли на фотографии кот, является ли изображение человеческим лицом, болен ли пациент раком.

3)**Задача кластеризации** – распределение данных на группы: разделение всех клиентов мобильного оператора по уровню платёжеспособности, отнесение космических объектов к той или иной категории (планета, звёзда, чёрная дыра и т. п.).

4)**Задача уменьшения размерности** – сведение большого числа признаков к меньшему (обычно 2–3) для удобства их последующей визуализации (например, сжатие данных).

5)**Задача выявления аномалий** – отделение аномалий от стандартных случаев. На первый взгляд она совпадает с задачей классификации, но есть одно существенное отличие: аномалии – явление редкое, и обучающих примеров, на которых можно натаскать машинно обучающуюся модель на выявление таких объектов, либо исчезающе мало, либо просто нет, поэтому методы классификации здесь не работают. На практике такой задачей является, например, выявление мошеннических действий с банковскими картами

Основная масса задач, решаемых при помощи методов машинного обучения, относится к двум разным видам: **обучение с учителем (supervised learning) либо без него (unsupervised learning)**.

**Машинное обучение с учителем**

Предположим, в нашем распоряжении оказались сведения о десяти тысячах московских квартир: площадь, этаж, район, наличие или отсутствие парковки у дома, расстояние от метро, цена квартиры и т. п. Нам необходимо создать модель, предсказывающую рыночную стоимость квартиры по её параметрам. Это идеальный пример машинного обучения с учителем: у нас есть исходные данные (количество квартир и их свойства, которые называются признаками) и готовый ответ по каждой из квартир – её стоимость. Программе предстоит решить задачу регрессии.

Ещё пример из практики: подтвердить или опровергнуть наличие рака у пациента, зная все его медицинские показатели. Выяснить, является ли входящее письмо спамом, проанализировав его текст. Это всё задачи на классификацию.

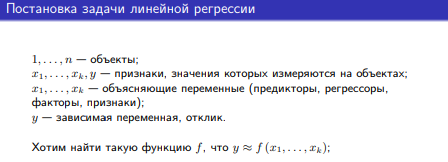
**Машинное обучение без учителя**

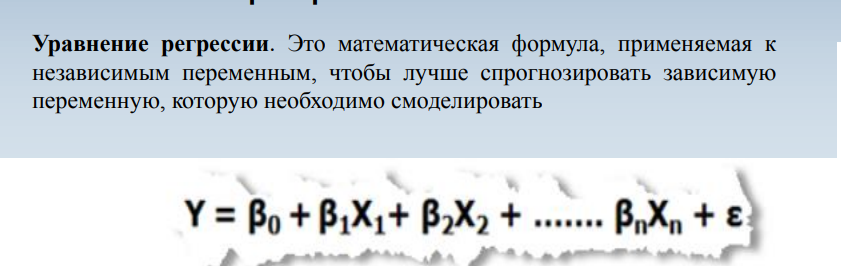
В случае обучения без учителя, когда готовых «правильных ответов» системе не предоставлено, всё обстоит ещё интереснее. Например, у нас есть информация о весе и росте какого-то количества людей, и эти данные нужно распределить по трём группам, для каждой из которых предстоит пошить рубашки подходящих размеров. Это задача кластеризации. В этом случае предстоит разделить все данные на 3 кластера (но, как правило, такого строгого и единственно возможного деления нет).

Если взять другую ситуацию, когда каждый из объектов в выборке обладает сотней различных признаков, то основной трудностью будет графическое отображение такой выборки. Поэтому количество признаков уменьшают до двух или трёх, и становится возможным визуализировать их на плоскости или в 3D. Это – задача уменьшения размерности.

# Задача регрессии: постановка, математическая формализация.

1)**Задача регрессии** – прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число (2, 35, 76.454 и др.), к примеру цена квартиры, стоимость ценной бумаги по прошествии полугода, ожидаемый доход магазина на следующий месяц, качество вина при слепом тестировании.





# Метод градиентного спуска для парной линейной регрессии.

**Метод градиентного спуска** — это способ нахождения локального минимума функции в процессе движения в направлении антиградиента. Он был предложен Огюстеном Луи Коши еще в 1847 году.

Итак, метод наименьших квадратов считает хорошо, и главное, он делает это быстро. Но постоянно его применять не получится, альтернатива: вместо метода наименьших квадратов (МНК) часто применяется градиентный спуск. Градиент это итеративный метод. Он показывает направление наискорейшего возрастания функции. Есть и обратная история: антиградиент, он показывает наибольшее убывание. У МНК есть проблема: он плох при большой размерности X, а точнее, когда много данных. Градиентный спуск не уступает по точности МНК. Принцип его работы простой, сначала он находит плохое решение, потом лучше, еще лучше, еще лучше, и так до наименьшего значения ошибки. Правило: много данных = градиентный спуск. В современном мире выбор алгоритма сильно завязан на производительность, МНК сразу оценивает весь объем данных, а стохастический градиентный спуск позволяет подавать данные батчами. Это помогает экономить оперативную память, которой всегда не хватает. Хотя и МНК, и градиентный спуск легко превращают любую железку в обогреватель. А стохастический градиентный спуск (SGD) нужен когда данные не помещаются в оперативку, и данные нужно разбивать. Это позволяет быстрее достичь глобального минимума, так как в итерациях будет участвовать не весь набор данных. Очевидный недостаток это не попадание в настоящую точку минимума, но в общем то, уровень точности приемлемый. Все по аналогии с веб-аналитикой: есть семплирование, есть и проблемы.

**(Коротеев)**

### **Метод градиентного спуска**

Таким образом, у нас есть наша функция гипотезы, и у нас есть способ оценить, насколько хорошо конкретная гипотеза вписывается в данные. Теперь нам нужно подобрать параметры в функции гипотезы. Вот где приходит на помощь метод градиентного спуска.

Представьте себе, что мы нарисуем нашу функцию гипотезы на основе ее параметров b0 и b1 (фактически мы представляем график функции стоимости как функцию оценок параметров).

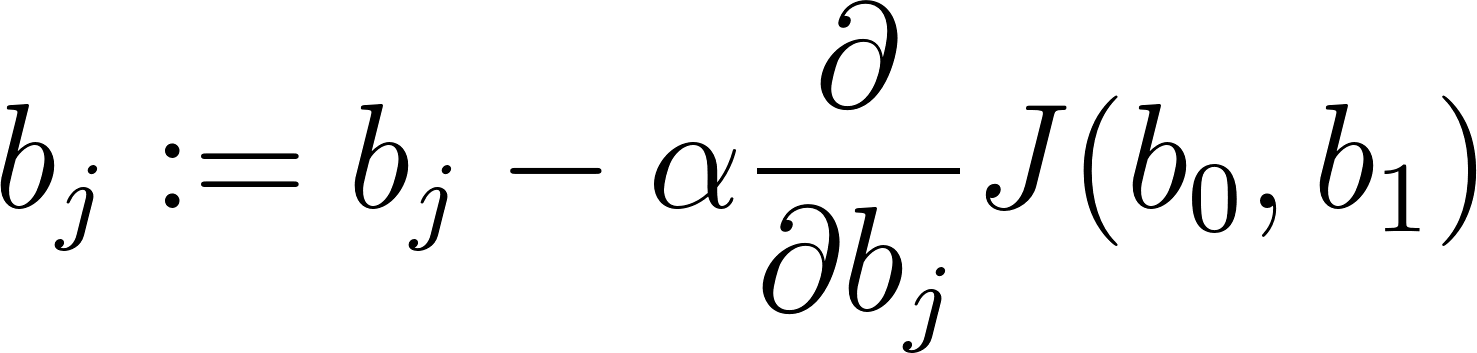
Отложим b0 на оси x и b1 на оси y, с функцией стоимости на вертикальной оси z. Точки на нашем графике будут результатом функции стоимости, используя нашу гипотезу с этими конкретными параметрами.

Мы будем знать, что нам удалось подобрать оптимальные параметры, когда наша функция стоимости находится в самом низу на нашем графике, то есть когда ее значение является минимальным.

То, как мы это делаем - это используя производную (касательную линию к функции) нашей функции стоимости. Наклон касательной является производной в этой точке, и это даст нам направление движения в сторону самого крутого уменьшения значения функции. Мы делаем шаги вниз по функции стоимости в направлении с самым крутым спусками, а размер каждого шага определяется параметром α, который называется скоростью обучения.

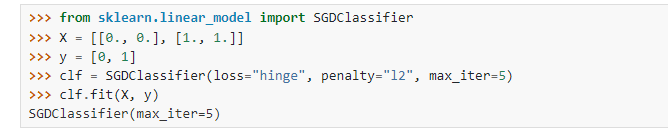
Алгоритм градиентного спуска:

повторяйте до сходимости:

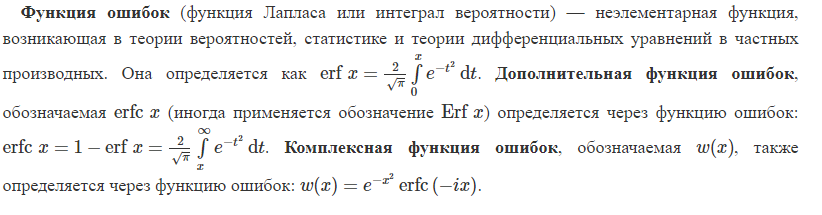
[](about:blank)

где j=0,1 - представляет собой индекс номера признака.

Для парной работает отлично , как в принципе и для множественной , в питоне есть отдельная библиотека . Вот код с ней , взято с scikit learn

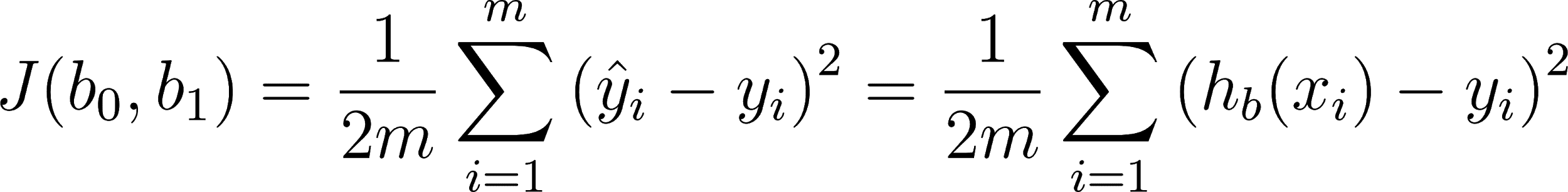


# Понятие функции ошибки: требования, использование, примеры.



(Из коротеева)

Мы можем измерить точность нашей функции гипотезы, используя функцию ошибки. Для этого требуется средняя (фактически чуть усложненная версия среднего арифметического) всех результатов вычисления гипотезы с входами x по сравнению с фактическим выходом y.

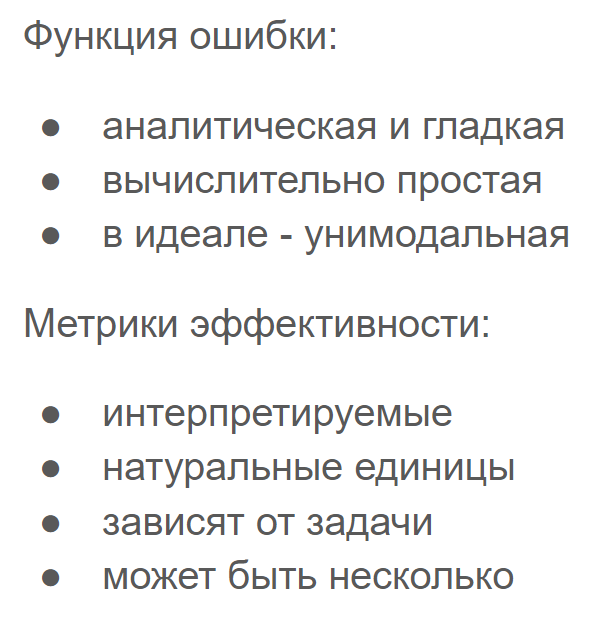
[](about:blank)

По сути своей, это половина среднего квадрата разницы между прогнозируемым и фактическим значением выходной переменной.

Эту функцию называют «функцией квадрата ошибки» или «среднеквадратичной ошибкой» (mean squared error, MSE). Среднее значение уменьшено вдвое для удобства вычисления градиентного спуска, так как производная квадратичной функции будет отменять множитель 1/2.

Теперь мы можем конкретно измерить точность нашей предсказывающей функции по сравнению с правильными результатами, которые мы имеем, чтобы мы могли предсказать новые результаты, которых у нас нет.

Если мы попытаемся представить это наглядно, наш набор данных обучения будет разбросан по плоскости x-y. Мы пытаемся подобрать прямую линию, которая проходит через этот разбросанный набор данных. Наша цель - получить наилучшую возможную линию. Лучшая линия будет такой, чтобы средние квадраты вертикальных расстояний рассеянных точек от линии были наименьшими. В лучшем случае линия должна проходить через все точки нашего набора данных обучения. В таком случае значение J будет равно 0.

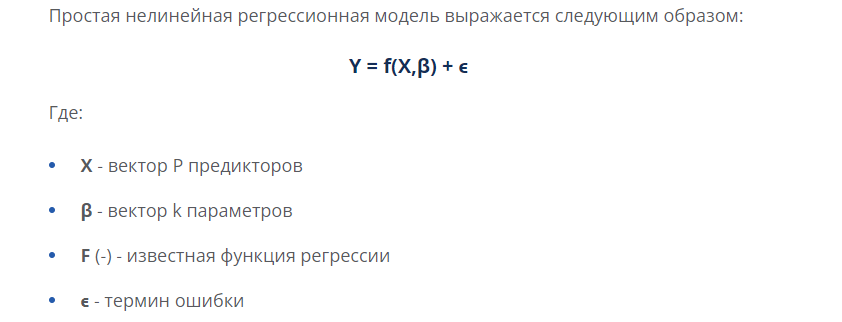


Таким образом обучение нейронной сети сводится к минимизации функции ошибки, путем корректировки весовых коэффициентов синаптических связей между нейронами. Под функцией ошибки понимается разность между полученным ответом и желаемым. Например, на вход был подан образ лица, предположим, что выход нейросети был 0.73, а желаемый результат 1 (т.к. образ лица), получим, что ошибка сети является разницей, то есть 0.27. Затем веса выходного слоя нейронов корректируются в соответствии с ошибкой. Для нейронов выходного слоя известны их фактические и желаемые значения выходов. Поэтому настройка весов связей для таких нейронов является относительно простой. Однако для нейронов предыдущих слоев настройка не столь очевидна. Долгое время не было известно алгоритма распространения ошибки по скрытым слоям.

# Множественная и нелинейная регрессии.

Множественная регрессия - уравнение связи с несколькими независимыми переменными: y = f (x1, x2,..., XP), где у – зависимая переменная (результативный признак); х 1, х2, …, хp - независимые переменные (факторы). Множественная регрессия применяется в ситуациях, когда из множества факторов, влияющих на результативный признак, нельзя выделить один доминирующий фактор и необходимо учитывать влияние нескольких факторов.

Нелинейная регрессия - это математическая модель, которая приспосабливает уравнение к определенным данным с помощью сгенерированной линии. Как и в случае с линейной регрессией, которая использует прямолинейное уравнение (например, Ỵ = c + m x), нелинейная регрессия показывает ассоциацию с использованием кривой, что делает ее нелинейной по [параметру](https://translated.turbopages.org/proxy_u/en-ru.ru.50094881-62a8ad1b-f034fd7c-74722d776562/https/corporatefinanceinstitute.com/resources/knowledge/other/parameter/).



Поскольку каждый параметр может быть оценен, чтобы определить, является ли он нелинейным или линейным, данная функция Y i может включать в себя сочетание нелинейных и линейных параметров. Функция h в модели рассматривается, так как ее нельзя записать как линейную по параметрам. Вместо этого функция выводится из теории.

Термин “нелинейный” относится к параметрам в модели, а не к [независимым переменным](https://translated.turbopages.org/proxy_u/en-ru.ru.50094881-62a8ad1b-f034fd7c-74722d776562/https/corporatefinanceinstitute.com/resources/knowledge/modeling/independent-variable/). Существуют неограниченные возможности для описания детерминированной части модели. Такая гибкость обеспечивает хорошую основу для статистических выводов.

# Задача классфикации: постановка, математическая формализация.

2)**Задача классификации** – получение категориального ответа на основе набора признаков. Имеет конечное количество ответов (как правило, в формате «да» или «нет»): есть ли на фотографии кот, является ли изображение человеческим лицом, болен ли пациент раком.

Классификация - системное распределение изучаемых предметов, явлений, процессов по родам, видам, типам, по каким-либо существенным признакам для удобства их исследования; группировка исходных понятий и расположение их в определенном порядке, отражающем степень этого сходства.

Пример задачи классификации. Пусть имеется БД о клиентах туристического агентства с информацией о возрасте и доходе за месяц. Есть рекламный материал двух видов: более дорогой и более дешевый отдых. Определены два класса клиентов: класс 1 и 2 ( в табл. 3.1).

Задача. Определить, к какому классу принадлежит новый клиент, и какой из двух видов рекламных материалов ему стоит отсылать.

Теперь мы переходим от регрессионных проблем к задачам классификации..

**Бинарная классификация**

Вместо того, чтобы наш выходной вектор y был непрерывным диапазоном значений, он будет равен только 0 или 1.

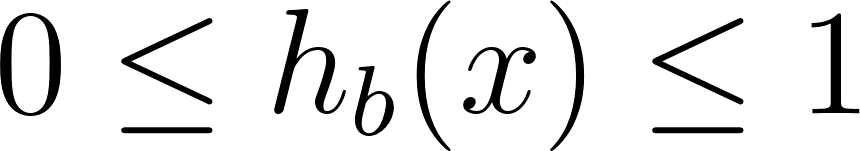
y∈ {0,1}

Где 0 обычно принимается как «отрицательный класс» и 1 как «положительный класс», но вы можете назначить ему какое-либо представление. На данный момент мы занимаемся только двумя классами, называемыми проблемой двоичной или бинарной классификации.

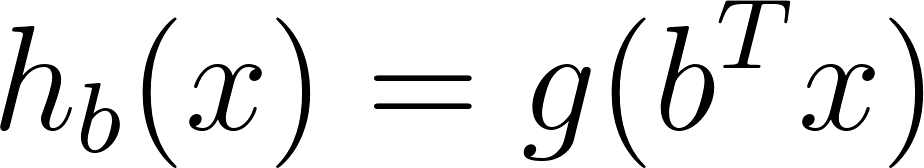
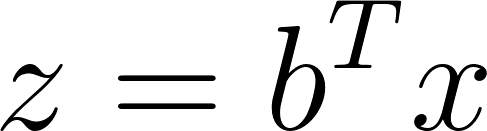
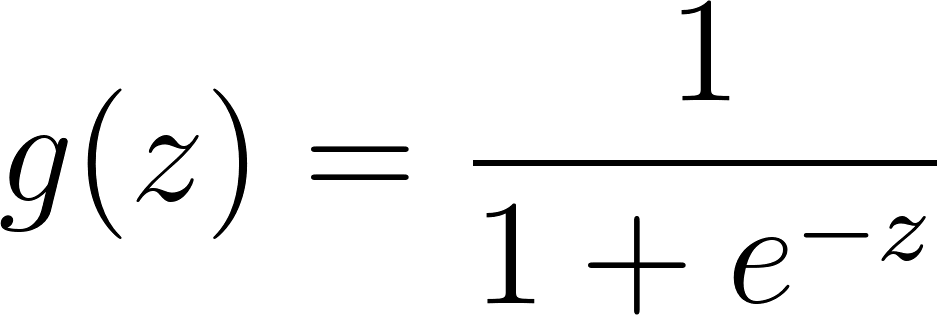
Один из методов - использовать линейную регрессию и отображать все предсказания, превышающие 0,5, как 1 и все меньше 0,5 в качестве 0. Этот метод не работает, потому что классификация на самом деле не является линейной функцией.

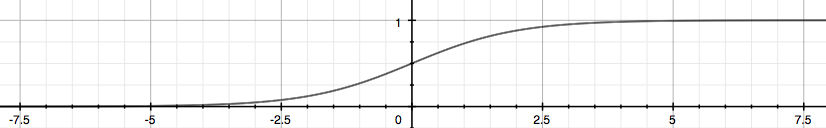
### **Представление гипотезы**

Наша гипотеза должна удовлетворять:

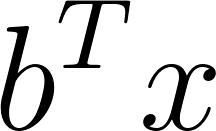
[](about:blank)

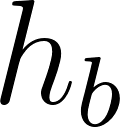
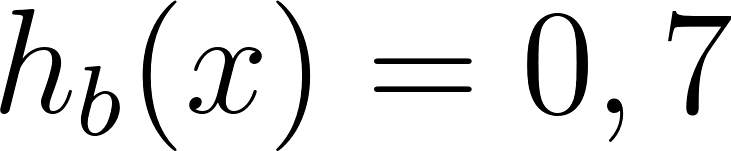
В этой новой форме задачи используется «сигмоидальная функция», также называемая «логистическая функция»:

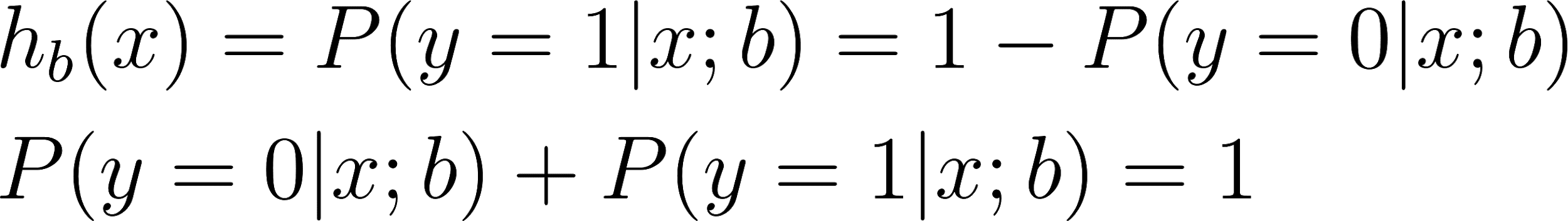
[](about:blank)   
 [](about:blank)  
[](about:blank)



Показанная здесь функция g(z) отображает любое вещественное число на интервал (0, 1), что делает его полезным для преобразования произвольнозначной функции в функцию, более подходящую для классификации.

Начнем с нашей старой гипотезы (линейной регрессии), за исключением того, что мы хотим ограничить диапазон до 0 и 1. Это достигается путем подключения [](about:blank) в логистическую функцию.

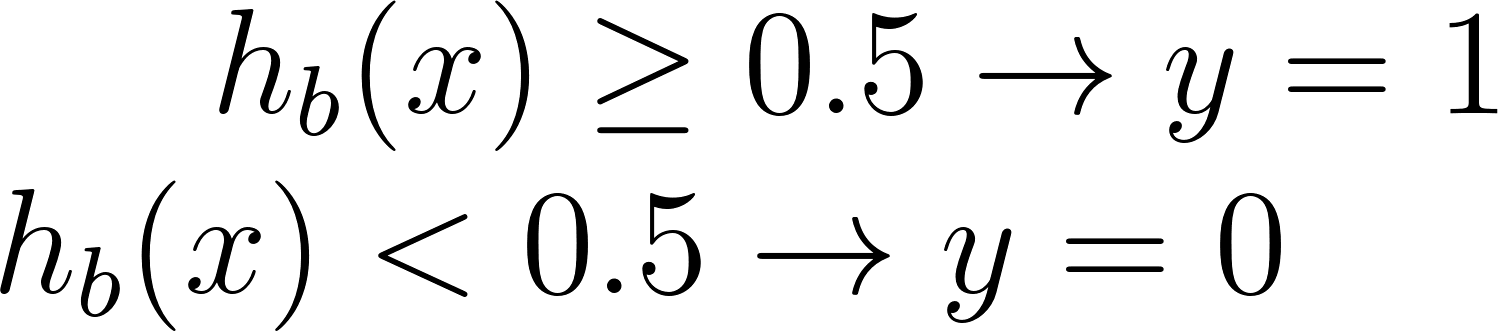
[](about:blank) даст нам вероятность того, что наш результат равен 1. Например,  дает нам вероятность 70%, что наш выход равен 1.

[](about:blank)

Наша вероятность того, что наше предсказание равна 0, является просто дополнением нашей вероятности того, что она равна 1 (например, если вероятность того, что она равна 1, равна 70%, то вероятность того, что она равна 0, равна 30%).

### **Граница принятия решений**

Чтобы получить нашу дискретную классификацию 0 или 1, мы можем перевести вывод функции гипотезы следующим образом:

[](about:blank)

**Вывод**

1. **Логистическая регрессия - это самый простой алгоритм бинарной классификации.**
2. **Можно взять регрессионную модель и ввести пороговое значение.**
3. **Обычная регрессия плохо работает в задачах классификации за счет своей чувствительности и неограниченности.**
4. **Метод логистической регрессии основан на применении логистической или сигмоидной функции.**
5. **Логистическая регрессия - это линейная модель.**
6. **Результат работы логичтической функции часто интерпретируется как вероятность отнесения объекта к положительному классу.**
7. **Для четкой классификации обычно выбирают некоторое пороговое значение, обычно - 0,5.**

# Метод градиентного спуска для задач классификации.

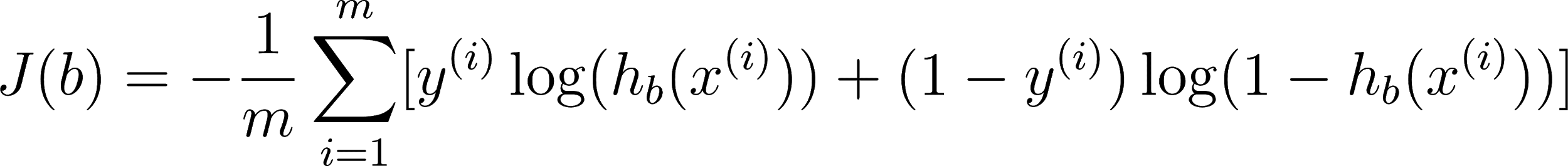
### **Градиентный спуск**

Мы можем сжать два условных случая функции стоимости в один случай:

[](about:blank)

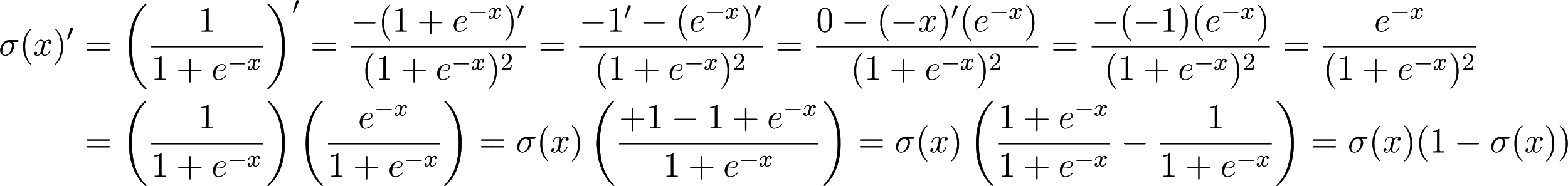
Обратите внимание, что когда y равно 1, то второй член будет равен нулю и не повлияет на результат. Если y равно 0, то первый член будет равен нулю и не повлияет на результат.

Мы можем полностью выписать всю нашу функцию затрат следующим образом:

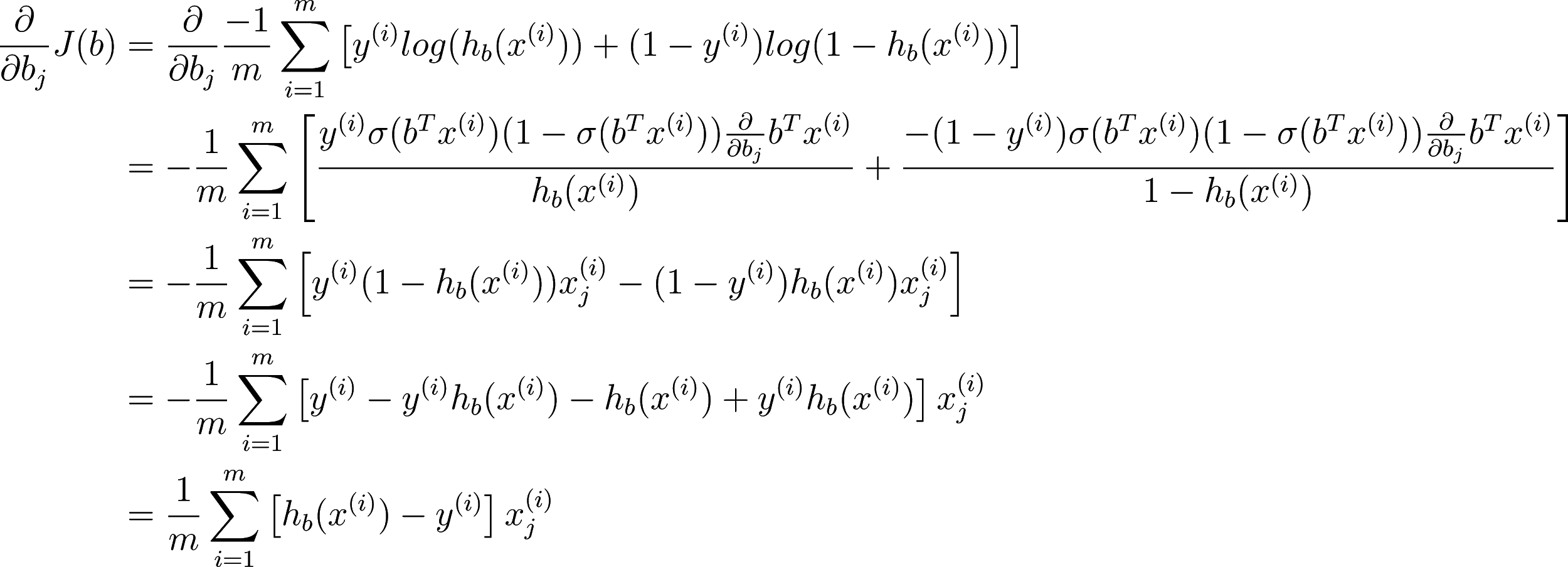
[](about:blank)

Частная производная от J (θ)

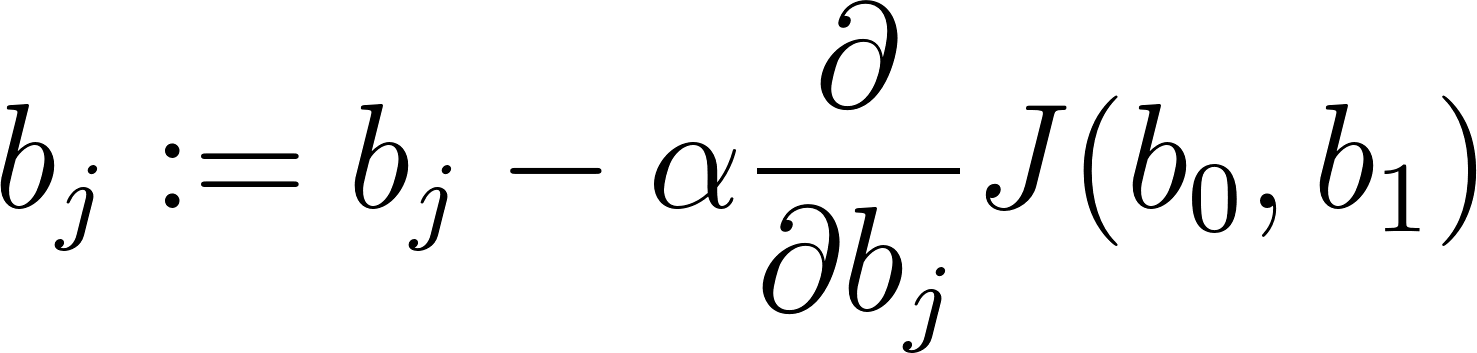
Сначала вычислим производную от сигмоидной функции (она будет полезна при нахождении частной производной от J(θ)):

[](about:blank)

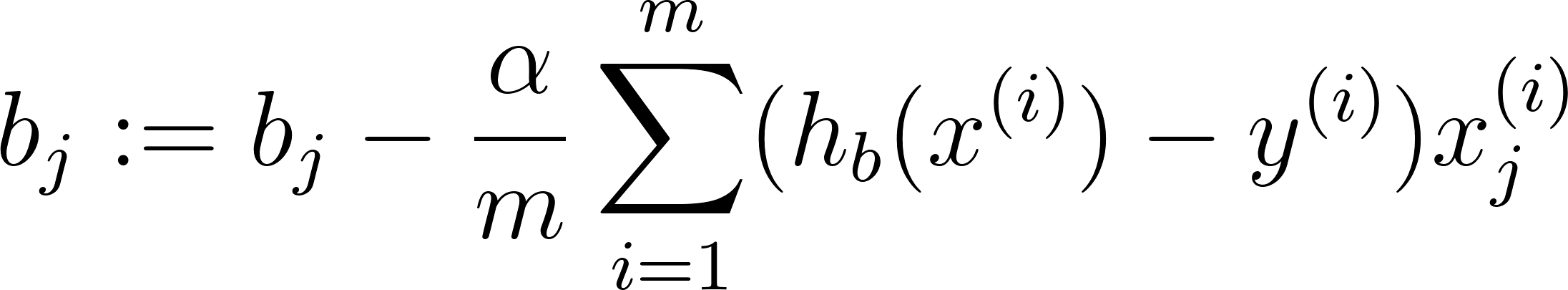
Теперь мы готовы найти полученную частную производную:

[](about:blank)

Помните, что общая форма градиентного спуска:

[](about:blank)

Мы можем разработать производную часть, используя вышеприведенное дифференцирование, чтобы получить:

[](about:blank)

Обратите внимание, что этот алгоритм идентичен тому, который мы использовали в линейной регрессии. Мы все равно должны одновременно обновлять все значения в b.

Сопряженный градиентный спуск, BFGS и L-BFGS - это более сложные и быстрые способы оптимизации b, которые можно использовать вместо градиентного спуска. Лучше не писать программные реализации этич более сложных алгоритмов самостоятельно (если только вы не являетесь экспертом в численных вычислениях), но вместо этого используйте библиотеки, поскольку они уже протестированы и сильно оптимизированы. Библиотека scikit learn реализует многие эти алгоритмы.

# Логистическая регрессия в задачах классфикации.

## ***Логистическая регрессия***

Теперь мы переходим от регрессионных проблем к задачам классификации. Пусть название «логистическая регрессия» не вводит в заблуждение; метод назван таким образом по историческим причинам и на самом деле является подходом к проблемам классификации, а не регрессионным проблемам.

### **Бинарная классификация**

Вместо того, чтобы наш выходной вектор y был непрерывным диапазоном значений, он будет равен только 0 или 1.

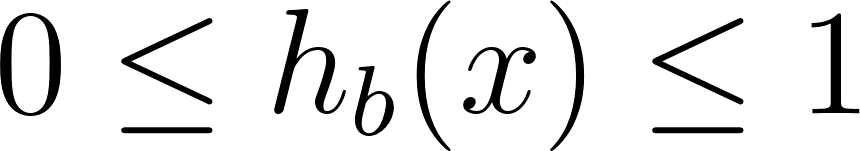
y∈ {0,1}

Где 0 обычно принимается как «отрицательный класс» и 1 как «положительный класс», но вы можете назначить ему какое-либо представление. На данный момент мы занимаемся только двумя классами, называемыми проблемой двоичной или бинарной классификации.

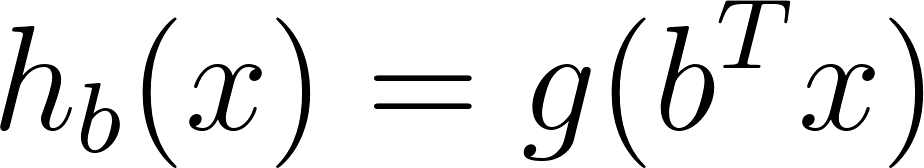
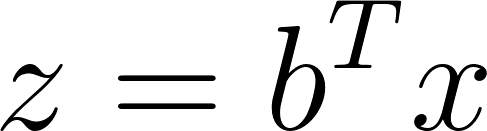
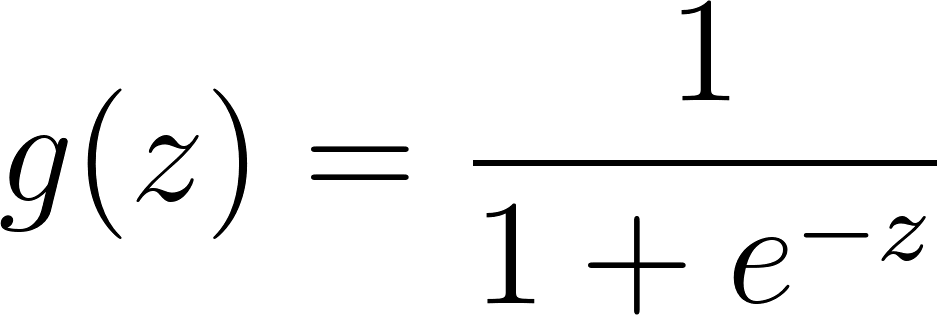
Один из методов - использовать линейную регрессию и отображать все предсказания, превышающие 0,5, как 1 и все меньше 0,5 в качестве 0. Этот метод не работает, потому что классификация на самом деле не является линейной функцией.

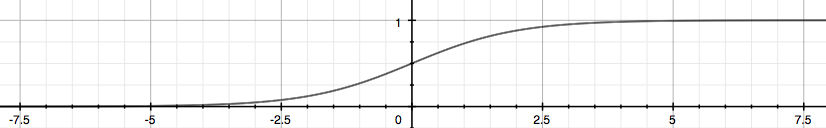
### **Представление гипотезы**

Наша гипотеза должна удовлетворять:

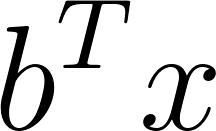
[](about:blank)

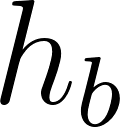
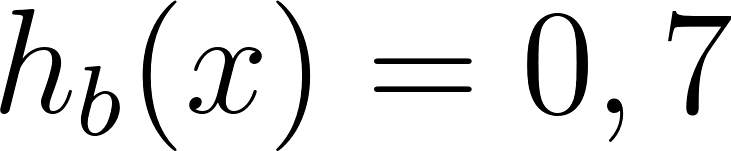
В этой новой форме задачи используется «сигмоидальная функция», также называемая «логистическая функция»:

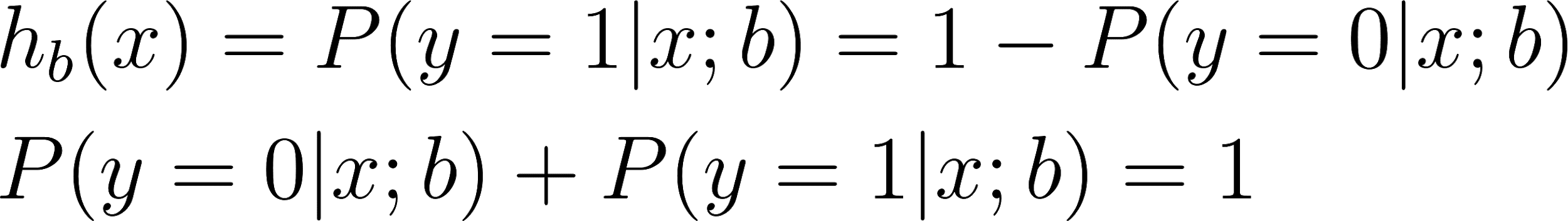
[](about:blank)   
 [](about:blank)  
[](about:blank)



Показанная здесь функция g(z) отображает любое вещественное число на интервал (0, 1), что делает его полезным для преобразования произвольнозначной функции в функцию, более подходящую для классификации.

Начнем с нашей старой гипотезы (линейной регрессии), за исключением того, что мы хотим ограничить диапазон до 0 и 1. Это достигается путем подключения [](about:blank) в логистическую функцию.

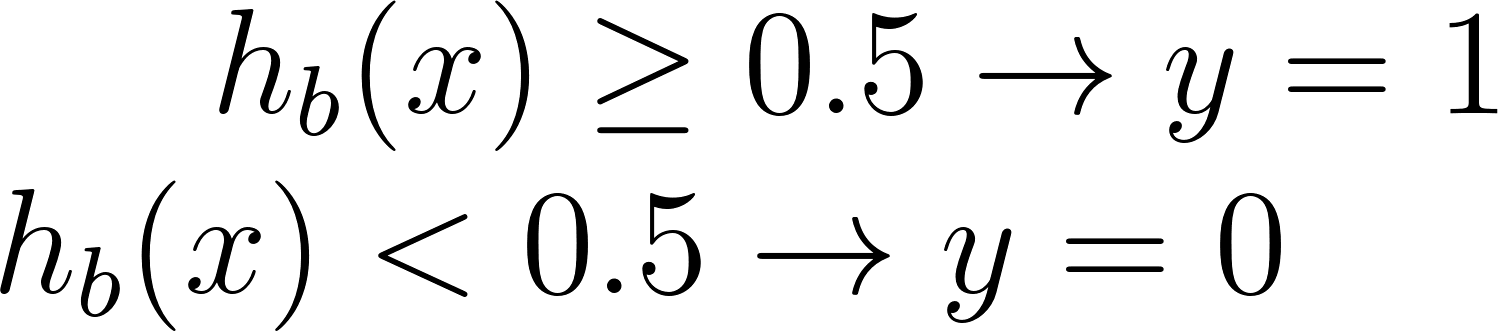
[](about:blank) даст нам вероятность того, что наш результат равен 1. Например,  дает нам вероятность 70%, что наш выход равен 1.

[](about:blank)

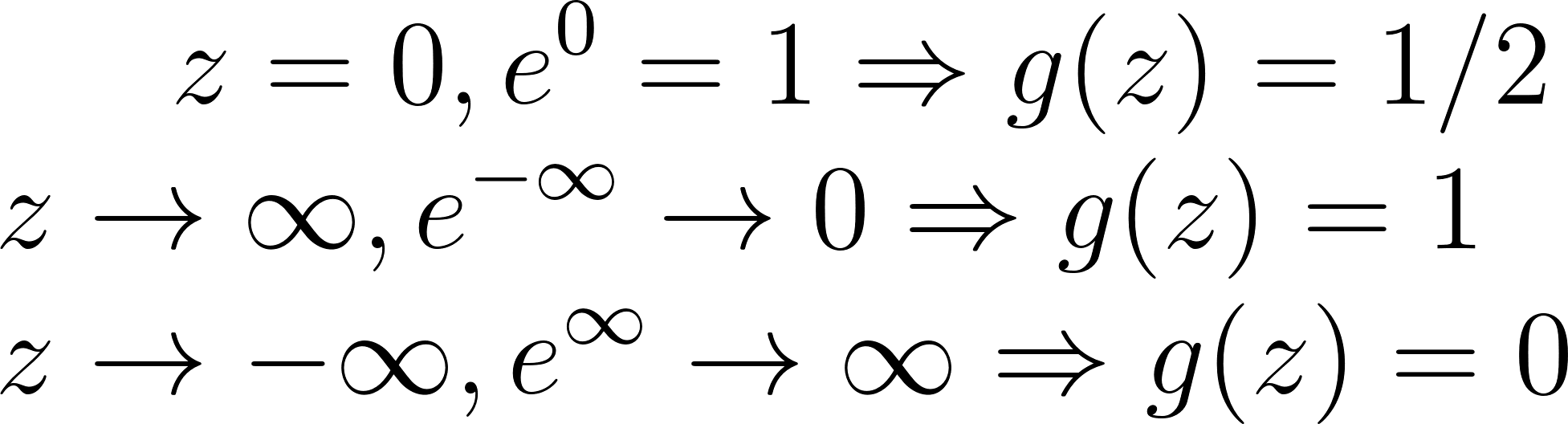
Наша вероятность того, что наше предсказание равна 0, является просто дополнением нашей вероятности того, что она равна 1 (например, если вероятность того, что она равна 1, равна 70%, то вероятность того, что она равна 0, равна 30%).

### **Граница принятия решений**

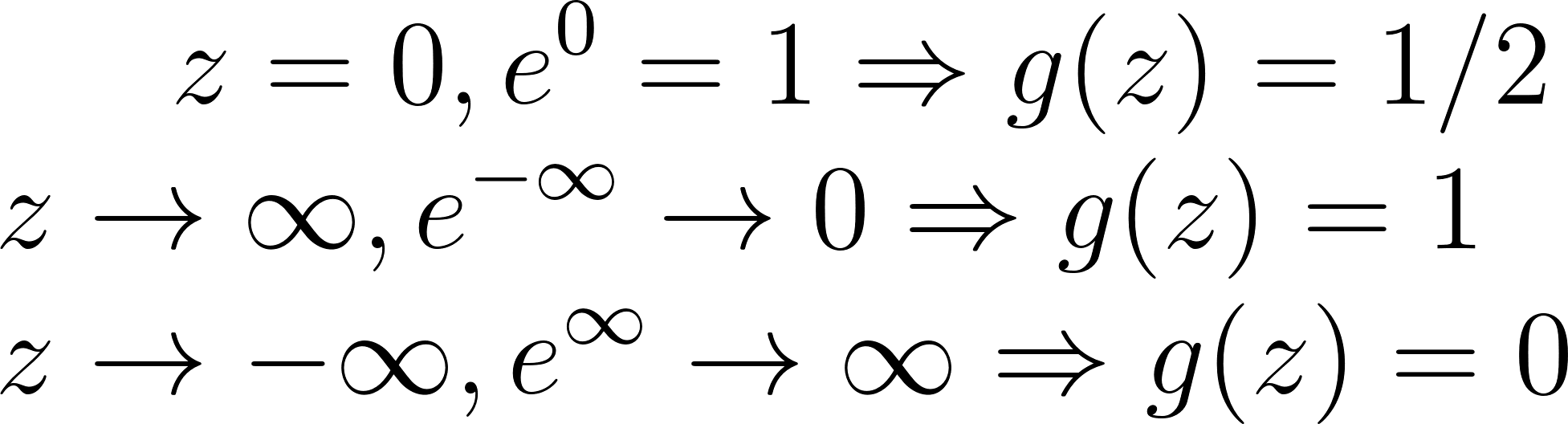
Чтобы получить нашу дискретную классификацию 0 или 1, мы можем перевести вывод функции гипотезы следующим образом:

[](about:blank)

Логистическая функция g ведет себя таким образом, что когда ее вход больше или равен нулю, его выход больше или равен 0,5. Следует запомнить:

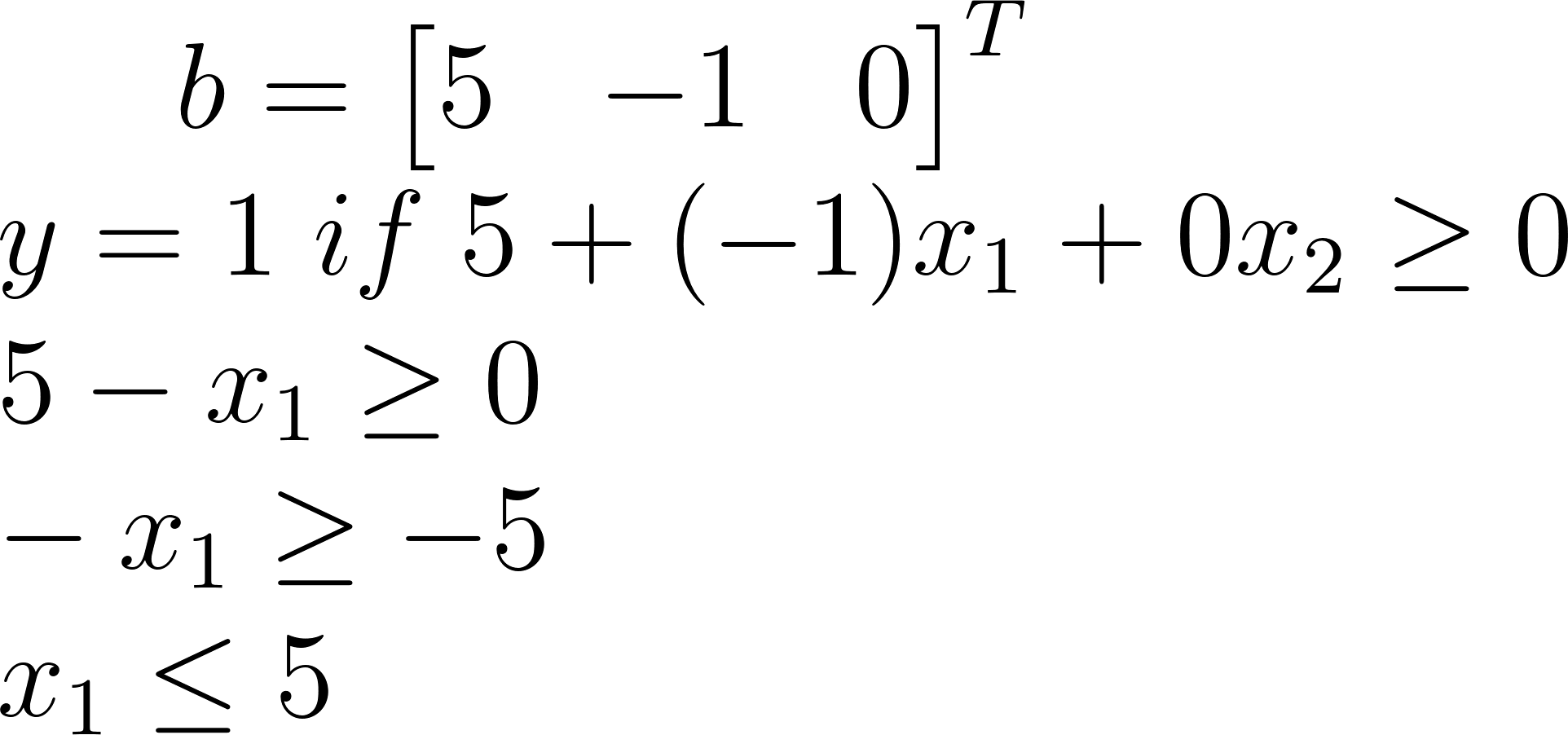
[](about:blank)

Отсюда:

[](about:blank)

Граница принятия решения - это линия, которая разделяет область, где y = 0 и где y = 1. Она создается нашей функцией гипотезы.

Пример:

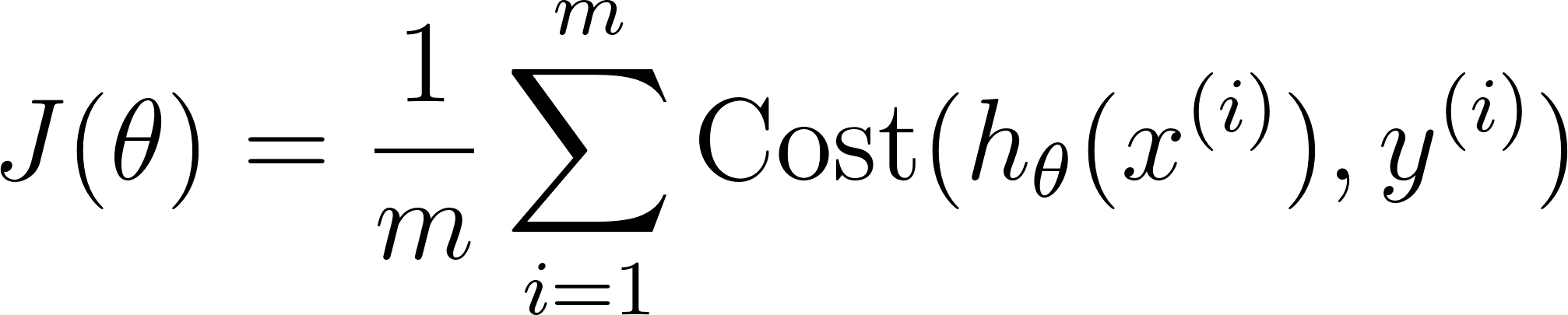
[](about:blank)

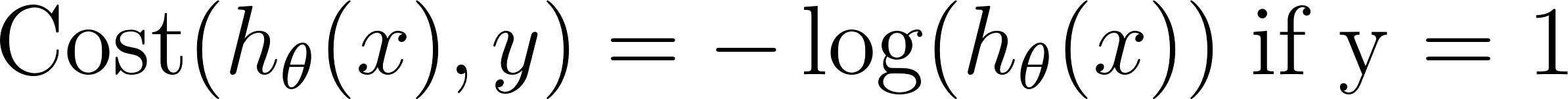
В этом случае наша граница решения является прямой вертикальной линией, расположенной на графике, где x1 = 5, а все слева от нее обозначает y = 1, а все вправо обозначает y = 0.

### **Функция ошибки**

Мы не можем использовать одну и ту же функцию ошибки, которую мы используем для линейной регрессии, потому что логистическая функция вызовет волнообразный вывод, вызывая множество локальных оптимумов. Другими словами, это не будет выпуклая функция.

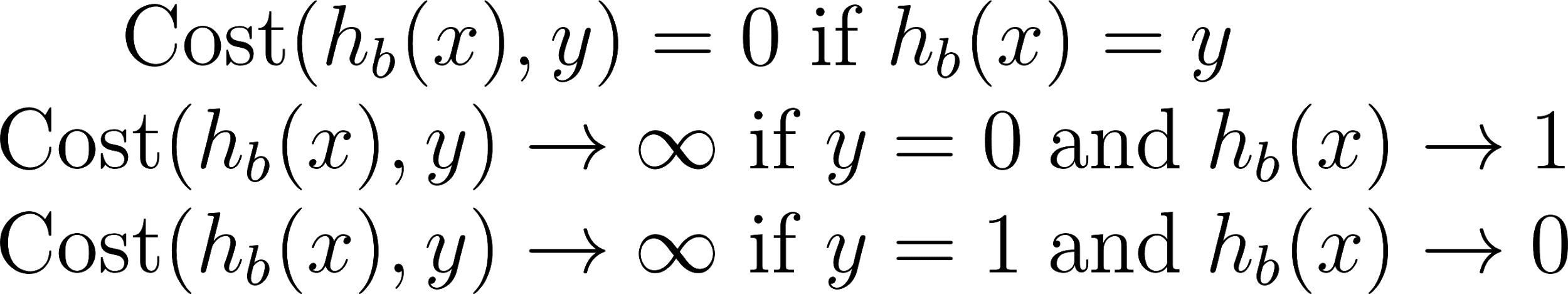
Вместо этого наша функция стоимости для логистической регрессии выглядит так:

[](about:blank), где

[](about:blank)

[](about:blank)

Чем больше наша гипотеза отключена от y, тем больше выходная функция стоимости. Если наша гипотеза равна y, то наша стоимость равна 0:

[](about:blank)

Если наш правильный ответ y равен 0, тогда функция стоимости будет равна 0, если наша функция гипотезы также выдает 0. Если наша гипотеза приближается к 1, то функция стоимости будет приближаться к бесконечности.

Если наш правильный ответ y равен 1, функция стоимости будет равна 0, если наша функция гипотезы выйдет 1. Если наша гипотеза приближается к 0, то функция стоимости приблизится к бесконечности.

Заметим, что запись функции стоимости таким образом гарантирует, что J(b) выпукла для логистической регрессии.

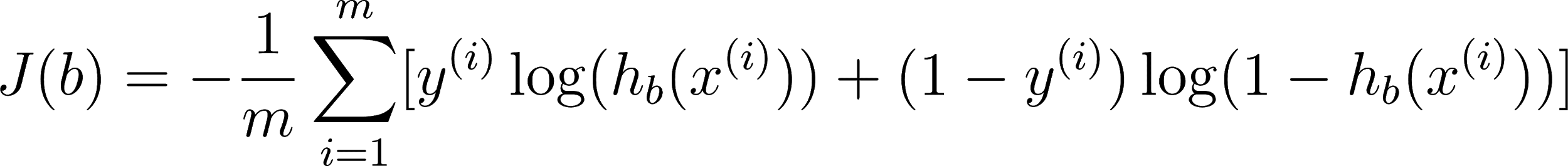
### **Градиентный спуск**

Мы можем сжать два условных случая функции стоимости в один случай:

[](about:blank)

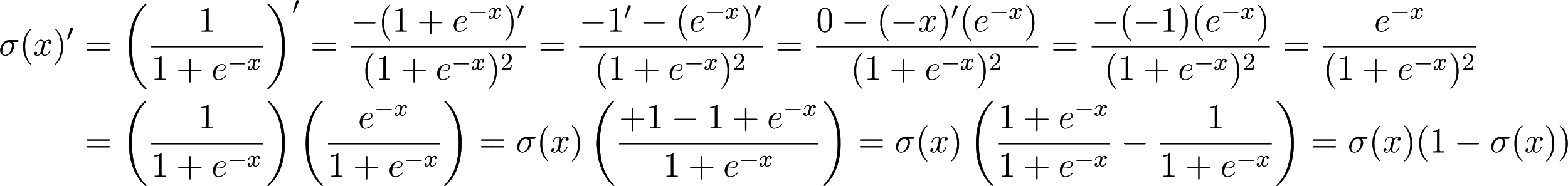
Обратите внимание, что когда y равно 1, то второй член будет равен нулю и не повлияет на результат. Если y равно 0, то первый член будет равен нулю и не повлияет на результат.

Мы можем полностью выписать всю нашу функцию затрат следующим образом:

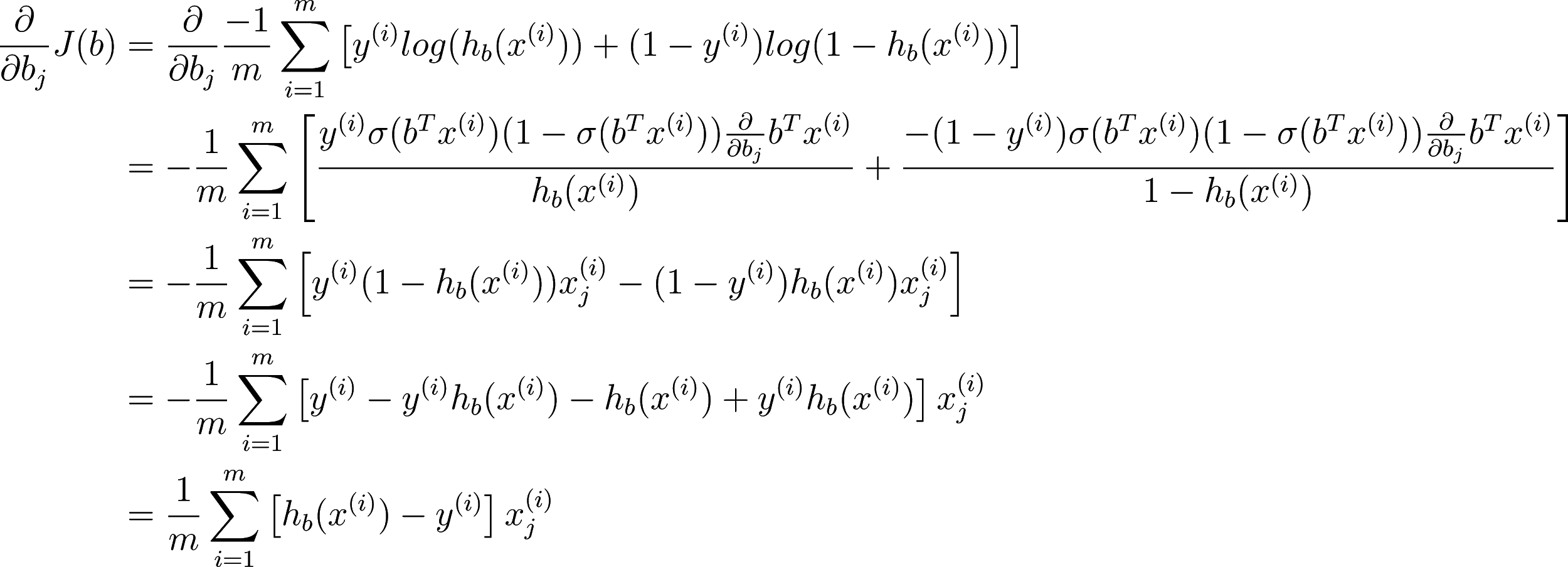
[](about:blank)

Частная производная от J (θ)

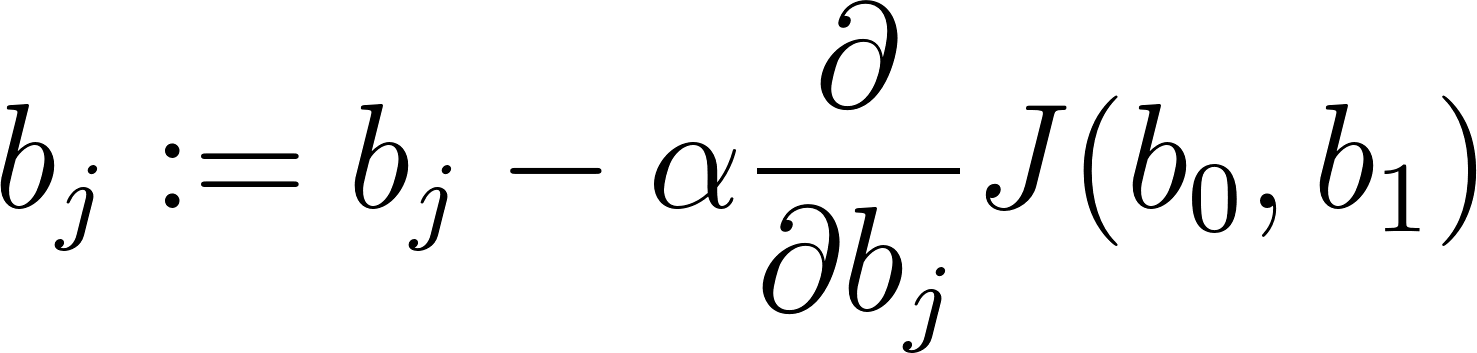
Сначала вычислим производную от сигмоидной функции (она будет полезна при нахождении частной производной от J(θ)):

[](about:blank)

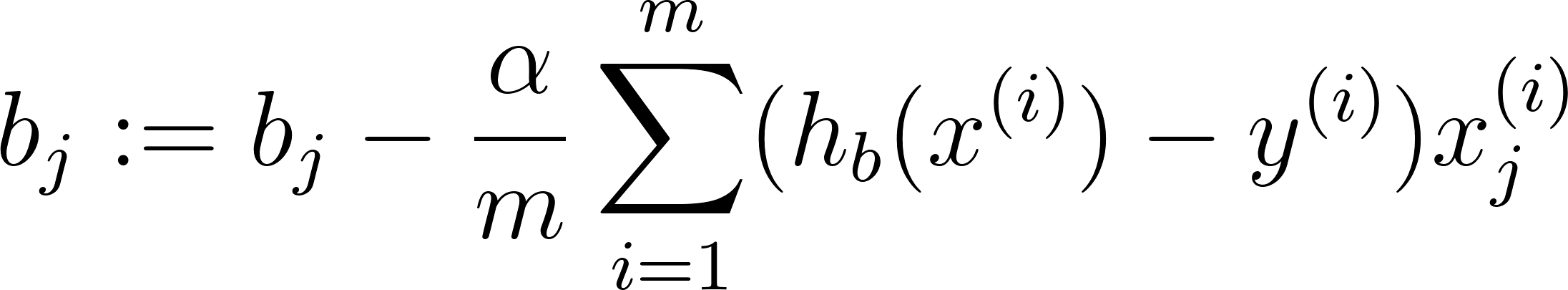
Теперь мы готовы найти полученную частную производную:

[](about:blank)

Помните, что общая форма градиентного спуска:

[](about:blank)

Мы можем разработать производную часть, используя вышеприведенное дифференцирование, чтобы получить:

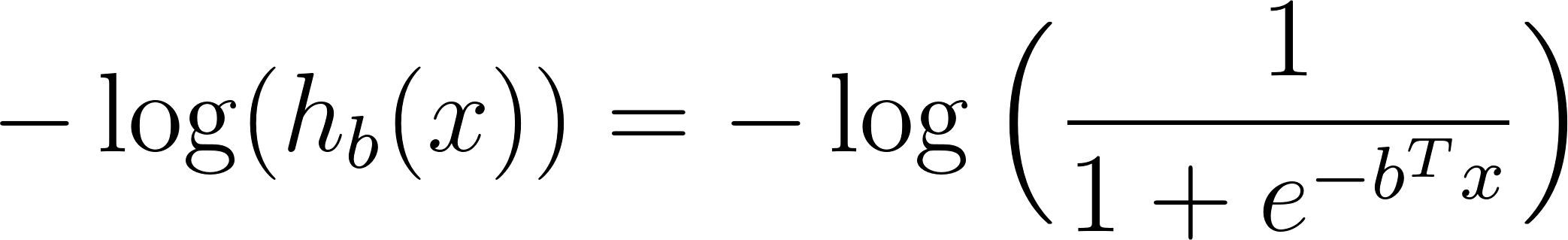
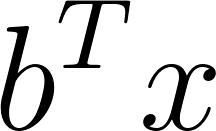
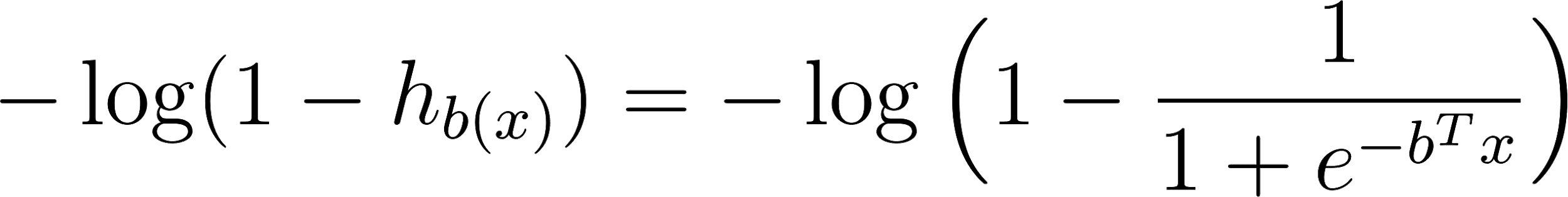
[](about:blank)

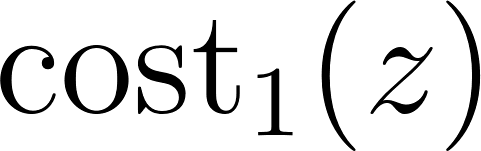
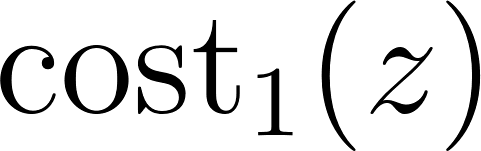
Обратите внимание, что этот алгоритм идентичен тому, который мы использовали в линейной регрессии. Мы все равно должны одновременно обновлять все значения в b.

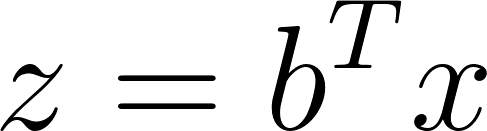
Сопряженный градиентный спуск, BFGS и L-BFGS - это более сложные и быстрые способы оптимизации b, которые можно использовать вместо градиентного спуска. Лучше не писать программные реализации этич более сложных алгоритмов самостоятельно (если только вы не являетесь экспертом в численных вычислениях), но вместо этого используйте библиотеки, поскольку они уже протестированы и сильно оптимизированы. Библиотека scikit learn реализует многие эти алгоритмы.

# Метод опорных векторов в задачах классификации.

## ***Метод опорных векторов***

Метод опорных векторов (support vector machines, SVM) является еще одним типом алгоритма машинного обучения с учителем для задач классификации. Иногда он работает быстрее и точнее логистической регрессии. Чтобы использовать метод опорных векторов, мы модифицируем первый член функции стоимости [](about:blank), так что при [](about:blank) (здесь и далее будем обозначать эту величину как z) больше 1, оно выводит 0. Кроме того, для значений z, меньших 1, мы будем использовать прямую убывающую линию вместо сигмовидной кривой (в англоязычной литературе это называется функцией hinge loss). Аналогично, мы модифицируем второй член функции стоимости [](about:blank), так что, когда z меньше -1, алгоритм выдает 0. Мы также модифицируем его так, чтобы при значениях z больше -1, вместо сигмоидной кривой мы используем прямую растущую линию.

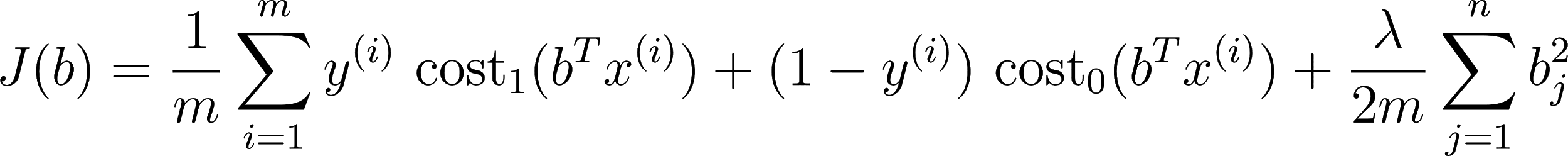
Мы будем обозначать соответствующие компоненты функции ошибки как [](about:blank) и [](about:blank) (соответственно, обратите внимание, что [](about:blank) является стоимостью для классификации при y = 1 и [](about:blank) - это стоимость классификации при y = 0), и мы можем определить их следующим образом (где k - произвольная константа, определяющая величину наклона линии):

[](about:blank)

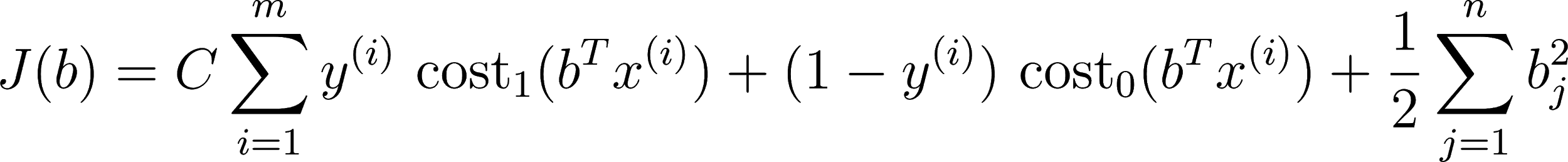
[](about:blank)

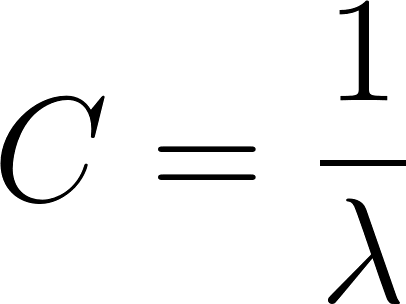
[](about:blank)

Вспомните полную функцию стоимости из регуляризованной логистической регрессии. Мы можем преобразовать ее в функцию стоимости для векторных машин поддержки, заменив cost0 и cost1.

[](about:blank)

Кроме того, общепринятое соглашение диктует, что мы регуляризуем с использованием фактора C вместо λ, тогда функция ошибки будет выглядеть следующим образом:

[](about:blank)

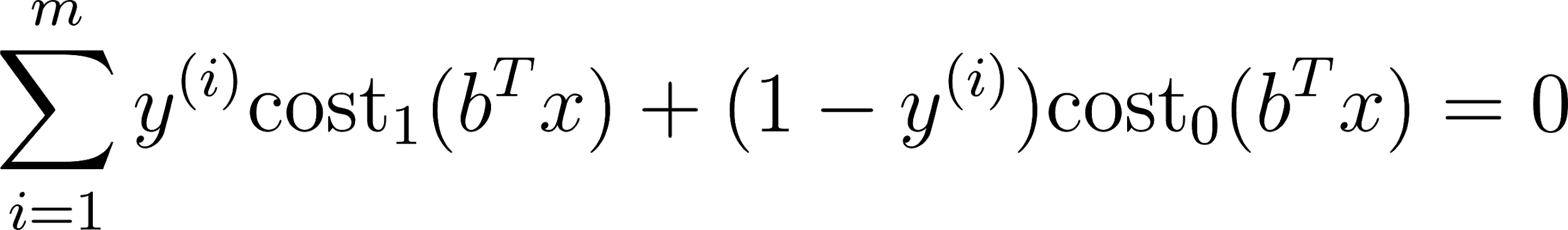
Это эквивалентно умножению уравнения на [](about:blank) и, таким образом, приводит к тем же значениям при оптимизации. Теперь, когда мы хотим еще больше регуляризовать (то есть сократить возможность переобучения), мы уменьшаем C, и когда мы хотим регуляризовать меньше (то есть уменьшать возможность недообучения), мы увеличиваем C.

Наконец, обратите внимание, что значение функции гипотезы в методе опорных векторов не интерпретируется как вероятность того, что y будет равен 1 или 0 (как для гипотезы логистической регрессии). Вместо этого он выводит либо 1, либо 0 дискретно.

Полезный способ думать о методе опорных векторов - как о методе, максимизирующем расстояние между классами и границей принятия решения. Когда мы установим нашу константу C в очень большое значение (например, 100 000), наша оптимизирующая функция будет ограничивать b так, чтобы уравнение суммы ошибки каждого примера равна 0. Мы накладываем следующие ограничения на b:

[](about:blank)

Если C очень велико, мы должны выбрать такие параметры b, что:

[](about:blank)

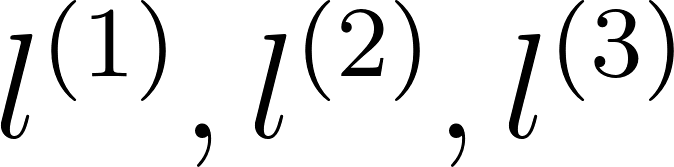
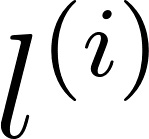
Напомним, что граница принятия решения в логистической регрессии - это линия, отделяющая положительные и отрицательные примеры. В SVM граница решения имеет особое свойство, заключающееся в том, что она как можно дальше от положительного и отрицательного примеров.

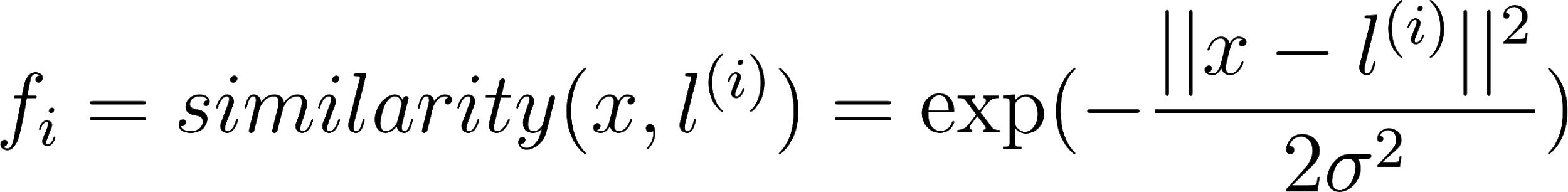
SVM будет отделять отрицательные и положительные примеры с большим отрывом. Этот большой запас достигается только тогда, когда C очень большой.

Данные называются линейно разделимыми, когда прямая линия, плоскости или гиперплоскость (в зависимости от размерности данных) может отделить положительные и отрицательные примеры. Увеличение и уменьшение C аналогично уменьшению и увеличению λ и может упростить вид границы принятия решения.

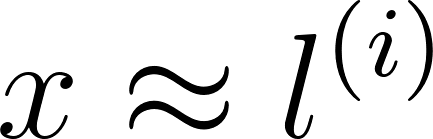
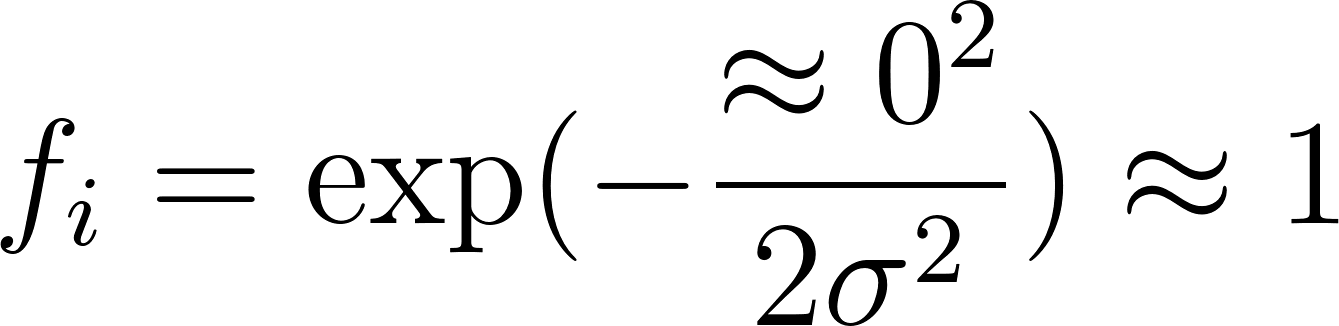
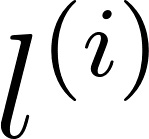
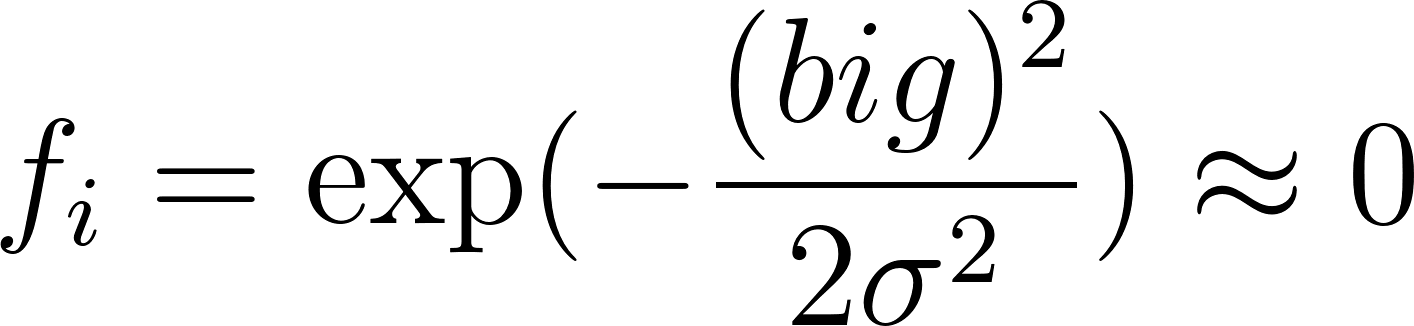
# 11. **Понятие ядра и виды ядер в методе опорных векторов.**

Ядра (kernels) позволяют нам создавать сложные нелинейные классификаторы с использованием метода опорных векторов.

При данном x, можно ввести новые признаки в зависимости от близости x к определенным заранее выбранным точкам (ориентирам), скажем [](about:blank). Для этого мы находим «подобие» x и некоторой точки [](about:blank):

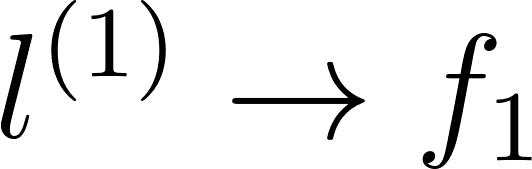
[](about:blank)

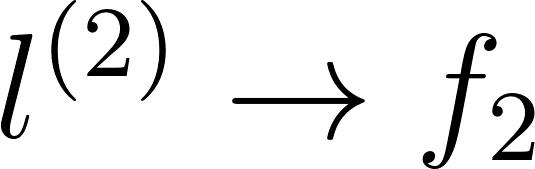
Эта функция «подобия» называется гауссовским ядром. Это конкретный пример ядра. Существует несколько свойств функции подобия:

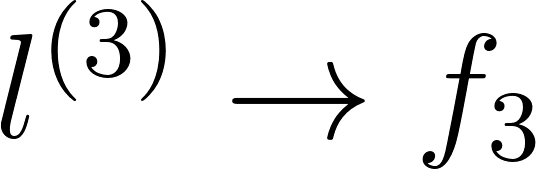
* Если [](about:blank), тогда [](about:blank)
* Если x далек от [](about:blank), тогда [](about:blank)

Другими словами, если x и ориентир близки, то сходство будет близким к 1, и если x и ориентир находятся далеко друг от друга, сходство будет близко к 0.

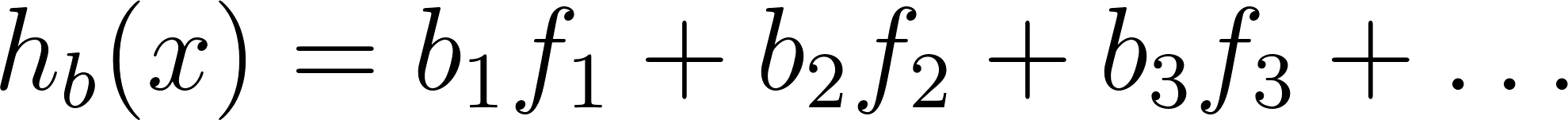
Каждый ориентир дает нам набор признаков для нашей модели:

[](about:blank)

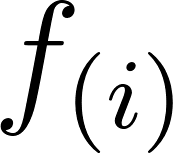
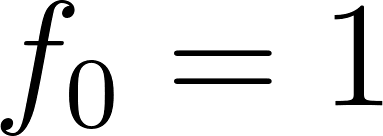
[](about:blank)

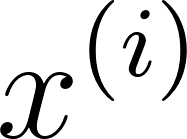
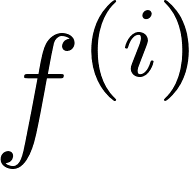
[](about:blank)

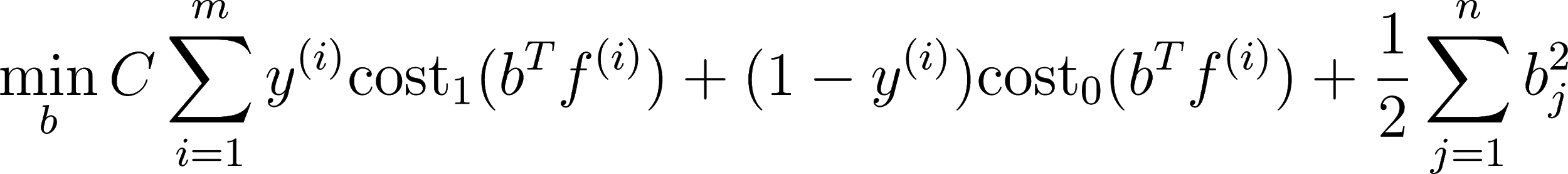
[](about:blank)

[](about:blank)

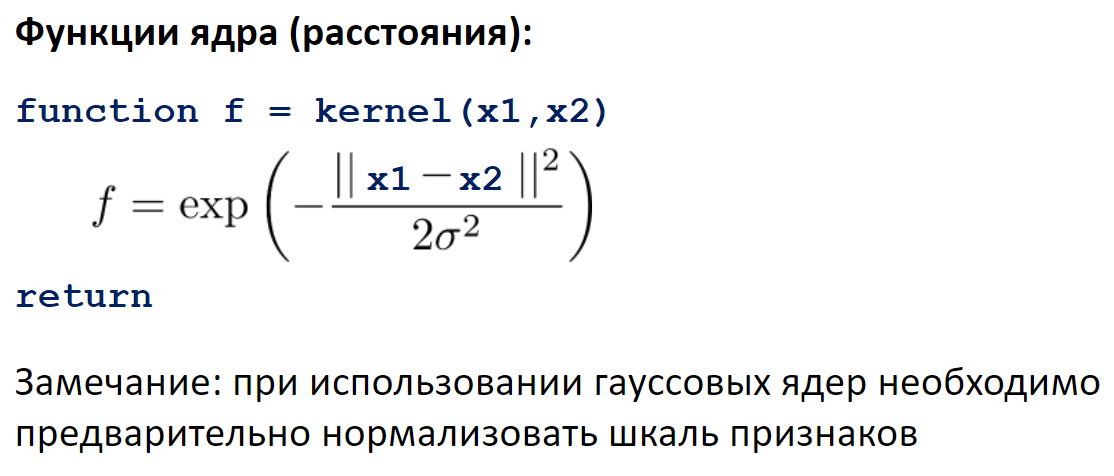
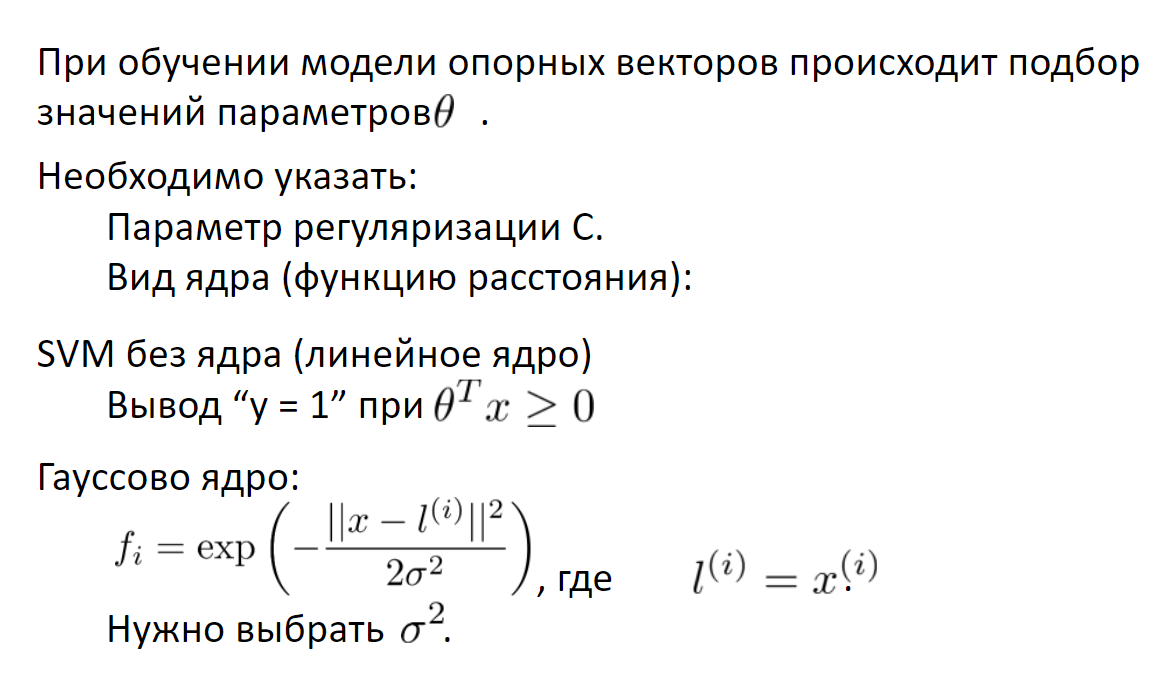
σ2 - параметр гауссовского ядра, и его можно модифицировать, чтобы увеличить или уменьшить область действия нашей функции fi. В сочетании с изменением значений внутри b, мы можем выбрать эти ориентиры, чтобы получить общую форму границы принятия решения.

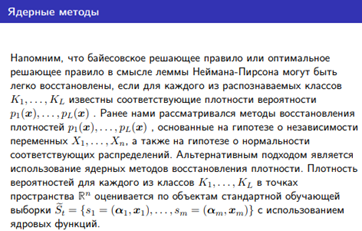
Один из способов получить ориентиры - разместить их в тех же точках гиперпространства, что и все учебные примеры. Это дает нам набор ориентиров, с одним ориентиром на каждый пример обучения. Это дает нам «вектор функций» [](about:blank) для всех наших изначальных признаков, например [](about:blank). Мы можем также установить [](about:blank), чтобы соответствовать b0. Таким образом, данный пример обучения [](about:blank).

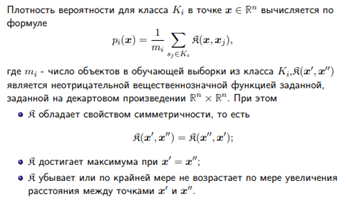
Теперь, чтобы получить параметры b, мы можем использовать алгоритм минимизации SVM, но с заменой [](about:blank) на [](about:blank):

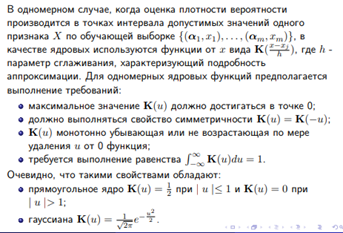
[](about:blank)

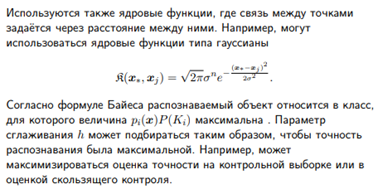
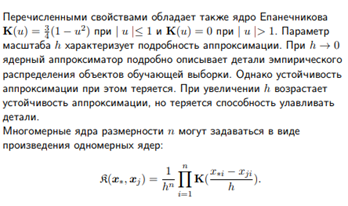
Использование ядер для генерации f(i) не является исключительным для SVM и может также применяться к логистической регрессии. Однако из-за вычислительной оптимизации в алгоритмах SVM, ядра работают с SVM, намного быстрее, чем с другими алгоритмами, поэтому ядра почти всегда встречаются вместе только с SVM.











# 12. **Метод решающих деревьев в задачах классификации.**

Как следует из определения, решающее дерево a(x) разбивает всё признаковое пространство на некоторое количество непересекающихся подмножеств {J1, . . . , Jn},

и в каждом подмножестве Jj выдаёт константный прогноз wj . Значит, соответствующий алгоритм можно записать аналитически: a(x) = Xn j=1 wj [x ∈ Jj ]. Обратим внимание, что это линейная модель над признаками ([x ∈ Jj ])n j=1 — а ведь в начале лекции мы хотели избавиться от линейности! Получается, что решающее дерево с помощью жадного алгоритма подбирает преобразование признаков для данной задачи, а затем просто строит линейную модель над этими признаками. Далее мы увидим, что многие нелинейные методы машинного обучения можно представить как совокупность линейных методов и хитрых способов порождения признаков.

Метода деревьев решений для задачи классификации состоит в том, чтобы осуществлять процесс деления исходных данные на группы, пока не будут получены однородные (или почти однородные) их множества. Совокупность правил, которые дают такое разбиение позволят затем делать прогноз (т.е. определять наиболее вероятный номер класса) для новых данных.

Метод деревьев решений применим для решения задач классификации, возникающих в самых разных областях, и считается одним из самых эффективных. Среди задач, успешно решаемых с помощью этого метода, можно назвать, например,

• скоринговые модели кредитования

• маркетинговые исследования, направленные на выявление предпочтений клиента или степени его удовлетворённости – обычно эти сведения бывают востребованы маркетинговыми агентствами или рекламными компаниями;

• диагностика (медицинская или техническая), где по набору значений факторов (симптомов, результатов анализов) нужно поставить диагноз или сделать вывод о динамике процесса. Определим основные понятия, используемые методом Деревьев решений.

**Основные понятия**

Итак, **дерево решений** – это модель, представляющая собой совокупность правил для принятия решений. Графически её можно представить в виде древовидной структуры, где моменты принятия решений соответствуют так называемым узлам. В узлах происходит ветвление процесса, т.е. деление его на так называемые ветви в зависимости от сделанного выбора. Конечные (или, что то же самое, терминальные) узлы называют листьями– каждый лист – это конечный результат последовательного принятия решений.

Данные, подлежащие классификации, находятся в так называемом «корне» дерева. В зависимости от решения, принимаемого в узлах, процесс в конце концов останавливается в одном из листьев, где переменной отклика (искомому номеру класса) присваивается то или иное значение.

Идея метода Метод деревьев решений реализует принцип так называемого «рекурсивного деления. Эта стратегия также называется «Разделяй и властвуй». В узлах, начиная с корневого, выбирается признак, значение которого используется для разбиения всех данных на 2 класса. Процесс продолжается до тех пор, пока не выполнится критерий остановки. Это возможно в следующих ситуациях:

· Все (или почти все) данные данного узла принадлежат одному и тому же классу;

· Не осталось признаков, по которым можно построить новое разбиение;

· Дерево превысило заранее заданный «лимит роста» (если таковой был заранее установлен).

# 13. **Метод k ближайших соседей в задачах классификации**.

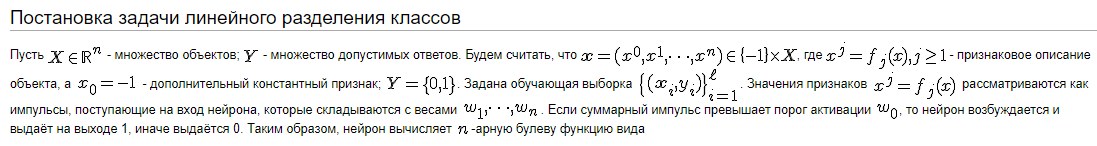
Метод K-ближайшего соседа (англ.: k-nearest neighbors method, k-NN) – один из методов решения задачи классификации. Предполагается, что уже имеется какое-то количество объектов с точной классификацией (т.е. для каждого них точно известно, какому классу он принадлежит). Нужно выработать правило, позволяющее отнести новый объект к одному из возможных классов (т.е. сами классы известны заранее). В основе k-NN лежит следующее правило: объект считается принадлежащим тому классу, к которому относится большинство его ближайших соседей. Под «соседями» здесь понимаются объекты, близкие к исследуемому в том или ином смысле. Заметим, что здесь необходимо уметь определять, насколько объекты близки друг к другу, т.е. уметь измерять «расстояние» между объектами. Это не обязательно евклидово расстояние. Это может быть мера близости объектов, например, по цвету, форме, вкусу, запаху, интересам (если речь идёт о формировании групп людей), особенностям поведения и т.д. Следовательно, для применения метода kNN в пространстве признаков объектов должна быть введена некоторая метрика (т.е. функция расстояния). Предполагается, что объекты с близкими значениями одних признаков будут близки и по другим признакам (т.е. относиться к одному и тому же классу).

https://edu.kpfu.ru/pluginfile.php/78207/mod\_resource/content/1/kNN.pdf

# 14. **Однослойный перцептрон в задачах классификации**.

**Однослойный персептрон** — это [линейный алгоритм классификации](http://www.machinelearning.ru/wiki/index.php?title=%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80), принцип работы которого основан на модели нервной клетки - [нейрона](http://www.machinelearning.ru/wiki/index.php?title=%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD&action=edit). Представляет собой пример [нейронной сети](http://www.machinelearning.ru/wiki/index.php?title=%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) с одним скрытым слоем.

<http://www.machinelearning.ru/wiki/index.php?title=%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D0%BB%D0%BE%D0%B9%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD>



проще взять по ссылке тк не дает нормально копировать. а скрины не вмещаются.

# 15. **Понятие набора данных (датасета) в машинном обучении. Требования, представление. Признаки и объекты**.

https://www.bigdataschool.ru/blog/dataset-data-preparation.html

**Dataset** для машинного обучения – это обработанная и структурированная информация в табличном виде. Строки такой таблицы называются **объектами**, а столбцы – **признаками**. Различают 2 вида признаков:

· независимые переменные – **предикторы**;

· зависимые переменные – **целевые признаки**, которые вычисляются на основе одного или нескольких предикторов.

Признаковое описание характерно для задач классификации, когда имеется выборка – конечное множество объектов, для которых известно, к каким классам они относятся. Классовая принадлежность остальных объектов неизвестна. В процессе машинного обучения строится модель, способная классифицировать произвольный объект из исходного множества. **Практический смысл задач классификации** состоит в предсказании возможных исходов на основе совокупности входных переменных, например, диагностика заболеваний, предварительная оценка эффективности месторождений полезных ископаемых, кредитный скоринг, распознавание речи, [прогнозирование](https://www.bigdataschool.ru/wiki/%d0%bf%d1%80%d0%be%d0%b3%d0%bd%d0%be%d0%b7%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5) оттока клиентов ([Churn Rate](https://www.bigdataschool.ru/wiki/churn-rate)) и т.д.

В зависимости от варианта задачи классификации, целевой признак может выглядеть по-разному:

· **один столбец с двоичными значениями** (1/0, TRUE/FALSE и пр.): двухклассовая [классификация](https://www.bigdataschool.ru/wiki/%d0%ba%d0%bb%d0%b0%d1%81%d1%81%d0%b8%d1%84%d0%b8%d0%ba%d0%b0%d1%86%d0%b8%d1%8f) (binary classification), когда каждый объект принадлежит только одному классу;

· **несколько столбцов с двоичными значениями**: задача классификации с пересекающимися классами (multi-label classification), когда один объект может принадлежать нескольким классам;

· **один столбец с действительными значениями**: регрессионный анализ, когда прогнозируется одна величина;

· **несколько столбцов с действительными значениями**: задача множественной регрессии, когда прогнозируется несколько величин.

## КАКИМ БЫВАЕТ DATASET: ТИПЫ ВЫБОРОК

Первичный набор исходных данных принято называть **генеральной совокупностью**. Процесс формирования выборок из генеральной совокупности называется **порождение данных. Выборка –** это конечное подмножество элементов генеральной совокупности, изучив которое можно понять поведение исходного множества. Например, генеральная совокупность состоит из 150 тысяч посетителей сайта, а в выборку попали 250 из них.

**Вероятностная модель порождения данных** предполагает, что выборка из генеральной совокупности формируется случайным образом. Если все ее элементы одинаково случайно и независимо друг от друга распределены по исходному множеству (генеральной совокупности), выборка называется простой. **Простая выборка** является математической моделью серии независимых опытов и, как правило, используется для машинного обучения. При этом для каждого этапа Machine Learning необходим свой набор данных:

· для непосредственного обучения модели нужна **обучающая выборка** (training sample), по которой производится настройка (оптимизация параметров) алгоритма;

· для оценки качества модели используется **тестовая (контрольная) выборка** (test sample), которая, в идеальном случае, не должна зависеть от обучающей;

· для выбора наилучшей модели машинного обучения понадобится **проверочная (валидационная) выборка** (validation sample), которая также не должна пересекаться с обучающей.

## КАК СФОРМИРОВАТЬ ВЫБОРКУ ДЛЯ DATA MINING

Методы формирования обучающих и оценочных выборок зависят от класса задачи, решаемой с помощью машинного обучения:

· **для задач классификации** данные следует разделить так, чтобы в полученных наборах численное соотношение объектов разных классов было таким же, как в исходной генеральной совокупности;

· **для задач регрессионного анализа** необходимо одинаковое распределение целевой переменной в полученных наборах, которые будут использоваться для обучения и контроля качества.

При соблюдении этих условий объемы обучающей и оценочных выборок могут существенно различаться. Например, размер валидационного датасета может составлять всего 10% генеральной совокупности. Главное в формировании выборок – ни в коем случае не объединять обучающий датасет и с оценочными (тестовым и валидационным), поскольку это грозит переобучением модели Machine Learning. В этом случае модель получит высокие оценки качества в процессе тренировки, но не покажет такого результата на реальных данных.

После того, как выборка сформирована, наступают следующие процессы [CRISP-DM](https://www.bigdataschool.ru/wiki/crisp-dm): [**очистка данных**](https://www.bigdataschool.ru/bigdata/%D0%BE%D1%87%D0%B8%D1%81%D1%82%D0%BA%D0%B0-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-data-preparation.html) и [**работа с признаками**](https://www.bigdataschool.ru/bigdata/feature-engineering-data-preparation.html): [генерация](https://www.bigdataschool.ru/bigdata/feature-extraction-text-data-preparation.html), трансформация, [нормализация](https://www.bigdataschool.ru/bigdata/%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-feature-transformation-data-preparation.html) и отбрасывание лишних переменных, чтобы исключить [**мультиколлинеарность**](https://www.bigdataschool.ru/wiki/%d0%bc%d1%83%d0%bb%d1%8c%d1%82%d0%b8%d0%ba%d0%be%d0%bb%d0%bb%d0%b8%d0%bd%d0%b5%d0%b0%d1%80%d0%bd%d0%be%d1%81%d1%82%d1%8c) **факторов** и понизить размерность модели Machine Learning.

# 16. **Шкалы измерения признаков. Виды шкал, их характеристика.**

Шкала измерения в [статистике](https://wiki.loginom.ru/articles/mathematical-statistics.html) — это способ представления переменных (признаков, [атрибутов](https://wiki.loginom.ru/articles/attribute.html)) и их группировки в различные категории. Она определяет характер значений, присвоенных переменным в наборе данных.

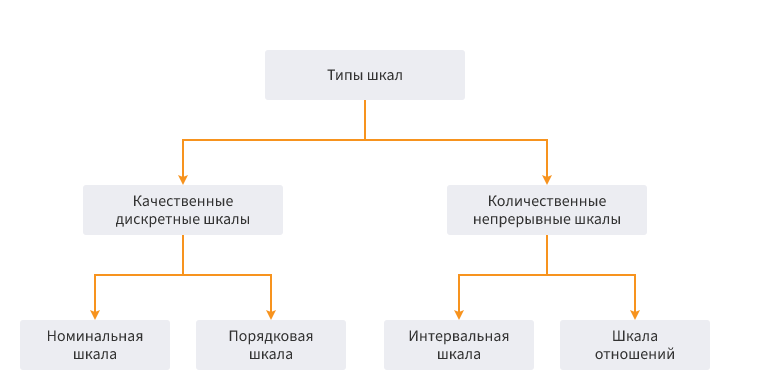
Шкала измерений формируется на основе двух ключевых понятий — измерение и масштабирование. Измерение — это процесс записи наблюдений, собранных в рамках исследования. Масштабирование — присвоение объектам числовых значений или определённой семантики. Эти два понятия, объединенные вместе, образуют связи между объектами и наблюдениями.

Шкала измерения используется для определения и описания переменных в наборах данных. Она определяет методы, которые могут быть использованы для их [анализа](https://wiki.loginom.ru/articles/data-analysis.html). В зависимости от типа анализируемых данных определяется тип шкалы измерения. Выделяют 4 основных вида шкал: номинальная, порядковая, интервальная и шкала отношений.

Шкалы измерения используются для представления как качественных, так и количественных данных. Номинальная и порядковая шкалы используются для измерения качественных данных, в то время как интервальная и шкала отношений используются для измерения количественных.

Основными свойствами шкал измерений являются:

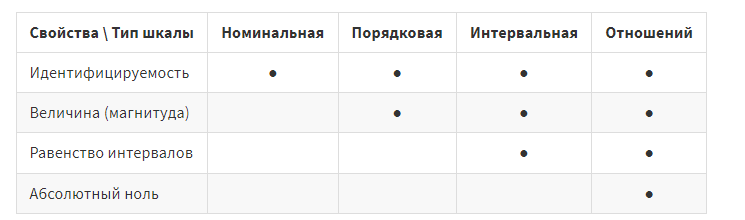
1. Идентифицируемость — возможность присвоения числовых значений каждой переменной в наборе данных. Например, в анкете запрашивается пол респондента — «Мужчина» и «Женщина». Для этих двух значений могут быть определены идентифицирующие значения — 1 и 2 соответственно. К таким значениям не могут быть применены арифметические операции, потому что они служат только для идентификации, а не описания.
2. Величина (магнитуда) — это размерность шкалы измерения, где значения могут быть упорядочены от наименьшего к наибольшему. Например, место в соревновании распределяется от 1-го, 2-го, 3-го до наименьшего.
3. Равенство интервалов — означают, что шкала имеет стандартизированный порядок, т.е. разность между двумя любыми соседними уровнями шкалы одинакова. Упорядоченность шкалы не гарантирует равенство интервалов. Например, в примере с местами в соревновании, каждая позиция имеет одинаковую разницу интервалов равную 1, но при этом 2-й участник может финишировать на 20 секунд позже, чем первый, а третий на 40 секунд позже, чем второй.
4. Абсолютный ноль — естественное и однозначное присутствие нулевой точки, изменение которой невозможно. Данная точка характеризует отсутствие измеряемого признака. Например, 0 градусов по Кельвину является абсолютным нулем на шкале, а 0 градусов по Цельсию — нет, т.к. за него принято одно из произвольно взятых физических явлений — температура плавления льда.



Зная различные уровни измерений данных, можно выбрать наилучший метод анализа.

1. **Номинальная шкала (категориальная, наименований)** — это шкала измерения, которая используется для идентификации. Она является самой «слабой» из четырех видов шкал в смысле возможности обработки данных. Она присваивает номера атрибутам для удобства идентификации, но может использоваться только как метка. Единственный вид статистического анализа, который можно выполнить с использованием номинальной шкалы, это вычисление процентных долей и частот. Данные в номинальной шкале можно проанализировать графически с помощью [гистограммы](https://wiki.loginom.ru/articles/histogram.html) и круговой диаграммы. Например, если измерить атрибут «Товар» в номинальной шкале, то она будет выглядеть так: 1 — мороженное; 2 — соки; 4 — выпечка. При этом значения шкалы не определяют какого-либо приоритета между товарами, а просто идентифицируют их. Очевидно, что такая шкала может использоваться только для самого просто анализа.
2. **Порядковая шкала (ординальная, ранговая)** — предполагает ранжирование (упорядочивание) значений переменной в зависимости от масштабирования. Атрибуты в порядковой шкале обычно располагаются в порядке возрастания или убывания. Порядковая шкала может быть использована в исследованиях рынка, рекламы и опросов удовлетворенности клиентов. Она использует квалификаторы, такие как «очень», «высоко», «больше», «меньше» и т. д. В порядковой шкале можно использовать для статистического анализа такие статистики как [медиана](https://wiki.loginom.ru/articles/median.html), но не среднее значение. Существуют и другие виды анализа, которые могут быть проведены с использованием порядковой шкалы. Например, компания-разработчик ПО может провести опрос пользователей для оценки нового приложения в шкале: «Отлично», «Очень хорошо», «Хорошо», «Плохо», «Очень плохо». Атрибуты в этом примере перечислены в порядке убывания.
3. **Интервальная шкала (разностей)** — это шкала, в которой уровни упорядочены, а интервалы между ними равны. Её можно рассматривать как расширение порядковой шкалы. Основным отличием является свойство равных интервалов. Интервальная шкала не только позволяет однозначно определить, какое значение больше (меньше), но и на сколько. Кроме того, в отличие от порядковой и номинальной шкал, в интервальной могут выполняться арифметические операции. Типичным примером является измерение температуры по шкале Фаренгейта. Интервальную шкалу можно использовать при расчете среднего значения, медианы, [моды](https://wiki.loginom.ru/articles/mode.html), [стандартного отклонения](https://wiki.loginom.ru/articles/mean-square-deviation.html) и других статистик.
4. **Шкала отношений (абсолютная)** является «наивысшим» уровнем представления данных. Она может рассматриваться как расширение интервальной шкалы, и следовательно, удовлетворяет четырем свойствам шкалы измерения: идентифицируемостью, величиной, равноинтервальностью и наличием абсолютного нуля. Примерами шкал отношения являются длина, вес, время и т. д. В исследованиях рынка примерами шкалы отношений являются цена, количество клиентов, суммы продаж и т. д. Она широко используется в маркетинге и рекламе. Шкала отношений совместима со всеми методами статистического анализа и может использовать как показатели центральной тенденции (среднее значение, медиана, мода и т. д.), так и разброса значения ([дисперсии](https://wiki.loginom.ru/articles/variance.html), размаха, стандартного отклонения и т. д.).

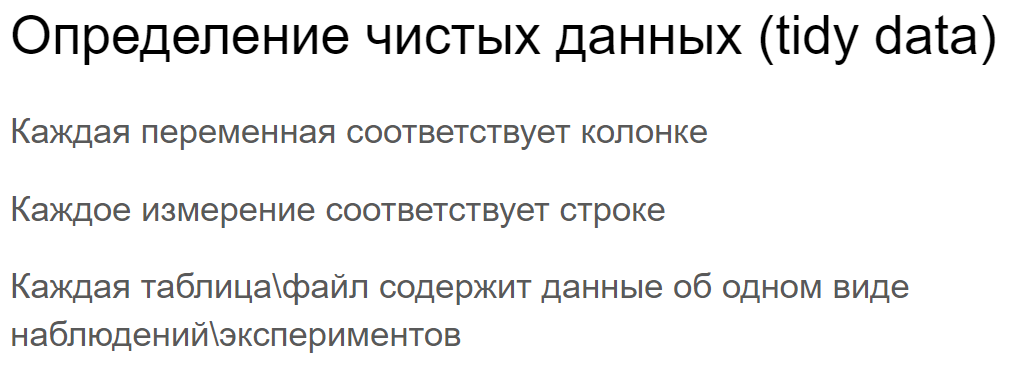
Сравнение типов шкал:

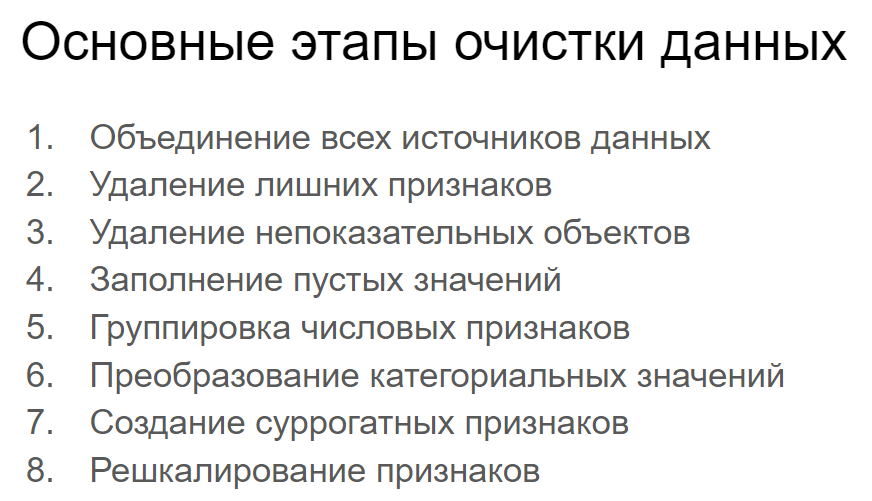


Кроме основных четырёх упомянутых типов, шкалы могут быть разделены на компаративные (сравнивающие) и некомпаративными (не сравнивающие). Компаративные шкалы позволяют устанавливать отношения сравнения между объектами (например, товар А продаётся в 5 раз чаще, чем товар Б). Иными словами, один объект оценивается путём прямого сравнения с другим. Некомпоративные шкалы позволяют оценивать объекты только по отдельности, без возможности сравнения с другими объектами.

Понимание концепции шкал измерений является необходимым условием для корректной обработки данных и проведения статистического анализа.

# 17. **Понятие чистых данных. Определение, очистка данных.**

****

****

# 18. **Основные этапы проекта по машинному обучению.**

[**http://neerc.ifmo.ru/wiki/index.php?title=%D0%96%D0%B8%D0%B7%D0%BD%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9\_%D1%86%D0%B8%D0%BA%D0%BB\_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8\_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE\_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F**](http://neerc.ifmo.ru/wiki/index.php?title=%D0%96%D0%B8%D0%B7%D0%BD%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%86%D0%B8%D0%BA%D0%BB_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F)

Жизненный цикл модели машинного обучения — это многоэтапный процесс, в течении которого исследователи, инженеры и разработчики обучают, разрабатывают и обслуживают модель машинного обучения. Разработка модели машинного обучения принципиально отличается от традиционной разработки программного обеспечения и требует своего собственного уникального способа разработки. Модель машинного обучения — это приложение искусственного интеллекта (ИИ), которое дает возможность автоматически учиться и совершенствоваться на основе собственного опыта без явного участия человека. Основная цель модели заключается в том, чтобы компания смогла использовать преимущества алгоритмов искусственного интеллекта и машинного обучения для получения дополнительных конкурентных преимуществ. Над каждым этапом работает SCRUM-команда. Сотрудничество команд организуется по методике SCRUM of SCRUMs

## Бизнес-анализ

## Анализ и подготовка данных

### **Анализ данных**

### **Сбор данных**

### **Нормализация данных**

### **Моделирование данных**

### **Конструирование признаков**

## Моделирование

### **Выбор алгоритма**

### **Планирование тестирования**

### **Обучение модели**

### **Оценка результатов**

## Оценка решения

## Внедрение

## Тестирование и мониторинг

### **Дифференциальные тесты**

### **Контрольные тесты**

### **Нагрузочные / стресс-тесты**

### **A/B-тестирование**

# 19. **Предварительный анализ данных: задачи, методы, цели**.

<http://elibrary.sgu.ru/uch_lit/1141.pdf>

**знать**

* • основные критерии классификации наборов данных и виды классификации;
* • основные виды графического представления данных и методы их группировки;
* • формулы расчета основных числовых характеристик количественных данных;

**уметь**

* определять тип шкалы измерения переменной и данных по упорядоченности во времени;
* таблично и графически изображать данные всех типов в наиболее удачной форме;
* строить таблицы частот и вариационные ряды – дискретные и интервальные;
* строить различные типы графиков и интерпретировать их;
* рассчитывать числовые характеристики количественных данных и интерпретировать их;
* находить основные показатели динамики временных рядов, строить на их основе прогнозы;

**владеть**

* категориями и понятиями современной классификации статистических данных;
* категориями, понятиями и методами современной описательной (дескриптивной) статистики и анализа временных рядов.

## Классификация статистических данных

### Критерии классификации данных

В процессе управления экономическими и техническими системами статистические методы позволяют выработать обоснованные решения, сочетающие интуицию и опыт специалиста с тщательным анализом имеющейся информации. И с каждым годом интерес к статистической обработке данных неуклонно возрастает, так как объемы окружающей нас информации угрожающе увеличиваются и без грамотной их обработки и представления, исследования закономерностей невозможно правильно принимать решения на их основе. При этом анализ данных может проводиться с целью:

* анализа и отображения конкретной собранной информации – в этом случае говорят о статистическом описании, описательной (дескриптивной) статистике (descriptive statistics);
* описания всего класса явлений по имеющимся выборочным данным, характеризующим только часть этого класса. Эти задачи относятся к аналитической статистике.

Как правило, любое статистическое исследование начинается с дескриптивной статистики, а потом уже при необходимости углубляется аналитической.

Под данными (data) в статистике понимают совокупность сведений, зафиксированных на определенном носителе в форме, пригодной для их постоянного хранения, передачи и обработки.

В статистике для характеристики изучаемых объектов используются различные типы данных, и к каждому типы применимы свои методы их обработки. Поэтому прежде всего необходимо определиться с их классификацией.

Основные критерии классификации наборов статистических данных следующие:

* 1) по числу переменных, характеризующих объект исследования, различают одномерные, двумерные и многомерные данные;
* 2) по наличию или отсутствию упорядочения во времени различают пространственные, временные и пространственно-временные данные;
* 3) по типу шкалы измерения каждого признака различают количественные (числовые) признаки, которые делятся на дискретные и непрерывные, и качественные (категориальные) признаки, которые делятся на номинальные и порядковые;
* 4) по способу получения данные делятся на первичные – если информация собиралась специально для данного анализа и вторичные – если используется информация из других источников, собранная для других целей.

# 20. **Проблема отсутствующих данных: причины, исследование, пути решения.**

В [статистике](https://www.hmong.press/wiki/Statistics) , **отсутствуют данные** , или **пропущенные значения** , происходят , когда нет [данных](https://www.hmong.press/wiki/Data) [значение](https://www.hmong.press/wiki/Value_(mathematics)) не сохраняется в памяти для [переменной](https://www.hmong.press/wiki/Variable_(mathematics)) в качестве [наблюдения](https://www.hmong.press/wiki/Unit_of_observation#Data_point) . Отсутствующие данные являются обычным явлением и могут существенно повлиять на выводы, которые можно сделать на основе данных.

Отсутствие данных может произойти из-за отсутствия ответа: информация не предоставляется ни по одному, ни по нескольким элементам, ни по всей единице («теме»). Некоторые вопросы с большей вероятностью вызовут отказ от ответа, чем другие: например, вопросы о частных предметах, таких как доход. [Истощение](https://www.hmong.press/wiki/Attrition_(epidemiology)) - это тип упущений, который может возникать в продольных исследованиях, например, при изучении развития, когда измерение повторяется через определенный период времени. Пропуск происходит, когда участники выбывают до окончания теста и одно или несколько измерений отсутствуют.

В исследованиях в области [экономики](https://www.hmong.press/wiki/Economics) , [социологии](https://www.hmong.press/wiki/Sociology) и [политологии](https://www.hmong.press/wiki/Political_science) данные часто отсутствуют, потому что правительства или частные организации предпочитают не предоставлять или не сообщать критическую статистику [[1]](https://www.hmong.press/wiki/Missing_data#cite_note-1) или потому, что информация недоступна. Иногда отсутствующие значения вызваны исследователем, например, когда сбор данных выполняется неправильно или при вводе данных допущены ошибки. [[2]](https://www.hmong.press/wiki/Missing_data#cite_note-AM_2008-2)

Эти формы отсутствия могут быть разных типов, что по-разному влияет на достоверность выводов исследования: полное отсутствие случайным образом, отсутствие случайного отсутствия и отсутствие случайного отсутствия. С отсутствующими данными можно обращаться так же, как с [цензурированными данными](https://www.hmong.press/wiki/Censored_data) .

Понимание причин отсутствия данных важно для правильной обработки оставшихся данных. Если значения отсутствуют полностью случайным образом, выборка данных, вероятно, все еще репрезентативна для генеральной совокупности. Но если значения систематически отсутствуют, анализ может быть необъективным. Например, при исследовании связи между IQ и доходом, если участники с IQ выше среднего, как правило, пропускают вопрос `` Какова ваша зарплата? '', Анализ, который не учитывает это случайное отсутствие (модель MAR) могут ошибочно не обнаружить положительной связи между IQ и зарплатой. Из-за этих проблем методологи обычно рекомендуют исследователям разработать исследования, чтобы свести к минимуму появление пропущенных значений. Графические модели могут использоваться для подробного описания механизма недостающих данных.

# 21. Проблема несбалансированных классов: исследование, пути решения.

В [машинном обучении](https://wiki.loginom.ru/articles/machine-learning.html) нередко возникают ситуации, когда в [обучающем наборе](https://wiki.loginom.ru/articles/training-set.html) данных доля [примеров](https://wiki.loginom.ru/articles/training-sample.html) некоторого [класса](https://wiki.loginom.ru/articles/class.html) оказывается слишком низкой (такой класс часто называют миноритарным), а другого — слишком большой (такой класс называют мажоритарным). Эта ситуация в теории машинного обучения известна как несбалансированность классов (class imbalance), а [классификация](https://wiki.loginom.ru/articles/classification.html) в условиях несбалансированности классов называется несбалансированной классификацией (unbalanced classification).

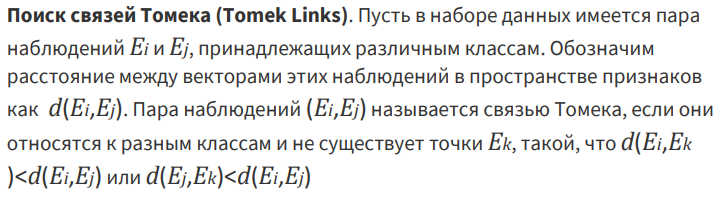
Несбалансированность классов как правило создаёт проблемы при решении [задач классификации](https://wiki.loginom.ru/articles/classification-problem.html), поскольку построенные на таких данных [модели](https://wiki.loginom.ru/articles/taught-model.html) имеют «перекос» в сторону мажоритарного класса, т.е. с большей вероятностью присваивают его [метку класса](https://wiki.loginom.ru/articles/class-label.html) новым наблюдениям при практическом использовании модели. Данное явление известно как переоценка (overestimation).

## Ребалансировка классов может происходить путём увеличения числа примеров миноритарного класса (undersampling), либо путём сокращения числа примеров мажоритарного (oversampling).

## **Сокращение числа примеров мажоритарного класса**

**Случайное удаление (random undesampling)**.

Сначала определяется число *K* примеров доминирующего класса, которое требуется удалить, чтобы достичь требуемого соотношения классов в обучающей выборке. Затем случайным образом выбираются *K* наблюдений доминирующего класса и удаляются.

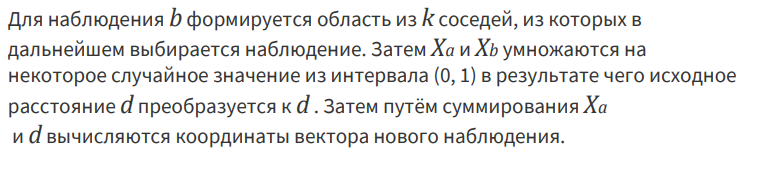
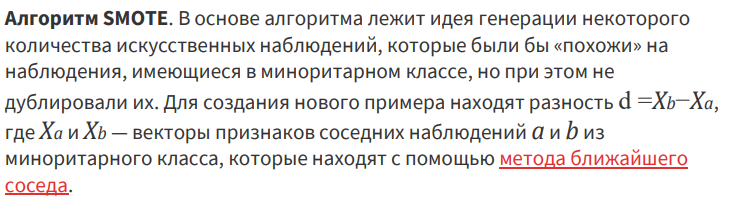


**Правило соcредточенного ближайшего соседа (Condensed Nearest Neighbor Rule)**.

Из исходного набора данных *L* извлекаются все примеры миноритарного класса и один мажоритарного (обозначим полученное подмножество как *S*). Затем производится классификация всех примеров из *L* по методу одного ближайшего соседа (1-NN), когда каждому, случайно выбранному наблюдению присваивается метка класса ближайшего соседа. При этом, если для наблюдения допущена ошибка классификации (найденный и фактический классы не совпадают), то оно добавляется в S.

## **Увеличение числа примеров миноритарного класса**

**Дублирование примеров миноритарного класса (Oversampling)**. Самый простой метод – это дублирование примеров миноритарного класса. В зависимости от того, какое соотношение классов необходимо получить в выборке, выбирается случайным образом соответствующее количество наблюдений для дублирования.

22. Понятие параметров и гиперпараметров модели. Обучение параметров и гиперпараметров. Поиск по сетке.

**Гиперпараметры модели** — параметры, значения которых задается до начала обучения модели и не изменяется в процессе обучения. У модели может не быть гиперпараметров.

**Параметры модели** — параметры, которые изменяются и оптимизируются в процессе обучения модели и итоговые значения этих параметров являются результатом обучения модели.

Наиболее важным гиперпараметром часто является скорость обучения, которая определяет размер шага, используемый при поиске следующего набора весов для оптимизации. Если скорость обучения слишком высока, крутейший спуск может быстро сойтись на плато или к неоптимальной точке. Если скорость обучения слишком низкая, спуск может остановиться и никогда полностью не сойтись.

Многие другие распространенные гиперпараметры зависят от используемых алгоритмов. Большинство алгоритмов имеют параметры остановки, такие как максимальное число эпох, максимальное время выполнения или минимальное улучшение от эпохи к эпохе. Определенные алгоритмы имеют гиперпараметры, которые управляют формой их поиска. Например, классификатор случайного леса имеет гиперпараметры для минимальных выборок на лист, максимальной глубины, минимальных выборок при расщеплении, минимальной массовой доли для листа и проч.

Поиск по сетке - это, алгоритм оптимизации, который позволяет вам выбирать лучшие параметры для вашей задачи оптимизации из списка параметров, которые вы предоставляете, тем самым автоматизируя метод «проб и ошибок». Его используют в машинном обучении для получения параметров, при которых модель дает наилучшую точность.

23. Понятие недо- и переобучения. Определение, пути решения.

Регуляризация

Проблема переобучения

Регуляризация предназначена для решения проблемы переобучения.

Недообучение - это проблема выбора гипотезы, когда форма нашей функции h плохо отражает тренд данных. Обычно это вызвано слишком простой

функцией или использует слишком мало функций. например. если взять то мы делаем первоначальное предположение о том, что линейная модель хорошо подгоняет учебные данные и сможет обобщить, но это может быть не так.

С другой стороны, чрезмерная или высокая дисперсия или переобучение вызвано функцией гипотезы, которая подходит к имеющимся данным, но не позволяет хорошо обобщать предсказание новых данных. Обычно это вызвано сложной функцией, которая создает много ненужных кривых и углов, не связанных с данными.

Эта терминология применяется как к линейной, так и к логистической регрессии. Существует два основных варианта решения проблемы переобучения:

1) Уменьшить количество признаков:

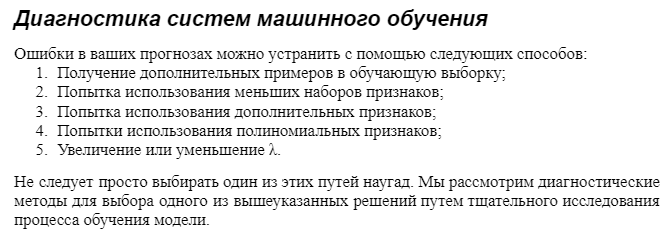
a) Вручную выберите, какие признаки сохранить.

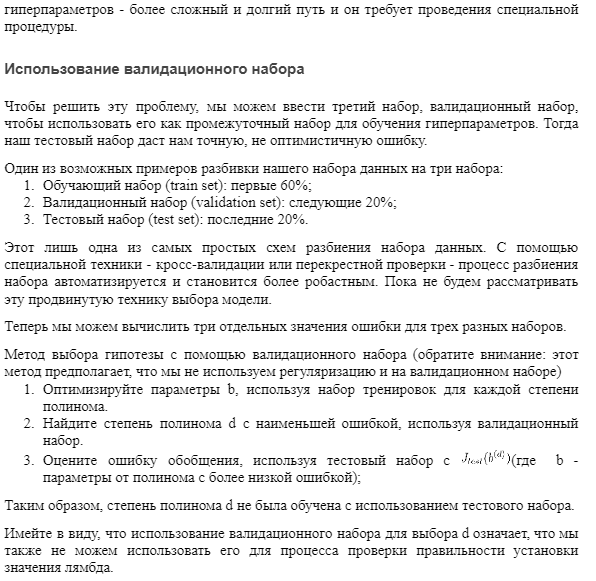
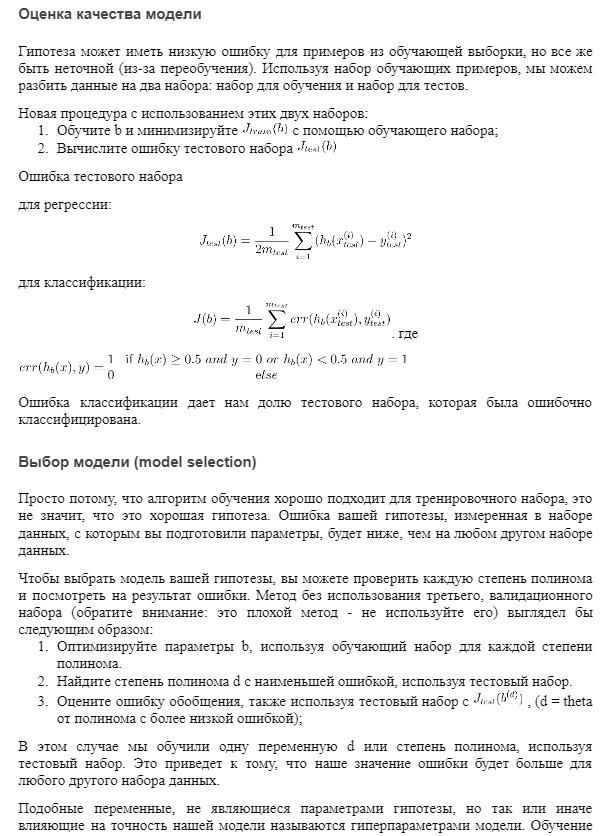
b) Использовать алгоритм выбора модели.

2) Регуляризация

Сохраните все признаки, но уменьшите параметры bj.

Регуляризация работает хорошо, когда у нас много полезных признаков.

24. Диагностика модели машинного обучения. Методы, цели.



# 25. Проблема выбора модели машинного обучения. Сравнение моделей.

# 26. Измерение эффективности работы моделей машинного обучения. Метрики эффективности.

вопросы 27-28

# 27. Метрики эффективности моделей классификации. Виды, характеристика, выбор.

# 28. Метрики эффективности моделей регрессии. Виды, характеристика, выбор.

# 29. Перекрестная проверка (кросс-валидация). Назначение, схема работы.

Перекрестная проверка — это процедура повторной выборки, используемая для оценки моделей машинного обучения на ограниченной выборке данных.

Процедура имеет один параметр, называемый k, который относится к числу групп, на которые должна быть разделена данная выборка данных. Таким образом, процедура часто называется перекрестной проверкой (кросс-валидацией) k-fold. При выборе определенного значения для k, оно может быть использовано вместо k в ссылке на модель, например, при k=10, становится 10-кратной перекрестной проверкой.

Перекрестная проверка в основном используется в прикладном машинном обучении для оценки квалификации модели машинного обучения на не используемых данных. То есть использовать ограниченную выборку (test sample) для оценки того, как модель будет работать в целом при использовании ее при прогнозирования на данных, не используемых во время обучения модели.

Общая процедура заключается в следующем:

1. Перемешайте датасет случайным образом

2. Разделите датасет на k-групп

3. Для каждой уникальной выборки:

* Возьмите группу в качестве тестирования датасета
* Возьмите остальные группы в качестве выборки учебных данных
* Приготовьте модель на обучаемых выборках и оцените ее на тестовой выборке
* Сохраняйте оценку модели и отбросьте модель

4. Обобщите параметры качества модели с помощью выборки оценки моделей

# 30. Методы векторизации текстов для задач машинного обучения.

# [**https://medium.com/@bigdataschool/4-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B0-%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8-%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2-f8ac90e4175a**](https://medium.com/@bigdataschool/4-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B0-%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8-%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2-f8ac90e4175a)

Обработка естественного языка или [NLP](https://python-school.ru/wiki/nlp/) (Natural Language Processing) занимается применением алгоритмов Machine Learning для текстовых данных. Как правило, модели машинного обучения работают с числами. В этой статье поговорим о 4-х наиболее применяемых методах для перевода текстов в числовые тензоры.

**Основные термины NLP: корпус, документ, токен, словарь**

Сначала текст разбивается на текстовые единицы (токены), например, символы, слова, словосочетания, предложения, абзацы и т.д. Чаще всего разбивают на слова. Токены образуют словарь, который может быть отсортирован по алфавиту.

Также в [NLP](https://python-school.ru/wiki/nlp/) применяются термины «документ» и «корпус». Документ — это совокупность токенов, которые принадлежат одной смысловой единице. В качестве документа может выступать предложение, комментарий или пост пользователя. Корпус — это генеральная совокупность всех документов.

**1. Прямое кодирование**

Прямое кодирование (one-hot encoding) считается самым простым способом преобразования токенов в тензоры и выполняется следующим образом:

каждый токен представляет бинарный вектор (значения 0 или 1); единица ставится тому элементу, который соответствует номеру токена в словаре.

Проблемой прямого кодирования является размерность. Каждое предложение состоит всего из 4 слов, но в итоге получилась большая матрица для каждого документа. Количество строк регулируется словарем, поэтому чем больше слов в словаре, тем больше будет матрица.

## **2. Bag of words**

В отличие от прямого кодирования, мешок слов (Bag of words) выделяет вектору весь документ, и каждый элемент кодируется 1 по порядку следования слов в словаре.

Bag of words решает проблему размерности по одной оси. Количество строк определяется количеством документов. Однако, этот метод не учитывает важность того или иного токена, ведь одно слово может повторятся по несколько раз. В этом случае пригодится альтернативный способ, рассмотренный далее.

## **3. TF-IDF**

TF-IDF состоит из двух компонентов: Term Frequency (частотность слова в документе) и Inverse Document Frequency (инверсия частоты документа). Они считаются следующим образом:

**TF** — это частотность термина

*TF термина* ***а*** *= (Количество раз, когда термин* ***а*** *встретился в тексте / количество всех слов в тексте)*

***IDF*** *— это обратная частотность документов. Она измеряет непосредственно важность термина.*

*IDF термина* ***а*** *= логарифм(Общее количество документов / Количество документов, в которых встречается термин* ***а****)*

*TF-IDF термина* ***а*** *= (TF термина* ***а****) \* (IDF термина* ***а****)*

# 31. Представление графической информации в моделях машинного обучения.

# 32. Задачи без учителя. Кластеризация. Метод k средних.

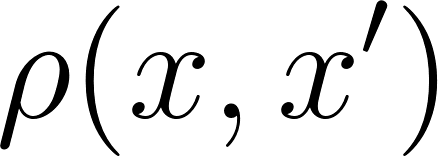
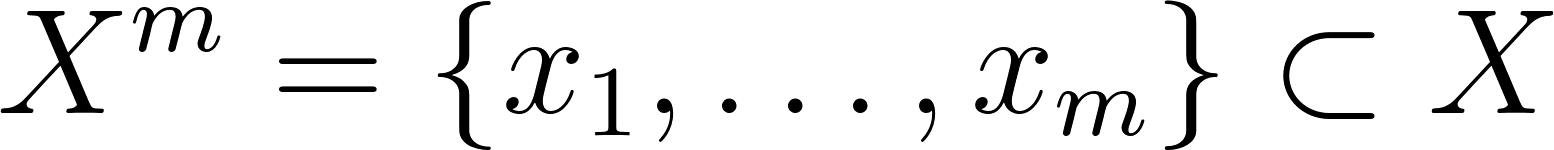
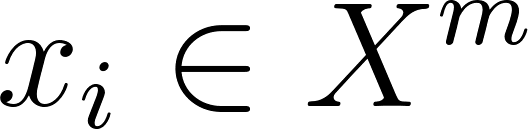
Обучение без учителя является противоположностью обучению с учителем, потому что оно использует немаркированный набор тренировок, а не размеченный. Другими словами, у нас нет вектора y ожидаемых результатов, у нас есть только набор данных, в которых мы можем найти структуру.

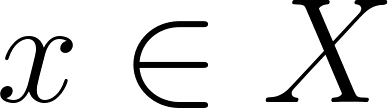
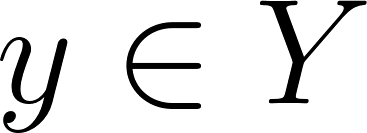
Кластеризация (или кластерный анализ) — это задача разбиения множества объектов на группы, называемые кластерами. Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных группы должны быть как можно более отличны. Главное отличие кластеризации от классификации состоит в том, что перечень групп четко не задан и определяется в процессе работы алгоритма.

Кластеризация хороша для:

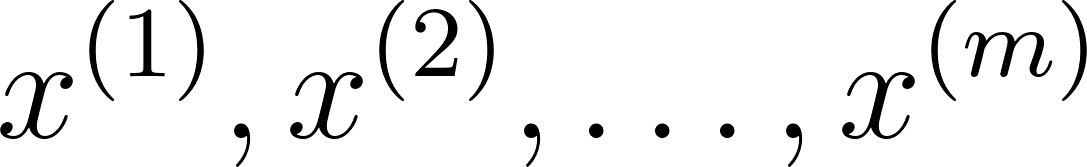
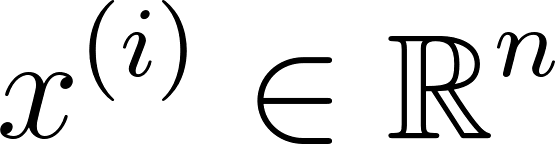
* Сегментация рынка
* Анализ социальной сети
* Организация компьютерных кластеров
* Астрономический анализ данных

### **Представление модели**

Пусть X — множество объектов, Y — множество номеров (имён, меток) кластеров. Задана функция расстояния между объектами [](about:blank). Имеется конечная обучающая выборка объектов [](about:blank). Требуется разбить выборку на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из объектов, близких по метрике [](about:blank), а объекты разных кластеров существенно отличались. При этом каждому объекту [](about:blank) приписывается номер кластера [](about:blank).

Алгоритм кластеризации — это функция [](about:blank), которая любому объекту [](about:blank) ставит в соответствие номер кластера [](about:blank). Множество Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.

Наши основные переменные:

* K (количество кластеров)
* Тренировочный набор [](about:blank)
* Где [](about:blank)

### **Алгоритм K средних**

Алгоритм k-средних (K-Means) является самым популярным и широко используемым алгоритмом для автоматической группировки данных в когерентные подмножества.

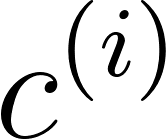
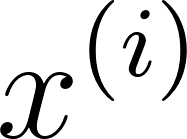
Основная идея алгоритма выражена в следующих шагах:

1. Случайно инициализируйте две точки в наборе данных, называемые центроидами кластера.
2. Назначение кластера: назначьте все примеры в одну из двух групп на основе того, какой кластер-центроид ближе всего к примеру.
3. Переместите центроид: вычислите средние значения для всех точек внутри каждой из двух групп кластеров кластеров, а затем переместите точки центроида кластера на эти средние.
4. Повторно запустите (2) и (3), пока мы не найдем наши кластеры.

Обратите внимание, что мы не будем использовать соглашение x0 = 1.

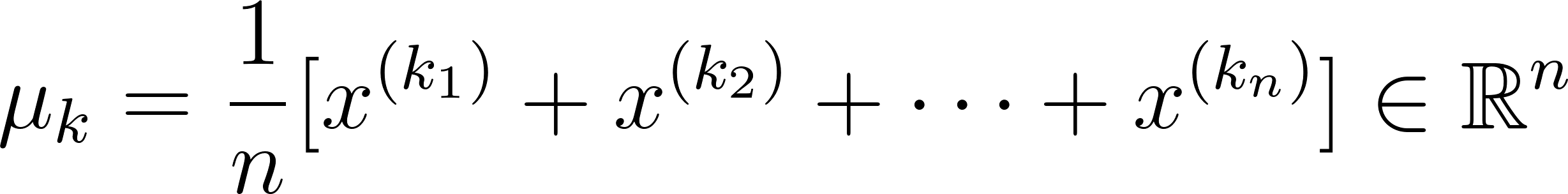
Первый цикл алгоритма - это назначение центроидов кластеров. Мы создаем вектор c, где c(i) представляет центроид, назначенный примеру x(i). Мы можем написать операцию шага назначения кластеров математически следующим образом:

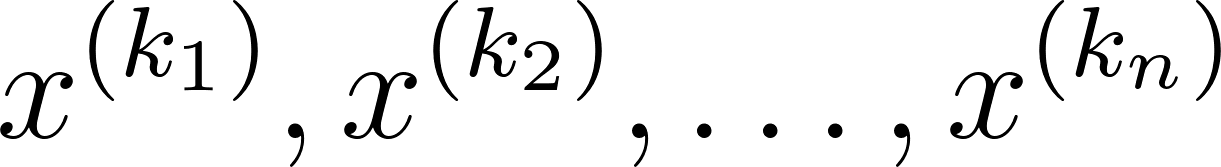
[](about:blank)

Каждый элемент [](about:blank) содержит индекс центроида, который имеет минимальное расстояние до [](about:blank).

По соглашению мы возводим правую сторону в квадрат, что делает функцию, которую мы пытаемся свести к минимуму, более быстро возрастающей. Это в основном просто удобное соглашение. Но соглашение, которое помогает уменьшить количество вычислений, потому что евклидово расстояние требует квадратного корня, но отменяется.

Второй цикл алгоритма - это перемещение центроидов, где мы перемещаем каждый центроид в среднем по своей группе. Более формально уравнение для этого цикла выглядит следующим образом:

[](about:blank)

Где каждый из [](about:blank) - примеры обучения, назначенные группе [](about:blank).

Если у вас есть центроид кластера, которому соответствуют 0 точек, вы можете случайно повторно инициализировать этот центроид для новой точки. Вы также можете просто исключить эту группу кластеров.

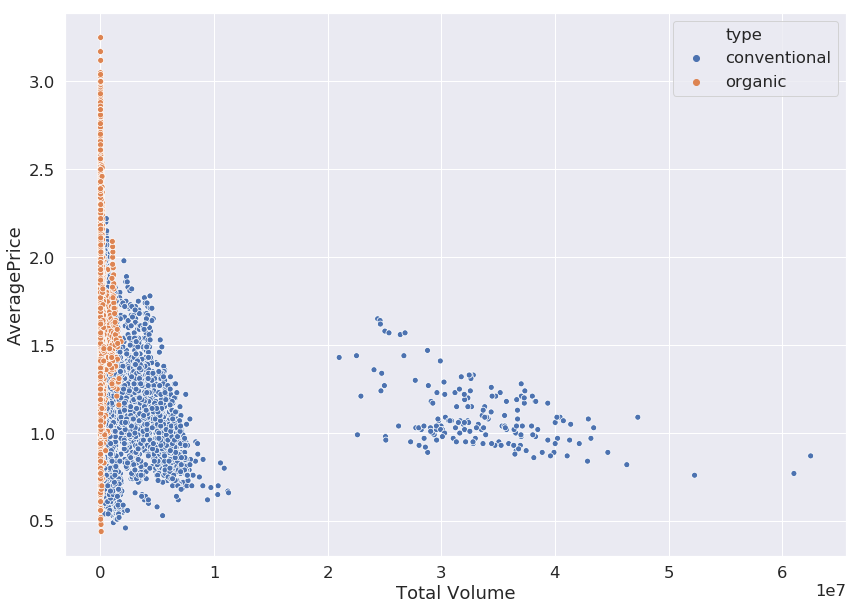
После ряда итераций алгоритм сходится, и новые итерации не влияют существенно на положение центроидов кластеров.

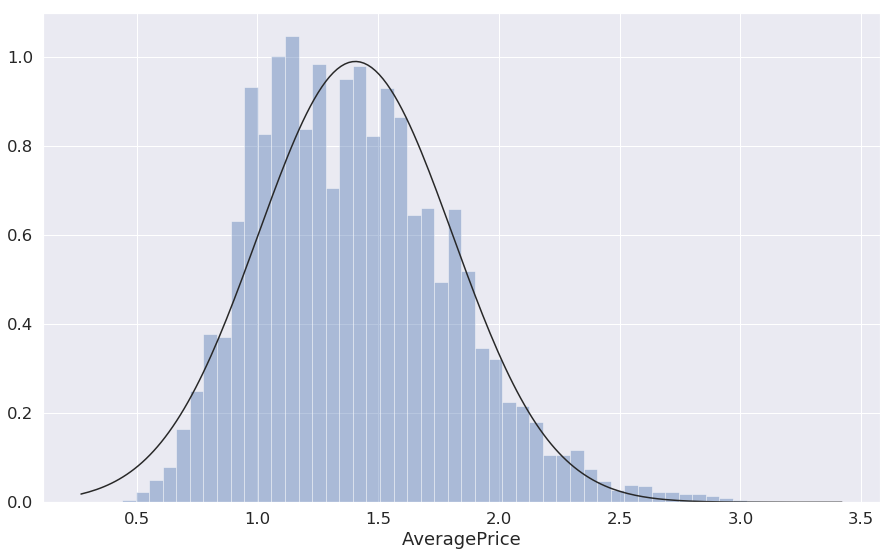
Некоторые наборы данных не имеют реального внутреннего разделения или естественной структуры. Алгоритмы типа К-средних могут равномерно сегментировать такие данные в подмножества, поэтому даже в этом случае могут быть полезны.

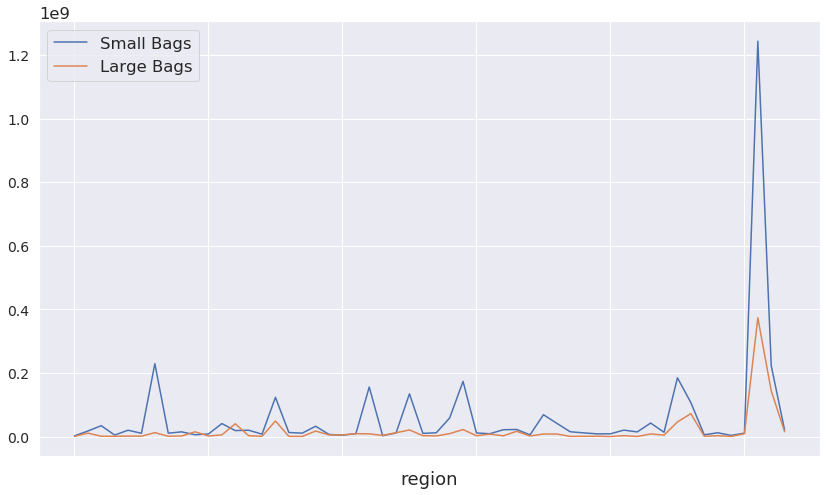
# 33. Задачи без учителя. Обнаружение аномалий.

Задача обнаружения аномалий сама по себе представляет собой область машинного обучения. Методы и инструменты данной сферы иногда применяются для предварительного анализа данных в целях идентификации выбросов, выбивающихся из общего ряда наблюдений значений конкретного признака. Несмотря на то, что задача обнаружения выбросов, несомненно, полезна для анализа данных, использование такого сложного и затратного инструмента редко может быть оправдано.

Основная задача:

Обнаружение аномальных объектов, потенциально ошибок в данных, потенциально нерелевантных членов выборки.



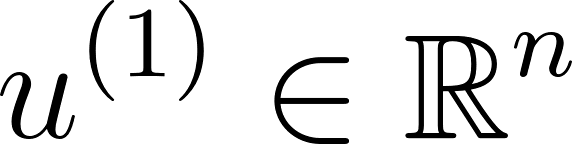


# 34. Задачи без учителя. Понижение размерности. Метод PCA.

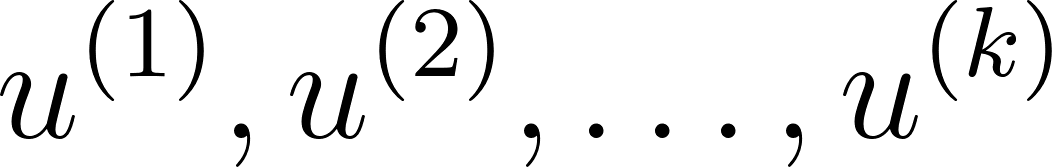
Наиболее популярным алгоритмом сокращения размерности является метод главных компонент (principal component analysis, PCA). В этом методе переменные преобразуются в новый набор переменных, которые являются линейной комбинацией исходных переменных. Эти новые переменные известны как основные или главные компоненты. Они получаются таким образом, что первый компонент учитывает большую часть возможного изменения исходных данных, после чего каждый последующий компонент имеет максимально возможную дисперсию.

Второй главный компонент должен быть ортогонален первому главному компоненту. Другими словами, он делает все возможное, чтобы зафиксировать дисперсию данных, которые не были захвачены первым основным компонентом. Для двумерного набора данных могут быть только два главных компонента.

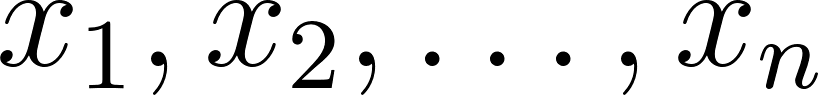
Учитывая две функции: x1 и x2, мы хотим найти одну такую переменную, которая эффективно описывает обе функции одновременно. Затем мы сопоставляем наши старые функции с этой новой строкой, чтобы получить новую отдельную функцию. То же самое можно сделать с тремя функциями, где мы сопоставляем их с плоскостью.

Цель PCA - уменьшить среднее значение всех расстояний каждой функции до линии проецирования. Это называется ошибка проецирования. Уменьшите от 2d до 1d: найдите направление (вектор [](about:blank), на который нужно проецировать данные, чтобы минимизировать ошибку проецирования.

Более общий случай таков:

Понижение от n-мерности до k-размерности: найдите k векторов [](about:blank) на которые нужно проецировать данные, чтобы свести к минимуму ошибка проектирования.

Если мы перейдем от 3d к 2d, мы будем проектировать наши данные на два направления (плоскость), поэтому k будет равно 2.

PCA не является линейной регрессией. В линейной регрессии мы минимизируем квадрат ошибки от каждой точки до нашей линии предиктора. Это вертикальные расстояния. В PCA мы минимизируем кратчайшее расстояние или кратчайшие ортогональные расстояния до наших точек данных. В более общем плане, в линейной регрессии, мы берем все наши примеры в x и применяем параметры в Θ для предсказания y. В PCA мы берем ряд функций [](about:blank) и находим среди них наиболее близкий общий набор данных. Мы не пытаемся предсказать какой-либо результат, и мы не применяем к ним тета-веса.

# 35. Конвейеры в библиотеке sklearn. Назначение, использование.

Конвейер — объединение трансформаторов и модели для последовательной обработки данных и предсказания на обработанных данных.

Трансформатор в sklearn — класс, в котором определены методы transform и fit\_transform.

Модель в sklearn — класс, в котором определен метод predict.

Конвейер в sklearn представлен классом Pipeline из под-модуля sklearn.pipeline.

Его создание будет продемонстрировано непосредственно на практике, когда мы перейдём к нашей задаче.

Задача — создать модель, которая сможет с высокой точностью предсказывать, является ли определённая звезда пульсаром, основываясь на некоторые показатели.

Использование:

1. Загрузка библиотек
2. Загрузка данных, используя pandas
3. Отделение исходных признаков от цели предсказания
4. Также необходимо разделить данные на тренировочный и тестовый набор. Для этого используется стандартный для этой задачи инструмент — функцию train\_test\_split из под-модуля sklearn.model\_selection.
5. Стандартизация. Алгоритм стандартизации реализован в классе StandardScaler. Так же создадим экземпляр класса классификатора.
6. Далее создаём массив последовательности «шагов» конвейера. Сначала необходимо стандартизировать данные, а уже затем выполнить предсказание.
7. Теперь пора выбирать гиперпараметры. Я осуществил поиск трёх параметров — максимального кол-ва итераций, параметра alpha для регуляризации и кол-ва скрытых слоёв сети.
8. Теперь используется метод fit(), чтобы адаптировать нашу «модель».
9. Сеточный поиск: С помощью метода score() проводится оценка точности модели на тестовых данных, в атрибуте best\_params\_ хранится словарь с лучшей комбинацией гиперпараметров.

# 36. Использование методов визуализации данных для предварительного анализа.

Аналитику требуется визуально оценить характер, тип и поведение данных, динамический диапазон значений, степень гладкости и наличие факторов, снижающих качество данных, таких как шумы, аномальные и пропущенные значения.

Визуальный анализ позволяет увидеть, соответствуют ли данные ожидаемым, оценить степень пригодности данных к анализу, выдвинуть гипотезы о закономерностях процессов, описываемых данными и определить, какие виды очистки и предобработки необходимо применить к данным

Кроме того, визуализация источников данных позволяет определить метод загрузки данных в аналитический контур и параметры, которые при этом должны быть использованы. Например, для корректной загрузки данных из текстового файла с разделителями необходимо правильно определить символразделитель, используемый формат даты и времени, расположение заголовков столбцов и так далее

.

Можно выделить следующие группы методов визуализации:

* общего назначения – применяются для решения типовых задач анализа данных визуальной оценки качества и характера данных, распределения значений признаков, статистических характеристик и т. д. В них можно выделить дез подвида – простые и сложные. К последним, в частности, относится OLAP-анализ – комплекс методов для визуализации многомерных данных;
* оценка качества моделей – позволяет оценивать различные характеристики моделей, такие как точность, эффективность, достоверность результатов, интерпретируемость, устойчивость и так далее;
* интерпретация результатов анализа – служаг для представления конечных результатов анализа в виде, наиболее удобном с точки зрения их интерпретации пользователем

# 37. Исследование коррелированности признаков: методы, цели, выводы.

Целью корреляционного анализа является выявление оценки силы связи между случайными величинами (признаками), которые характеризует некоторый реальный процесс.

Задачи корреляционного анализа:

а) Измерение степени связности (тесноты, силы, строгости, интенсивности) двух и более явлений.

б) Отбор факторов, оказывающих наиболее существенное влияние на результативный признак, на основании измерения степени связности между явлениями. Существенные в данном аспекте факторы используют далее в регрессионном анализе.

в) Обнаружение неизвестных причинных связей.

Формы проявления взаимосвязей весьма разнообразны. В качестве самых общих их видов выделяют функциональную (полную) и корреляционную (неполную) связи.

Корреляционная связь проявляется в среднем, для массовых наблюдений, когда заданным значениям зависимой переменной соответствует некоторый ряд вероятностных значений независимой переменной. Связь называется корреляционной, если каждому значению факторного признака соответствует вполне определенное неслучайное значение результативного признака.

Наглядным изображением корреляционной таблицы служит корреляционное поле. Оно представляет собой график, где на оси абсцисс откладываются значения X, по оси ординат – Y, а точками показываются сочетания X и Y. По расположению точек можно судить о наличии связи.

Показатели тесноты связи дают возможность охарактеризовать зависимость вариации результативного признака от вариации признака-фактора.

Более совершенным показателем степени тесноты корреляционной связи является линейный коэффициент корреляции. При расчете этого показателя учитываются не только отклонения индивидуальных значений признака от средней, но и сама величина этих отклонений.

Ключевыми вопросами данной темы являются уравнения регрессионной связи между результативным признаком и объясняющей переменной, метод наименьших квадратов для оценки параметров регрессионной модели, анализ качества полученного уравнения регрессии, построение доверительных интервалов прогноза значений результативного признака по уравнению регрессии.

# 38. Решкалирование данных. Виды, назначение, применение. Нормализация и стандартизация данных.

Нормализация (normalization) и стандартизация (standardization) являются методами изменения диапазонов значений — шкалирования. Шкалирование особенно полезно в машинном обучении (Machine Learning), поскольку разные атрибуты могут измеряться в разных диапазонах, или значения одного атрибута варьируются слишком сильно. Например, один атрибут имеет диапазон от 0 до 1, а второй — от 1 до 1000. Для задачи регрессии второй атрибут оказывал бы большое влияние на обучение, хотя не факт, что он является более важным, чем первый. Нормализация и стандартизация отличаются своими подходами:

Нормализация подразумевает изменение диапазонов в данных без изменения формы распределения,

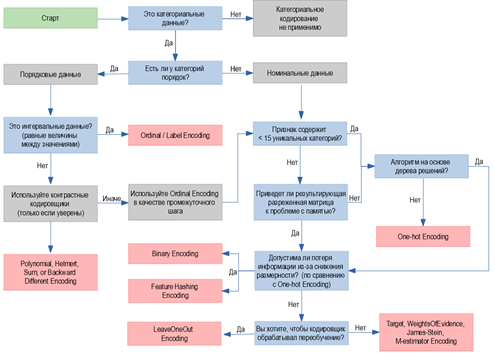
Стандартизация изменяет форму распределения данных (приводится к нормальному распределению).

Обычно достаточно нормализовать данные. Например, в глубоком обучении (Deep Learning) требуется перевести цвета изображений RGB из диапазона 0–255 к диапазону 0–1. А вот стандартизацию стоит применять при использование алгоритмов, которые основываются на измерении расстояний, например, k ближайших соседей или метод опорных векторов (SVM).

# 39. Преобразование категориальных признаков в числовые.

Существует несколько способов преобразовать категории в числа, каждый из них имеет свои плюсы и минусы. Выбор метода зависит от типа и смысла ваших данных, мощности множества категорий, алгоритма машинного обучения.

Ниже приведена схема, как выбрать подходящий метод кодирования.



Рассмотрим наиболее популярные методы преобразования категорий в числа.

Самый простой способ – обычная нумерация значений (0, 1, 2, …). У данного подхода есть существенный недостаток. Обычно он ведет к плохому результату так как, алгоритмы начинают учитывать бессмысленную упорядоченность значений признаков. Однако данный метод имеет преимущество с точки зрения памяти.

Метод реализован в классе sklearn.preprocessing.LabelEncoder.

Следующий способ – dummy-кодирование, также называемое — one-hot. Суть заключается в создании дополнительных N признаков (столбцов), где N – количество уникальных категорий. Новые признаки принимают значения 0 или 1 в зависимости от принадлежности к категории. One-hot encoder значительно увеличивает объем данных, что делает его неэффективным с точки зрения памяти, частично эту проблему решает применение разреженных матриц. Метод реализован в классе sklearn.preprocessing.OneHotEncoder.

Далее рассмотрим метод среднего кодирования (Mean/Target Encoding). Метод предполагает кодирование категорий средним арифметическим от суммы целевых меток (Target). Вариация этого метода рассматривалась нами ранее в задаче предсказания спроса, вместо целевых меток, мы брали другой признак — цену на товар, таким образом категории товаров упорядочивались их дороговизной. Однако, при кодировании средним по Target велика вероятность переобучения модели, эту проблему поможет решить регуляризация.

методов кодирования существует гораздо больше, и среди них нет универсальных. Выбирая метод, стоит отталкиваться от ваших данных, мощности множества категорий и алгоритма машинного обучения.

# 40. Ансамблевые модели машинного обучения. Виды ансамблирования.

Ансамблевые методы - это мощный инструмент для построения моделей машинного обучения. Команды, которые используют их в соревнованиях на kaggle, занимают победные места. Ансамбли позволяют увеличить точность модели до 90+, при этом они довольно просты в понимании.

Что такое ансамбль?

Метод машинного обучения, где несколько моделей обучаются для решения одной и той же проблемы и объединяются для получения лучших результатов называется ансамблевым методом. Основная предпосылка заключается в том, что результат работы нескольких моделей будет более точен, чем результат только одной модели.

Когда говорится об ансамблях, то вводится понятие слабого ученика(обычные модели вроде линейной регрессии или дерева решений). Множество слабых учеников являются строительными блоками для более сложных моделей. Объединение слабых учеников для улучшения качества модели, уменьшения смещения или разброса, называется сильным учеником.

Наиболее популярными ансамблевыми методами являются: стекинг, бэггинг, бустинг.

* Стекинг. Используется несколько разнородных слабых учеников. Их обучают и объединяют для построения прогноза, основанного на результатах различных слабых моделей.

Работа этого типа ансамблей довольно проста. На вход всех слабых прогнозаторов подаётся обучающий набор, каждый прогноз идёт к финальной модели, которая называется смеситель, мета-ученик или мета-модель, после чего та вырабатывает финальный прогноз.\

* Бэггинг. В этом случае однородные модели обучают на разных наборах данных и объединяют. Получают прогноз путём усреднения. Если использовать в качестве слабого ученика деревья решений, то получится случайный лес RandomForestClassifier / RandomForestRegressor.

Основная идея бэггинга заключается в том, чтобы обучить несколько одинаковых моделей на разных образцах. Распределение выборки неизвестно, поэтому модели получатся разными.

Для начала генерируется несколько бутстрэп-выборок. Бутстрэп - это случайный выбор данных из датасета и представление их в модель, затем данные возвращаются в датасет и процесс повторяется. После модели делают свои прогнозы на основе бутстрэп-выборок. В случае регрессии прогнозы просто усредняются. В случае же классификации применяется голосование.

* Бустинг. При использовании данного метода несколько однородных моделей последовательно обучаются, исправляя ошибки друг друга.

Метод бустинга в чём то схож с методом бэггинга: берётся множество одинаковых моделей и объединяется, чтобы получить сильного ученика. Но разница заключается в том, что модели приспосабливаются к данным последовательно, то есть каждая модель будет исправлять ошибки предыдущей.

Базовые модели для бустинга - это модели с низким разбросом и высоким смещением. Например неглубокие деревья решений. Одна из причин такого выбора моделей - они требуют меньше вычислительных затрат. Ещё бустинг (в отличии от бэггинга) нельзя распараллелить.

Помимо библиотеки scikit-learn в python есть библиотека XGBoost, которая предоставляет более обширный набор ансамблевых моделей с более точной реализацией.