# 

[**Теория**](#_heading=h.b3rid15lwgz0) **3**

[1. Ансамблевые модели машинного обучения. Виды ансамблирования.](#_heading=h.tle9jjyqgarm) 3

[2. Диагностика модели машинного обучения. Методы, цели.](#_heading=h.xlcf3bkssgn7) 5

[3. Задача классификации: постановка, математическая формализация.](#_heading=h.7vgb8wb4s0qo) 7

[4. Задача регрессии: постановка, математическая формализация.](#_heading=h.3goramkoszz0) 8

[5. Задачи без учителя. Кластеризация. Метод k средних.](#_heading=h.rk7gqmfuetmd) 9

[6. Задачи без учителя. Обнаружение аномалий.](#_heading=h.si528mdgq8lg) 11

[7. Задачи без учителя. Понижение размерности. Метод PCA.](#_heading=h.2dh1qdopzgdd) 13

[8. Измерение эффективности работы моделей машинного обучения. Метрики эффективности.](#_heading=h.dd0o1p1bn384) 14

[9. Использование методов визуализации данных для предварительного анализа.](#_heading=h.7ri5tjyjosih) 15

[10. Исследование коррелированности признаков: методы, цели, выводы.](#_heading=h.1u5ecrid3few) 16

[11. Классификация задач машинного обучения. Примеры задач из различных классов.](#_heading=h.vgaiz99wbex1) 18

[12. Конвейеры в библиотеке sklearn. Назначение, использование.](#_heading=h.quncuorfkvpr) 20

[13. Логистическая регрессия в задачах классификации.](#_heading=h.u1pcnkivx8hw) 22

[14. Метод k ближайших соседей в задачах классификации.](#_heading=h.8p2dfgps2m6o) 23

[15. Метод градиентного спуска для задач классификации.](#_heading=h.vcx2ntxyinzn) 25

[16. Метод градиентного спуска для парной линейной регрессии.](#_heading=h.mfda7cocdxj0) 27

[17. Метод опорных векторов в задачах классификации.](#_heading=h.fkv875mlp5n) 29

[18. Метод решающих деревьев в задачах классификации.](#_heading=h.iz20nk8ys7tq) 31

[19. Методы векторизации текстов для задач машинного обучения.](#_heading=h.165xr3b61bol) 33

[20. Метрики эффективности моделей классификации. Виды, характеристика, выбор.](#_heading=h.2du06y121ace) 34

[21. Метрики эффективности моделей регрессии. Виды, характеристика, выбор.](#_heading=h.61d4s9etzqoz) 37

[22. Множественная и нелинейная регрессии.](#_heading=h.wnmwu52ghvh1) 39

[23. Однослойный перцептрон в задачах классификации.](#_heading=h.jd6srbe4ewtu) 41

[24. Основные этапы проекта по машинному обучению.](#_heading=h.hc80pgzg4y3c) 43

[25. Перекрестная проверка (кросс-валидация). Назначение, схема работы.](#_heading=h.5ezow8akgyh7) 45

[26. Понятие машинного обучения. Отличие машинного обучения от других областей программирования.](#_heading=h.gst1p7qky0o1) 47

[27. Понятие набора данных (датасета) в машинном обучении. Требования, представление. Признаки и объекты.](#_heading=h.k0q6v2p5k1vj) 48

[28. Понятие недо- и переобучения. Определение, пути решения.](#_heading=h.kl0rv7sd74dg) 49

[29. Понятие параметров и гиперпараметров модели. Обучение параметров и гиперпараметров. Поиск по сетке.](#_heading=h.d0gx89saxfok) 52

[30. Понятие функции ошибки: требования, использование, примеры.](#_heading=h.mxwy7u8pfufk) 53

[31. Понятие чистых данных. Определение, очистка данных.](#_heading=h.kidcpjybmnct) 55

[32. Понятие ядра и виды ядер в методе опорных векторов.](#_heading=h.ab0wi2pxtrfs) 56

[33. Предварительный анализ данных: задачи, методы, цели.](#_heading=h.ajc9kuvss0o1) 57

[34. Представление графической информации в моделях машинного обучения.](#_heading=h.3szz88mvxa24) 59

[35. Преобразование категориальных признаков в числовые.](#_heading=h.urr0fbcjo3o) 60

[36. Проблема выбора модели машинного обучения. Сравнение моделей.](#_heading=h.rxtpfm1d743u) 61

[37. Проблема несбалансированных классов: исследование, пути решения.](#_heading=h.gepi1cn5uani) 63

[38. Проблема отсутствующих данных: причины, исследование, пути решения.](#_heading=h.2wzkjj3ooyns) 64

[39. Решкалирование данных. Виды, назначение, применение. Нормализация и стандартизация данных.](#_heading=h.iqmwexvh1rg) 65

[40. Шкалы измерения признаков. Виды шкал, их характеристика.](#_heading=h.lkw4y749jkaq) 66

[**Практика**](#_heading=h.kkyd5o1a3goo) **67**

[**Дополнительная информация**](#_heading=h.xhrfve5zmxl9) **69**

# 

# 

# Теория

## 1. 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.

**Виды ансамблевых методов**

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. При обучении мета-модели используется приём удерживаемого набора. Сначала набор разделяется на 2 части. Слабые ученики обучаются на первой половине обучающего набора, затем на второй. Затем создаётся новый обучающий набор на основе прогнозов, сделанных на прогнозах первой и второй части набора. Таким образом, на каждый образец из входного набора приходится столько прогнозов, сколько слабых учеников в ансамбле. Мета-модель учится прогнозировать значения на основе нового набора.

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.

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.

Существует два наиболее распространенных алгоритма бустинга - адаптивный бустинг и градиентный бустинг.

## 2. Диагностика модели машинного обучения. Методы, цели.

**Диагностика моделей машинного обучения** - это неточная наука, здесь нужно принимать в расчет и задачу, и выбор признаков и многие другие факторы.

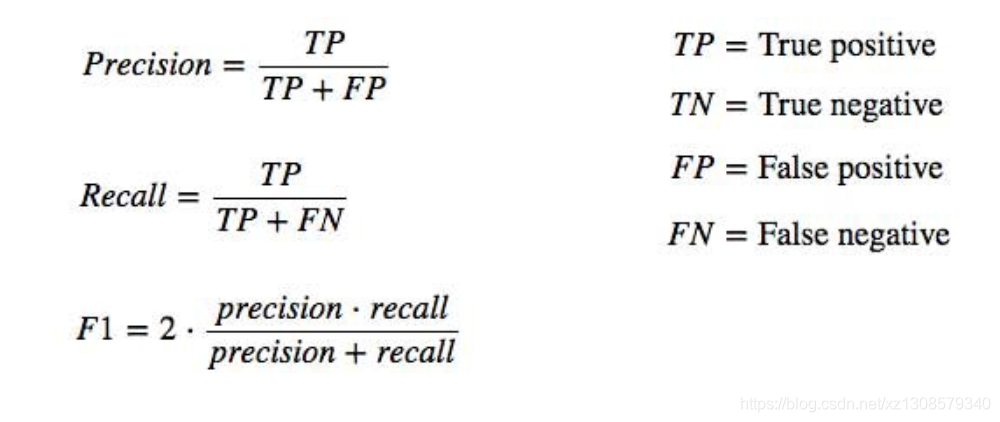
Обычно определить адекватность какой-либо модели возможно с помощью ряда метрик, которые зависят от характера решаемой задачи.

**При классификации** на два класса (по сути да/нет) у нас есть 4 различных исхода:

* True Positive (TP) — истинное значение было "да" и мы предсказали "да"
* True Negatives (TN) — истинное значение было "нет" и мы предсказали "нет"
* False Positive (FP) — истинное значение было "нет", а мы предсказали "да". Ложное срабатывание, ошибка I рода.
* False Negatives (FN) — истинное значение было "да", а мы предсказали "нет". Пропуск цели, ошибка II рода. (ред.) ща

Также для классификации сущенствуют следующие метрики:

* Accuracy (аккуратность) - процент верных предсказаний
* Precision (точность) - сколько верных среди предсказанных как "Класс 1/да"
* Recall (полнота) - сколько из настоящих "Класс 1/да" мы определили верно (то же самое, что True Positive Rate, Sensitivity)
* F-мера (гармоническое среднее P и R)
* Log-loss (учитывает значение вероятности) (ред.)



**При регрессии** у нас работают другие метрики:  
Общая идея: насколько хорошо вписывается в данные линия регрессии. Например, как далеко вокруг неё разбросаны все наблюдения:

* MAE/MAD (Mean Absolute Error, Mean Absolute Deviation) - средний модуль ошибки
* MSE/MSD (Mean Squared Error/Deviation) - среднеквадратическая ошибка
* RMSE/RMSD (Root Mean Squared Error) - лучше, чем MSE, потому что выражается в тех же единицах, что и измеряемая величина
* MAPE/MAPD (Mean Absolute Percentage Error) - ошибка в процентах от самой величины

**Более подробно тут:**

<https://koroteev.site/text/ml4/>

## 3. Задача классификации: постановка, математическая формализация.

**См. коротеев сайт если что**

Классификацией называется процедура, в которой объекты распределяются по группам (классам) в соответствии с численными значениями их переменных, характеризующими свойства этих объектов. Исходными данными для классификации является матрица **X**, в которой каждая строка представляет один объект, а каждый столбец – одну из переменных. Эта матрица называется исходным набором данных.

Классификацией называют не только саму процедуру распределения, но и ее результат. Употребляется также термин распознавание образов (pattern recognition) , который можно считать синонимом. В математической статистике классификацию часто называют дискриминацией.

Метод (алгоритм), которым проводят классификацию, называют классификатором. Классификатор переводит вектор признаков объекта x в целое число, 1, 2, … , соответствующее номеру класса, в который он помещает этот объект.

Если для всех объектов исходного набора известно, к какому классу они принадлежат, то такая постановка задачи называется классификацией с учителем (или с обучением). Обучение без учителя происходит тогда, когда принадлежность объектов в исходном наборе нам заранее не известна.

Первым делом следует очистить датасет от ненужных данных - построить корреляционную матрицу, проанализировать зависимости и удалить ненужные столбцы данных. Затем делим на тестовую и обучающую выборки и строим модели, например -

Модель логистической регрессии,

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.

Метод К-ближайший соседей и другие. Анализируем, какая модель показала себя лучше всего и делаем вывод.

Так же Задача классификации — это задача присвоения меток объектам. Например, если объекты — это фотографии, то метками может быть содержание фотографий: содержит ли изображение пешехода или нет, изображен ли мужчина или женщина, какой породы собака изображена на фотографии. Обычно есть набор взаимоисключающих меток и сборник объектов, для которых эти метки известны. Имея такую коллекцию данных необходимо автоматически расставлять метки на произвольных объектах того же типа, что были в изначальной коллекции.

## 4. Задача регрессии: постановка, математическая формализация.

**См. коротеев сайт если что**

Задача регрессии в машинном обучении — это предсказание одного параметра (Y) по известному параметру X, где X — набор параметров, характеризующий наблюдение.

Напомним, что в задачах регрессии мы принимаем входные переменные и пытаемся подогнать выход на непрерывную ожидаемую функцию результата. Линейная регрессия с одной переменной также известна как «парная линейная регрессия».

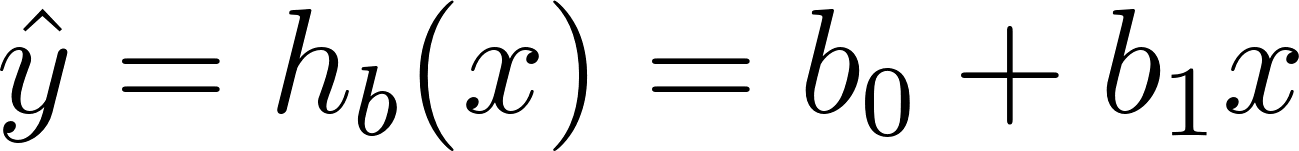
Одномерная линейная регрессия используется, когда вы хотите предсказать одно выходное значение y, зависящее от одного входного значения x. Мы проводим обучение с учителем, это означает, что мы уже имеем представление о том, какие значения выходной переменной соответствуют некоторым значениям входной переменной.

**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.

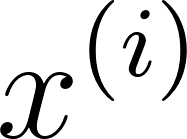
**Функция гипотезы**

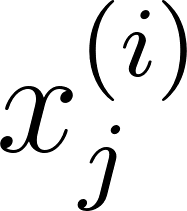
Наша прогностическая функция (функция гипотезы, модель) имеет общий вид:

[](about:blank)

Обратите внимание, что это похоже на уравнение прямой. в данном случае, мы пытаемся подобрать функцию h(x) таким образом, чтобы отобразить данные нам значения x в данные значения y.

Линейная регрессия с несколькими переменными также известна как «множественная линейная регрессия». Введем обозначения для уравнений, где мы можем иметь любое количество входных переменных:

[](about:blank) - вектор-столбец всех значений признаков i-го обучающего примера;

[](about:blank) - значение j-го признака i-го обучающего примера;

m - количество примеров в обучающей выборке;

n - количество признаков;

X - матрица признаков;

b - вектор параметров регрессии.

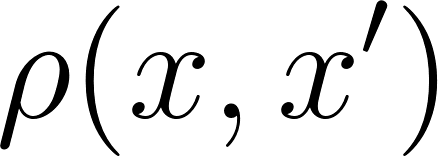
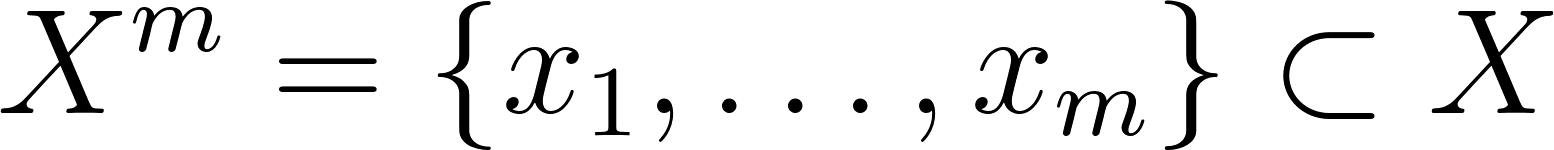
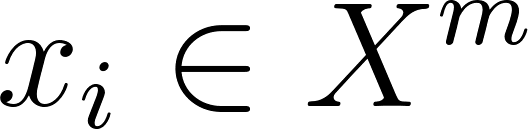
## 5. Задачи без учителя. Кластеризация. Метод k средних.

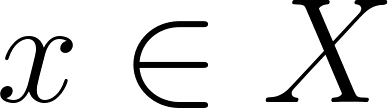
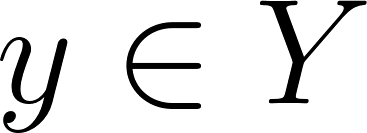
Обучение без учителя – это метод машинного обучения, при котором модель обучается на неразмеченных данных. Испытуемая система спонтанно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора

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.

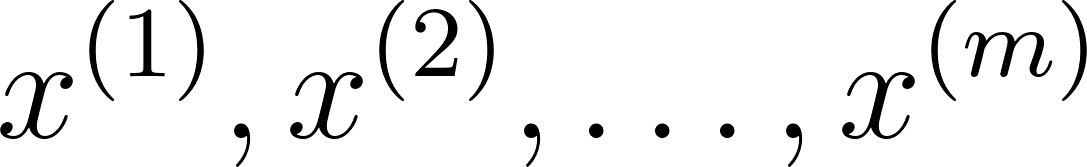
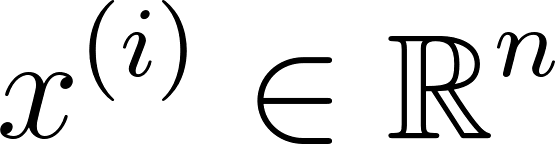
Кластеризация хороша для:

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

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

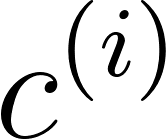
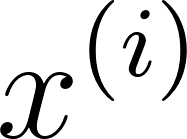
Основная идея алгоритма выражена в следующих шагах:

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.

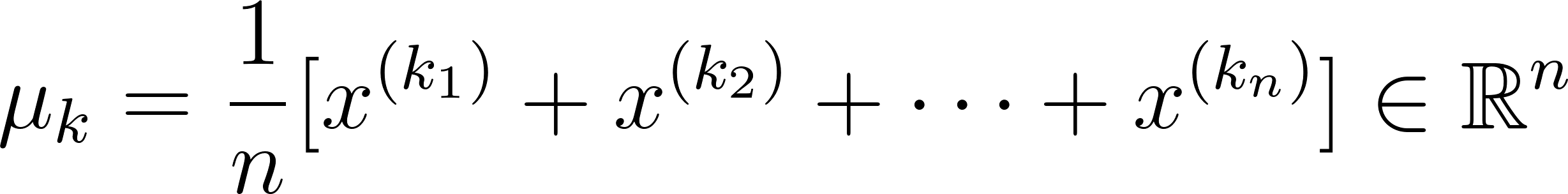
Первый цикл алгоритма - это назначение центроидов кластеров. Мы создаем вектор c, где c(i) представляет центроид, назначенный примеру x(i). Мы можем написать операцию шага назначения кластеров математически следующим образом:

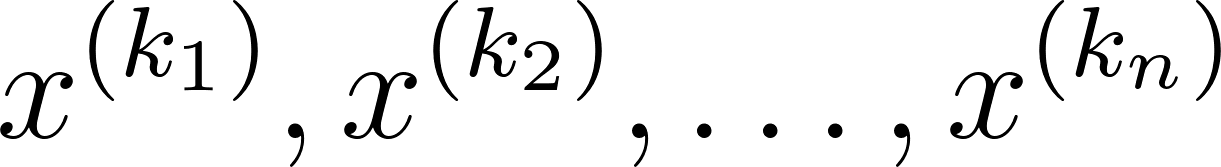
[](about:blank)

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.

## 6. Задачи без учителя. Обнаружение аномалий.

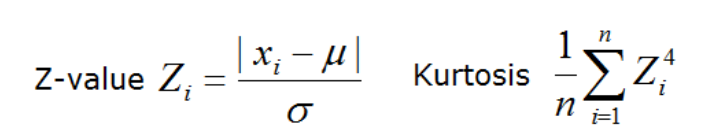
Выбросы являются следствием:

* ошибок в данных (неточности измерения, округления, неверной записи и т.п.)
* наличия шумовых объектов (неверно классифицированных объектов)
* присутствия объектов «других» выборок (например, показаниями сломавшегося датчика).

Методы обнаружения выбросов

**1. Статистические тесты**

Как правило, применяют для отдельных признаков и отлавливают экстремальные значения (Extreme-Value Analysis). Для этого используют, например, Z-value или Kurtosis measure.



Многие методы визуализации, например *ящик с усами*, имеют встроенные средства для детектирования и показа таких экстремальных значений.

**2. Модельные тесты**

Идея очень простая – мы строим модель, которая описывает данные. Точки которые сильно отклоняются от модели (на которых модель сильно ошибается) и есть аномалии (см. рис. 2). При выборе модели мы можем учесть природу задачи, функционал качества и т.п.

Применение модельного подхода. У нас есть матрица и требуется найти в ней выбросы. Мы используем *неполное сингулярное разложение (SVD)*, чтобы найти матрицу небольшого ранга максимально похожую на нашу (для наглядности все числа округлены). Элементы, которые сильно отличаются от соответствующих элементов матрицы небольшого ранга, будем считать выбросами.

**3. Итерационные методы**

Методы, которые состоят из итераций, на каждой из которых удаляется группа «особо подозрительных объектов». Например, в n-мерном признаковом пространстве можно удалять выпуклую оболочку наших точек-объектов, считая её представителей выбросами. Как правило, методы этой группы достаточно трудоёмки.

**4. Метрические методы**

Интуитивно понятно, что у выброса мало соседей, а у типичной точки много. Поэтому хорошей мерой аномальности может служить, например «расстояние до k-го соседа» (см. метод Local Outlier Factor). Здесь используются специфические метрики, например расстояние Махаланобиса.

**5. Методы машинного обучения**

Самые популярные алгоритмы здесь:

* Метод опорных векторов для одного класса (OneClassSVM)
* Изолирующий лес (IsolationForest)
* Эллипсоидальная аппроксимация данных (EllipticEnvelope)

**6. Ансамбли алгоритмов**

В методы решения задач обнаружения аномалий также проникла идея «один алгоритм хорошо, а сто лучше», поэтому часто строят много разных алгоритмов. Каждый из них даёт оценку аномальности и эти оценки потом «усредняют».

Поскольку ключевым моментов в реальных задачах обнаружения аномалий является выбор признаков, которые характеризуют те или иные отклонения от нормы, алгоритмы из ансамбля строят пытаясь угадать хорошие пространства. Здесь популярны:

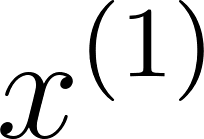
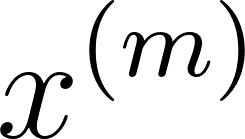
* Feature Bagging – для каждого алгоритма берут случайное признаковое подпространство,
* Rotated Bagging – в выбранном случайном признаковом подпространстве совершают случайный поворот.

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

Обучение без учителя – это метод машинного обучения, при котором модель обучается на неразмеченных данных. Испытуемая система спонтанно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора

Если данных для анализа слишком много, вычисления для машинного обучения могут занять годы. Мы можем существенно уменьшить количество имеющихся признаков, если у нас много избыточных данных. Для этого мы найдем две сильно коррелированные признаки, на их основе создадим новый признак, который точно описывает оба.

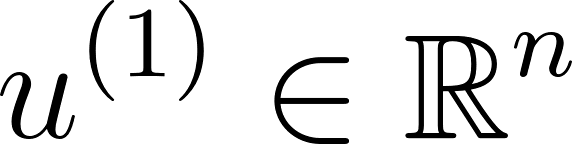
Выполнение уменьшения размерности уменьшит общие данные, которые мы должны хранить в памяти компьютера, и ускорит алгоритм обучения.

При уменьшении размерности мы уменьшаем количество признаков, а не наше количество примеров. Наша переменная m останется прежнего размера; n, число признаков, каждый из которых переносится из [](about:blank) в [](about:blank), будет уменьшено.

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.

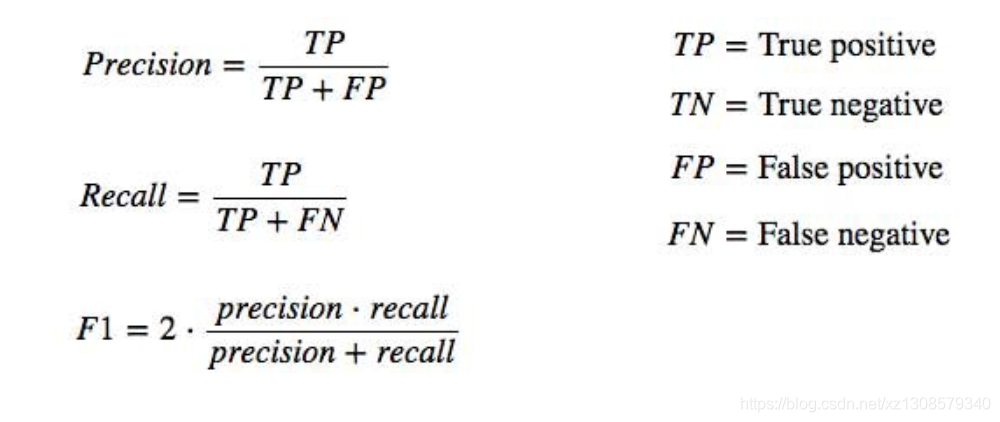
## 8. Измерение эффективности работы моделей машинного обучения. Метрики эффективности.

**При классификации** на два класса (по сути да/нет) у нас есть 4 различных исхода:

* True Positive (TP) — истинное значение было "да" и мы предсказали "да"
* True Negatives (TN) — истинное значение было "нет" и мы предсказали "нет"
* False Positive (FP) — истинное значение было "нет", а мы предсказали "да". Ложное срабатывание, ошибка I рода.
* False Negatives (FN) — истинное значение было "да", а мы предсказали "нет". Пропуск цели, ошибка II рода. (ред.)

Также есть следующие метрики:

* Accuracy (аккуратность) - процент верных предсказаний
* Precision (точность) - сколько верных среди предсказанных как "Класс 1/да"
* Recall (полнота) - сколько из настоящих "Класс 1/да" мы определили верно (то же самое, что True Positive Rate, Sensitivity)
* F-мера (гармоническое среднее P и R)
* Log-loss (учитывает значение вероятности) (ред.)



При регрессии у нас работают другие метрики:  
Общая идея: насколько хорошо вписывается в данные линия регрессии. Например, как далеко вокруг неё разбросаны все наблюдения:

* MAE/MAD (Mean Absolute Error, Mean Absolute Deviation) - средний модуль ошибки
* MSE/MSD (Mean Squared Error/Deviation) - среднеквадратическая ошибка
* RMSE/RMSD (Root Mean Squared Error) - лучше, чем MSE, потому что выражается в тех же единицах, что и измеряемая величина
* MAPE/MAPD (Mean Absolute Percentage Error) - ошибка в процентах от самой величины

## 9. 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 символ разделитель, 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 два 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 – служат для представления конечных результатов анализа в виде, наиболее удобном с точки зрения их интерпретации пользователем

## 10. 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:**

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

## 

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

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

* 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.
* 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.
* Задача кластеризации – распределение данных на группы: разделение всех клиентов мобильного оператора по уровню платежеспособности, отнесение космических объектов к той или иной категории (планета, звезда, чёрная дыра и т. п.).
* Задача уменьшения размерности – сведение большого числа признаков к меньшему (обычно 2–3) для удобства их последующей визуализации (например, сжатие данных).
* 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.

**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 ее 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: 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 трем группам, для каждой из которых предстоит сшить 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.

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

Рабочий процесс машинного обучения выполняется по принципу конвейера, т.е. выходные данные первых шагов становятся входными данными второго шага. **Scikit-learn** предоставляет функцию обработки таких каналов в sklearn.pipeline под названием **Pipeline**.

**Pipeline** можно использовать для объединения нескольких оценщиков в одну. Это полезно, поскольку часто существует фиксированная последовательность шагов при обработке данных, например, выбор функций, нормализация и классификация. Вам нужно только один раз вызвать fit и predict свои данные, чтобы они соответствовали всей последовательности оценщиков.

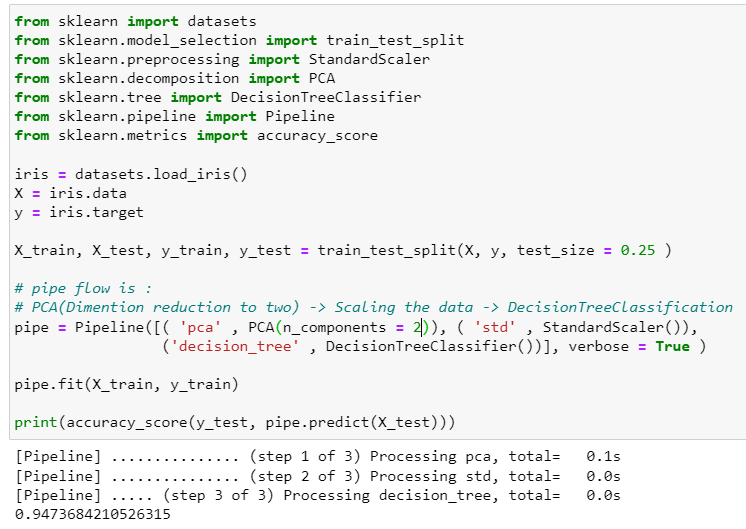
Конвейеры помогают избежать утечки статистики из ваших тестовых данных в обученную модель при перекрестной проверке, гарантируя, что одни и те же образцы используются для обучения преобразователей и предикторов.

Все оценщики в конвейере, кроме последнего, должны быть преобразователями (должны иметь метод transform ). Последний оценщик может быть любого типа (преобразователь, классификатор и т. д.).

ПРИМЕР:

model\_pipeline = Pipeline(steps=[ ("dimension\_reduction", PCA(n\_components=10)), ("classifiers", RandomForestClassifier())])  
model\_pipeline.fit(train\_data.values, train\_labels.values)  
predictions = model\_pipeline.predict(predict\_data.values)

ЕЩЕ ПРИМЕР:



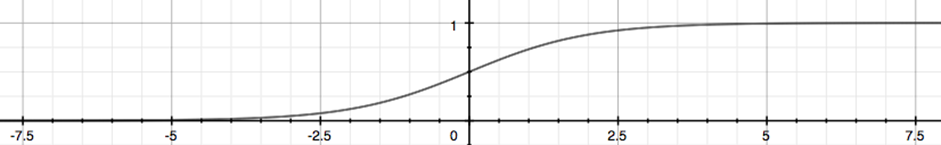
## 13. Логистическая регрессия в задачах классификации.

Логистическая регрессия – представляет из себя статистический инструмент, смысл, которого заключается в вычислении вероятности того, что исходное значение принадлежит определенному классу.

Цель использования логистической регрессии – это классификация набора данных. Функция на входе принимает признаки набора данных и высчитывает их взаимосвязь с зависимой переменной. Представляет из себя “сигмоидную функцию” с интервалом от 0 до 1 и является моделью обучения с учителем. Для бинарной классификации вероятность меньше 0,5 будет предсказывать 0, а вероятность больше 0 будет предсказывать 1. В результате работы функции получаем вероятности принадлежности к группе.

**Типы логистической регрессии:**

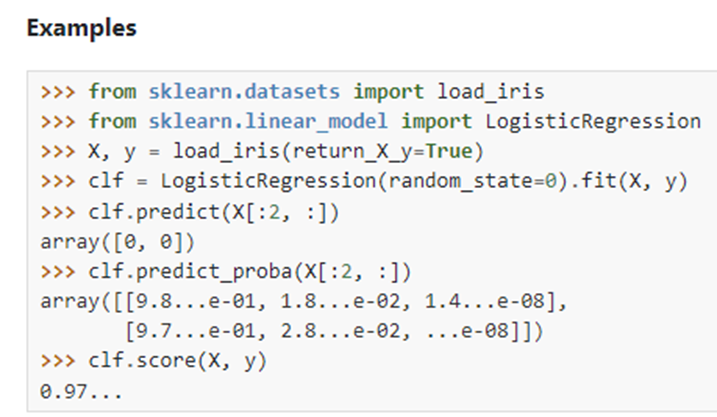
* Бинарная логистическая регрессия называется так, потому что есть только два возможных результата 0 или 1. Например, является ли электронное письмо спамом или нет.
* Порядковая логистическая регрессия применяется в случае, когда есть определенный порядок, это могут быть шкалы оценок от 1 до 10.
* Полиномиальная логистическая регрессия если нет строго определенного порядка, но есть больше трех переменных.



**Примеры применения логистической регрессии:**

1. Прогнозирование заболеваниями
2. Стоит ли банку выдавать кредит заемщику

Scikit-learn:



## 14. Метод k ближайших соседей в задачах классификации.

Метод k-ближайших соседей (k-nearest neighbors/KNN) – алгоритм машинного обучения с учителем, который используется в решении задач классификации. Основной смысл в том, что алгоритм классифицирует объект и относит его к классу, которому принадлежит большинство из его соседей, т. е. k ближайших объектов обучающей выборки.

**Как работает KNN:**

Данный алгоритм требователен к нормализации данных, когда значения расстояния могут сильно зависеть от атрибутов с большими диапазонами, применяют minmax-нормализация или z-нормализация.

**MinMax:**



Z:



Основой алгоритма является евклидово расстояние, которое рассчитывает расстояние между двумя точками по теореме Пифагора (вычисляется наименьшее расстояние между двумя точками)

Очень важным является выбор значения параметра k для получения корректных результатов, если выбрать маленькое k, то будет возникать эффект переобучения. Если выбрать k=1, то метод ближайших соседей будет присваивать новому объекту метку класса ближайшего к нему объекта. В случае, когда значение выбрано слишком большое, при классификации будет приниматься много объектов, что приведет к плохой выраженности границ классов.

**Задачи, решаемые данным методом:**

* Классификация машин по виду модели/скорости
* Классификация клиентов по их физическим данным
* Классификация животных по видам

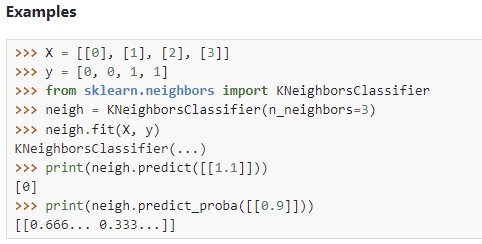
**Преимущества алгоритма:**

* Простая реализация
* Высокая точность
* Устойчивость к аномальным значениям

**Недостатки алгоритма:**

* Затраты в вычислениях, их-за необходимости вычисления расстояний до каждого объекта выборки.
* Необходимость хранить обучающую выборку целиком

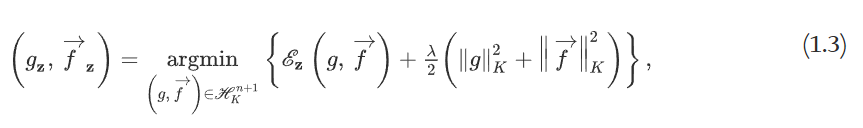
**Scikit-learn:**



## 

## 15. Метод градиентного спуска для задач классификации.

Алгоритм градиентного спуска в настройках классификации



Алгоритм (1.3) реализуется путем решения задачи выпуклой оптимизации квадратичного программирования с помощью теоремы о представителях. Однако размер проблемы оптимизации огромен, когда либо размер выборки, либо количество переменных велико. Та же трудность для соответствующих алгоритмов в условиях регуляризованной регрессии наименьших квадратов преодолевается за счет снижения вычислительной сложности методом градиентного спуска.

**Стохастический градиентный спуск (SGD)** — это простой, но очень эффективный подход к подгонке линейных классификаторов и регрессоров под выпуклые функции потерь, такие как (линейные) Метод опорных векторов и логистическая регрессия . Несмотря на то, что SGD существует в сообществе машинного обучения уже давно, совсем недавно он привлек значительное внимание в контексте крупномасштабного обучения.

SGD успешно применяется для решения крупномасштабных и разреженных задач машинного обучения, часто встречающихся при классификации текста и обработке естественного языка. Учитывая, что данные немногочисленны, классификаторы в этом модуле легко масштабируются для решения задач с более чем 10^5 обучающими примерами и более чем 10^5 функциями.

Строго говоря, SGD — это просто метод оптимизации и не соответствует конкретному семейству моделей машинного обучения. **Это всего лишь способ обучить модель**. Часто экземпляр SGDClassifier или SGDRegressor будет иметь эквивалентный оценщик в scikit-learn API, потенциально используя другой метод оптимизации.

Например, использование SGDClassifier(loss='log') результатов в логистической регрессии, т. е. Модель, эквивалент LogisticRegression которой подбирается через SGD, а не подгоняется одним из других решателей в LogisticRegression. Точно так же SGDRegressor(loss=’squared\_loss’, penalty=’l2′) и Ridge решают одну и ту же задачу оптимизации разными способами.

Преимущества стохастического градиентного спуска:

* Эффективность.
* Простота реализации (множество возможностей для настройки кода).

К недостаткам стохастического градиентного спуска можно отнести:

* SGD требует ряда гиперпараметров, таких как параметр регуляризации и количество итераций.
* SGD чувствителен к масштабированию функций.

## 16. Метод градиентного спуска для парной линейной регрессии.

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

Алгоритм градиентного спуска предназначен для решения экстремального значения в непрерывном пространстве решений.

Этот метод заимствован из оптимизации. Алгоритм позволяет найти минимум функции. Выбирает случайное начальное значение. Делает шаги пропорционально градиенту в данной точке, до тех пор, пока не достигнет предела. Проблема может возникнуть в достижении только локального минимума. Это проблема градиентного спуска, устранить ее можно подбором изначальных параметров и аналитических расчетов.

Исходное значение градиента - это вектор, что означает, что производная функции по направлению в этой точке принимает максимальное значение вдоль направления, то есть функция находится вдоль направления в этой точке ( Направление этого градиента изменяется быстрее всего, а скорость изменения самая большая (по модулю градиента).

Выбор параметра скорости обучения (learning rate) выполняется экспериментальным путем. Если минимизируемая функция достаточно гладкая, то параметр можно выбрать достаточно большим, тогда время работы алгоритма обучения будет малым. Если целевая функция имеет сильную кривизну, то в случае больших значений скорости обучения возможна ситуация, при которой вектор весов w будет «перескакивать» свое оптимальное значение. Для таких случаев следует брать малые значения параметра — скорость сходимости к минимуму функции (локальному!) будет малой, однако меньше шанс его пропустить.

Для реализации алгоритма обучения возможны два подхода:

* 1) пакетный {batch) — на каждой итерации алгоритма обучающая выборка просматривается целиком и после этого рассчитывается новое значение вектора w;
* 2) стохастический (stochastic) — на каждой итерации алгоритма случайным образом выбирается только один объект из обучающей выборки.

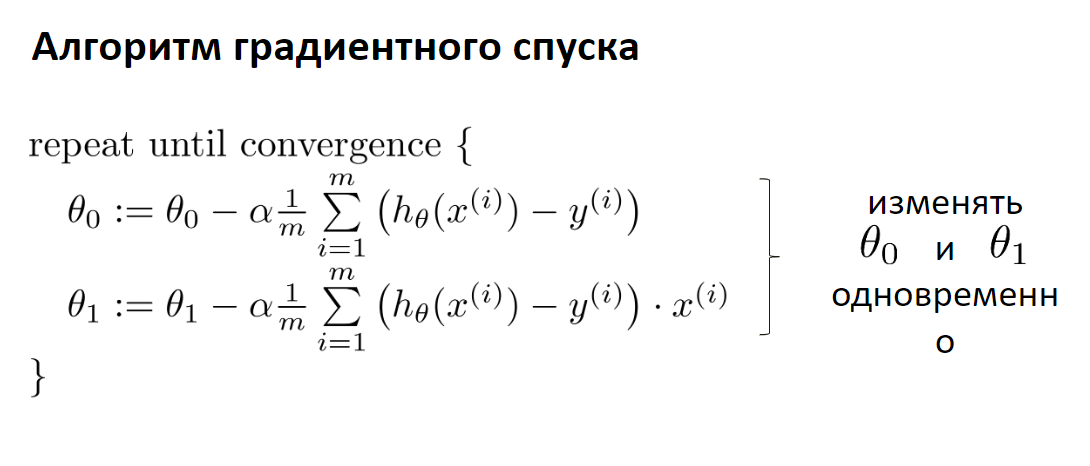
Первый метод вычислительно более сложен, однако он быстрее сходится к минимуму благодаря просмотру всей выборки. Второй метод предполагает менее трудоемкие вычисления, но требует разработки алгоритма выбора следующего элемента для обучения.

**Описание проблемы**

Существующая функция y = f (x1, x2, x3, ..., xn), теперь необходимо запросить минимальное значение этой функции.

**Идеи алгоритмов**

1. Знай, где ты сейчас
2. Найдите направление самого быстрого спуска относительно местоположения
3. Сделайте небольшой шаг в направлении, указанном на втором шаге, чтобы добраться до нового места, которое должно быть ниже исходного.
4. Вернуться к первому шагу
5. Конец в самой низкой точке



## 17. Метод опорных векторов в задачах классификации.

Задача классификации состоит в определении к какому классу из, как минимум, двух изначально известных относится данный объект. Обычно таким объектом является вектор в n-мерном вещественном пространстве . Координаты вектора описывают отдельные аттрибуты объекта.

**Метод опорных векторов** (англ. support vector machine, SVM) — один из наиболее популярных методов обучения, линейный алгоритм, который применяется для решения задач классификации и регрессии. Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора. SVM — алгоритм обучения с учителем.

**Теория**

Основной задачей алгоритма является найти наиболее правильную линию, или гиперплоскость разделяющую данные на два класса. SVM это алгоритм, который получает на входе данные, и возвращает такую разделяющую линию.

**Как SVM находит лучшую линию**

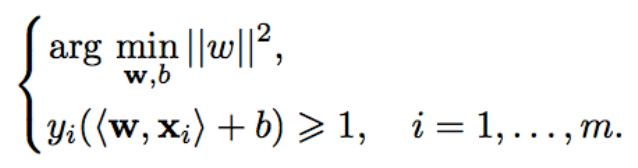
Алгоритм SVM устроен таким образом, что он ищет точки на графике, которые расположены непосредственно к линии разделения ближе всего. Эти точки называются опорными векторами. Затем, алгоритм вычисляет расстояние между опорными векторами и разделяющей плоскостью. Это расстояние которое называется зазором. Основная цель алгоритма — максимизировать расстояние зазора. Лучшей гиперплоскостью считается такая гиперплоскость, для которой этот зазор является максимально большим.

Данный метод изначально относится к бинарным классификаторам, хотя существуют способы заставить его работать и для задач мультиклассификации.

Пусть имеется обучающая выборка: .

Метод опорных векторов строит классифицирующую функцию F в виде , где  — скалярное произведение, w — нормальный вектор к разделяющей гиперплоскости,b — вспомогательный параметр. Те объекты, для которых F(x) = 1 попадают в один класс, а объекты с F(x) = -1 — в другой. Выбор именно такой функции неслучаен: любая гиперплоскость может быть задана в виде  для некоторых w и b.

Далее, мы хотим выбрать такие w и b которые максимизируют расстояние до каждого класса. Можно подсчитать, что данное расстояние равно . Проблема нахождения максимума  эквивалентна проблеме нахождения минимума . Запишем все это в виде задачи оптимизации:

**

которая является стандартной задачей квадратичного программирования и решается с помощью множителей Лагранжа.

**Линейная неразделимость**

На практике случаи, когда данные можно разделить гиперплоскостью, или, как еще говорят, линейно, довольно редки.

В этом случае поступают так: все элементы обучающей выборки вкладываются в пространство X более высокой размерности с помощью специального отображения . При этом отображение  выбирается так, чтобы в новом пространстве X выборка была линейно разделима.

Классифицирующая функция F принимает вид .

## 18. Метод решающих деревьев в задачах классификации.

**Решающее дерево** (Decision tree) — решение задачи обучения с учителем, основанный на том, как решает задачи прогнозирования человек. В общем случае — это k-ичное дерево с решающими правилами в нелистовых вершинах (узлах) и некотором заключении о целевой функции в листовых вершинах (прогнозом). Решающее правило — некоторая функция от объекта, позволяющее определить, в какую из дочерних вершин нужно поместить рассматриваемый объект. В листовых вершинах могут находиться разные объекты: класс, который нужно присвоить попавшему туда объекту (в задаче классификации), вероятности классов (в задаче классификации).

Деревья решений представляют собой иерархические древовидные структуры, состоящие из решающих правил вида «Если ..., то ...». Правила автоматически генерируются в процессе обучения на обучающем множестве и, поскольку они формулируются практически на естественном языке, деревья решений как аналитические модели более вербализуемы и интерпретируемы.

Дерево решений для классификации — отнесение объектов к одному из заранее известных классов. Целевая переменная должна иметь дискретные значения.

**Процесс построения**

Процесс построения деревьев решений заключается в последовательном, рекурсивном разбиении обучающего множества на подмножества с применением решающих правил в узлах. Процесс разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями. Объявление узла листом может произойти естественным образом (когда он будет содержать единственный объект, или объекты только одного класса), или по достижении некоторого условия остановки, задаваемого пользователем (например, минимально допустимое число примеров в узле или максимальная глубина дерева).

Алгоритмы построения деревьев решений относят к категории так называемых жадных алгоритмов. Жадными называются алгоритмы, которые допускают, что локально-оптимальные решения на каждом шаге, приводят к оптимальному итоговому решению. В случае деревьев решений это означает, что если один раз был выбран атрибут, и по нему было произведено разбиение на подмножества, то алгоритм не может вернуться назад и выбрать другой атрибут, который дал бы лучшее итоговое разбиение. Поэтому на этапе построения нельзя сказать обеспечит ли выбранный атрибут, в конечном итоге, оптимальное разбиение.

В настоящее время разработано значительное число алгоритмов обучения деревья решений: ID3, CART, C4.5, C5.0, NewId, ITrule, CHAID, CN2 и т.д. Но наибольшее распространение и популярность получили следующие:

* **ID3 (Iterative Dichotomizer 3)** — алгоритм позволяет работать только с дискретной целевой переменной, поэтому деревья решений, построенные с помощью данного алгоритма, являются классифицирующими. Число потомков в узле дерева не ограничено. Не может работать с пропущенными данными.
* **CART (Classification and Regression Tree)** — алгоритм обучения деревьев решений, позволяющий использовать как дискретную, так и непрерывную целевую переменную, то есть решать как задачи классификации, так и регрессии. Алгоритм строит деревья, которые в каждом узле имеют только два потомка.

**Основные этапы построения**

В ходе построения дерева решений нужно решить несколько основных проблем, с каждой из которых связан соответствующий шаг процесса обучения:

1. Выбор атрибута, по которому будет производиться разбиение в данном узле (атрибута разбиения).
2. Выбор критерия остановки обучения.
3. Выбор метода отсечения ветвей (упрощения).
4. Оценка точности построенного дерева.

## 

## 19. Методы векторизации текстов для задач машинного обучения.

Векторизация текста в машинном обучении – это процесс превращения текста в набор числовых признаков. В библиотеку sklearn встроены несколько простых методов векторизации текстов. Они собраны в модуле sklearn.feature\_extraction.text.

Рассмотрим следующие виды наиболее популярных векторизаторов:

* CountVectorizer
* TfidfVectorizer
* HashingVectorizer

**CountVectorizer** – токенизирует входные данные и строит словарь известных слов, а затем представляет документ, используя этот словарь.

**Пример:**  
text = ["She sells seashells in the seashore"]

vectorizer = CountVectorizer()

X = vectorizer.fit\_transform(text)

**TfidfVectorizer**. Одна из проблем при работе с Countvectorizer заключается в том, что частые слова, такие как "the" будут встречаться много раз, не представляя большой значимости. В таких случаях используется TfidfVectorizer (Term Frequency Inverse Document Frequency)

Частота слова (Term Frequency) — подсчитывает, как часто выбранное слово появляется в документе.

Обратная частота документа (Inverse Document Frequency) — снижает вес слов, которые часто встречаются в документах.

**Пример:**

Использование аналогично CountVectorizer.

**HashingVectorizer**. HashingVectorizer комбинирует 2 метода: FeatureHasher, который преобразовывает строку в числовой массив заданной длинной с помощью хэш-функции и CountVectorizer, который преобразовывает входной текст в матрицу, значениями которой, являются количества вхождения данного ключа(слова) в текст.

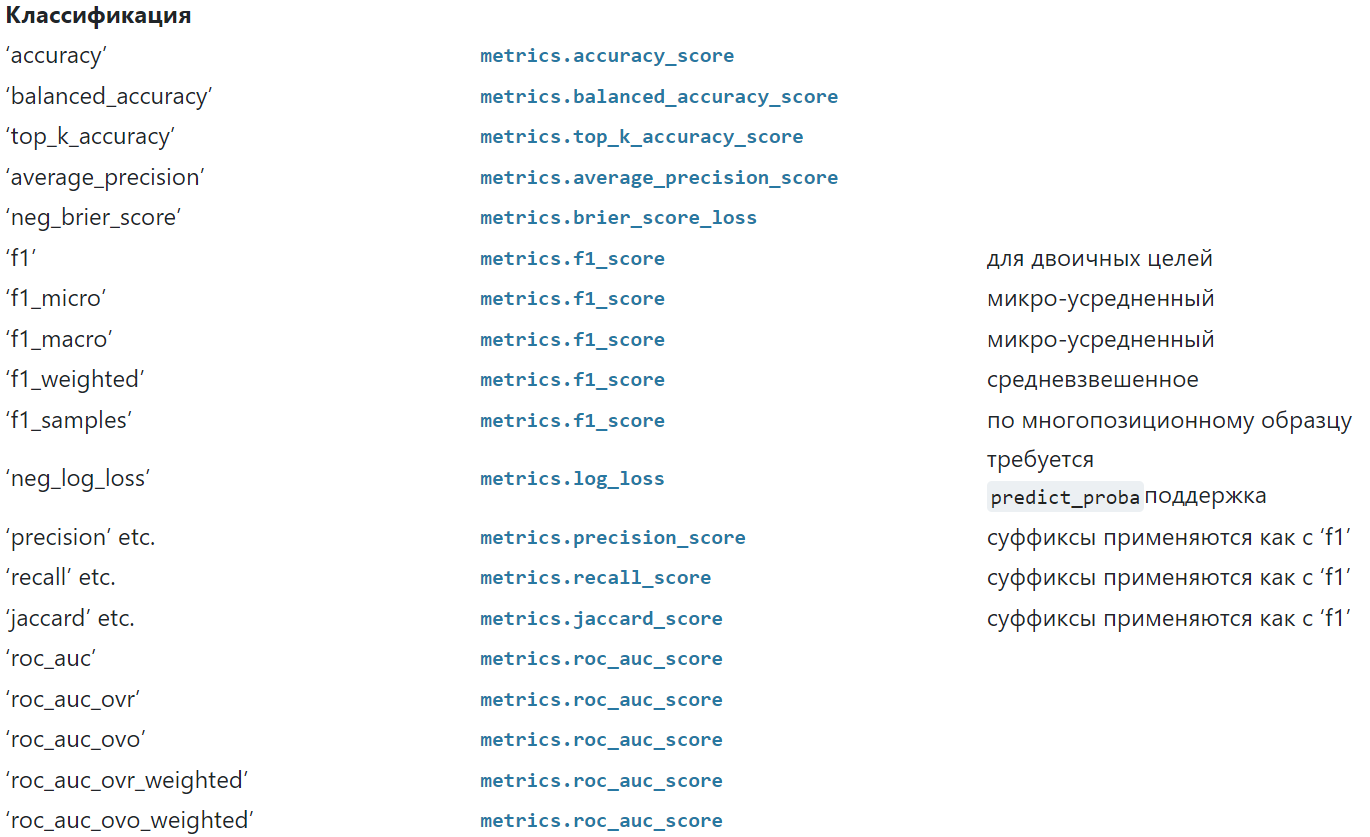
**Пример:**

hv = HashingVectorizer(n\_features=10)

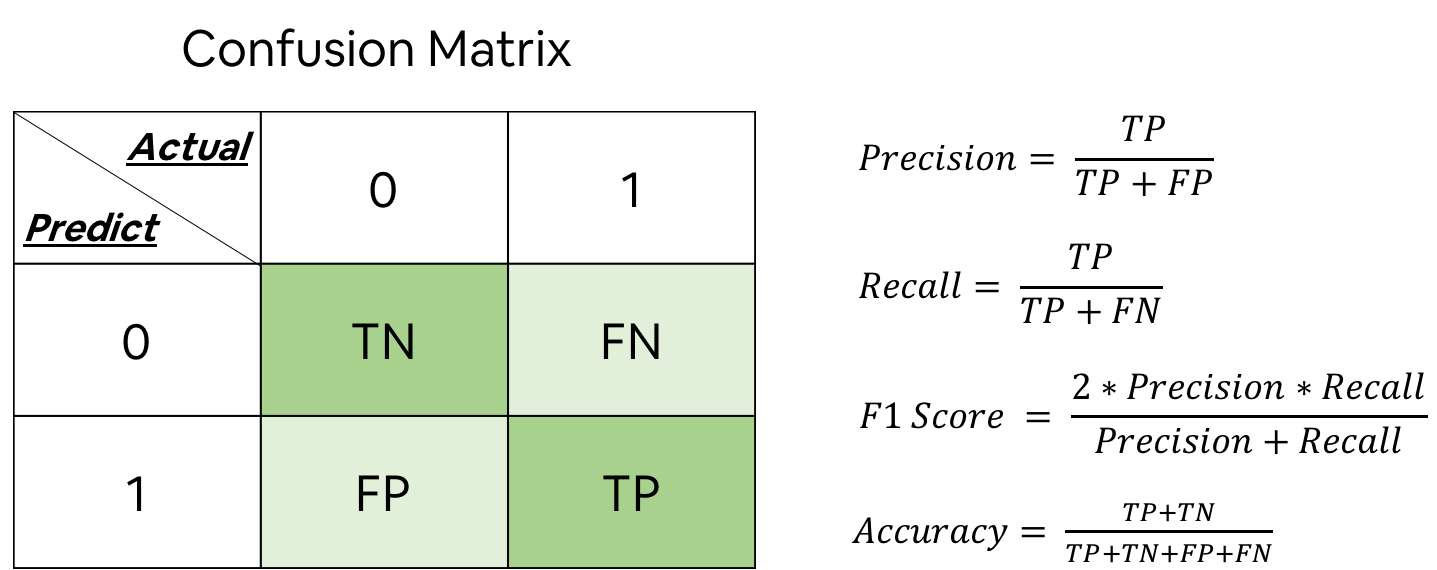
hv.transform(corpus)

Метод fit вызывать уже не нужно.

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

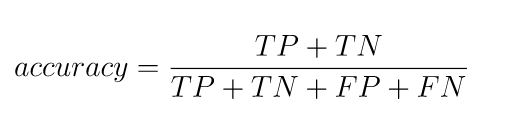
****

Перед переходом к самим метрикам необходимо ввести важную концепцию для описания этих метрик в терминах ошибок классификации — **confusion matrix** (матрица ошибок). **Матрица ошибок** – это показатель успешности классификации, где классов два или более. Это таблица с различными комбинациями сочетаний прогнозируемых и фактических значений.



**Accuracy**

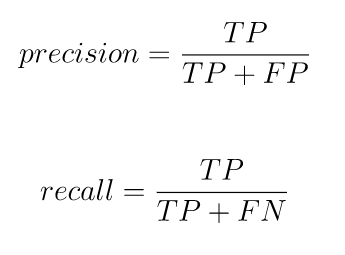
Интуитивно понятной, очевидной и почти неиспользуемой метрикой является accuracy — доля правильных ответов алгоритма:



Эта метрика бесполезна в задачах с неравными классами.

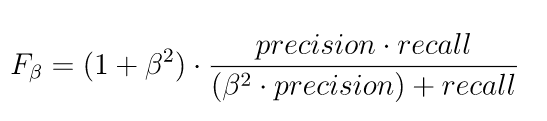
**Precision, recall и F-мера**

Для оценки качества работы алгоритма на каждом из классов по отдельности введем метрики precision (точность) и recall (полнота).



Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм. Именно введение precision не позволяет нам записывать все объекты в один класс, так как в этом случае мы получаем рост уровня False Positive. Recall демонстрирует способность алгоритма обнаруживать данный класс вообще, а precision — способность отличать этот класс от других классов. Precision и recall не зависят, в отличие от accuracy, от соотношения классов и потому применимы в условиях несбалансированных выборок.

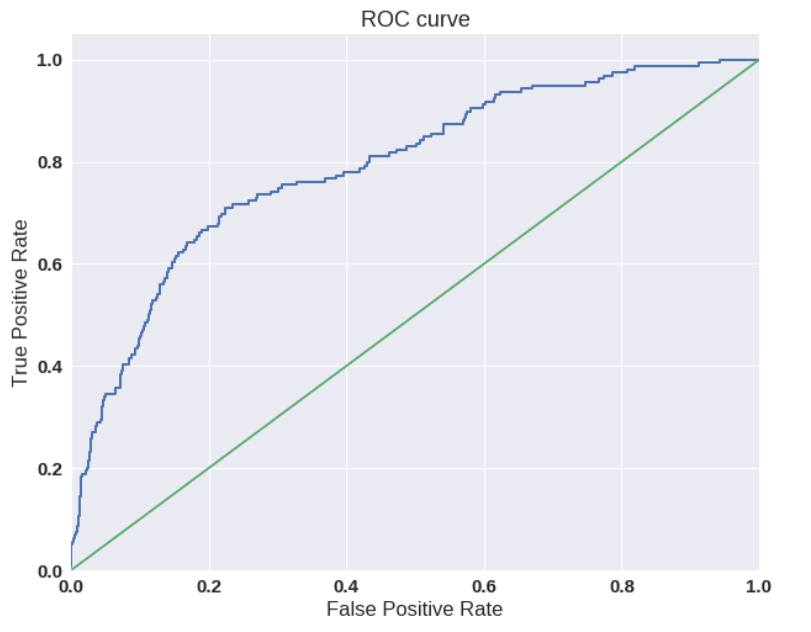
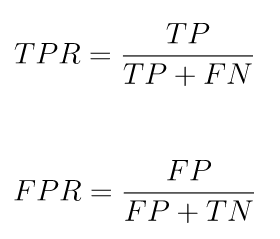
Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. F-мера — среднее гармоническое precision и recall:



**AUC-ROC и AUC-PR**

При конвертации вещественного ответа алгоритма в бинарную метку, мы должны выбрать какой-либо порог, при котором 0 становится 1. Естественным и близким кажется порог, равный 0.5, но он не всегда оказывается оптимальным, например, при вышеупомянутом отсутствии баланса классов.

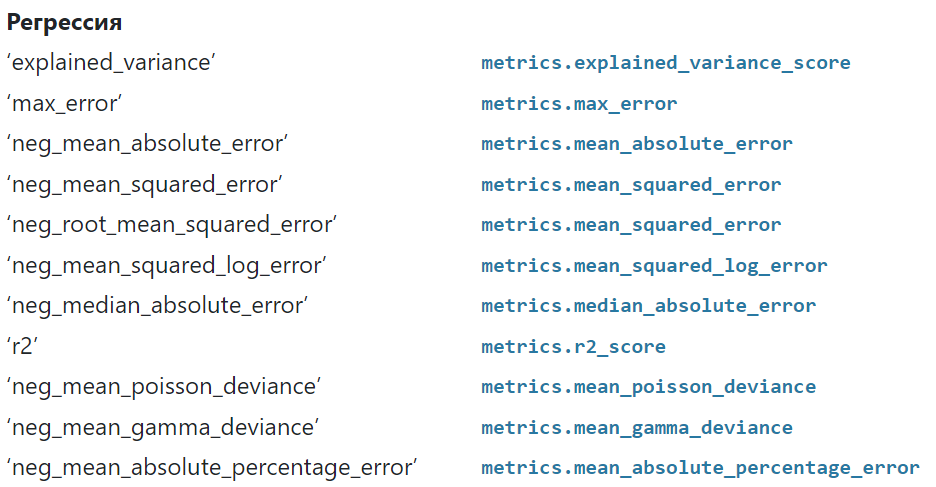
Одним из способов оценить модель в целом, не привязываясь к конкретному порогу, является AUC-ROC (или ROC AUC) — площадь (Area Under Curve) под кривой ошибок (Receiver Operating Characteristic curve ). Данная кривая представляет из себя линию от (0,0) до (1,1) в координатах True Positive Rate (TPR) и False Positive Rate (FPR):



* В случае многоклассовой классификации нужно внимательно следить за метриками каждого из классов и следовать логике решения задачи, а не оптимизации метрики;
* В случае неравных классов нужно подбирать баланс классов для обучения и метрику, которая будет корректно отражать качество классификации;
* Выбор метрики нужно делать с фокусом на предметную область, предварительно обрабатывая данные и, возможно, сегментируя (как в случае с делением на богатых и бедных клиентов).

## 

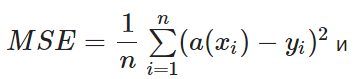
## 21. Метрики эффективности моделей регрессии. Виды, характеристика, выбор.

****

Наиболее типичными мерами качества в задачах регрессии являются:

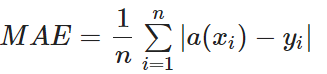
**Средняя квадратичная ошибка (англ. Mean Squared Error, MSE)**

MSE применяется в ситуациях, когда нам надо подчеркнуть большие ошибки и выбрать модель, которая дает меньше больших ошибок прогноза. Грубые ошибки становятся заметнее за счет того, что ошибку прогноза мы возводим в квадрат. И модель, которая дает нам меньшее значение среднеквадратической ошибки, можно сказать, что что у этой модели меньше грубых ошибок.

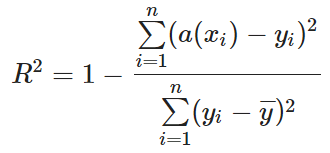


**Средняя абсолютная ошибка (англ. Mean Absolute Error, MAE)**

Среднеквадратичный функционал сильнее штрафует за большие отклонения по сравнению со среднеабсолютным, и поэтому более чувствителен к выбросам. При использовании любого из этих двух функционалов может быть полезно проанализировать, какие объекты вносят наибольший вклад в общую ошибку — не исключено, что на этих объектах была допущена ошибка при вычислении признаков или целевой величины. Среднеквадратичная ошибка подходит для сравнения двух моделей или для контроля качества во время обучения, но не позволяет сделать выводов о том, на сколько хорошо данная модель решает задачу.

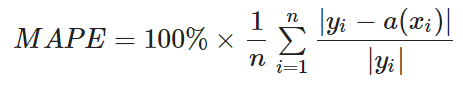


**Коэффициент детерминации**



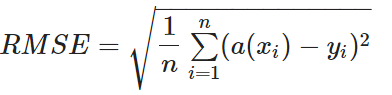
Коэффициент детерминации измеряет долю дисперсии, объясненную моделью, в общей дисперсии целевой переменной. Фактически, данная мера качества — это нормированная среднеквадратичная ошибка. Если она близка к единице, то модель хорошо объясняет данные, если же она близка к нулю, то прогнозы сопоставимы по качеству с константным предсказанием.

**Средняя абсолютная процентная ошибка (англ. Mean Absolute Percentage Error, MAPE)**



Это коэффициент, не имеющий размерности, с очень простой интерпретацией. Его можно измерять в долях или процентах. Если у вас получилось, например, что MAPE=11.4%, то это говорит о том, что ошибка составила 11,4% от фактических значений. Основная проблема данной ошибки — нестабильность.

**Корень из средней квадратичной ошибки (англ. Root Mean Squared Error, RMSE)**

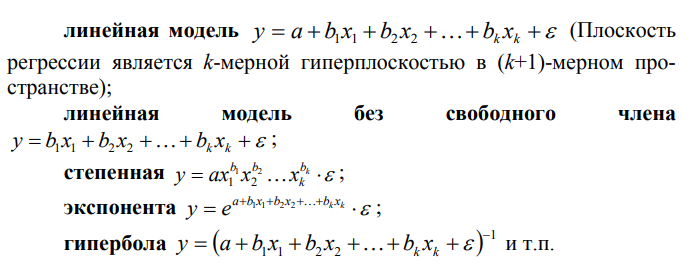


Примерно такая же проблема, как и в MAPE: так как каждое отклонение возводится в квадрат, любое небольшое отклонение может значительно повлиять на показатель ошибки. Стоит отметить, что существует также ошибка MSE, из которой RMSE как раз и получается путем извлечения корня.

## 22. Множественная и нелинейная регрессии.

**Множественной регрессией** называется уравнение связи с несколькими регрессорами X2,…,Xk. Модель множественной линейной регрессии имеет вид: Yi = β1 + β 2 X 2i + ...+ β k X ki + ε i , i = 1,…, n, где n – число наблюдений, X 2 ,..., X k - независимые переменные, Y – зависимая переменная, ε – случайная составляющая, β1 ,β 2 ,...,β k - коэффициенты регрессии.

**Нелинейная регрессия** - это тип полиномиальной регрессии. Это метод моделирования нелинейной зависимости между зависимыми и независимыми переменными. Он используется на месте, когда данные показывают извилистую тенденцию, и линейная регрессия не даст очень точных результатов по сравнению с нелинейной регрессией.



Множественная линейная регрессия (MLR) или множественная регрессия - это статистический метод, который использует несколько подготовительных переменных для прогнозирования результата переменной ответа. Целью множественной линейной регрессии (MLR) является моделирование линейной зависимости между объясняющими (независимыми) переменными и ответной (зависимой) переменной.

По сути, множественная регрессия - это расширение обычной регрессии наименьших квадратов (OLS), которая включает более одной объясняющей переменной.

Модель множественной регрессии основана на следующих предположениях:

* **Линейность**: существует линейная связь между зависимыми переменными и независимыми переменными.
* **Корреляция**: независимые переменные не слишком сильно коррелируют друг с другом.
* **Наблюдения** yi выбираются независимо и случайным образом из популяции.
* **Нормальное распределение:** остатки должны быть нормально распределены со средним значением 0 и дисперсией σ.

**Пример:**



## 23. Однослойный перцептрон в задачах классификации.

**Однослойный персептрон** — это линейный алгоритм классификации, принцип работы которого основан на модели нервной клетки - нейрона. Представляет собой пример нейронной сети с одним скрытым слоем. Однослойный персептрон имеет простой алгоритм обучения и способен решать лишь самые простые задачи.

**Постановка задачи линейного распределения классов:**

Пусть  - множество объектов; Y - множество допустимых ответов. Будем считать, что , где  - признаковое описание объекта, а  - дополнительный константный признак; . Задана обучающая выборка . Значения признаков  рассматриваются как импульсы, поступающие на вход нейрона, которые складываются с весами . Если суммарный импульс превышает порог активации , то нейрон возбуждается и выдаёт на выходе 1, иначе выдаётся 0. Таким образом, нейрон вычисляет n-арную булеву функцию вида . Для настройки вектора весов воспользуемся методом стохастического градиента. Возьмем квадратичную функцию потерь: , а в качестве функции активации возьмем сигмоидную функцию: . Согласно принципу минимизации эмпирического риска задача сводится к поиску вектора, доставляющего минимум функционалу .

**Описание алгоритма:**

В соответствии с методом градиентного спуска: , где \eta > 0 величина шага в направлении антиградиента, называемая также темпом обучения (learning rate). Темп обучения выбираем, решая задачу одномерной минимизации:. Будем выбирать прецеденты  по одному в случайном порядке, для каждого делать градиентный шаг и сразу обновлять вектор весов:



**Значение функционала оцениваем:**

, где , параметр  берем равным . Процедура останавливается после того, как изменение значения функционала функционала Q становится меньше заданной константы: 

## 24. Основные этапы проекта по машинному обучению.

**Жизненный цикл модели машинного обучения (ML)** – это процесс, охватывающий сразу от идентификации исходных данных до разработки модели, развертывания модели и обслуживания модели. На высоком уровне вся деятельность подпадает под две широкие категории, такие как разработка моделей машинного обучения и операции с моделями машинного обучения.

Шаги жизненного цикла разработки модели машинного обучения можно в целом разделить на следующие:

1. исследование данных
2. построение модели
3. настройка гиперпараметров модели
4. выбор модели с оптимальной производительностью.

**Исследовательский анализ данных**– важный шаг, который начинается после того, как бизнес-гипотеза готова. Исследовательский анализ данных включает идентификацию атрибутов данных, предварительную обработку данных и проектирование функций. Идентификация атрибутов включает в себя классификацию функций на категориальные и непрерывные переменные, что помогает в применении соответствующей обработки, чтобы быть задается переменной алгоритмом при построении модели.

**Предварительная обработка данных** включает в себя идентификацию недостающих значений и выбросов и заполнение этих пробелов путем вычисления среднего значения для количественных атрибутов и режима для качественных атрибутов данных для повышения предварительной силы модели. Функциональная инженерия– это следующий по важности шаг в исследовательском анализе данных, когда необработанный набор данных обрабатывается для преобразования типов данных строковых, datetime или числовых в числовые векторы для алгоритма машинного обучения для понимания и построения эффективной модели прогнозирования.

В качестве последнего шага необходимо выбрать соответствующие функции, которые помогают повысить точность модели, с помощью таких методов, как одномерный выбор , важность функции и матрица корреляции . Эти методы обнаруживают коллинеарность между двумя переменными, где они сильно коррелированы и содержат аналогичную информацию о дисперсии в пределах данного набора данных.

Построение модели машинного обучения требует разделения данных на два набора, такие как «обучающий набор» и «набор для тестирования».

**Вычисление производительности модели** является следующим логическим шагом для выбора правильной модели. Показатели производительности модели будут определять окончательный выбор модели, которые включают вычисление точности, точности, отзыва, оценки F1 (средневзвешенное значение точности и отзыва), а также матрицу путаницы для моделей классификации и коэффициент определения для регрессионных моделей.

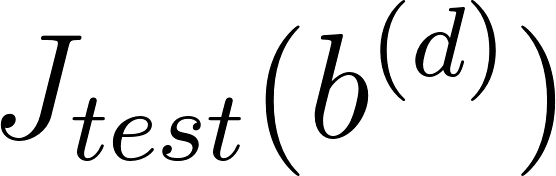
**Настройка гиперпараметров модели** является настоятельно рекомендуемым шагом в этом процессе, продолжайте, пока производительность модели не достигнет примерно 80-85%. Рекомендуется использовать метод поиска по сетке для поиска оптимальных гиперпараметров модели, что приводит к наиболее “точным” прогнозам. Кроме того, также рекомендуется выполнить перекрестную проверку, так как иногда повышение точности модели может быть связано с переоснащением (слишком чувствительная модель, неспособная к обобщению) или недостаточной подгонкой (очень обобщенная модель) модели с использованием метода перекрестной проверки k-кратной. Чтобы избежать переоснащения, увеличьте размер выборки обучающих данных, чтобы ввести больше шаблонов, или уменьшите количество функций, чтобы избежать сложности.

**Выберите модель с оптимальной производительностью.**

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

Если алгоритм обучения хорошо подходит для тренировочного набора, это не значит, что это хорошая гипотеза. Ошибка вашей гипотезы, измеренная в наборе данных, с которым вы подготовили параметры, будет ниже, чем на любом другом наборе данных.

Чтобы выбрать модель вашей гипотезы, вы можете проверить каждую степень полинома и посмотреть на результат ошибки. Метод без использования третьего, валидационного набора (обратите внимание: это плохой метод - не используйте его) выглядел бы следующим образом:

1. Оптимизируйте параметры b, используя обучающий набор для каждой степени полинома.
2. Найдите степень полинома d с наименьшей ошибкой, используя тестовый набор.
3. Оцените ошибку обобщения, также используя тестовый набор с [](about:blank) , (d = theta от полинома с более низкой ошибкой);

В этом случае мы обучили одну переменную d или степень полинома, используя тестовый набор. Это приведет к тому, что наше значение ошибки будет больше для любого другого набора данных.

Подобные переменные, не являющиеся параметрами гипотезы, но так или иначе влияющие на точность нашей модели называются гиперпараметрами модели. Обучение гиперпараметров - более сложный и долгий путь и он требует проведения специальной процедуры.

Чтобы решить эту проблему, мы можем ввести третий набор, валидационный набор, чтобы использовать его как промежуточный набор для обучения гиперпараметров. Тогда наш тестовый набор даст нам точную, не оптимистичную ошибку.

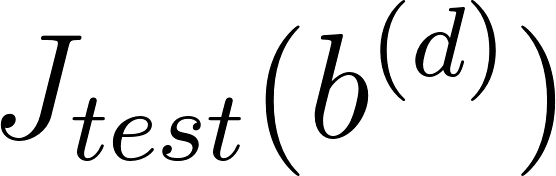
Один из возможных примеров разбивки нашего набора данных на три набора:

1. Обучающий набор (train set): первые 60%;
2. Валидационный набор (validation set): следующие 20%;
3. Тестовый набор (test set): последние 20%.

Этот лишь одна из самых простых схем разбиения набора данных. С помощью специальной техники - кросс-валидации или перекрестной проверки - процесс разбиения набора автоматизируется и становится более робастным. Пока не будем рассматривать эту продвинутую технику выбора модели.

Теперь мы можем вычислить три отдельных значения ошибки для трех разных наборов.

Метод выбора гипотезы с помощью валидационного набора (обратите внимание: этот метод предполагает, что мы не используем регуляризацию и на валидационном наборе)

1. Оптимизируйте параметры b, используя набор тренировок для каждой степени полинома.
2. Найдите степень полинома d с наименьшей ошибкой, используя валидационный набор.
3. Оцените ошибку обобщения, используя тестовый набор с [](about:blank)(где b - параметры от полинома с более низкой ошибкой);

Таким образом, степень полинома d не была обучена с использованием тестового набора.

Имейте в виду, что использование валидационного набора для выбора d означает, что мы также не можем использовать его для процесса проверки правильности установки значения лямбда.

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

Артур Самуэль дал следующее определение:

Машинное обучение - это способ заставить компьютер решить определенную задачу без описания явного алгоритма решения задачи.

Том Митчелл предлагает более современное определение: “Говорят, что компьютерная программа учится на опыте E по отношению к некоторому классу задач T и показателю производительности P, если его производительность при задачах в T, измеряемая через P, улучшается с опытом E.”

Пример: игра в шашки. В такой задаче, чтобы рассматривать ее как потенциальную задачу машинного обучения необходимо формализовать три главные части задачи, например, таким образом:

* E = опыт игры во многие игры шашек, их исходы, стратегия игры в каждой партии;
* T = задача игры в шашки, правила игры, их точное определение, все возможные ограничения;
* P = вероятность того, что программа выиграет следующую игру, доля игр, которые программа выигрывает.

Таким образом, если мы можем сформулировать три этих аспекта в виде математических формул или программных алгоритмов, мы можем начать решать эту задачу методами машинного обучения.

Машинное обучение в частности и ИИ дают возможность подступиться к решению таких задач, которые считались попросту невозможными для классического программирования. Вместо того чтобы писать явный алгоритм машинное обучение предлагает другой вариант. Нам просто надо собрать много данных о том, в каких ситуациях какие правильные действия нужно сделать и подобрать достаточно сложную, “емкую” модель, которая сможет подстроить свое поведение под эти данные. Если данных будет достаточно для “покрытия” всех возможных ситуаций, то и модель сможет принимать правильные решения в таких ситуациях и в похожих на те, что она видела при своем “обучении”.

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

**Набор данных (датасет)** – это структурированная информация, представленная в табличном виде.

При этом строки такой таблицы называются объектами, а столбцы – признаками.

**Признаки** - это некоторые характеристики объекта (не обязательно числовые).

**Объект** - это экземпляр некоторой сущности, которую мы изучаем.

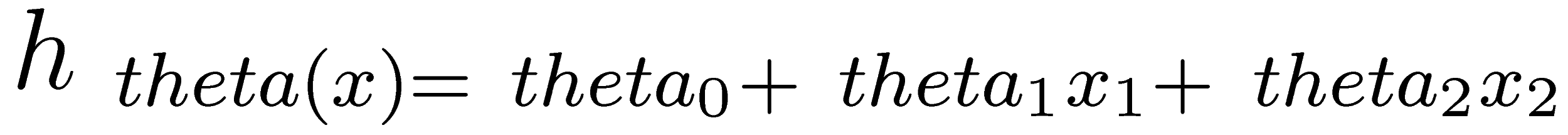
**Например:**

При изучение успеваемости студентов нашими объектами будут являться сами студенты, которые в свою очередь имеют признаки такие как Имя, Возраст, Оценка по математике и так далее.

Качество и релевантность набора данных напрямую влияет на эффективность обученной системы. Другими словами, если у вас плохой, недостаточный, нерелевантный набор данных, в них есть пропуски, выбросы, систематические ошибки - то уже совершенно неважно, насколько хорошую модель вы используете, все равно на выходе получится неэффективная система. Модели подстраиваются под данные, поэтому обработка данных, их анализ, сбор, очистка - это очень важные части технологий машинного обучения.

## 28. Понятие недо- и переобучения. Определение, пути решения.

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 [](about:blank), 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.

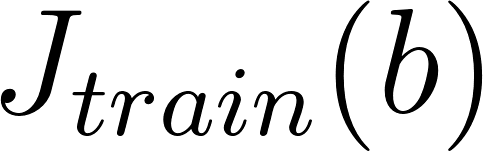
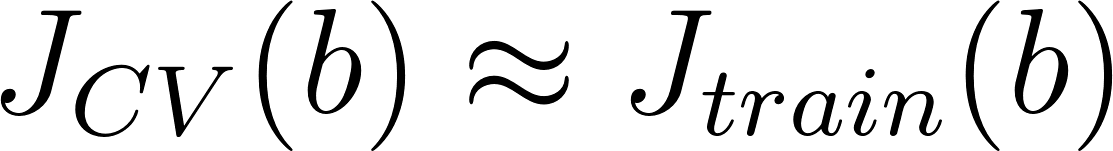
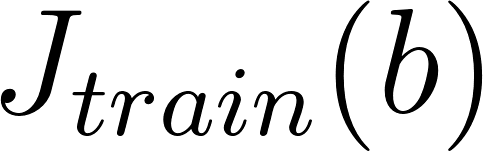
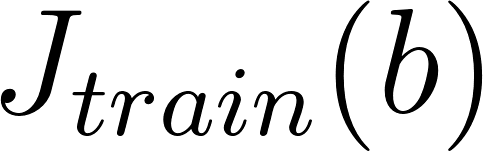
b) Использовать алгоритм выбора модели.

2) Регуляризация

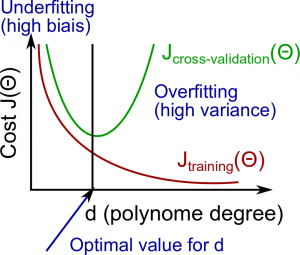
Сохраните все признаки, но уменьшите параметры bj.

Регуляризация работает хорошо, когда у нас много полезных признаков. Нам нужно различать, является ли проблемой, которая способствует плохим предсказаниям, смещение (bias) или дисперсия (variance) функции гипотезы. Высокое смещение чаще недообучается, а высокая дисперсия чаще переобучается. Нам нужно найти золотую середину между этими двумя крайностями.

Ошибка обучения будет уменьшаться по мере увеличения степени d многочлена. В то же время ошибка перекрестной проверки будет уменьшаться по мере увеличения d до точки, а затем она увеличивается с увеличением d, образуя выпуклую кривую.

* Высокое смещение (недообучение): как [](about:blank), так и [](about:blank) будут высокими. Кроме того, [](about:blank).
* Высокая дисперсия (переобучение):[](about:blank) будет низкой, а [](about:blank) будет намного больше, чем [](about:blank).

При выборе из многих значений d можно построить зависимость ошибок на обучающем и валидационных наборах на графике зависимости от d. Такой график наглядно показывает области недо- и переобучения и выглядит подобно следующему:

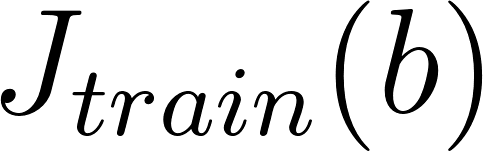
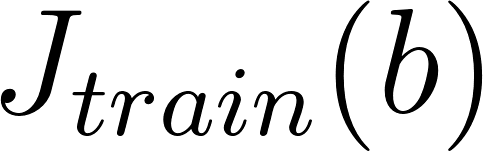
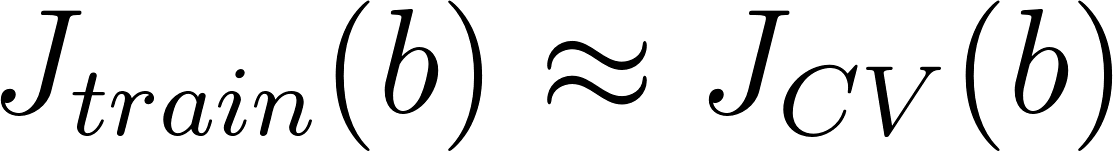
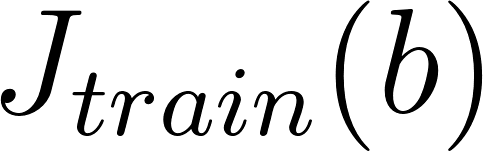


Вместо того, чтобы смотреть на степень d, способствующую смещению / дисперсии, теперь мы рассмотрим параметр регуляризации λ.

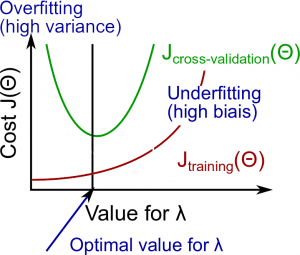
* Большой λ: высокое смещение (недообучение);
* Малый λ: высокая дисперсия (переобучение).

Большая лямбда сильно штрафует все параметры b, что значительно упрощает линию нашей результирующей функции, поэтому вызывает недообучение.

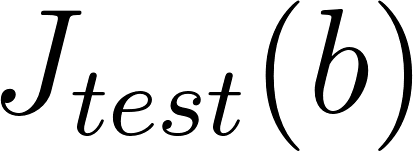
Отношение λ к набору тренировок и набору дисперсий выглядит следующим образом:

* Низкий λ: [](about:blank) низкий, а [](about:blank) высокий (высокая дисперсия / переобучение);
* Оптимальное значение λ: [](about:blank) и [](about:blank) низки и [](about:blank);
* Большой λ: оба [](about:blank) и [](about:blank) будут высокими (недообучение / высокое смещение).

Опять же, можно построить зависимость ошибок от параметра регуляризации для наглядного поиска оптимального значения λ:



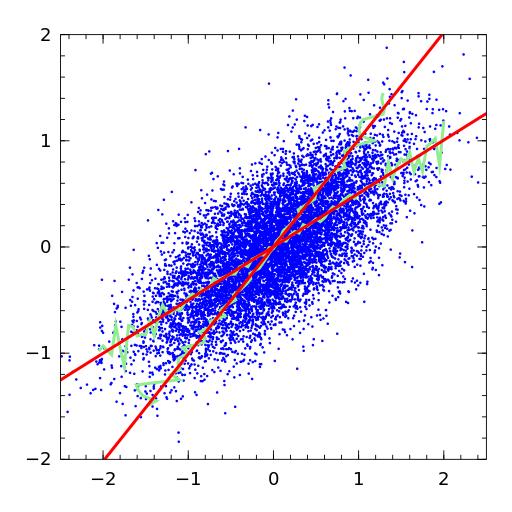
Чтобы выбрать вид модели и параметр регуляризации λ одновременно, нам нужно:

1. Создайте список возможных значения λ (например, λ∈ {0, 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, 1.28, 2.56 ,5.12, 10.24});
2. Создайте набор моделей с разной степенью или любыми другими вариантами;
3. Начните итерировать по λ и для каждого λ итерировать по всем моделям, чтобы обучать параметры b;
4. Вычислите ошибку валидационного набора, используя обученные b (вычисленные с λ) на [](about:blank) без регуляризации или λ = 0;
5. Выберите лучшую комбинацию, которая дает самую низкую ошибку на валидационном наборе;
6. Используя наилучшую комбинацию b и λ, примените ее на [](about:blank), чтобы увидеть, дает ли она хорошее обобщение задачи.

## 29. Понятие параметров и гиперпараметров модели. Обучение параметров и гиперпараметров. Поиск по сетке.

**Параметры** - это численные коэффициенты, которые изменяются в процессе обучения.

Давайте рассмотрим для примера в качестве модели самую простую математическую функцию - линейную. Ее формула y = θ0 + θ1x. Здесь y - выходная переменная, значение или целевая переменная, x - входная переменная, аргумент или фактор, а θ - параметры модели, их мы уже сразу обозначим именно как вектор. В данном случае это двумерный вектор параметров или можно просто сказать, что у этой модели два параметра. Понятно, что в зависимости от значения вектора параметров линейная функция будет принимать различные значения. Если добавить сюда данные, то может получиться картина типа такой:



**Гиперпараметры** - это параметры, которые определяют саму структуру модели. Они устанавливаются программистом перед началом обучения и не меняются в процессе.

При изменении (оптимизации) гиперпараметров обучение нужно запускать заново.

Есть несколько методов оптимизации гиперпараметров наиболее известным из которых является “поиск по сетке”, или в оригинале Grid search.

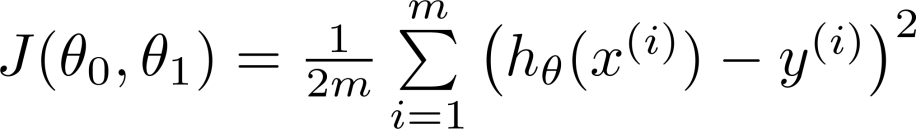
Данный метод перебирает все возможные комбинации гиперпараметров, обучает на каждой такой комбинации модель и считает её качество. Затем путем сравнения результатов выдает набор гиперпараметров, который дал наилучшие результаты.

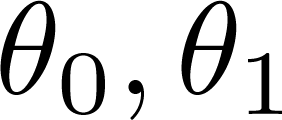
## 

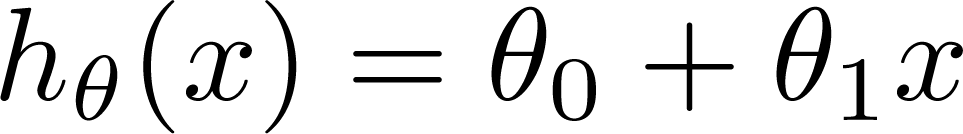
## 30. Понятие функции ошибки: требования, использование, примеры.

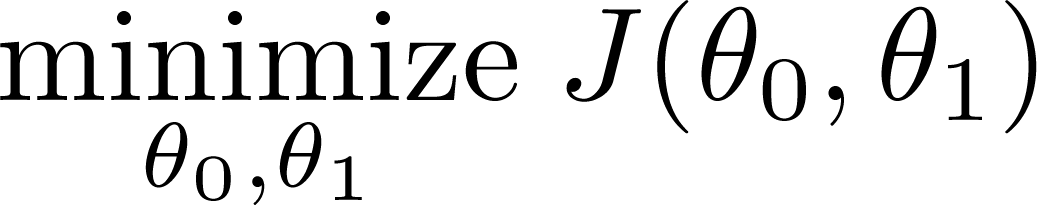
Чтобы оценить ML-алгоритм (параметры модели), нужно вычислить, насколько результат его работы далек от ожидаемого. Эту задачу решает функция потерь (Loss/ функция ошибки). Она определяет расстояние между фактическим выходом алгоритма и ожидаемым. Чем меньше результат, который выдает функция потерь, тем лучше. ​​Функции потерь не являются фиксированными, они меняются в зависимости от поставленной задачи и цели, которую необходимо достичь. Функция потерь возникает в тот момент, когда мы сводим задачу построения модели к задаче оптимизации.

Для парной линейной регрессии:

, где

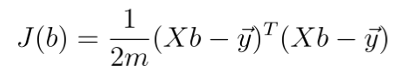
 - параметры

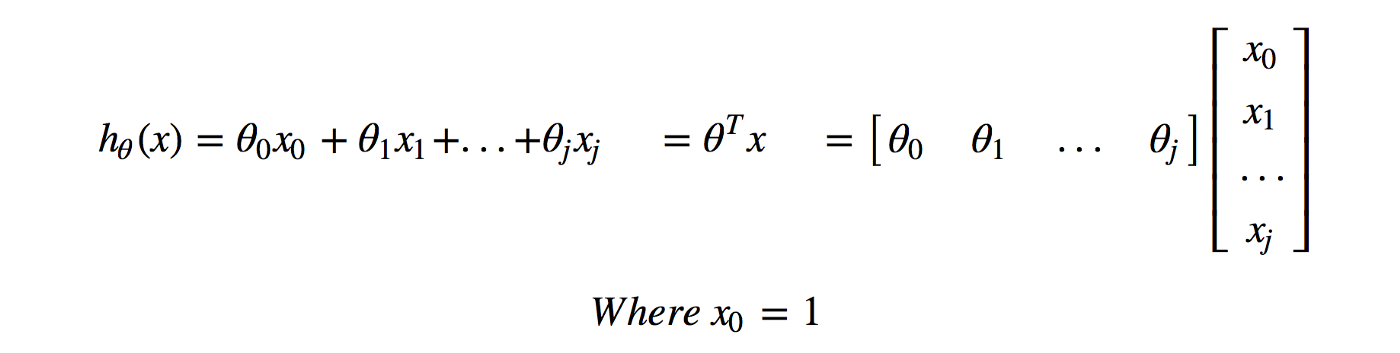
 - гипотеза

цель: 

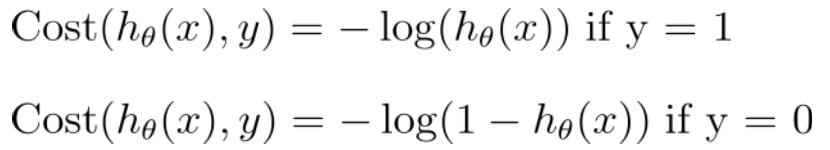
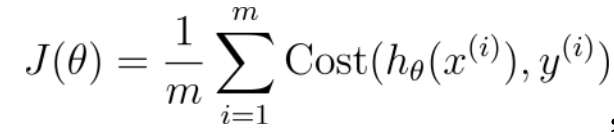
По сути своей, это половина среднего квадрата разницы между прогнозируемым и фактическим значением выходной переменной.

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.

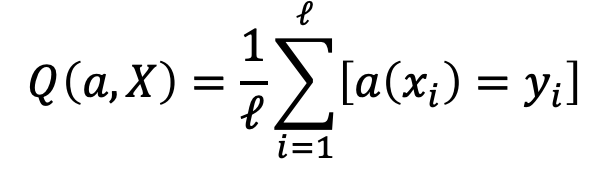
Для **множественной регрессии функция** ошибки от вектора параметров b в матричной форме выглядит следующим образом: 

(гипотеза ->) 

**Логистическая регрессия**

****

Accuracy (доля правильных ответов):

****

**Выбор функции ошибки**

* Среднеквадратичная ошибка (MSE) (он же L2 Loss)*.*Данный метод расчёта в значительной мере чувствителен к выбросам в выборке, или к выборкам где разброс значений очень большой. В основном, данная функция применяется для переменных, распределение которых близко к распределению Гаусса.
* Средняя абсолютная ошибка (MAE) (он же L1 Loss) – это усреднённая сумма модулей разницы между реальным и предсказанным значениями. MAE во многом похожа на MSE, но она отличается меньшей чувствительностью к выбросам значений (так как не берётся квадрат отклонения).
* Среднеквадратичная логарифмическая ошибка (MSLE) – усреднённая сумма квадратов разностей между логарифмами значений. Благодаря большому гасящему эффекту логарифма она более применима к моделям, строящимся на данных, которые имеют большой разброс значений на несколько порядков.

## 

## 31. Понятие чистых данных. Определение, очистка данных.

Определение чистых данных (tidy data):

* Каждая переменная соответствует колонке
* Каждое измерение соответствует строке
* Каждая таблица\файл содержит данные об одном виде наблюдений\экспериментов

Одна из самых распространенных проблем в данных, использующихся в машинном обучении - наличие отсутствующих измерений. Данные с пропусками нельзя без предварительной обработки подавать на вход почти никаких обучаемых моделей, поэтому практически непременным этапом предварительного анализа данных является выявление отсутствующих значений. Главным образом цель этого метода - выявить лучший подход к избавлению от пропущенных значений.

**Основные задачи:**

* Выбор стратегии борьбы с пропущенными данными.

**Основные методы борьбы с пропусками значений**

* Удаление признака
* Удаление объекта
* Заполнение средним значением
* Заполнение медианным, модальным, средним по логарифмической шкале
* Заполнение групповым средним
* Заполнение специальным значением
* Заполнение случайным значением
* Введение нового признака

## 

## 32. Понятие ядра и виды ядер в методе опорных векторов.

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

Классифицирующая функция F принимает вид . Выражение  называется ядром классификатора. С математической точки зрения ядром может служить любая положительно определенная симметричная функция двух переменных.

В практическом применении выбор, который вам нужно сделать, это:

* Выбор параметра C
* Выбор ядра (функция подобия)
  + Нет ядра («линейное» ядро) - дает стандартный линейный классификатор. Выберем, когда n велико, а m мало
  + Гауссовское ядро ​​(описанное выше) - нужно выбрать σ2. Выберите, когда n мало и m велико
  + Полиномиальное: 
  + Радиальная базисная функция: 
  + Гауссова радиальная базисная функция: 
  + Сигмоид: 

Выполняйте масштабирование признаков перед использованием гауссовского ядра.

## 33. Предварительный анализ данных: задачи, методы, цели.

**1. Подробное описание каждого признака, шкалы, вида распределения;**

На первоначальном этапе знакомства с набором данных для подготовки к обучению моделей очень важно понимать шкалы, по которым измеряется каждый фактор. Принципиальное различие в методах работы с категориальными и числовыми, дискретными и непрерывными признаками обосновывает необходимость получить эту информацию как можно раньше.

**Основные задачи:**

* Нахождение ошибок в шкалах;
* Разделение признаков на категориальные, числовые дискретные и непрерывные;
* Выбор инструментальных средств и средств визуализации данных.

**2. Исследование выборки на отсутствующие значения;**

Одна из самых распространенных проблем в данных, использующихся в машинном обучении - наличие отсутствующих измерений. Данные с пропусками нельзя без предварительной обработки подавать на вход почти никаких обучаемых моделей, поэтому практически непременным этапом предварительного анализа данных является выявление отсутствующих значений. Главным образом цель этого метода - выявить лучший подход к избавлению от пропущенных значений.

**Основные задачи:**

* Выбор стратегии борьбы с пропущенными данными.

**Основные методы борьбы с пропусками значений:**

* Удаление признака
* Удаление объекта
* Заполнение средним значением
* Заполнение медианным, модальным, средним по логарифмической шкале
* Заполнение групповым средним
* Заполнение специальным значением
* Заполнение случайным значением
* Введение нового признака

**3. Описание вида распределения каждого признака и целевой переменной;**

Также на начальном этапе зачастую строят индивидуальное эмпирическое распределение каждого признака, что позволяет выдвинуть предварительную гипотезу о виде распределения соответствующей переменной в генеральной совокупности. Вид распределения может оказать влияние на выбор метода нормализации данных (по среднему, по разбросу или стандартизация), выявить проблему смещенных классов, особенно острую в распределении по целевой переменной, индицировать системные ошибки выборки. Анализ индивидуального распределения может также помочь выявить аномальные выбросы, ошибки измерений, или опечатки в данных.

**Основные задачи:**

* Нахождение временных трендов по факторам;
* Обнаружение смещенных классов;
* Оценка степени вариации каждого признака;
* Гипотеза о виде эмпирического распределения признаков.

**Подробный анализ**

1. Исследование отношения выборки к генеральной совокупности;
2. Формальное тестирование гипотез о виде распределения каждого признака с возможным обнаружением аномалий;
3. Построение корреляционной матрицы и аллювиальных диаграмм;
4. Исследование выборки на однородность распределения.

**Основные этапы очистки данных**

1. Объединение всех источников данных
2. Удаление лишних признаков
3. Удаление непоказательных объектов
4. Заполнение пустых значений
5. Группировка числовых признаков
6. Преобразование категориальных значений
7. Создание суррогатных признаков
8. Решкалирование признаков

## 34. Представление графической информации в моделях машинного обучения.

Для работы с графической информацией в моделях машинного обучения необходимо преобразовать исходные данные (матрицу пикселей). Методы машинного обучения библиотеки sklearn рассчитаны, что на вход будет подаваться плоский массив признаков. Можно разными способами превратить изображение в набор признаков.

**Первый способ** - просто представить изображение как линейную последовательность пикселей.

Если изображение имеет разрешение 64 на 64 пикселя, то всего пикселей получается 4096. Если изображение черно-белое (в градациях серого), то значением признака, соответствующего каждому пикселю можно взять яркость этого пикселя. Если изображение цветное, то каждый пиксель в изображении состоит из красного, зеленого и синего. Эти три цвета образуют три цветовых канала. Каждый канал изменяется и накладывается, чтобы сформировать различные цвета. Диапазоны значений трех цветовых каналов R, G и B - все 0–255, а наложенное цветовое пространство - 256 ^ 3.

Обычно такие значения нормируются к шкале долей единицы. Такой набор признаков уже можно использовать как исходные данные для машинного обучения. Далее алгоритм машинного обучения полностью совпадает с работой с численными данными.

**Второй способ**, используемый для нейронных моделей – свертки.

## 35. 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.

Способ 1

Нумерация категориальных значений. 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.

Способ 2

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.

Способ 3

Метод среднего кодирования (Mean/Target Encoding). Метод предполагает кодирование категорий средним арифметическим от суммы целевых меток (Target). Однако, при кодировании средним по Target велика вероятность переобучения модели, эту проблему поможет решить регуляризация.

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

Вариант 1

Проверить каждую степень полинома и посмотреть на результат ошибки.

Этот метод без использования третьего, валидационного набора выглядел бы следующим образом:

1. Оптимизировать параметры b, используя обучающий набор для каждой степени полинома.
2. Найти степень полинома d с наименьшей ошибкой, используя тестовый набор.
3. Оценить ошибку обобщения, также используя тестовый набор с, (d = theta от полинома с более низкой ошибкой).

В этом случае мы обучили одну переменную d или степень полинома, используя тестовый набор. Это приведет к тому, что наше значение ошибки будет больше для любого другого набора данных.

Подобные переменные, не являющиеся параметрами гипотезы, но так или иначе влияющие на точность нашей модели называются гиперпараметрами модели. Обучение гиперпараметров - более сложный и долгий путь и он требует проведения специальной процедуры.

Вариант 2

Для решения проблемы, описанной выше, мы можем ввести третий набор, валидационный набор, чтобы использовать его как промежуточный набор для обучения гиперпараметров. Тогда наш тестовый набор даст нам точную, не оптимистичную ошибку.

Один из возможных примеров разбивки нашего набора данных на три набора:

Обучающий набор (train set): первые 60%; Валидационный набор (validation set): следующие 20%; Тестовый набор (test set): последние 20%.

Теперь мы можем вычислить три отдельных значения ошибки для трех разных наборов.

Метод выбора гипотезы с помощью валидационного набора (этот метод предполагает, что мы не используем регуляризацию и на валидационном наборе)

1. Оптимизировать параметры b, используя набор тренировок для каждой степени полинома.
2. Найти степень полинома d с наименьшей ошибкой, используя валидационный набор.
3. Оценить ошибку обобщения, используя тестовый набор с (где b - параметры от полинома с более низкой ошибкой);

Таким образом, степень полинома d не была обучена с использованием тестового набора.

Использование валидационного набора для выбора d означает, что мы также не можем использовать его для процесса проверки правильности установки значения лямбда.

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

В задаче классификации данные называются несбалансированными (Imbalanced Data), если в обучающей выборке доли объектов разных классов существенно различаются, также говорят, что «классы не сбалансированы». Есть понятие несбалансированности и для задач регрессии, но там оно граничит с наличием аномалий в данных. Важно не надо путать дисбаланс с разреженностью.

При исследовании проблемы важно провести анализ имеющихся данных. Например, очень часто к дисбалансу приводит наличие дубликатов. Также часто дисбаланс возникает в задаче с очень большим числом классов и некоторые классы малы по естественным причинам (например это генотипы представителей малых народов).

При выборе пути решения важно опираться на природу задачи: в чём причина дисбаланса, сколько классов, насколько серьёзный дисбаланс, какими данными и мета-данными мы располагаем.

Варианты решения проблемы несбалансированных классов:

* Удаление лишних данных из обучающей выборки
* Ресемплирование
* Добавление других объектов
* Придание весов классам
* Иерархические классификаторы
* Ансамблевые модели
* Анализ ошибок
* Пересмотр постановки задачи
* Изменение функции ошибки

## 38. Проблема отсутствующих данных: причины, исследование, пути решения.

Одна из самых распространенных проблем в данных, использующихся в машинном обучении - наличие отсутствующих измерений. Данные с пропусками нельзя без предварительной обработки подавать на вход почти никаких обучаемых моделей, поэтому практически непременным этапом предварительного анализа данных является выявление отсутствующих значений.

Причинами отсутствующих данных могут быть ошибки ввода данных, сокрытие информации, фрод.

Исследование на предмет отсутствующих значений можно провести за счет проверки данных с помощью агрегации - data.isnull().sum(); проверки данных с помощью визуализации (столбчатые диаграммы, дендрограммы).

Основные методы борьбы с пропусками значений:

* Удаление признака
* Удаление объекта
* Заполнение средним значением
* Заполнение медианным, модальным, средним по логарифмической шкале
* Заполнение групповым средним
* Заполнение специальным значением
* Заполнение случайным значением
* Введение нового признака

Важно выбирать метод основываясь на природе задачи. Неудачный выбор метода заполнения пропусков может не только не улучшить, но и сильно ухудшить результаты.

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

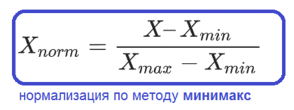
Нормализация (normalization) и стандартизация (standardization) являются методами изменения диапазонов значений – решкалирования.

Нормализация — это преобразование данных к неким безразмерным единицам. Иногда — в рамках заданного диапазона, например, [0..1] или [-1..1]. Иногда — с какими-то заданным свойством, как, например, стандартным отклонением равным 1.

Ключевая цель нормализации — приведение различных данных в самых разных единицах измерения и диапазонах значений к единому виду, который позволит сравнивать их между собой или использовать для расчёта схожести объектов.

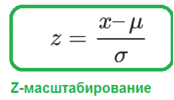
**Методы нормализации:**

* Минимакс – линейное преобразование данных в диапазоне [0..1], где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно (реализовано в sklearn.preprocessing.MinMaxScaler)
* Десятичное масштабирование путем удаления десятичного разделителя значения переменной.



**Стандартизация** – техника преобразования значений признака, адаптирующая признаки с разными диапазонами значений к моделям, использующих дистанцию для прогнозирования.

**Z-масштабирование** - стандартизация данных на основе среднего значения и стандартного отклонения: деление разницы между переменной и средним значением на стандартное отклонение



Такое преобразование необходимо, поскольку признаки датасета могут иметь большие различия между своими диапазонами, и для моделей, основанных на вычислении дистанции между точками на графике как основу прогнозирования (Метод k-ближайших соседей (kNN), Метод опорных векторов (SVM), Дерево решений (Decision Tree)) это спровоцирует искаженное восприятие данных.

## 40. Шкалы измерения признаков. Виды шкал, их характеристика.

На первоначальном этапе знакомства с набором данных для подготовки к обучению моделей очень важно понимать шкалы, по которым измеряется каждый фактор. Принципиальное различие в методах работы с категориальными и числовыми, дискретными и непрерывными признаками обосновывает необходимость получить эту информацию как можно раньше.

Шкалы:

* Категориальная шкала - это шкала измерения, которая используется для идентификации. Она присваивает номера атрибутам для удобства идентификации, но может использоваться только как метка.
* Порядковая шкала - предполагает ранжирование (упорядочивание) значений переменной в зависимости от масштабирования. Атрибуты в порядковой шкале обычно располагаются в порядке возрастания или убывания. Она использует квалификаторы, такие как «очень», «высоко», «больше», «меньше» и т. д.
* Целочисленная шкала (шкала интервалов) - это шкала, в которой уровни упорядочены, а интервалы между ними равны. Целочисленная шкала не только позволяет однозначно определить, какое значение больше (меньше), но и на сколько.
* Абсолютная шкала (шкала отношений) - является «наивысшим» уровнем представления данных. Эта шкала отличается от целочисленной шкалы только тем, что в ней строго определено положение нулевой точки. Благодаря этому шкала отношений не накладывает никаких ограничений на математический аппарат, используемый для обработки результатов наблюдений.

# Практика

1. Очистка данных и обучение моделей. Датасет: <https://www.kaggle.com/akshayksingh/kidney-disease-dataset>

2. Очистка данных и обучение моделей. Датасет: <https://www.kaggle.com/elikplim/forest-fires-data-set>

3. Описательный анализ и визуализация данных. Датасет: <https://www.kaggle.com/shebrahimi/financial-distress>

4. Описательный анализ и визуализация данных. Датасет: <https://www.kaggle.com/akshayksingh/kidney-disease-dataset>

5. Построение модели и оптимизация гиперпараметров. Датасет: <https://www.kaggle.com/kaushiksuresh147/customer-segmentation>

6. Построение модели и оптимизация гиперпараметров. Датасет: <https://www.kaggle.com/elikplim/forest-fires-data-set>

7. Выбор признаков. Датасет: <https://www.kaggle.com/amir75/caesarean-section-classification>

8. Выбор признаков. Датасет: <https://www.kaggle.com/amir75/caesarean-section-classification>

9. Исследование влияния обучения без учителя на эффективность обучения. Датасет: <https://www.kaggle.com/veer06b/marrket-mix-dataset>

10. Исследование влияния обучения без учителя на эффективность обучения. Датасет: <https://www.kaggle.com/sachinsharma1123/performance-prediction>

# Дополнительная информация

**Для каждого задания будет необходимо:**

1. Загрузить датасет в Python.
2. Описать набор данных и решаемую задачу.
3. Выделить целевую переменную и факторные переменные.
4. Удалить ненужные данные, проанализировать отсутствующие значения.
5. Прокомментировать количественные параметры датасета.
6. Разбить выборку на обучающую и тестовую.

**Работа по вариантам.**

Вариант 1. Очистка данных и обучение моделей.

Данный вариант предполагает фокусировку на обучении нескольких

видов моделей обучения с учителем. В зависимости от набора

данных, может предполагаться задача классификации и регрессии.

Необходимо после минимальной подготовки датасета к обучению

обучить несколько моделей и сравнить их эффективность.

Вариант 2. Описательный анализ и визуализация данных.

Данный вариант предполагает фокусировку на исследовании данных

и визуализации. При решении этого варианта следует провести как

можно более подробный описательный анализ данных с

использованием максимального спектра средств визуализации. При

этом следует делать значимые выводы об обнаруженных в данных

закономерностях.

Вариант 3. Построение модели и оптимизация гиперпараметров.

Данный вариант предполагает фокусировку на процессе улучшения

эффективности модели обучения с учителем. Студенту следует

подготовить датасет к обучению, обучить одну из моделей с учителем

со значениями гиперпараметров по умолчанию, получить значение

эффективности. После этого вручную или автоматически подобрать

значения гиперпараметров таким образом, чтобы получить

максимальный прирост эффективности.

Вариант 4. Выбор признаков.

Данный вариант предполагает фокусировку на улучшении модели

путем ввода новых признаков в модель. Следует подготовить модель

к обучению, обучить модель и зафиксировать начальный уровень

эффективности. Затем следует исследовать влияние исключения

существующих и введения новых признаков в модель на

эффективность. Как вариант можно рассматривать введение

полиномиальных признаков. Следует стремиться к максимальному

увеличению эффективности модели.

Вариант 5. Исследование влияния обучения без учителя на

эффективность обучения.

Данный вариант предполагает фокусировку на использовании

методов обучения без учителя для ускорения или повышения

эффективности обучения с учителем. Следует подготовить модель к

обучению, обучить модель и зафиксировать начальный уровень

эффективности. Затем следует попробовать применить понижение

размерности, обнаружение аномалий или кластеризацию (в любой

комбинации) для трансформации исходного датасета. В конце работы

следует сделать значимый вывод об изменении скорости и

эффективности обучения с учителем.

\*Каждый билет состоит из одного теоретического вопроса (20 баллов) и одного практического вопроса (40 баллов), в сумме дают 60 баллов.