# [Основы машинного обучения на Python - Book.ru](https://reader.new.book.ru/?t=eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo5ODY2MjMsImdyb3VwX2lkIjozMDYsImJvb2tfaWQiOjk1Mjc1MSwiYm9va19hY2Nlc3MiOjEsInVzZXJfZW1haWwiOiIyMjI3MThAZWR1LmZhLnJ1IiwidXNlcl90eXBlIjoxLCJleHAiOjE3MTcyODMwODEsImlhdCI6MTcxNzI2MTQ1MX0.AyOOW1yZN_Wy7Y2O56EncPQIA9w-hJMb5_VfVm295YzVAjzMYB2SsSizNqzAAhAlQUpyDPdqXcOZpJewghWzKQ&v=0)

Написать до 15 июня

**чёрный - Макс**

**красный - Аня**

**синий - Алёна**

**зелёный - Таня**

**фиолетовый - Егор**

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

Машинное обучение - это численная оптимизация параметри-

ческих моделей для описания определенного набора данных.

Отличительная черта машинного обучения, которая выделяет

его среди других методов искусственного интеллекта - это то, что

модели машинного обучения работают на основе подстройки к

определенному набору данных. Этот набор должен быть предвари-

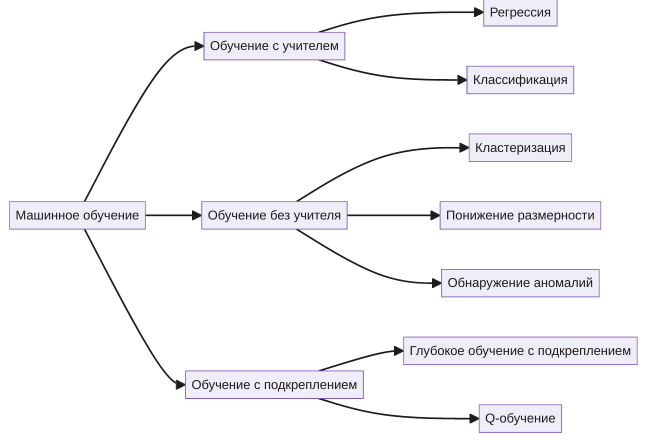
тельно собран, приведен к определенному виду, и, желательно, про-

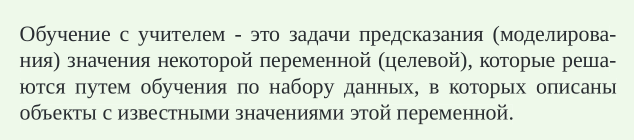
анализирован. Поэтому машинное обучение, как никакая другая

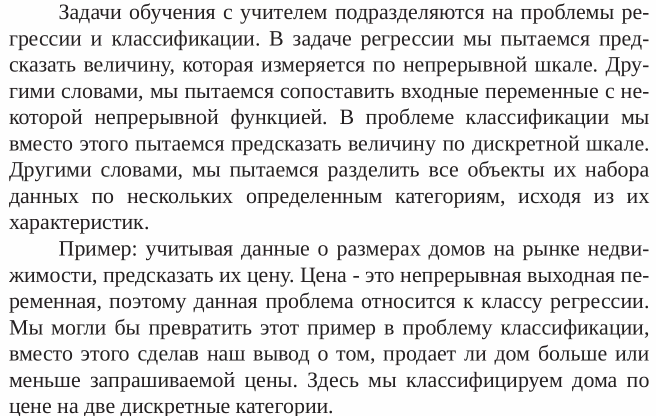
сфера искусственного интеллекта, работает с данными и связана с

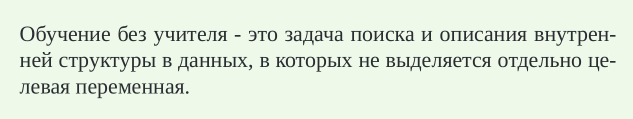
такой областью человеческой деятельности, как науки о данных.

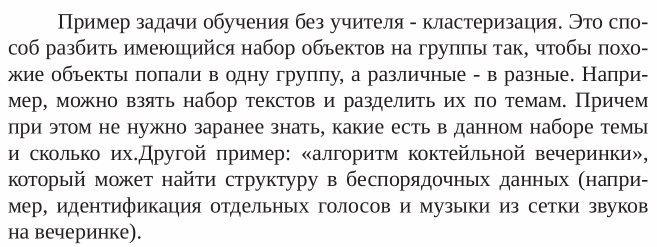
# **Классификация задач машинного обучения. Примеры задач из различных классов.**











# **Основные понятия машинного обучения: набора данных, объекты, признаки, атрибуты, модели, параметры.**

Набор данных - это набор информации, который используется для обучения модели машинного обучения. Он состоит из объектов и их признаков.

Объекты - это конкретные элементы данных, например, отдельные наблюдения, изображения или текстовые документы. Каждый объект имеет набор признаков.

Признаки, также называются атрибутами, это свойства или характеристики объектов, которые описывают их. Например, если объект - это изображение, признаками могут быть цвета пикселей или текстуры.

Модели - это алгоритмы или структуры данных, которые используются для обучения на наборе данных и предсказания результатов для новых данных. Модель может быть обучена распознавать закономерности или шаблоны в данных.

Параметры - это переменные, которые определяют поведение модели. В процессе обучения модель настраивает параметры, чтобы минимизировать ошибку предсказания. Например, веса в нейронной сети являются параметрами.

# Структура и представление данных для машинного обучения.

Структура данных является важным компонентом машинного обучения, и правильная структура данных может помочь в достижении более быстрой обработки, более легкого доступа к данным и более эффективного хранения.

Вот некоторые часто используемые структуры данных для машинного обучения −

### Массивы

Массивы представляют собой набор похожих типов данных, к которым можно получить доступ с помощью индекса. Они обычно используются в машинном обучении для хранения данных в виде таблиц, таких как CSV-файлы. Массивы просты в использовании и обеспечивают быструю индексацию, но их размер фиксирован, что может быть ограничением при работе с большими наборами данных.

### Списки

Списки представляют собой наборы разнородных типов данных, к которым можно получить доступ с помощью итератора. Они обычно используются в машинном обучении для хранения сложных структур данных, таких как вложенные списки, словари и кортежи. Списки обладают гибкостью и могут обрабатывать данные разного размера, но они работают медленнее, чем массивы, из-за необходимости итерации.

### Словари

Словари представляют собой набор пар ключ-значение, к которым можно получить доступ с помощью ключей. Они обычно используются в машинном обучении для хранения метаданных или меток, связанных с данными. Словари обеспечивают быстрый доступ к данным и полезны для создания справочных таблиц, но они могут занимать много памяти при работе с большими наборами данных.

### Связанные списки

Связанные списки представляют собой наборы узлов, каждый из которых содержит элемент данных и ссылку на следующий узел в списке. Они обычно используются в машинном обучении для хранения последовательных данных и манипулирования ими, таких как данные временных рядов. Связанные списки обеспечивают эффективные операции вставки и удаления, но они медленнее, чем массивы и списки, когда дело доходит до доступа к данным.

### Деревья

Деревья - это иерархические структуры данных, которые обычно используются в машинном обучении для алгоритмов принятия решений, таких как деревья решений и случайные леса. Деревья предлагают эффективные алгоритмы поиска и сортировки, но они могут быть сложными в реализации и страдать от переоснащения.

### Графы

Графы представляют собой наборы узлов и ребер, которые обычно используются в машинном обучении для представления сложных взаимосвязей между точками данных. Графы предлагают мощные алгоритмы кластеризации, классификации и прогнозирования, но они могут быть сложными в реализации и иметь проблемы с масштабируемостью.

В дополнение к вышеупомянутым структурам данных, многие библиотеки и фреймворки машинного обучения предоставляют специализированные структуры данных для конкретных случаев использования, такие как матрицы и тензоры для глубокого обучения. Важно выбрать правильную структуру данных для решаемой задачи с учетом таких факторов, как размер данных, скорость обработки и использование памяти.

Ниже приведены некоторые способы использования структур данных в машинном обучении −

### Хранение данных и доступ к ним

Алгоритмы машинного обучения требуют больших объемов данных для обучения и тестирования. Структуры данных, такие как массивы, списки и словари, используются для эффективного хранения данных и доступа к ним. Например, массив может использоваться для хранения набора числовых значений, в то время как словарь может использоваться для хранения метаданных или меток, связанных с данными.

### Предварительная обработка данных

Перед обучением модели машинного обучения необходимо предварительно обработать данные для их очистки, преобразования и нормализации. Структуры данных, такие как списки и массивы, могут использоваться для хранения данных и манипулирования ими во время предварительной обработки. Например, список может использоваться для фильтрации пропущенных значений, в то время как массив может использоваться для нормализации данных.

### Создание векторов признаков

Векторы объектов являются важным компонентом моделей машинного обучения, поскольку они представляют объекты, которые используются для составления прогнозов. Структуры данных, такие как массивы и матрицы, обычно используются для создания векторов объектов. Например, массив может использоваться для хранения значений пикселей изображения, в то время как матрица может использоваться для хранения частотного распределения слов в текстовом документе.

### Построение деревьев решений

Деревья решений - это распространенный алгоритм машинного обучения, который использует древовидную структуру данных для принятия решений на основе набора входных характеристик. Деревья решений полезны для задач классификации и регрессии. Они создаются путем рекурсивного разделения данных на основе наиболее информативных характеристик. Древовидная структура данных упрощает процесс принятия решений и делает прогнозы.

### Построение графов

Графы используются в машинном обучении для представления сложных взаимосвязей между точками данных. Структуры данных, такие как матрицы смежности и связанные списки, используются для создания графов и управления ими. Графы используются для кластеризации, классификации и задач прогнозирования.

# **Инструментальные средства машинного обучения**

Инструментальные средства машинного обучения представляют собой программные инструменты и библиотеки, которые упрощают задачи машинного обучения, такие как преобразование данных, выбор модели и настройка гиперпараметров.

Базовые инструменты, которые используются для обращения с данными в проектах по машинному обучению - библиотеки numpy, pandas, средства визуализации matplotlib, seaborn.

Программные инструменты, такие как **TensorFlow, PyTorch, scikit-learn**, являются широко используемыми для разработки и обучения моделей машинного обучения. Они предоставляют различные алгоритмы машинного обучения, удобный интерфейс для работы с данными и возможности для настройки моделей.

Для работы с данными и построения моделей машинного обучения часто применяются библиотеки программирования, такие как **NumPy, Pandas и Matplotlib**. Они предоставляют мощные инструменты для работы с математическими операциями, обработки данных и визуализации результатов

Инструменты визуализации данных играют важную роль в процессе машинного обучения, поскольку позволяют визуализировать и анализировать данные, обнаруживать закономерности и паттерны, а также визуализировать результаты моделей

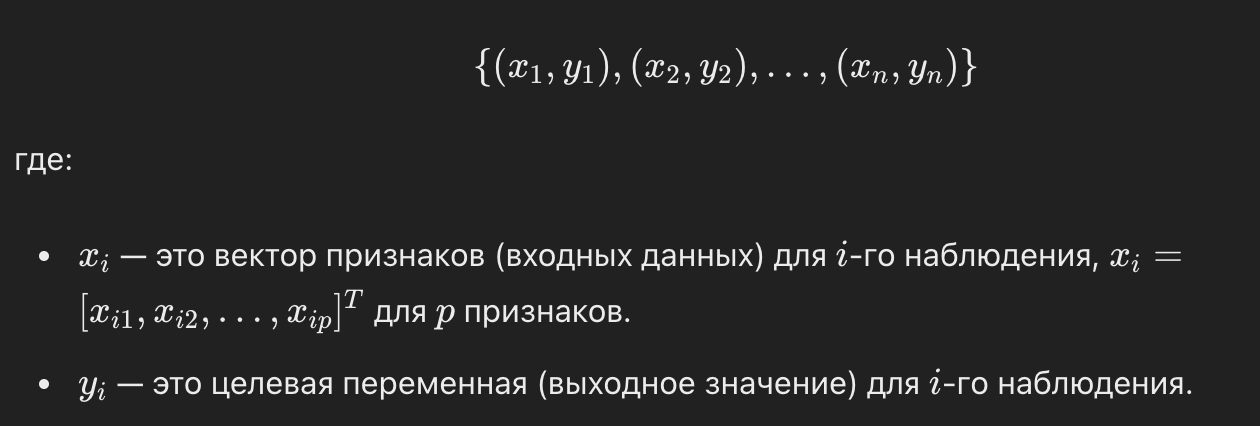
# **Задача регрессии: постановка, математическая формализация.**

#### **Постановка задачи**

Задача регрессии в машинном обучении заключается в том, чтобы предсказать непрерывную целевую переменную yyy на основе одного или нескольких входных признаков XXX. Примеры задач регрессии включают предсказание цены дома на основе его характеристик, предсказание температуры на следующий день на основе погодных данных и т.д.

#### **Математическая формализация**

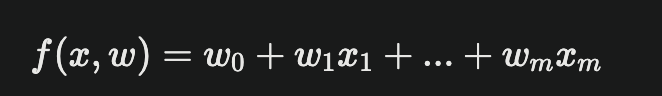
Пусть у нас есть обучающая выборка, состоящая из n наблюдений:



Наша цель — построить функцию f, которая предсказывает значение y на основе входных данных xxx:

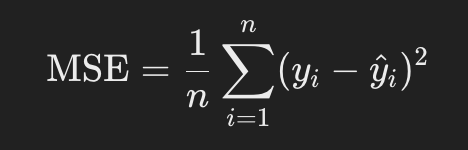
#### **Линейная регрессия**

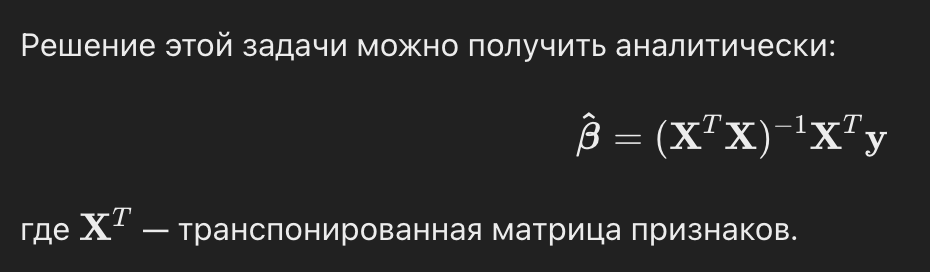
Одним из простейших и наиболее часто используемых методов регрессии является линейная регрессия. В линейной регрессии предполагается, что зависимость между входными признаками и целевой переменной можно аппроксимировать линейной функцией:



#### **Критерий качества модели**

Для оценки качества модели регрессии часто используется среднеквадратичная ошибка (Mean Squared Error, MSE):



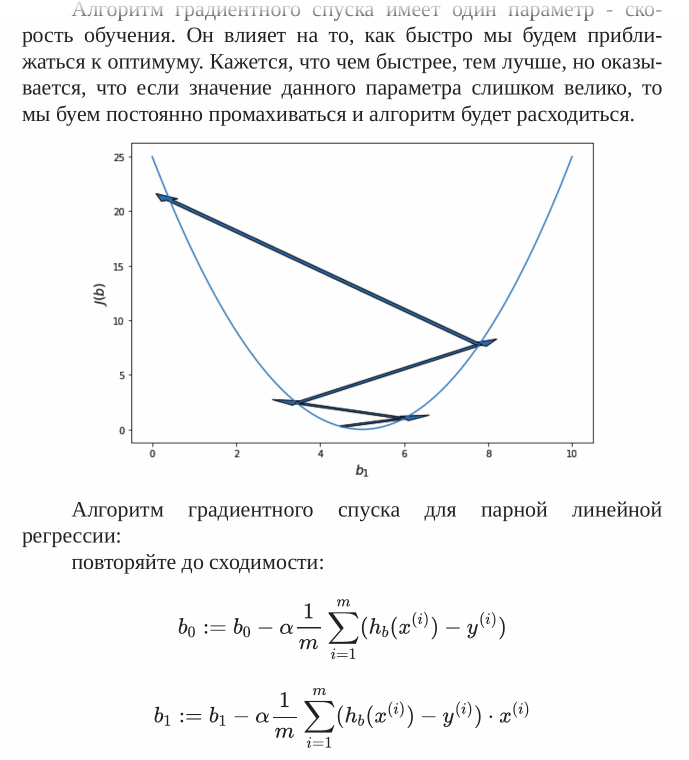


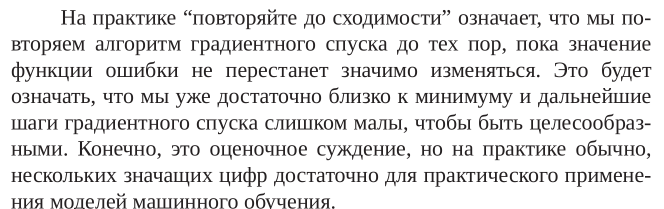
Бетта штрих - матрица коэффициентов

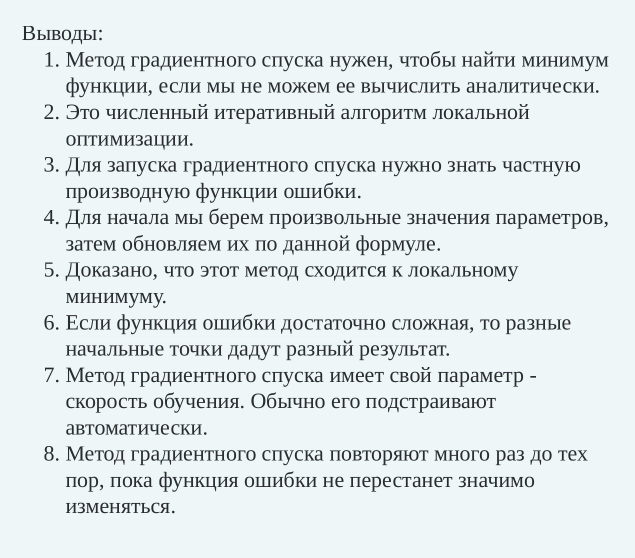
# **Метод градиентного спуска для парной линейной регрессии.**

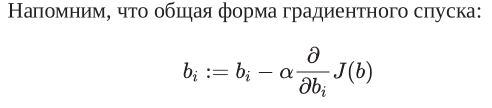
Градиентный спуск - это итеративный алгоритм оптимизации, который пытается найти оптимальное значение (минимальное / максимальное) целевой функции. Это один из наиболее часто используемых методов оптимизации в проектах машинного обучения для обновления параметров модели с целью минимизации функции затрат.

Основная цель градиентного спуска - найти наилучшие параметры модели, которые обеспечивают наивысшую точность при обучении, а также при тестировании наборов данных. В градиентном спуске градиент - это вектор, который указывает в направлении наибольшего увеличения функции в определенной точке. Движение в направлении, противоположном градиенту, позволяет алгоритму постепенно спускаться к более низким значениям функции и в конечном итоге достигать минимума функции.









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

Функция ошибки является важным понятием в машинном обучении. Она представляет собой метрику для измерения того, насколько хорошо модель работает на задаче обучения. Цель функции ошибки - минимизировать разницу между прогнозируемыми значениями и реальными данными.

Критерии для хорошей функции ошибки включают в себя следующие требования:

1. Должна быть дифференцируемой - это позволяет использовать методы оптимизации для нахождения оптимальных параметров модели.

2. Должна быть неотрицательной - так как ошибка не может быть отрицательной величиной.

3. Должна быть монотонно возрастающей - чем ближе прогноз к реальным данным, тем меньше должна быть ошибка.

Некоторые распространенные функции ошибки включают в себя:

1. Среднеквадратичную ошибку (Mean Squared Error, MSE) - сумма квадратов разности между прогнозируемыми значениями и реальными данными, деленная на количество наблюдений.

2. Средняя абсолютная ошибка (Mean Absolute Error, MAE) - сумма абсолютных разностей между прогнозируемыми значениями и реальными данными, деленная на количество наблюдений.

3. Кросс-энтропия - используется в задачах классификации и измеряет расхождение между предсказанными вероятностями и реальными метками классов.

Функция ошибки является ключевым компонентом процесса обучения моделей машинного обучения, поэтому важно выбирать подходящую функцию для конкретной задачи и модели.

# **Множественная и нелинейная регрессии.**

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

х(i)- вектор-столбец всех значений признаков і-го обучающего примера;

хj(i) - значение ј-го признака і-го обучающего примера;

хj - вектор ј-го признака всех обучающих примеров;

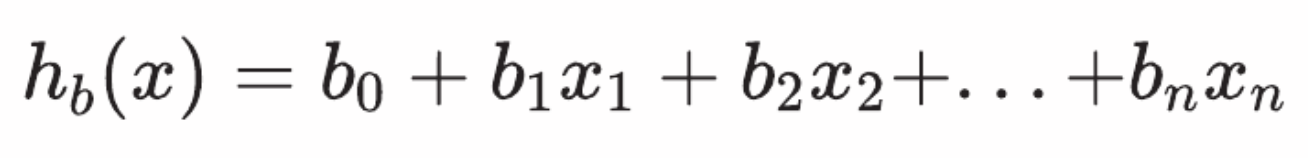
m - количество примеров в обучающей выборке;

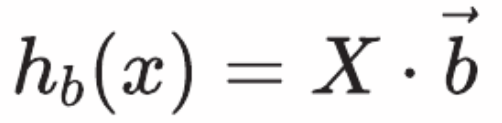
n - количество признаков;

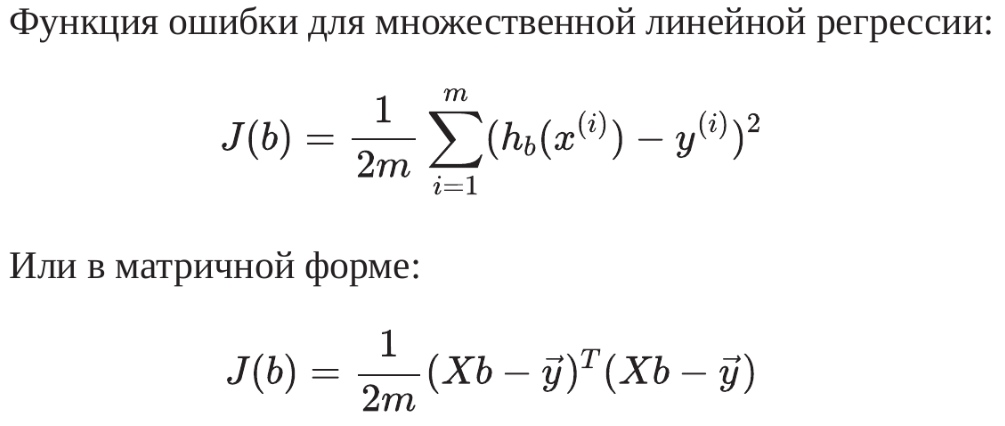
Х - матрица признаков;

b - вектор параметров регрессии.

Общий вид модели множественной линейной регрессии:

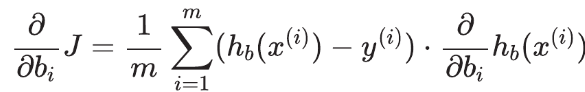


Или в матричной форме: 



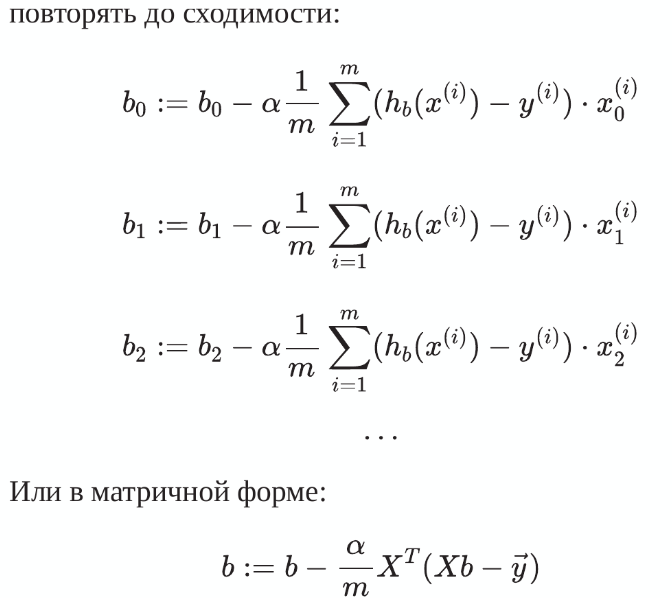
Обратите внимание, что мы специально не раскрываем выражение hb(x(i)). Это нужно, чтобы подчеркнуть, что форма функции ошибки не зависит от функции гипотезы, она выражается через нее.

Теперь нам нужно взять производную этой функции ошибки. Здесь уже нужно знать производную самой функции гипотезы, так как:



В такой формулировке мы представляем частные производные функции ошибки (градиент) через частную производную функции гипотезы. Это так называемое моделе-независимое представление градиента. Ведь для этой формулы совершенно неважно, какой функцией будет наша гипотеза. Пока она является дифференцируемой, мы можем использовать градиент ее функции ошибки. Именно

поэтому метод градиентного спуска работает с любыми аналитическими моделями, и нам не нужно каждый раз заново "переизобретать” математику градиентного спуска, адаптировать ее к каждой конкретной модели машинного обучения. Достаточно изучить этот метод один раз, в общей форме. Метод градиентного спуска для множественной регрессии определяется следующими уравнениями:



Выводы:

1. Множественная регрессия очень похожа на парную, но с большим количеством признаков.

2. Для удобства и однообразия, почти всегда обозначают xo =1.

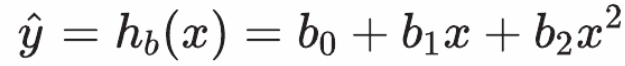
3. Признаки образуют матрицу, поэтому уравнения множественной регрессии часто приводят в матричной форме, так короче.

4. Алгоритм градиентного спуска для множественной регрессии точно такой же, как и для парной.

**Полиномиальная регрессия или нелинейная регрессия.**

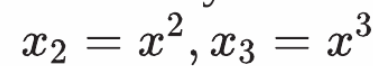
Функция гипотезы не обязательно должна быть линейной, если это не соответствует данным. На практике вы не всегда будете иметь данные, которые можно хорошо аппроксимировать линейной функцией.

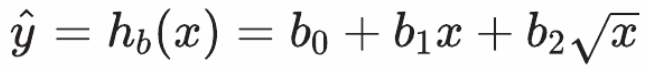
Например, если наша функция гипотезы , то мы можем добавить еще один признак, основанный на х1, получив квадратичную функцию



или кубическую:



В кубической функции мы по сути ввели два новых признака: . Точно таким же образом, мы можем создать, например, такую функцию:



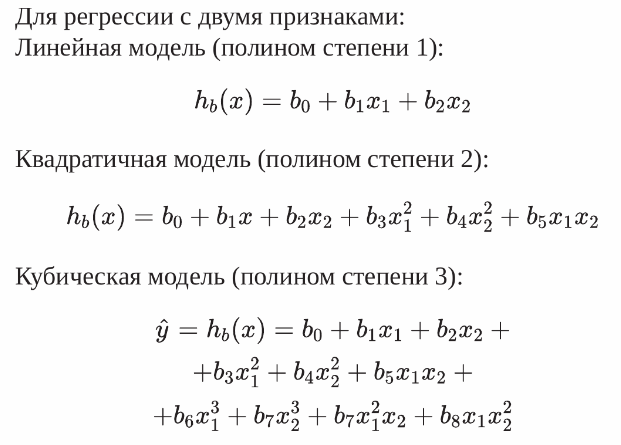
В любом случае, мы из парной линейной функции сделали какую-то другую функцию. И к этой нелинейной функции можно относиться по разному. С одной стороны, это другой класс функций, который обладает нелинейным поведением, а следовательно, может описывать более сложные зависимости в данных. С другой стороны, это линейная функция от нескольких переменных. Только сами эти переменные оказываются в функциональной зависимости друг от друга. Но никто не говорил, что признаки должны быть независимы. И вот такое представление нелинейной функции как множественной линейной позволяет нам без изменений воспользоваться алгоритмом градиентного спуска для множественной линейной регрессии. Только вместо х2, х3, ... , хn нам нужно будет подставить соответствующие функции от х1.

Очевидно, что нелинейных функций можно придумать бесконечное количество. Поэтому встает вопрос, как выбрать нужный класс функций для решения конкретной задачи. В случае парной регрессии мы можем взглянув на график точек обучающей выборки сделать предположение о том, какой вид нелинейной зависимости

связывает входную и целевую переменные. Но если у нас множество признаков, просто так проанализировать график нам не удастся. Поэтому по умолчанию используют полиномиальную регрессию, когда в модель добавляют входные переменные второго, третьего, четвертого и так далее порядков.

Порядок полиномиальной регрессии подбирается в качестве компромисса между качеством получаемой регрессии, и вычислительной сложностью. Ведь чем выше порядок полинома, тем более сложные зависимости он может аппроксимировать. И вообще, чем выше степень полинома, тем меньше будет ошибка при прочих равных. Если степень полинома на единицу меньше количества точек - ошибка будет нулевая. Но одновременно с этим, чем выше степень полинома, тем больше в модели параметров, тем она сложнее и занимает больше времени на обучение. Есть еще вопросы переобучения, но про это мы поговорим позднее.

А что делать, если изначально в модели было несколько признаков? Тогда обычно для определенной степени полинома берутся все возможные комбинации признаком соответствующей степени и ниже. Например:



При этом количество признаков и, соответственно, количество параметров растет экспоненциально с ростом степени полинома. Поэтому полиномиальные модели обычно очень затратные в обучении при больших степенях. Но полиномы высоких степеней более универсальны и могут аппроксимировать более сложные данные

лучше и точнее.

Выводы:

1. Данные в датасете не всегда располагаются так, что их хорошо может описывать линейная функция.

2. Для описания нелинейных зависимостей нужна более сложная, нелинейная модель.

3. Чтобы не изобретать алгоритм обучения заново, можно просто ввести в модель суррогатные признаки.

4. Суррогатный признак - это новый признак, который считается из существующих атрибутов.

5. Чаще всего используют полиномиальную регрессию - это когда в модель вводят полиномиальные признаки - степени существующих атрибутов.

6. Обычно берут все комбинации факторов до какой-то определенной степени полинома.

7. Полиномиальная регрессия может аппроксимировать любую функцию, нужно только подобрать степень полинома.

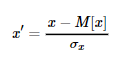
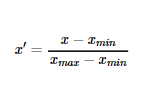
8. Чем больше степень полиномиальной регрессии, тем она сложнее и универсальнее, но вычислительно сложнее (экспоненциально).

# **Нормализация признаков в задачах регрессии**

# **Нормализация** — это приведение всех значений признака к новому диапазону. Например, к диапазону [0, 1]. Это полезно, поскольку значения признаков могут изменяться в очень большом диапазоне. Причем, значения разных признаков могут отличаться на несколько порядков. А после нормализации они все будут находиться в узком (и, часто, едином) диапазоне. Собственно, нормализации подлежат только признаки, выраженные в числовой форме.

Проще говоря, нормализация необходима для корректной работы модели машинного обучения, чтобы не было искажения результатов из-за разного масштаба разных признаков.

**Формула:**

(Считается по признаку)

**Нормализация нужна:**

* Нормализация нужна для ускорения метода градиентного спуска.
* Есть два основных метода нормализации - минимаксная и стандартизация.
* Параметры нормализации высчитываются по обучающей выборке.
* Нормализация встроена в большинство библиотечных методов.
* Некоторые методы более чувствительны к нормализации, чем другие.
* Нормализацию лучше сделать, чем не делать.

**В коде:**

**from sklearn.preprocessing import MinMaxScaler**

**scaler = MinMaxScaler()**

**features\_scaled = scaler.fit\_transform(features)**

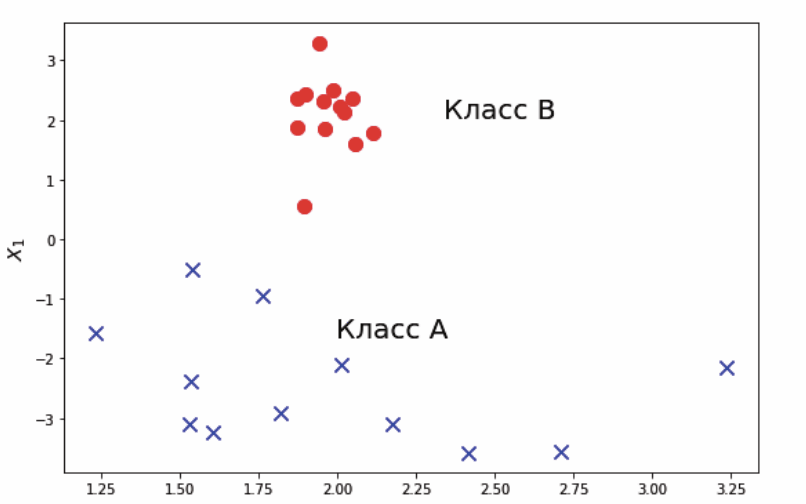
**from sklearn.preprocessing import StandardScaler**

**scaler = StandardScaler()**

**features\_scaled = scaler.fit\_transform(features)**

**(Выше формулы представлены соответственно для этих 2х видов нормализации)**

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



Задача классификации — это задача, в которой множество объектов (ситуаций) необходимо разделить на классы. При этом задано конечное множество объектов, для которых известно, к каким классам они относятся (выборка), но классовая принадлежность остальных объектов неизвестна. Для решения задачи требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества, то есть указать, к какому классу он относится.

## Примеры задач классификации

* Бинарная классификация: задача, в которой объекты делятся на два класса. Примеры включают обнаружение спама в электронной почте (спам или нет), прогнозирование оттока клиентов (отток или нет), прогноз конверсии (купит или нет).
* Мультиклассовая классификация: задача, в которой объекты делятся на более чем два класса. Примеры включают классификацию лиц, классификацию видов растений, оптическое распознавание символов.
* Классификация по нескольким меткам: задача, в которой объекты могут принадлежать нескольким классам одновременно. Примером может быть классификация объектов на фотографии, где изображение может содержать несколько объектов, таких как «велосипед», «яблоко», «человек»

## Математическая формализация задачи классификации

## Основные понятия

Объекты: множество объектов 𝑋={𝑥1,𝑥2,…,𝑥𝑛}

Классы: множество классов

Обучающая выборка: множество пар

𝐷={(𝑥𝑖,𝑦𝑖)}𝑖=1𝑁

## Цель задачи

Цель задачи классификации — построить функцию

𝑓:𝑋→𝑌 которая для любого объекта 𝑥∈𝑋 предсказывает его класс 𝑦∈𝑌

## Примеры моделей классификации

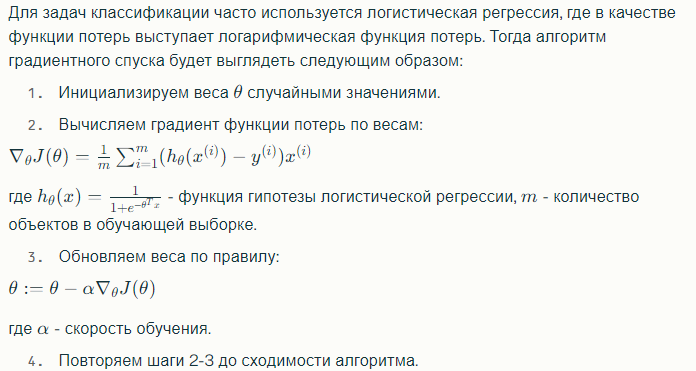
* Линейный классификатор: Модель, в которой разделяющая поверхность между классами является гиперплоскостью. Примером является логистическая регрессия.
* Наивный байесовский классификатор: Простой вероятностный классификатор, основанный на применении теоремы Байеса с предположением о независимости признаков.
* Искусственные нейронные сети: Мощные модели, способные решать сложные задачи классификации за счет многослойной архитектуры и нелинейных активационных функций.

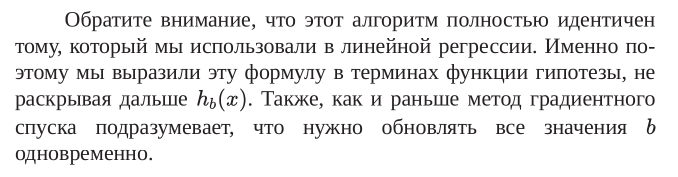
# **Метод градиентного спуска для задач классификации.**

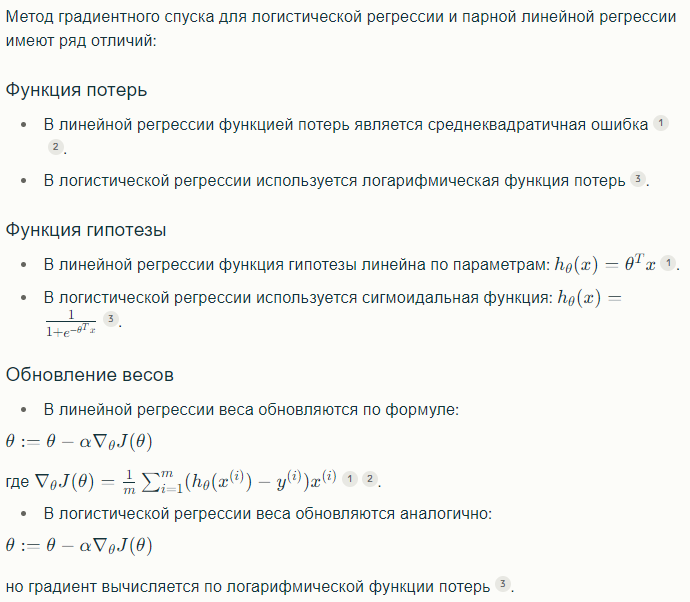
Градиентный спуск - это итеративный алгоритм оптимизации, который пытается найти оптимальное значение (минимальное / максимальное) целевой функции. Это один из наиболее часто используемых методов оптимизации в проектах машинного обучения для обновления параметров модели с целью минимизации функции затрат.

Основная цель градиентного спуска - найти наилучшие параметры модели, которые обеспечивают наивысшую точность при обучении, а также при тестировании наборов данных. В градиентном спуске градиент - это вектор, который указывает в направлении наибольшего увеличения функции в определенной точке. Движение в направлении, противоположном градиенту, позволяет алгоритму постепенно спускаться к более низким значениям функции и в конечном итоге достигать минимума функции.

Метод градиентного спуска также может быть применен для решения задач классификации. Основная идея заключается в минимизации функции потерь, которая зависит от разницы между предсказанными классами и истинными классами объектов обучающей выборки







# **Логистическая регрессия в задачах классификации.**

**Логистическая регрессия** — это популярный метод машинного обучения для решения задач бинарной классификации, то есть когда объекты нужно разделить на два класса. Она позволяет построить модель, которая по входным данным предсказывает вероятность принадлежности объекта к одному из двух классов.

Основные особенности логистической регрессии:

* Возвращает вероятность принадлежности к классу, а не просто метку класса. Это позволяет калибровать классификатор и получать более корректные оценки вероятностей.
* Использует логистическую функцию потерь для оценки качества предсказаний вероятностей.
* Проста в реализации и эффективна по сравнению с другими классификаторами, что делает её одной из самых популярных моделей.

Логистическая регрессия применяется в различных областях, где требуется решить задачу бинарной классификации, например:

* Определение вероятности отклика на рекламу
* Оценка риска невозврата кредита
* Классификация спама в почте
* Распознавание рукописного текста

*Выводы:*

1. Логистическая регрессия - это самый простой алгоритм

бинарной классификации.

2. Можно взять регрессионную модель и ввести пороговое

значение.

3. Обычная регрессия плохо работает в задачах

классификации за счет своей чувствительности и

неограниченности.

4. Метод логистической регрессии основан на применении

логистической или сигмоидной функции.

5. Результат работы логистической функции часто

интерпретируется как вероятность отнесения объекта к

положительному классу.

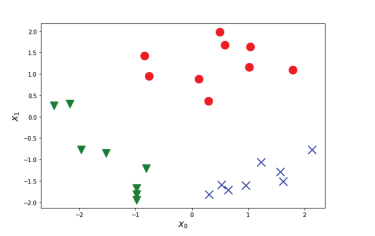
6. Для четкой классификации обычно выбирают некоторое

пороговое значение, обычно - 0,5.

# **Множественная и многоклассовая классификация. Алгоритм “один против всех”.**

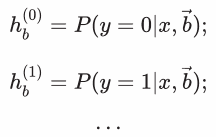
Теперь мы рассмотрим классификацию данных более чем в двух категориях. Вместо у = {0,1} мы расширим наше определение так, чтобы y = {0,1 ... n}.

Пример набора данных для задачи множественной классификации, содержащих два признака и три класса можно увидеть на рисунке:



Алгоритм классификации в данном случае очень прост. Мы берем последовательно каждый имеющийся класс в данных, делаем его "положительным", а все остальные - "отрицательными", и обучаем модель, которая стремится отделить данный класс от остальных.

В этом случае мы делим нашу задачу на n + 1 (потому что индекс начинается с 0) бинарных задач классификации. В каждом из них мы прогнозируем вероятность того, что у является членом одного из наших классов. То есть мы обучаем сразу множество моделей, столько, сколько у нас есть классов. Так что на каждый класс в задаче будет своя собственная модель, которая для определенного объекта выдает вероятность принадлежности этого объекта к соответствующему классу. Для каждого конкретного объекта все модели выдают такой вектор вероятностей:





Другими словами, построив несколько моделей бинарной классификации мы можем использовать их, чтобы получить оценки вероятности принадлежности любого объекта ко всем имеющимся классам. После этого для окончательной классификации выбирается тот класс, чья модель дала наивысший результат. Другими словами, мы с помощью нескольких моделей оцениваем, к какому классу вероятнее всего принадлежит данный объект и выбираем для окончательной классификации тот класс, чья вероятность выше остальных.

Данный метод называется "один против всех" (one vs all или one vs rest). Это название подчеркивает тот факт, что для каждого класса оценивается вероятность принадлежности объекта к нему, в сравнении с вероятностью принадлежности к любому из всех остальных классов.

Формулировка данного алгоритма не предполагает использование порогового значения. Вероятности принадлежности объекта к классам могут быть любые. Нам важно выбрать среди них максимум. Даже есть все эти вероятности

меньше 50%, все равно мы выберем тот класс, к которому данный объект больше всего подходит. Более того, в среднем, эти вероятности обратно пропорциональны количеству классов в задаче. То есть если в конкретной проблеме, например, 50 000 классов, то не стоит ожидать, что объект будет принадлежать одному из них с вероятностью 90%. Здесь скорее речь пойдет о выборе между 0,028% и 0,025%.

У этого алгоритма есть еще одна характерная черта. С его помощью можно решать задачи мультиклассификации (множественная классификация). Напомним, это такие, в которых один конкретный объект может принадлежать нескольким классам одновременно. Такие задачи часто формулируются как придание объекту набора меток. Например, добавление релевантных тегов к посту в социальных сетях. Естественно, что у отдельно взятому посту можно придать достаточно много тегов. В данном случае теги служат классами, но задача мультиклассовая.

Другой пример - распознавание объектов на изображении. На конкретной картинке может же быть не один объект, а несколько, произвольное количество. Так вот, при использовании алгоритма "один против всех" можно брать как итоговый не один класс с максимальной вероятностью, а сразу несколько. Стратегии отбора классов бывают разные: в одной задаче можно брать фиксированное количество лучше подходящих классов, в других - брать любое количество классов, вероятности которых больше определенного порога.

Конечно, есть у данного алгоритма и оборотная сторона. Его применение не очень целесообразно, если в задачу уж очень большое количество классов. Ведь тогда придется обучить и использовать точно такое же количество моделей. А это может быть как затратно по процессорному времени, то есть обучение будет проходить слишком медленно, так и затратно по памяти, ведь все эти модели, их параметры надо хранить для осуществления предсказания.

Может, стоит рассмотреть другие виды моделей классификации, которые решают множественные задачи сами по себе.

Выводы:

1. Существуют методы классификации, которые сами по себе могут решать задачи множественной классификации.

2. Для тех, которые не умеют, существует алгоритм "один против всех".

3. В нем строится столько бинарных моделей, сколько классов существует в задаче.

4. Данный алгоритм уже не зависит от выбора порогового значения.

5. Этот алгоритм еще может решать проблемы мультиклассификации.

6. Для задач с очень большим количеством классов этот алгоритм может быть неэффективен.

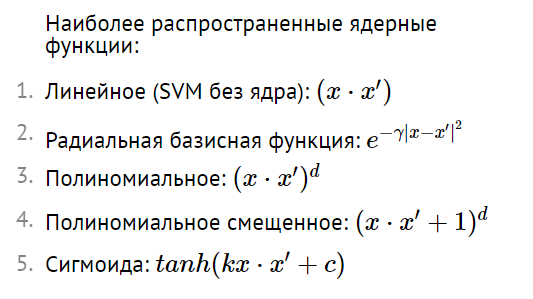
# **Метод опорных векторов в задачах классификации.**

**Метод опорных векторов** - модель машинного обучения с учителем.

Может применяться как для задач регрессии так и для задач классификации

Метод опорных векторов основан на идее нахождения опорных точек (support vectors) в пространстве признаков, которые имеют решающее значение для классификации или регрессии. Эти опорные точки называются опорными векторами.

Способ задачи опорных векторов зависит от ядра и выбриаеться под определенные данные. Ядро - это функция, используемая в методе опорных векторов и других алгоритмах машинного обучения, чтобы позволить им работать в пространствах высокой размерности.





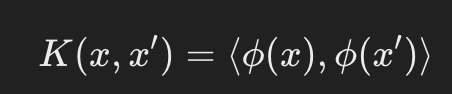
# **Понятие ядра и виды ядер в методе опорных векторов.**

**Ядро** — это функция, используемая в методе опорных векторов и других алгоритмах машинного обучения, чтобы позволить им работать в пространствах высокой размерности, не вычисляя преобразований в этих пространствах явно. Это основано на том, что некоторые алгоритмы, включая SVM, могут быть записаны в терминах скалярных произведений.

Пусть

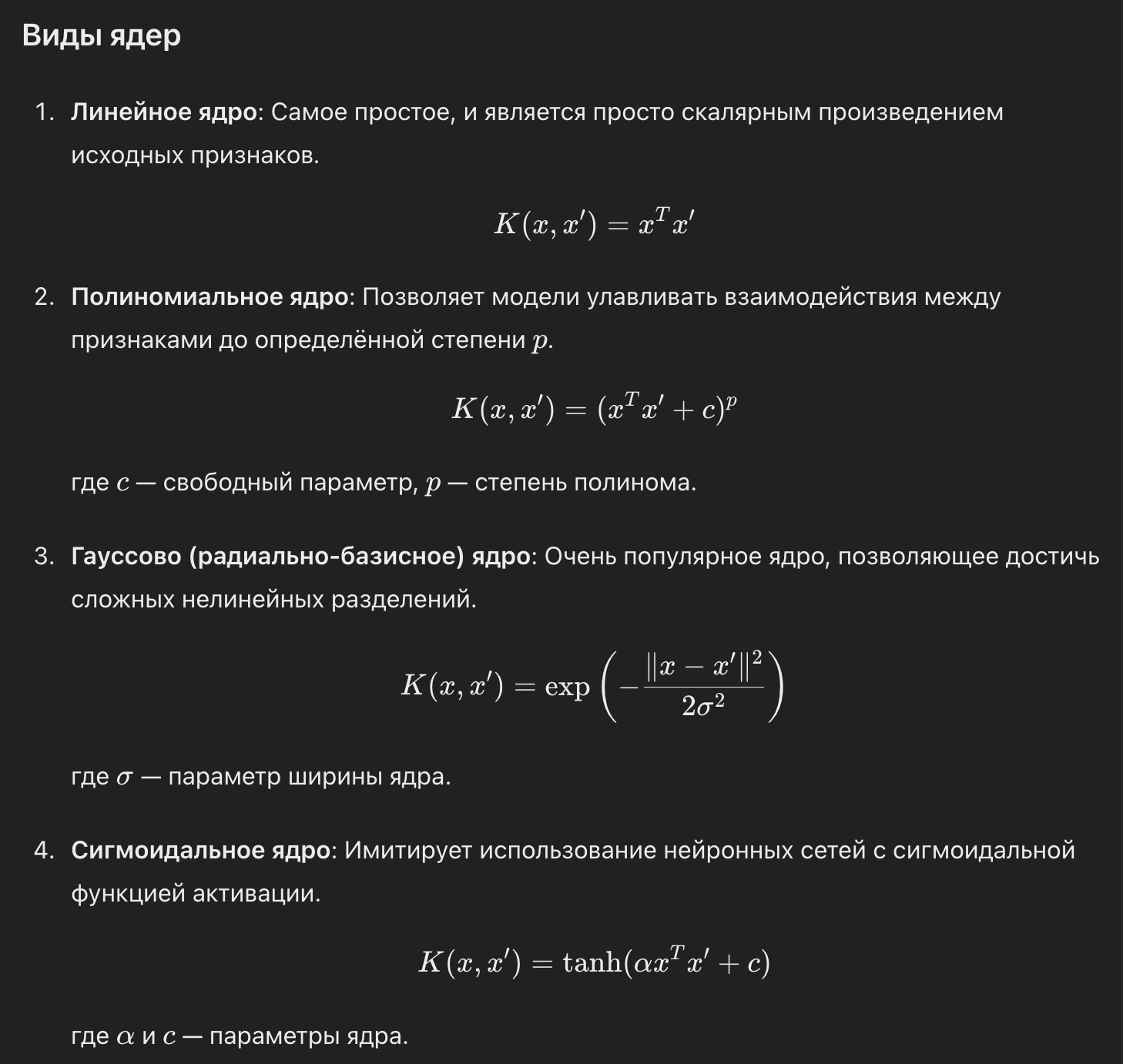


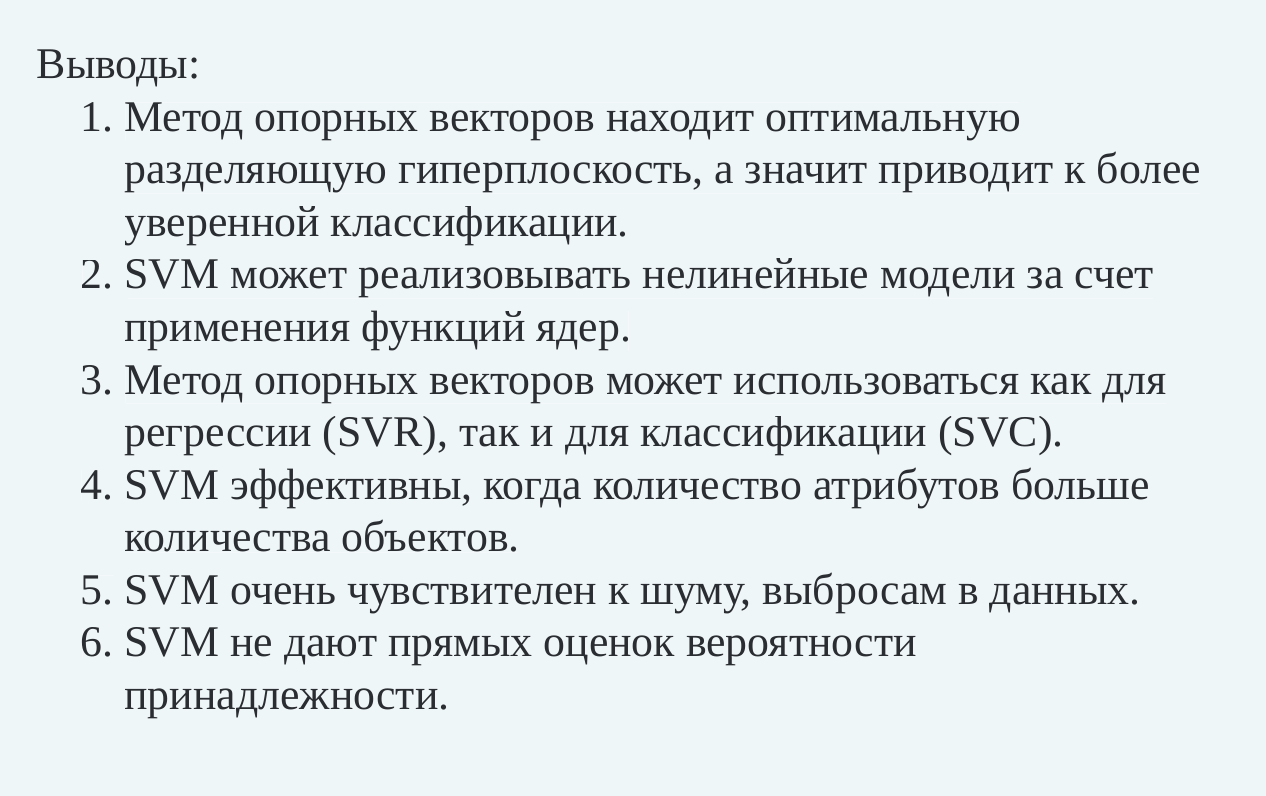
отображение исходных векторов признаков в (высокоразмерное) пространство признаков. Тогда функция ядра KKK определяется как скалярное произведение в этом пространстве:



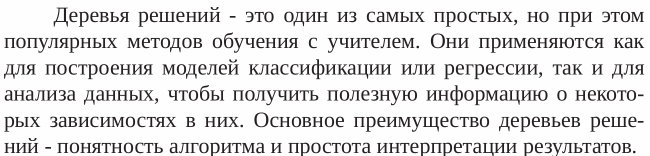
(Угловые скобочки - скалярное произведение в пространстве F)

где 𝜙 — функция отображения в пространство признаков.



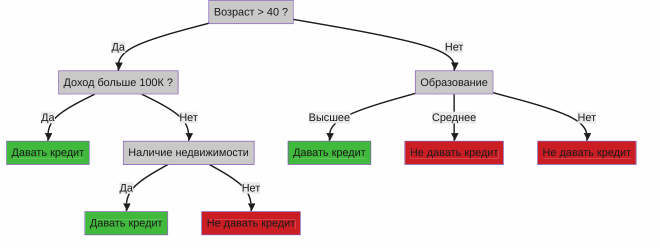


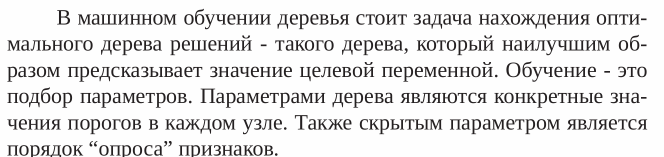
# **Метод решающих деревьев в задачах классификации.**

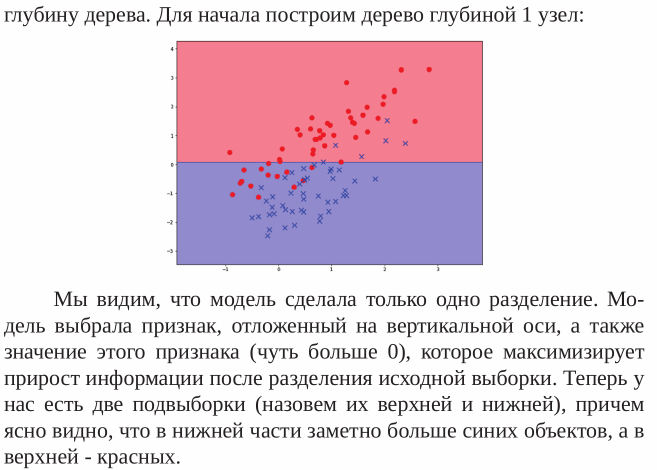


Дерево решений - это иерархическая структура, состоящая из

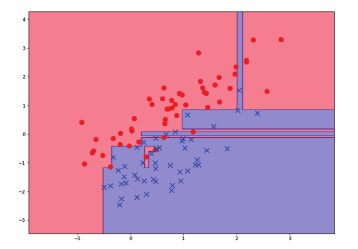
правил вида “Если ..., то...”.







Чем больше будет узлов, тем точнее модель, однако стоит опасаться переобучнения, чтобы ваша модель не выглядела вот так



Дело в том, что с ростом глубины дерева модель становиться более сложной и чувствительной, что не всегда может быть хорошо. Плюс к этому у нас не так много данных (точек выборки), чтобы строить такие глубокие деревья. Именно поэтому, максимальную глубину дерева стоит ограничивать. Причем сколько именно уровней использовать зависит только от структуры данных.

**Преимущества деревьев решений в задачах классификации:**

* Интерпретируемость: Дерево решений обеспечивает понятную и интерпретируемую модель, которая помогает понять, какие признаки и правила используются для классификации объектов.
* Управляемость: Дерево решений может быть легко изменено и настроено для конкретной задачи, что обеспечивает высокую управляемость.
* Автоматическое отбор признаков: Дерево решений может автоматически выбирать наиболее информативные признаки для классификации, что улучшает качество модели.

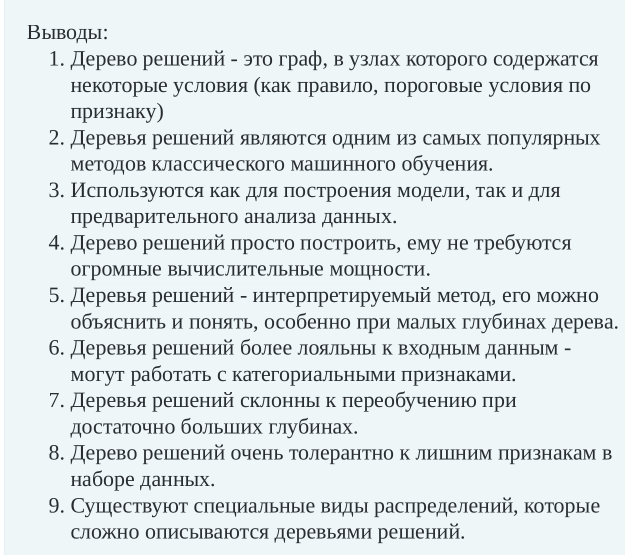
**Недостатки деревьев решений:**

* Низкая обобщающая способность: Дерево решений может не обобщаться на новые объекты, если оно было обучено на ограниченной выборке.
* Чувствительность к параметрам: Дерево решений может быть чувствительно к выбору параметров, таких как пороговая функция и критерий остановки, что может привести к переобучению.

**Практика и Приложения**

Дерево решений широко используется в различных приложениях, таких как:

* Распознавание текста и речи: Дерево решений может быть использовано для классификации текстов и речи, анализируя признаки, такие как частота использования слов и фонетические признаки.
* Распознавание изображений: Дерево решений может быть использовано для классификации изображений, анализируя признаки, такие как размер, форма и цвет.
* Обнаружение спама: Дерево решений может быть использовано для обнаружения спама, анализируя признаки, такие как текстовые признаки и признаки отправителя



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

***Метод k ближайших соседей (K-Nearest Neighbors, KNN)*** является простым и понятным алгоритмом машинного обучения, который широко применяется в задачах классификации. Он основан на принципе "большинство голосов": для классификации нового объекта алгоритм находит k ближайших соседей из обучающего набора данных и относит новый объект к классу, который является наиболее распространенным среди этих соседей.

**Принцип работы метода K ближайших соседей:**

1. Выбор параметра K: Определение количества ближайших соседей, которые будут использоваться для классификации. Значение K влияет на производительность алгоритма: маленькое K может сделать модель чувствительной к шуму, а большое K может сгладить границы между классами.
2. Расчет расстояний: Вычисление расстояний между новым объектом и всеми объектами в обучающем наборе данных. Расстояния могут быть рассчитаны с использованием различных метрик, таких как евклидово расстояние, манхэттенское расстояние и косинусное расстояние.
3. Сортировка соседей: Нахождение K ближайших соседей и определение классов, к которым они принадлежат.

**Примеры использования метода K ближайших соседей в задачах классификации:**

* Классификация объектов: На основе близости к соседям новый объект классифицируется в соответствующий класс.
* Определение класса: Если большинство из K соседей принадлежат к определенному классу, то новый объект также относится к этому классу.

Метод K ближайших соседей прост в реализации и понимании, что делает его привлекательным для новичков в машинном обучении. Однако важно учитывать его недостатки, такие как высокие вычислительные затраты для больших наборов данных, чувствительность к шуму и выбросам, а также необходимость правильного выбора параметра K для оптимальной производительности модели.

*Выводы:*

1. Метод ближайших соседей заключается в предсказании

значения исходя из значения ближайших к нему объектов

обучающей выборки.

2. Опирается на гипотезу компактности: схожие объекты

чаще принадлежат одному классу, чем разным.

3. Чаще всего используется евклидова метрика расстояния

между объектами.

4. Не имеет внутренних параметров. Можно считать, что

параметры - это сама обучающая выборка.

5. Нужно выбрать k - количество используемых соседей.

6. Чем больше к, тем проще модель и больше вероятность

недообучения.

7. Самый простой алгоритм для классификации.

8. Данные обязательно надо нормализовать.

9. Алгоритм не чувствителен к выбросам.

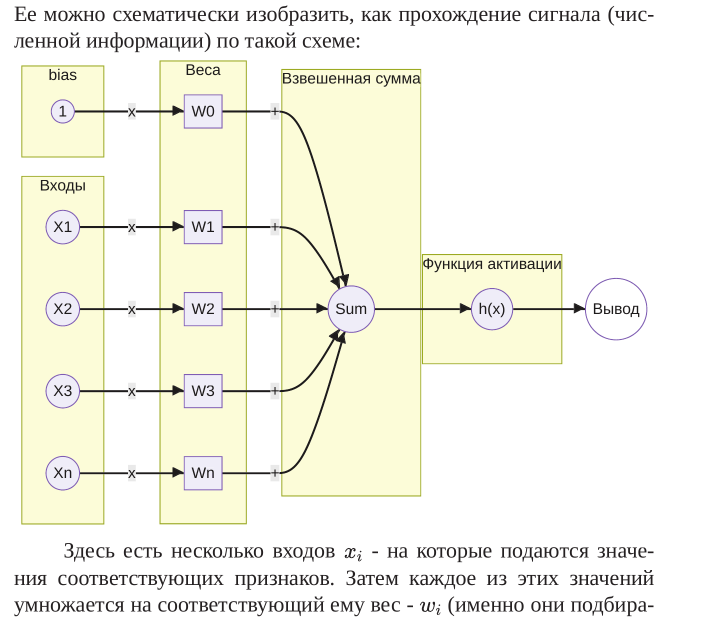
10. Алгоритм работает медленно при большом объеме

обучающей выборки.

# **Однослойный перцептрон в задачах классификации.**

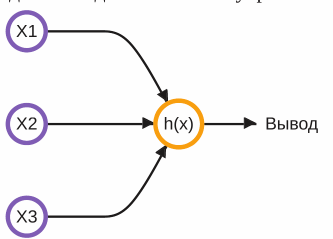
Перцептрон - это самый простой вид искусственных нейронных сетей. Рассмотрим сначала принцип действия одного искусственного нейрона.

Для этого возвратимся к обычной логистической регрессии. Ее можно схематически изобразить, как прохождение сигнала (численной информации) по такой схеме:

Здесь есть несколько входов xi - на которые подаются значения соответствующих признаков. Затем каждое из этих значений умножается на соответствующий ему вес - w; (именно они подбираются в процессе обучения), затем они суммируются, а после этого эта взвешенная сумма подается на вход некоторой нелинейной функции (в случае логистической функции - сигмоидальной). Обратите внимание, что на схеме присутствует один постоянный вход, на который подается всегда значение 1 - это искусственный константный признак, который нужен для учета свободного слагаемого

в взвешенной сумме.

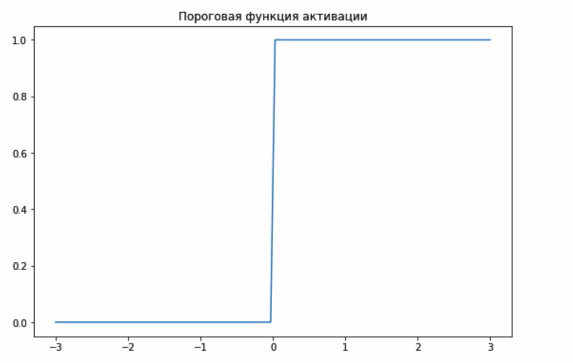
Таким образом модель логистической регрессии представляется в виде некоторого объекта, у которого есть несколько входов, один выход и какие-то внутренние параметры:



Мы можем пользоваться этим нейроном, как моделью классификации. Для этого на

его входы надо подать значения, соответствующие значениям признака классифицируемого объекта. Тогда на выходе мы получим некоторое значение, которое можно интерпретировать, как значение бинарной классификации.

В искусственном нейроне можно использовать разные функции активации. Исторически, чаще всего использовалась так называемая ступенчатая функция. Она принимает значение 0, если на вход ей подается значение меньше либо равно нулю, или 1, если подается положительное значение. Другими словами эта функция активирует нейрон, если значение взвешенной суммы входных сигналов больше какого-то значения. Причем, это пороговое значение может регулироваться свободным параметром wi.

Сам по себе один нейрон представляет собой довольно простую модель, но именно такое представление позволяет очень просто объединять несколько нейронов - то есть соединять выход одного нейрона со входом другого, другими словами, подавать результат работы одного нейрона как значение признака другому. \

Выводы:

1. Нейронные сети являются самым популярным методом обучения с учителем.

2. Нейронные сети позволяют строить самые сложные и масштабируемые модели.

3. Обучение и применение нейронных сетей очень хорошо распараллеливается. За счет этого можно ускорять их работу с помощью графических ядер и других средств.

4. Перцептрон - самая простая архитектура нейронной сети, состоящая из нескольких полносвязных слоев.

5. Нейронные сети работают по принципу черного ящика, их очень сложно интерпретировать.

6. При применении нейронных сетей встает вопрос выбора архитектуры, количества нейронов.

7. Главный параметр нейронной сети - количество скрытых слоев. Если их больше одного сеть называется глубокой.

8. Нейронные сети очень естественно решают задачу множественной и мультиклассовой классификации.

# **Метрики эффективности и функции ошибки: назначение, примеры, различия.**

В задачах машинного обучения для оценки качества моделей и сравнения различных алгоритмов используются **метрики эффективности**, а их выбор и анализ — непременная часть работы датасатаниста.

**Назначение:** Оценка качества эффективности полученной модели, сравнение моделей между собой, их может быть несколько и они должны выбиратся исходя из задачи машинного обучения

**Примеры:**

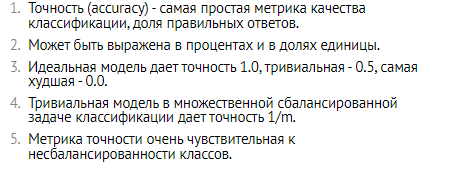
Классификация:



**Accuracy**

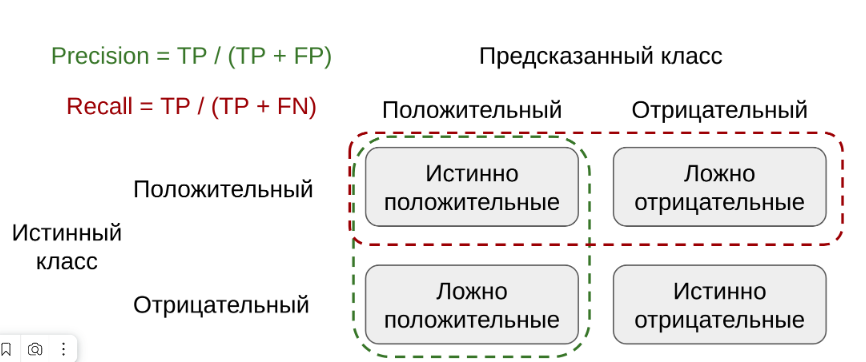
from sklearn.metrics import accuracy\_score

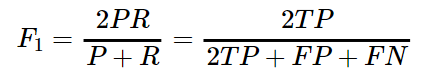


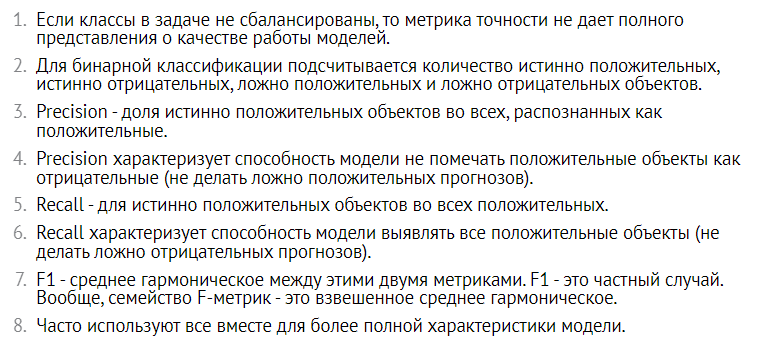


**F1-score, recall, precision**

from sklearn.metrics import F1\_score, recall, precision

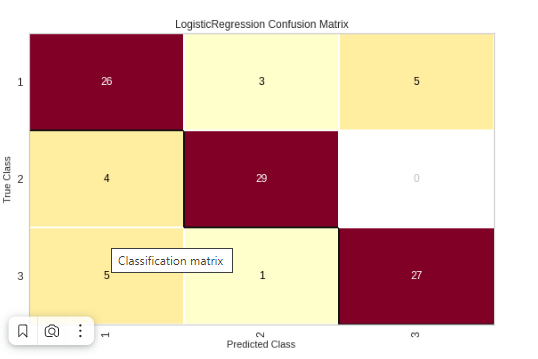


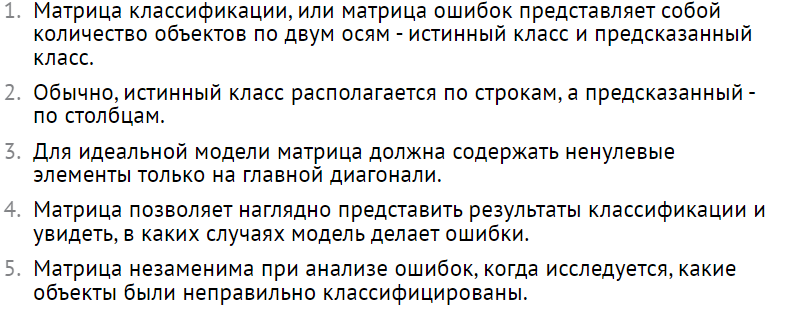




**Матрица классификации**

**from sklearn.metrics import confusion\_matrix**

****

****

**МЕТРИКИ МНОЖЕСТВЕННОЙ КЛАССИФИКАЦИИ (ДО ЭТОГО БЫЛА БИНАРНАЯ)**

**from sklearn.metrics import classification\_report**

**target\_names = ['class 0', 'class 1', 'class 2']**

**print(classification\_report(y\_true, y\_pred, target\_names=target\_name)**

****

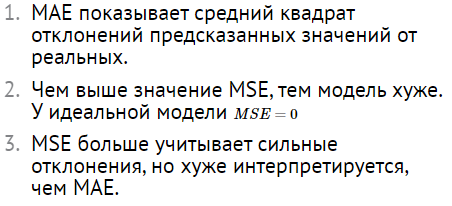
Регрессия: (обычно анализируют отклонений значений реальных от предсказанных)



**Mean Squared Error (MSE**)

from sklearn.metrics import mean\_squared\_error

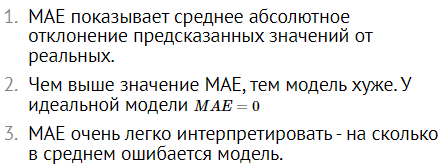




**Mean Absolute Error (MAE)**

**from sklearn.metrics import mean\_absolute\_error**

****

****

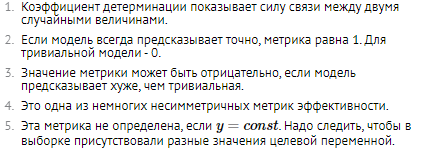
**)**

**r2-score**

Именно коэффициент детерминации чаще всего используется как **метрика** по умолчанию, которую можно посмотреть при помощи метода score() у модели регрессии.

**from sklearn.metrics import r2\_score**

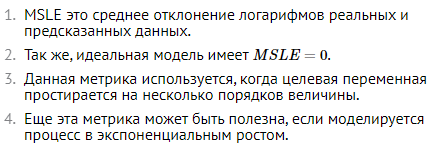
****

****

**MSLE**

**from sklearn.metrics import mean\_squared\_log\_error**

****

****

**Функции ошибки** в машинном обучении — это функции, которые оценивают «ошибку» или «расхождение» между предсказаниями модели и фактическими значениями.

**Назначение:** Используется для огранизации процесса обучения, нахождения оптимума, может быть только одна и должна быть легко вычислимой и конструироваться исходя из типа модели

**Примеры:**

**MAE**

**MSE**

**Cross-Entropy Loss**

**Функция кросс-энтропии - общепринятая функция потерь для классификационных задач, где цель - предсказать категориальный маркер.**

**Функция ошибки: E = -Σ(y\_true \* log(y\_pred))**

**где y\_true - реальный маркер, y\_pred - предсказанная вероятностная распределениевеми классами.**

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

**Набор данных (датасет)** в машинном обучении — это совокупность данных, используемая для обучения, валидации и тестирования моделей. Датасет состоит из объектов (или экземпляров), каждый из которых описывается набором признаков и, в случае задач обучения с учителем, содержит метки классов или значения целевой переменной.

### **Требования к набору данных**

1. **Полнота**: Данные должны быть достаточно полными и репрезентативными для рассматриваемой задачи, чтобы модель могла обучиться на всех значимых случаях и корректно обобщать на новых данных.
2. **Качество**: Данные должны быть чистыми и аккуратными, минимизировано количество ошибок, выбросов и пропущенных значений.
3. **Баланс**: В задачах классификации желательно, чтобы классы были сбалансированы — это уменьшает риск создания предвзятых моделей.
4. **Разнообразие**: Данные должны охватывать разнообразные случаи, сценарии и условия, чтобы модель могла эффективно работать в различных ситуациях.

### **Представление данных**

Данные обычно представлены в форме таблицы, где строки соответствуют объектам, а столбцы — признакам. В зависимости от типа данных, признаки могут быть:

* **Числовыми**: непрерывными или дискретными.
* **Категориальными**: номинальными или порядковыми.
* **Бинарными**: принимающими только два значения, например, да/нет.
* **Текстовыми**: требующими предобработки для преобразования в числовую форму, например, с использованием методов векторизации текста.
* **Временными**: даты и времена, которые могут потребовать специального обращения.

### **Признаки и объекты**

**Признаки (features)** — это характеристики, по которым описываются объекты. Они играют ключевую роль в процессе обучения, так как именно на основе признаков модель делает выводы и принимает решения. Качественный подбор и инжиниринг признаков могут значительно улучшить производительность модели.

**Объекты (instances, samples)** — это конкретные единицы данных в датасете. В контексте машинного обучения каждый объект — это ситуация или пример, на котором модель учится, чтобы понять, как реагировать на аналогичные ситуации в будущем.

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



**В машинном обучении выделяют четыре основных типа шкал:**

*Номинальная шкала:*

* Используется для классификации объектов по категориям.
* Числа не имеют конкретной величины и используются только для обозначения и классификации.
* Примеры: номера телефонов, автомашин, пол людей.

*Порядковая шкала:*

* Используется для сравнения объектов по порядку, но не по величине разницы между ними.
* Числа характеризуют порядок расположения объектов.
* Примеры: оценки знаний учащихся, силы землетрясений.

*Интервальная шкала:*

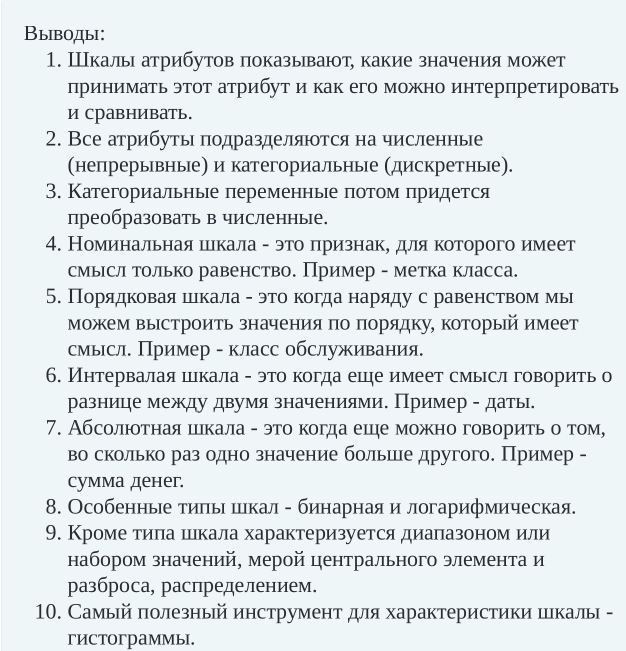
* Используется для измерения величины разницы между объектами.
* Числа характеризуют величину разницы между объектами.
* Примеры: измерения мощностей системы в децибелах.

*Шкала отношений:*

* Используется для сравнения объектов по величине.
* Числа характеризуют величину объектов.
* Примеры: измерения отношения мощностей на выходе и на входе системы

В машинном обучении шкалы измерения важны для определения методов анализа и преобразования данных. Например, номинальные шкалы могут быть преобразованы в порядковые или количественные, что может помочь в более эффективном анализе данных.

В целом, выбор типа шкалы измерения зависит от типа анализируемых данных и от целей исследования. Правильный выбор шкалы измерения может помочь в получении более точных результатов и улучшении качества модели машинного обучения.



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

**Чистые данные** - это набор данных, который представлен в

форме реляционной таблицы, все значения в которой выражены числом, не имеет пропущенных значений, имеет внутреннюю согласованность. Такие данные формально можно

применять для использования в любых моделей машинного

обучения.

**Основные этапы очистки данных:**

1. Преобразование типов данных - приведение данных к числовым типам, чтобы алгоритмы машинного обучения могли применять математические операции.
2. Обработка пропущенных значений - заполнение или удаление строк с пропусками в зависимости от задачи.
3. Обнаружение и удаление выбросов - выявление аномальных значений, которые могут негативно влиять на модель. Один из эффективных методов - алгоритм изолирующего леса.
4. Преобразование категориальных признаков - кодирование строковых признаков в числовые.
5. Масштабирование признаков - приведение признаков к одному масштабу, чтобы алгоритмы работали корректно.

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

Основные этапы:

1. постановка задачи машинного обучения (выбираем, что будет исследоваться и какому классу задач относится наша задача регрессия/классификация)
2. Анализ и обработка дата-сета

* Первичный анализ данных:Это этап, на котором происходит изучение и понимание структуры и содержания собранных данных
* Предобработка данных:Это этап, на котором данные обрабатываются и готовятся к использованию для обучения модели. Он может включать в себя удаление пропущенных значений, преобразование типов данных и другие операции
* Разведывательный анализ данных (Exploratory Data Analysis):Это этап, на котором модель исследуется и понимается лучше, чтобы выявить паттерны и зависимости в данных. Он включает в себя визуализацию данных, изучение выбросов.
* Создание признаков (Feature Engineering):Это этап, на котором новые признаки создаются из существующих, чтобы улучшить качество модели. Он может включать в себя преобразование данных, создание новых признаков на основе существующих, избавление от категориальных признаков.
* Отбор признаков (Feature selection):Это этап, на котором из существующих признаков выбираются те, которые наиболее важны для модели. Он может включать в себя методы, такие как корреляция и другие
* Подготовка данных (Data Preparation):Это этап, на котором данные готовятся к использованию для обучения модели. Он может включать в себя нормализацию и другие операции.

1. Разработка модели:

Разработка модели — это этап, на котором создается и настраивается модель машинного обучения

1. Оценка модели:

Оценка модели — это этап, на котором модель проверяется на эффективность и адекватность для предполагаемого применения

1. Развертывание модели:

Развертывание модели — это этап, на котором модель готовится к использованию в производственной среде

1. Завершающий этап

Это этап, на котором модель проверяется на ее способность работать должным образом с новыми, невиданными ранее данными.

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

**Задачи** предварительного анализа данных:

* Очистка данных от ошибок и выбросов
* Поиск и заполнение пропущенных значений
* Преобразование данных в удобный формат для моделирования
* Выделение важных признаков для построения модели

**Методы** предварительного анализа данных:

* Визуализация данных (гистограммы, диаграммы рассеяния)
* Статистические методы анализа данных (корреляция, t-тесты)
* Методы машинного обучения для поиска закономерностей в данных

**Цели** предварительного анализа данных:

* Повышение качества и точности модели машинного обучения
* Уменьшение времени обучения модели
* Повышение интерпретируемости модели и выводов на основе данных

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

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

Отсутствующие данные, или пропущенные значения, являются распространенной проблемой в исследованиях и анализе данных. Они могут возникать по различным причинам и оказывать существенное влияние на достоверность и обобщаемость результатов анализа. В этом ответе мы рассмотрим основные причины отсутствующих данных, их типы, а также способы обработки пропущенных значений.

## Причины отсутствующих данных

Отсутствующие данные могут возникать по следующим причинам:

1. Ошибки при вводе данных: Человеческие ошибки, такие как пропуск вопросов в опросах или неправильный ввод данных.
2. Технические сбои: Сбои оборудования, потеря файлов или повреждение данных.
3. Проблемы с дизайном исследования: Некорректно сформулированные вопросы, неподходящие методы сбора данных.
4. Отказ от ответа: Респонденты могут отказываться отвечать на определенные вопросы по личным причинам.
5. Выбытие из исследования: В лонгитюдных исследованиях участники могут выбывать до завершения, что приводит к пропущенным данным.

## Типы отсутствующих данных

Существует три основных типа отсутствующих данных:

1. Отсутствующие полностью случайным образом (MCAR): Вероятность отсутствия данных не зависит от самих данных или других переменных. Например, случайные сбои оборудования.
2. Отсутствующие случайным образом (MAR): Вероятность отсутствия данных зависит от других наблюдаемых переменных, но не от самих пропущенных значений. Например, респонденты определенной демографической группы чаще пропускают некоторые вопросы.
3. Отсутствующие не случайным образом (MNAR): Вероятность отсутствия данных зависит от самих пропущенных значений. Например, пациенты с более тяжелым состоянием чаще выбывают из исследования.

## Пути решения проблемы отсутствующих данных

Существует несколько подходов к обработке отсутствующих данных:

Принятие отсутствующих данных: Оставление пропущенных значений как есть. Применимо, если данные отсутствуют случайным образом (MCAR или MAR) и их немного.

Удаление наблюдений или переменных: Удаление строк или столбцов с пропущенными значениями. Простой подход, но может привести к потере статистической мощности и смещению результатов.

Импутация данных: Замена пропущенных значений на оценки, полученные с помощью статистических методов (среднее, регрессия, метод ближайших соседей и др.). Позволяет сохранить размер выборки, но может внести дополнительную неопределенность.

Модельные подходы: Использование статистических моделей (EM-алгоритм, максимальное правдоподобие) для оценки пропущенных значений и учета неопределенности. Более сложные, но потенциально более точные методы.

Анализ чувствительности: Проверка устойчивости результатов к различным предположениям об отсутствующих данных.

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

Проблема несбалансированных классов в машинном обучении - это распространенная проблема, которая возникает, когда в наборе данных один класс (или несколько классов) значительно превосходит другие по количеству объектов. Это может привести к смещению модели в сторону преобладающего класса, что может вести к низкой чувствительности модели к объектам меньшего класса и ошибочным предсказаниям

**Причины и последствия несбалансированных классов**

Несбалансированные классы могут возникать в различных областях, включая медицинскую диагностику, фильтрацию спама и обнаружение мошенничества.

*Они могут возникать из-за различных факторов, таких как:*

* Неправильное отбор данных
* Недостаточное количество данных для меньших классов
* Неправильное представление данных

*Проблема несбалансированных классов может привести к следующим последствиям:*

* Смещение модели в сторону преобладающего класса
* Низкая чувствительность модели к объектам меньшего класса
* Ошибочные предсказания
* Низкая точность модели

Методы решения проблемы несбалансированных классов

*Для решения проблемы несбалансированных классов используются различные методы, включая:*

* Повторное сэмплирование (oversampling): повторное сэмплирование меньших классов для достижения равного количества выборок.
* Undersampling: уменьшение количества выборок для преобладающего класса.
* Synthetic data generation: генерация синтетических данных для меньших классов.
* Class weighting: весовые коэффициенты для классов, чтобы уменьшить влияние преобладающего класса.
* Cross-validation: кросс-валидация для оценки модели на различных подмножествах данных.

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

**Параметры модели** — это внутренние переменные, которые изменяются в процессе обучения модели машинного обучения. Они получаются автоматически и не настраиваются вручную. Примерами параметров модели могут быть веса узлов в нейронной сети или коэффициенты в линейной модели.

*Параметры модели* обучаются автоматически в ходе процесса машинного обучения. Они получаются на основе данных и не настраиваются вручную. Обучение параметров происходит в ходе оптимизации функции потерь, когда модель адаптируется к данным.

**Гиперпараметры модели** — это внешние переменные конфигурации, которые фиксируются до начала обучения модели. Они управляют процессом обучения и не изменяются в ходе обучения. Примерами гиперпараметров модели могут быть количество узлов и слоев в нейронной сети, а также количество ветвей в дереве принятия решений.

*Гиперпараметры модели* настраиваются вручную или с помощью автоматизированных методов перед началом обучения модели. Они влияют на архитектуру модели, темп обучения и сложность модели.

Метод поиска по сетке — это алгоритм, который перебирает все возможные сочетания гиперпараметров и оценивает производительность модели для каждого из них. Он эффективен, когда количество гиперпараметров не очень велико и влияние каждого параметра на результат модели можно оценить.

*Выводы:*

1. Поиск по сетке - полный перебор всех комбинаций

значений гиперпараметров для поиска оптимальных

значений.

2. Для его организации надо задать список гиперпараметров

и их конкретных значений.

3. Непрерывные гиперпараметры надо дискретизировать.

4. Для непрерывных гиперпараметров часто используется

логарифмическая шкала.

5. Поиск по сетке имеет экспоненциальную сложность.

6. Чем больше параметров и значений задать, тем лучше

получится модель, но дольше поиск.

7. Можно задать критерии поиска - целевые метрики.

8. Рекомендуется использовать кросс-валидацию.

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

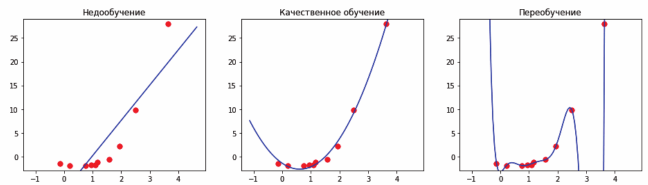
При решении задачи методами машинного обучения всегда встает задача выбора вида модели.

Перед аналитиком стоит задача выбора параметрического семейства модели, которую он будет обучать на имеющихся данных. Причем разные семейства дадут модели разного уровня качества после обучения. К сожалению, очень сложно заранее предугадать, какое семейство моделей после завершения обучения даст наилучшее качество предсказания по данной выборке.

Как показывает практика, самое существенное влияние на эффектив-

ность оказывает уровень сложности модели. Любое параметрическое семейство моделей имеет определенное количество степеней свободы, которое определяет то, насколько сложное и изменчивое поведение может демонстрировать получившаяся функция.

Влияние уровня сложности на поведение модели относительно данных наиболее наглядно можно проследить на примере модели полиномиальной модели. Степень полинома - это очень показательная характеристика уровня сложности модели. Давайте рассмотрим три модели регрессии - линейную (которую можно рассматривать как полином первой степени), полином второй и восьмой степени. Мы обучили эти модели на одном и том же датасете и вот что получилось:



Естественно предположить, что модель, изображенная на втором графике показывает наилучшее описание точек данных. Но по любой метрике качества третья модель будет показывать более высокий результат. Человек, глядя на график третьей модели, сразу сделает вывод, что она "слишком" хорошо подстроилась под имеющиеся данные. Сравните это поведение с первым графиком, который демонстрирует самую низкую эффективность на имеющихся данных. Можно проследить, как именно сложность модели влияет на ее применимость. Если модель слишком простая, то она может не выявить имеющиеся сложные зависимости между признаками и целевой переменной. Говорят, что у простых моделей низкая вариативность (variance). Слишком же сложная модель имеет слишком высокую вариативность, что тоже не очень хорошо.

Слишком сложные модели избыточно подстраиваются под малейшие выбросы в данных. Это увеличивает значение метрик эффективности, но снижает пригодность модели на практике, так как очевидно, что модель будет делать большие ошибки на новых данных из той же выборки. Такая ситуация называется **переобучением**.

Переобучение - это очень коварная проблема моделей машинного обучения, ведь на "бумаге" все метрики показывают отличный результат. Это происходит потому, что в практически любой выборке данных конкретное положение точек, их совместное распределение определяется как существенной зависимостью между признаками и

целевой переменной, так и случайными отклонениями. Эти случайные отклонения, выбросы, аномалии не позволяют сделать однозначный вывод, что модель, которая лучше описывает имеющиеся данные, является лучшей в глобальном смысле.

Выводы:

1. Прежде чем обучать модель, нужно выбрать ее вид (параметрическое семейство функций).

2. Разные модели при своих оптимальных параметрах будут давать разный результат.

3. Чем сложнее и вариативнее модель, тем больше у нее параметров.

4. Простые модели быстрые, но им недостает вариативности, изменчивости, у них высокое смещение (bias).

5. Сложные модели могут описывать больше зависимостей, но вычислительно более трудоемкие и имеют большую дисперсию (variance).

6. Слишком вариативные (сложные) модели алгоритм может подстраиваться под случайный шум в данных - переобучение.

7. Слишком смещенные (простые) модели алгоритм может пропустить связь признака и целевой переменной - недообучение.

8. Не всегда модель, которая лучше подстраивается под данные (имеет более высокие метрики эффективности) лучше.

**Для решения проблем переобучения и недообучения** моделей машинного обучения можно использовать следующие пути:

Переобучение:

1. Увеличение объема данных:Увеличение объема обучающих данных может помочь модели обобщить свои навыки и не специализироваться на конкретных данных
2. Добавление новых функций:Добавление новых признаков может помочь модели лучше обрабатывать новые данные и не специализироваться на конкретных признаках
3. Изменение архитектуры модели:Изменение архитектуры модели может помочь модели лучше обрабатывать новые данные и не специализироваться на конкретных признаках

Недообучение:

1. Увеличение объема данных:Увеличение объема обучающих данных может помочь модели лучше обрабатывать новые данные и не недообучиваться
2. Добавление шума в данные:Добавление шума в данные может помочь модели лучше обрабатывать новые данные и не недообучиваться
3. Улучшение качества данных:Улучшение качества данных может помочь модели лучше обрабатывать новые данные и не недообучиваться
4. Проверка модели на тестовом множестве:Проверка модели на тестовом множестве может помочь обнаружить недообучение и улучшить качество модели
5. Улучшение метрик оценки:Улучшение метрик оценки может помочь обнаружить недообучение и улучшить качество модели

Для того, чтобы обнаруживать недо и переобучение:

1. Разделение данных на обучающий и тестовый наборы:Разделение данных на обучающий и тестовый наборы помогает оценить качество модели и обнаруживать проблемы переобучения и недообучения
2. Визуализация данных:Визуализация данных помогает обнаруживать проблемы и улучшать качество модели
3. Проверка модели на новых данных:Проверка модели на новых данных помогает обнаруживать проблемы переобучения и недообучения и улучшать качество модели

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

**Диагностика модели машинного обучения** - это процесс анализа и оценки модели машинного обучения для обнаружения ошибок, неэффективности или неприемлемых результатов.и.

**Методы** диагностики модели машинного обучения:

* Кросс-валидация (подходы: leave-one-out, k-fold cross validation)
* ROC-кривая и площадь под ROC-кривой (AUC)
* Матрица ошибок (confusion matrix)
* Метрики оценки качества модели (точность, полнота, F1-мера)
* Визуализация результатов работы модели (графики, диаграммы)

**Цели** диагностики модели машинного обучения:

* Оценка качества модели и ее способности к обобщению на новые данные
* Выявление проблем и слабых сторон модели для их исправления
* Оптимизация параметров модели для улучшения ее работы

В глобальном понимании **Цель** - обеспечить качественный и надежный результат моделирования, чтобы она могла быть использована в практическом применении.ии.и

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

Выбор подходящей модели машинного обучения — ключевой этап в проекте по анализу данных. Он зависит от множества факторов, включая тип задачи (классификация, регрессия, кластеризация, уменьшение размерности и др.), размер и вид данных, требуемую точность, скорость работы, доступность вычислительных ресурсов, а также от того, насколько важна интерпретируемость результатов.

### **Критерии для сравнения моделей**

1. **Точность и производительность**: Оценка качества моделей с помощью метрик, таких как точность, F1-мера, ROC-AUC для классификации или MSE, RMSE для регрессии.
2. **Переобучение и обобщающая способность**: Важно оценить, насколько хорошо модель работает не только на тренировочных данных, но и на новых, невиданных данных. Используются методы валидации, такие как кросс-валидация.
3. **Вычислительная сложность**: Время обучения и время предсказания могут быть критичными, особенно в больших системах и приложениях в реальном времени.
4. **Потребность в данных**: Некоторые модели требуют больших объемов данных для эффективного обучения, в то время как другие могут хорошо работать с меньшим количеством данных.
5. **Интерпретируемость**: В некоторых приложениях (например, в медицине или финансах) важно, чтобы результаты модели можно было легко интерпретировать и объяснить.

### **Методы сравнения моделей**

**Эмпирические тесты**: Используют различные метрики производительности для оценки моделей на одних и тех же наборах данных.

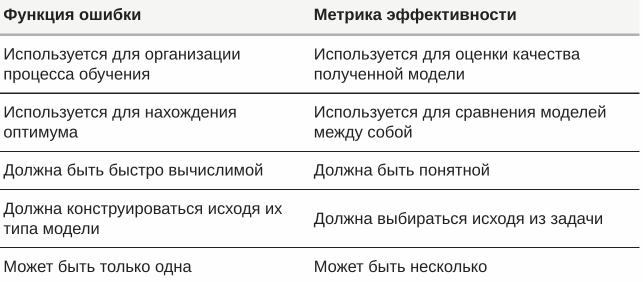
**Статистические тесты**: Такие как t-тест или ANOVA, используются для оценки значимости различий в производительности моделей.

**Валидация**: Кросс-валидация, Leave-One-Out или другие методы валидации помогают убедиться в стабильности и надежности моделей.

**Анализ ошибок**: Изучение типов ошибок, которые совершает модель, может дать понимание ограничений модели и возможных путей её улучшения.

**Ансамблевые методы**: Сравнение отдельных моделей может также включать их комбинации, такие как бэггинг, бустинг или стекинг, чтобы повысить производительность и устойчивость.

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



Для измерения эффективности работы моделей машинного обучения используются различные метрики, которые позволяют оценить качество предсказаний модели на новых данных. **Основные метрики включают:**

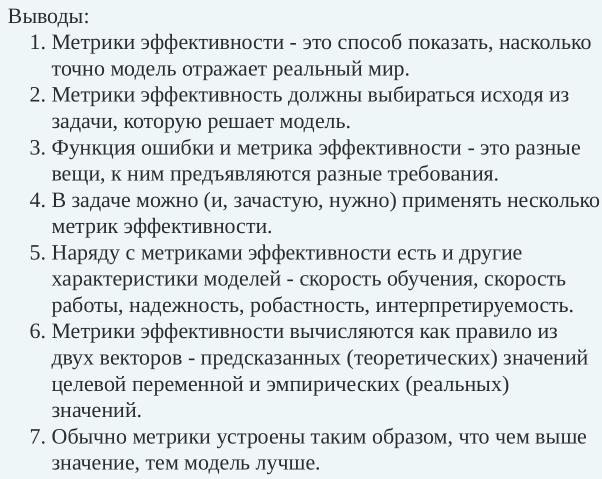
*Метрики для задач классификации*

* Точность (Accuracy): доля правильно классифицированных объектов.
* Полнота (Recall): доля объектов класса, которые были правильно классифицированы.
* Точность (Precision): доля объектов, отнесенных к классу, которые действительно принадлежат этому классу.
* F1-мера: гармоническое среднее между точностью и полнотой.
* ROC-кривая и AUC: показывает соотношение между истинно-положительными и ложно-положительными срабатываниями.

*Метрики для задач регрессии*

* Среднеквадратичная ошибка (MSE): среднее значение квадрата разницы между предсказанными и истинными значениями.
* Среднее абсолютное отклонение (MAE): среднее значение абсолютной разницы между предсказанными и истинными значениями.
* R-квадрат (R^2): показывает долю дисперсии зависимой переменной, объясняемую регрессионной моделью.

Для оценки моделей также используются методы кросс-валидации, которые позволяют оценить обобщающую способность модели на новых данных. Важно учитывать не только метрики качества, но и практические и экономические соображения при выборе наилучшей модели.



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

**Правильность (Accuracy)**

Правильность - это доля правильных предсказаний модели среди всех предсказаний. Она вычисляется как отношение количества верных предсказаний к общему количеству предсказаний.

Однако правильность не всегда является лучшей метрикой, особенно на несбалансированных данных, когда один класс значительно преобладает над другим. В таких случаях модель может достичь высокой правильности, просто предсказывая большинство класс, но при этом неправильно классифицировать меньшинство классов.

**Матрица ошибок (Confusion Matrix)**

Матрица ошибок - это таблица, показывающая количество верных и ошибочных предсказаний для каждого класса. Она позволяет детально проанализировать, какие именно ошибки совершает модель.

**Полнота (Recall) и точность (Precision)**

Полнота показывает, какую долю объектов целевого класса модель смогла правильно идентифицировать. Точность показывает, какая доля предсказаний модели для целевого класса оказалась верной.

Полнота и точность часто используются совместно, так как они дополняют друг друга. Например, модель может достичь высокой полноты, предсказывая целевой класс для всех объектов, но при этом будет иметь низкую точность из-за большого количества ложных срабатываний.

**ROC-кривая и AUC**

ROC-кривая показывает соотношение между долей верно классифицированных положительных примеров (полнотой ) и долей ложно классифицированных отрицательных примеров (1 - специфичность) при различных порогах.

Площадь под ROC-кривой (AUC) является агрегированной метрикой, которая показывает общее качество модели бинарной классификации. Значение AUC лежит в диапазоне от 0 до 1, причем 1 соответствует идеальной модели.

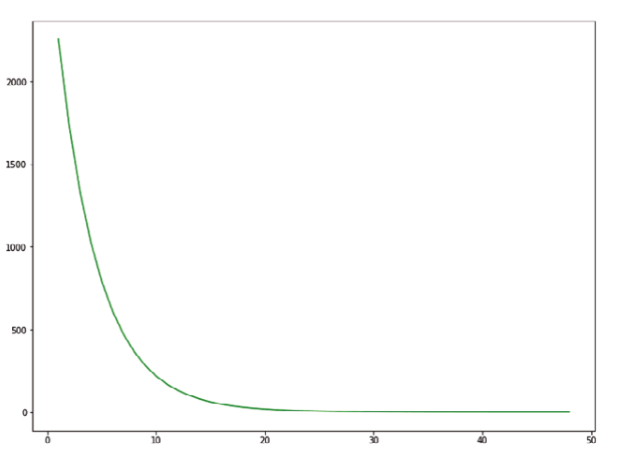
Выбор метрик зависит от конкретной задачи и требований к модели. Например, в задачах обнаружения мошенничества важнее высокая полнота, чтобы не пропустить ни одного мошенника, даже ценой некоторого количества ложных срабатываний. В медицинских задачах, наоборот, важнее высокая точность, чтобы минимизировать ложные диагнозы.

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

При оценки качества модели:

В простых случаях качество модели можно оценить визуально на графике. Но если у вас многомерная задача, это уже не представляется возможным. Кроме того, если ошибка и сама модель меняется незначительно, то очень сложно определить, стало хуже или лучше. Поэтому для диагностики моделей машинного обучения используют кривые.

Самая простая кривая обучения - зависимость ошибки от времени (итерации градиентного спуска).



На этом графике наглядно видно, что в начале обучения ошибка падала быстро, но в ходе градиентного спуска она вышла на плато. Учитывая, что мы используем гладкую функцию ошибки второго порядка, это свидетельствует о том, что мы достигли локаль-

ного оптимума и дальнейшее повторение алгоритма не принесет улучшения модели.

Если бы мы наблюдали на графике обучения ситуацию, когда по достижении конца обучения ошибка все еще заметно снижалась, это значит, что мы рано прекратили обучение, и нужно продолжить его еще на какое-то количество итераций.

При анализе графиков с библиотечными моделями не получится таких гладких графиков, они больше напоминают случайные колебания. Это из-за того, что в готовых реализациях используется очень оптимизированный вариант метода градиентного спуска. А он может работать с произвольными флуктуациями. В любом случае, нас интересует общий вид этой кривой.

Наиболее часто используемые метрики:

1. Коэффициент детерминации (R2) - это метрика, показывает на сколько расчетные параметры модели, то есть сама модель, объясняют зависимость и изменения изучаемого параметра -Y от исследуемых факторов - X. Она варьируется от 0 до 1, где 0 означает, что модель не объясняет никакую вариацию, а 1 — что модель объясняет все вариации.
2. Средняя абсолютная ошибка MAE - это метрика, которая измеряет среднюю абсолютную ошибку прогнозов модели. Она варьируется от 0 до бесконечности, где меньшее значение означает лучшую модель.
3. Среднеквадратичная ошибка MSE - это метрика, которая измеряет среднюю квадратическую ошибку прогнозов модели. Она варьируется от 0 до бесконечности, где меньшее значение означает лучшую модель.
4. Среднеквадратичная логарифмическая ошибка - это метрика, которая измеряет корень из средней квадратической ошибки прогнозов модели. Она варьируется от 0 до бесконечности, где меньшее значение означает лучшую модель.
5. Средняя абсолютная процентная ошибка MAPE

Mean Absolute Percentage Error (MAPE) — это метрика, которая измеряет среднюю абсолютную процентную ошибку прогнозов модели. Она варьируется от 0 до бесконечности, где меньшее значение означает лучшую модель.

1. Среднеквадратичная ошибка в процентах Mean Squared Percentage Error (MSPE) — это метрика, которая измеряет среднюю квадратическую процентную ошибку прогнозов модели. Она варьируется от 0 до бесконечности, где меньшее значение означает лучшую модель.

Выбор метрики эффективности модели регрессии зависит от конкретной задачи и требований к модели. Например, если модель должна прогнозировать значения с высокой точностью, то MSE или RMSE могут быть более подходящими. Если модель должна прогнозировать значения с высокой точностью и низкой абсолютной ошибкой, то MAE или MAPE могут быть более подходящими.

Факты о метриках:

1. Метрики эффективности для регрессий обычно анализируют отклонения предсказанных значений от реальных.

2. Большинство метрик пришло в машинное обучение из математической статистики.

3. Результаты работы модели можно исследовать более продвинутыми статистическими методами.

4. Обычно метрики сравнивают данную модель с тривиальной - моделью, которая всегда предсказывает среднее реальное значение целевой переменной.

5. Модель могут быть точны на 100%, но плохи они могут быть без ограничений.

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

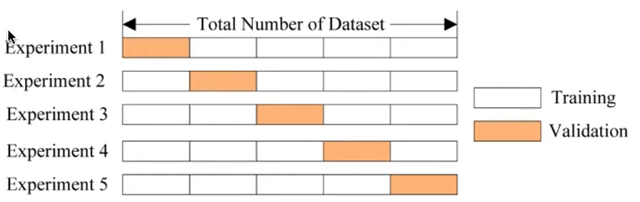
**Кросс-валидация** - это метод оценки качества алгоритма машинного обучения, который помогает уменьшить вероятность переобучения модели. Она заключается в разделении обучающего набора данных на несколько фолдов (частей) и последующем обучении модели на каждом из них, при этом тестирование происходит на оставшихся фолдах. Таким образом, каждый фолд используется как тестовый набор данных, а оставшиеся фолды - как обучающий.

from sklearn.model\_selection import cross\_val\_score

scores = cross\_val\_score(model, X, y, cv=5)

**Схема работы:**

* Данные разделяются на подмножества из k элементов (обычно k=5 или k=10).
* Модель обучается на k-1 элементах и оценивается на оставшемся подмножестве.
* Этот процесс повторяется k раз, каждый раз выбирается другое тестовое подмножество.
* После завершения всех k итераций вычисляется средняя оценка производительности модели.



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

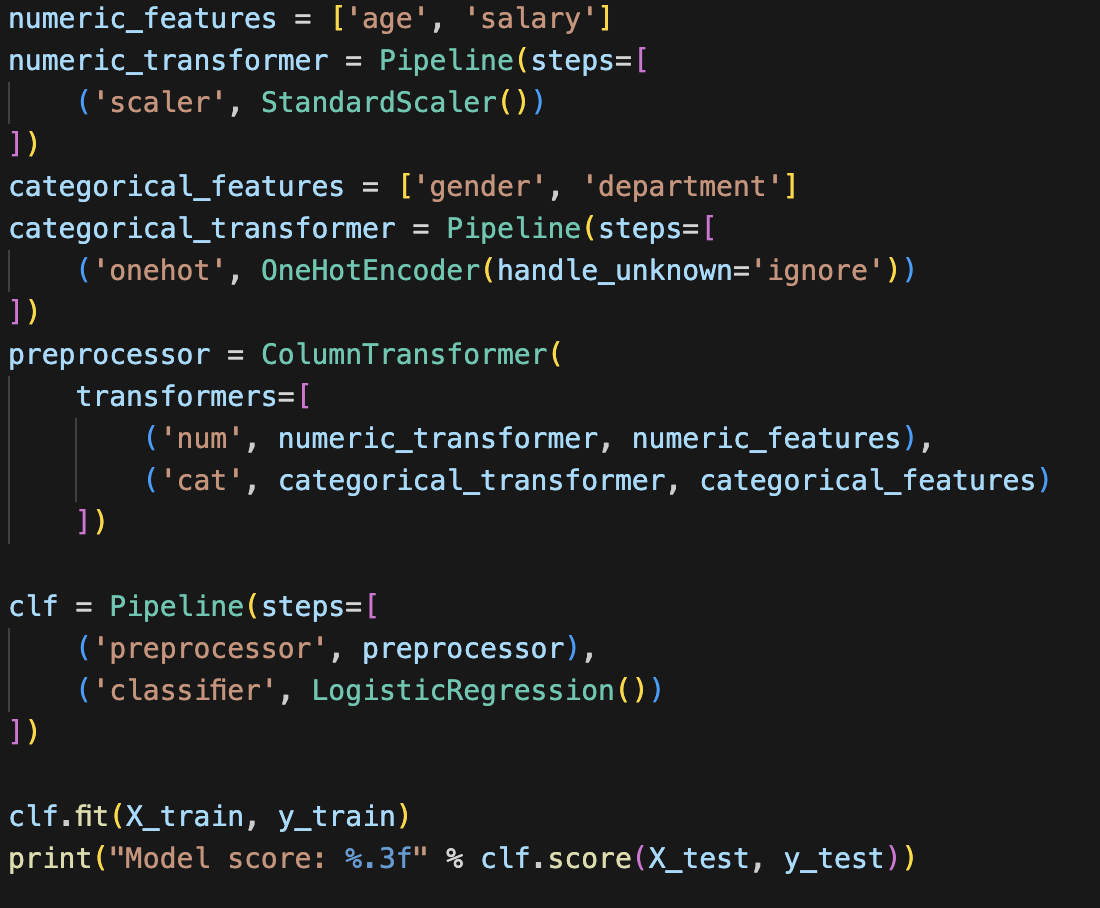
Конвейеры (Pipeline) в библиотеке sklearn предназначены для инкапсуляции последовательности шагов обработки данных и моделирования в единый объект. Это позволяет стандартизировать процесс обучения и предсказания, делает код более читаемым и уменьшает риск ошибок, связанных с неправильным порядком выполнения операций.

### **Назначение конвейеров**

1. **Автоматизация рабочего процесса**: Конвейеры позволяют автоматизировать последовательные шаги обработки данных, от масштабирования и кодирования признаков до обучения модели.
2. **Согласованность предобработки**: Обеспечивают согласованную предобработку данных как на этапе обучения, так и на этапе предсказания, что устраняет расхождения в обработке данных.
3. **Удобство использования и повторяемость**: Упрощают экспериментирование с различными комбинациями шагов предобработки и моделей, обеспечивая легкую повторяемость исследований.
4. **Помощь в валидации**: Упрощает процесс кросс-валидации, поскольку каждый шаг предобработки выполняется внутри каждой итерации валидации, предотвращая утечку данных.

### **Пример использования конвейера в sklearn**

Давайте рассмотрим пример конвейера, который выполняет стандартизацию данных, преобразование категориальных признаков и обучение модели логистической регрессии:



# **Использование методов визуализации данных для предварительного анализа.**

Визуализация данных может быть выполнена с помощью различных методов и инструментов, таких как:

* Графики: для отображения распределения значений по оси координат
* Диаграммы: для сравнения значений между категориями
* Матрицы корреляции: для определения взаимосвязей между переменными
* Кластеризация: для группировки объектов по схожим признакам

**Примеры визуализации данных**

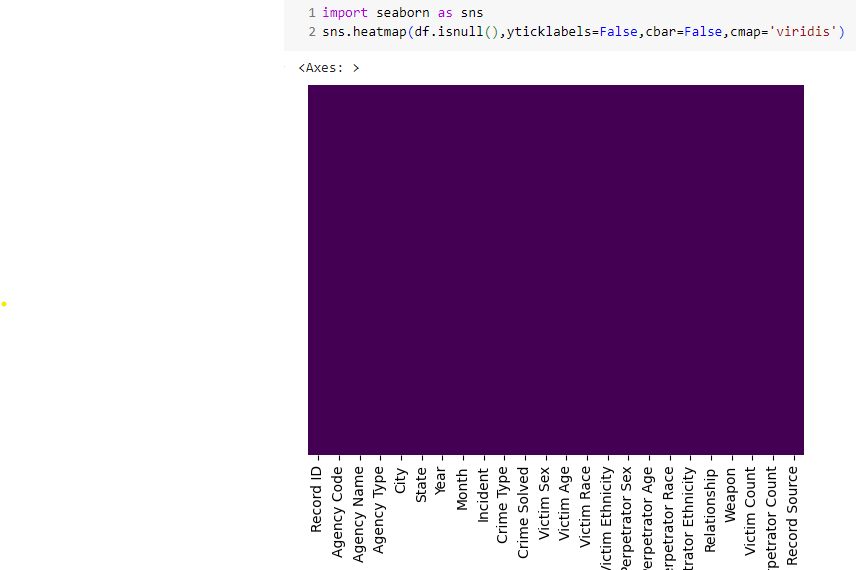
* *Визуализация распределения значений*: для отображения распределения значений по оси координат, что может помочь в обнаружении аномалий и паттернов.
* *Визуализация взаимосвязей*: для отображения взаимосвязей между переменными, что может помочь в определении важных признаков и исключении не важных.
* *Визуализация кластеризации*: для группировки объектов по схожим признакам, что может помочь в обнаружении групп и кластеров в данных.

**Влияние визуализации данных на машинное обучение**

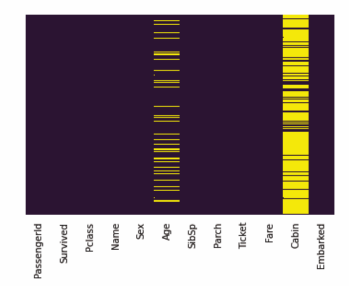
*Визуализация данных может иметь следующие влияния на машинное обучение:*

* Улучшение качество модели: визуализация данных может помочь в обнаружении аномалий и паттернов, что может улучшить качество модели и уменьшить риск ошибок.
* Улучшение понимания данных: визуализация данных может помочь в понимании структуры и свойств данных, что может улучшить выбор алгоритма и параметров модели.
* Улучшение интерпретации результатов: визуализация данных может помочь в интерпретации результатов модели, что может улучшить понимание и использование результатов.

Пример визуализации нулевых значений в датасете (тут их нет):



(Тут есть)



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

**Цели корреляционного анализа**

Основными целями корреляционного анализа признаков в задачах машинного обучения являются:

* Измерение силы связи между признаками для выявления наиболее значимых факторов, влияющих на целевую переменную.
* Отбор наиболее информативных признаков для построения регрессионных и классификационных моделей.
* Выявление мультиколлинеарности (сильной корреляции) между предикторами, что может негативно сказываться на качестве модели.

**Методы исследования коррелированности**

Существует несколько основных методов оценки коррелированности признаков:

* Коэффициент корреляции Пирсона - показывает линейную зависимость между двумя признаками. Вычисляется как отношение ковариации к произведению стандартных отклонений.
* Ранговые коэффициенты корреляции (Спирмена, Кендалла) - используются для оценки монотонной зависимости, в том числе нелинейной. Основаны на рангах (порядковых местах) значений признаков.
* Взаимная информация - показывает, насколько хорошо можно предсказать значения одного признака, зная значения другого. Позволяет выявлять нелинейные зависимости.
* Дисперсионный анализ - оценивает, насколько значимо влияние одного признака на другой. Основан на сравнении дисперсий.

**Выводы и применение**

На основе анализа коррелированности признаков можно сделать следующие выводы и применения в машинном обучении:

* Удалить сильно коррелированные признаки, оставив только наиболее информативные, чтобы избежать мультиколлинеарности.
* Отобрать наиболее значимые признаки для построения регрессионных и классификационных моделей.
* Выявить скрытые факторы, объясняющие связи между признаками (факторный анализ).
* Построить диаграммы рассеяния для визуализации зависимостей между признаками.
* Использовать коэффициенты корреляции в качестве весов при линейных комбинациях признаков.

Таким образом, исследование коррелированности признаков является важным этапом подготовки данных для машинного обучения, позволяющим повысить качество и интерпретируемость моделей.

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

Решкалирование данных — это процесс приведения данных к определенным форматам или масштабам для удобства анализа и обработки. В машинном обучении это особенно важно, так как модели могут быть чувствительны к масштабу данных и могут быть склонны к переобучению или недообучению из-за неадекватного масштаба.

В данных упоминаются следующие виды решкалирования:

1. Порядковые шкалы: используются для данных, которые имеют определенный порядок или ранжирование.
2. Интервальные шкалы: используются для данных, которые имеют определенный интервал или диапазон.
3. Метрические шкалы: используются для данных, которые имеют количественные значения и могут быть измерены.
4. Номинальные шкалы: используются для данных, которые не имеют количественных значений и не могут быть измерены.
5. Ранговые шкалы: используются для данных, которые имеют ранжирование или сравнение.

Решкалирование данных имеет важное значение в машинном обучении, так как оно позволяет:

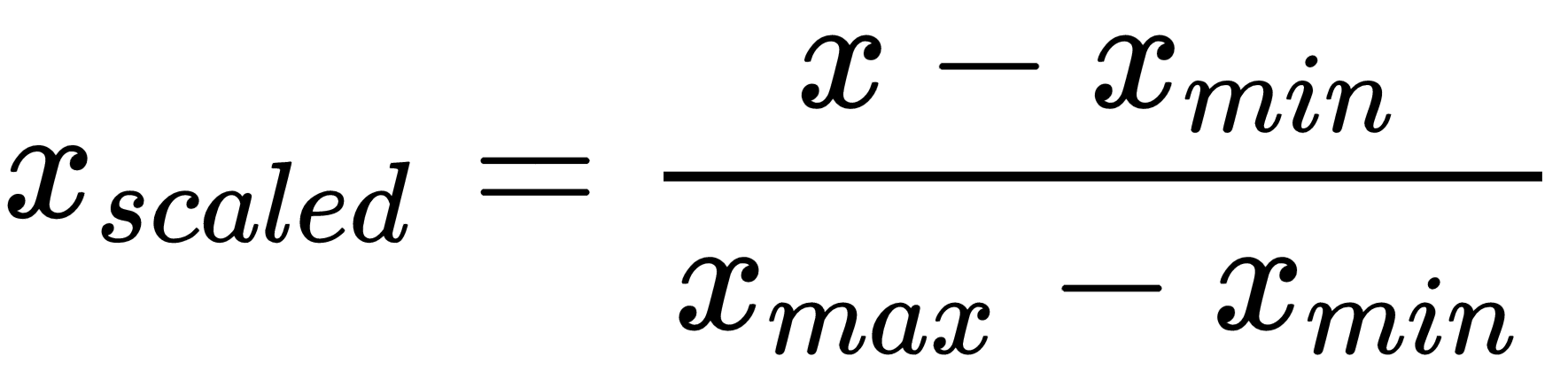
1. Улучшить качество модели: правильное масштабирование данных может улучшить качество модели и уменьшить риск переобучения или недообучения.

2. Упростить анализ данных: решкалирование данных может упростить анализ данных и помочь в обнаружении важных признаков и закономерностей.

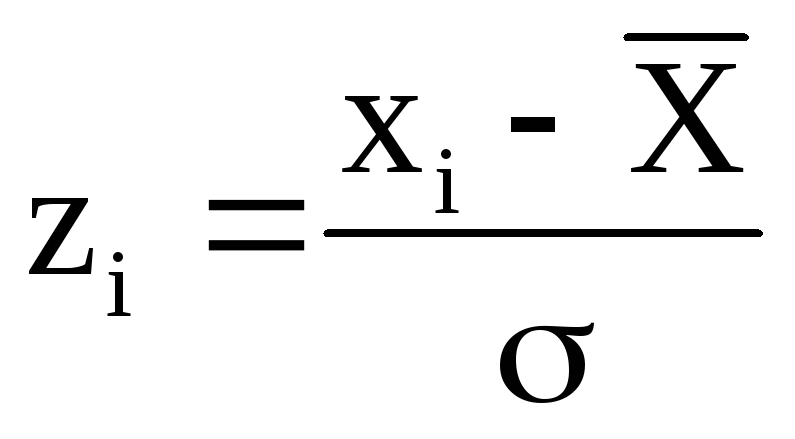
3. Увеличить эффективность модели: правильное масштабирование данных может увеличить эффективность модели и уменьшить время обучения.

Нормализация и стандартизация данных необходимы для приведения данных к единому масштабу, что особенно важно для алгоритмов машинного обучения.

Мин-Макс нормализация приводит значения к диапазону от 0 до 1. Этот метод полезен для данных, которые имеют разные масштабы и диапазоны значений. Нормализация помогает избежать доминирования признаков с большими значениями над признаками с меньшими значениями.



Стандартизация приводит данные к нормальному распределению с средним 0 и стандартным отклонением 1. Этот метод полезен для данных, которые следуют нормальному распределению, и помогает улучшить работу алгоритмов машинного обучения, которые чувствительны к масштабу данных.



где q - среднеквадратическое отклонение

 среднее

# **Преобразование категориальных признаков в числовые.**

Категориальные признаки часто выражаются строковыми данными и не подходят для использования в машинном обучении, их преобразуют в численные.

Методы которые используются для преобразования:

**LabelEncoder –** Категориальные признаки (слова) просто нумеруются

**Binary Encoding –** Бинарное кодирование признаков ( 0 или 1)

**One-hot-encoding** – Слова, которые находятся в определенном алфавите кодируются вектором, размерность которого составляет колличесво слов в заданном алфавите, состоит из всех 0 и одной 1, которая ставится на место, соответсвующего слова в алфавите. Таким образом в датасет добавляются новые столбцы для того чтобы корректно представить вектор.

Этот метод имеет много недостатков, такие как:

· Не отражает смысл слова

· Не дает понять степень схожести слов

· Много лишней памяти

· Словарь ограничен, проблема добавляения новых слов

**Hash Encoding** – принцип такой же как и у OHE, но колличество новых столбцов для представления вектора мы задаем сами. Всегда используется для преобразования объекта произвольной размерности в объект фиксированной размерности.

**Target encoding** – Катеогриальные признаки группируются и вычисляется среднее значение целевой переменной в этих группах, которым в последствии и заманяется категориальный признак. Однако такой метод приводит к сильной корреляции с целевой переменной и переобучению, так что его нужно использовать с осторожностью.

# **Методы визуализации данных для машинного обучения.**

### **1. Гистограммы и ящики с усами (Boxplots)**

* **Гистограммы** позволяют анализировать распределение числовых данных, показывая частоту значений данных в различных интервалах.
* **Ящики с усами** иллюстрируют распределение данных через пять числовых характеристик: минимум, первый квартиль, медиану, третий квартиль и максимум. Они также помогают выявлять выбросы.

### **2. Диаграммы рассеяния (Scatter Plots)**

* Диаграммы рассеяния отлично подходят для визуализации взаимосвязей между двумя числовыми переменными и могут быть расширены с использованием цветов или размеров точек для представления дополнительных переменных или классов.

### **3. Парные диаграммы (Pair Plots)**

* Парные диаграммы позволяют одновременно отображать несколько диаграмм рассеяния для пар признаков. Это особенно полезно при исследовании корреляций и взаимосвязей между несколькими переменными.

### **4. Тепловые карты корреляции (Correlation Heatmaps)**

* Тепловые карты используются для визуализации матрицы корреляции, показывая степень корреляции между различными признаками. Цвета карты помогают легко идентифицировать высокие и низкие корреляции.

### **5. Визуализация многомерных данных**

* **Метод главных компонент (PCA)** применяется для уменьшения размерности данных, после чего результаты можно визуализировать как двумерную или трехмерную диаграмму.
* **t-SNE (t-distributed Stochastic Neighbor Embedding)** и **UMAP (Uniform Manifold Approximation and Projection)** — современные методы для визуализации высокоразмерных данных в низкоразмерных пространствах, часто используемые для визуальной кластеризации сложных наборов данных.

### **6. Оценка производительности моделей**

* **Кривые ошибок (ROC curves)** и **Precision-Recall curves** используются для оценки качества классификационных моделей на разных порогах.
* **Диаграмма важности признаков (Feature Importance Plots)** помогает оценить, какие признаки оказывают наибольшее влияние на прогнозы модели.

### **7. Визуализация структуры деревьев решений**

**Деревья решений** могут быть визуализированы для понимания, как модель принимает решения, что помогает в интерпретации модели.

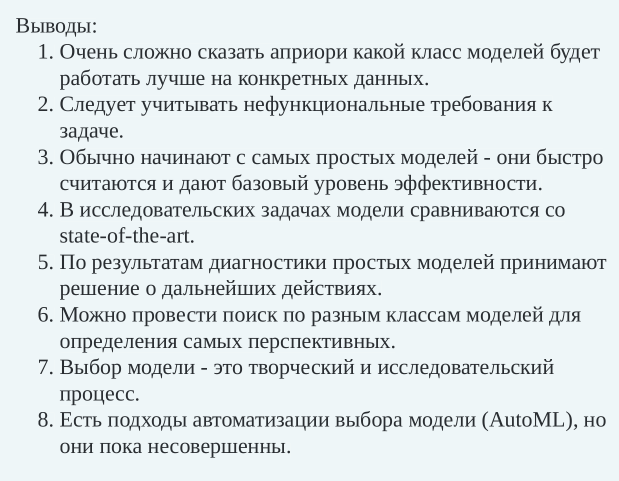
# **Задача выбора модели. Оценка эффективности, валидационный набор.**

Проблема выбора модели чаще всего решается перебором.

Если решается известная задача, то обязательно надо проводить анализ

литературы и прошлой истории решения подобных проблем.

Если же никакой информации о предпочтительных видах моделей у исследователя нет, то самым естественным первым шагом будет построение самых простых моделей. Здесь имеются в виду простые как по классу сложности, так и по вычислительным затратам. Логичным выбором будет использование линейной или логистической регрессии (в зависимости от поставленной задачи) и дерева решения (иногда используют его ансамблевый вариант - случайный лес). Эти модели хороши тем, что довольно быстро работают, даже на относительно больших массивах данных.



Валидационный набор в машинном обучении — это часть данных, используемых для проверки и оценки качества модели, построенной на основе машинного обучения.

Валидационный набор отличается от обучающего и тестового наборов следующим образом:

* Обучающий набор: используется для обучения модели, то есть для настройки параметров и весов модели, чтобы она могла наиболее точно предсказывать результаты.
* Тестовый набор: используется для оценки точности модели на независимом от обучающего набора данных. Если модель хорошо подходит для тестового набора, то это указывает на минимальное переобучение.
* Валидационный набор: используется для оценки качества модели на независимом от обучающего и тестового наборов данных. Валидационный набор помогает определить, насколько модель может быть полезной в реальных условиях, не только на обучающем и тестовом данных.

Валидация модели машинного обучения включает в себя проверку на:

* соответствие модели требованиям задачи;
* качество используемых данных;
* влияние факторов на результаты;
* соответствие глубины исторических данных нормативным требованиям;
* определение целевого события.

Валидационный набор важен для определения, насколько модель может быть полезной в реальных условиях, и для предотвращения переобучения модели

# **Кривые обучения для диагностики моделей машинного обучения.**

**Кривые обучения** - это диаграммы, которые отображают зависимость производительности модели машинного обучения от объема обучающих данных. Они помогают анализировать, как модель изменяется при увеличении размера набора данных и обнаруживать пороговые значения, после которых качество модели начинает стагнировать или ухудшаться.

**Кривые обучения могут быть полезны для следующих целей:**

1. Оценка качества модели: Кривые обучения позволяют оценить, насколько модель улучшается с увеличением размера набора данных и когда она достигает оптимального уровня.
2. Выбор порогового размера данных: Пороговый размер набора данных, после которого качество модели начинает стагнировать, может быть использован для оптимизации размера обучающего набора.
3. Диагностика моделей: Кривые обучения могут помочь обнаруживать проблемы, такие как переобучение, мультиколлинеарность или низкое качество разметки данных.
4. Выбор методов регуляризации: Регуляризация может быть использована для предотвращения переобучения, и кривые обучения помогают оценить эффективность различных методов регуляризации.
5. Выбор подходящей архитектуры модели: Кривые обучения могут помочь в выборе подходящей архитектуры модели, учитывая, какие изменения в модели приводят к наибольшей улучшению производительности.

# **Регуляризация моделей машинного обучения. Назначение, виды, формализация.**

Регуляризация - очень полезный и распространенный на практике математический прием, который позволяет гибко и очень удобно управлять сложностью параметрических моделей машинного обучения.

Допустим, у нас есть набор данных и мы не знаем, модель какого вида подойдет к нему лучше - линейная или квадратичная. При прочих равных, квадратичная модель всегда даст меньшую величину ошибки и поэтому будет предпочтительнее. Но малая ошибка - не всегда показатель качества модели.

Функция ошибки - это по сути система штрафов. Каждый раз, когда модель ошибается, к значению ошибки прибавляется величина, пропорциональная отклонению. Это - штраф за ошибку модели. Но мы можем включить в ошибку и уровень сложности модели. Мы добавляем к ошибке некоторые штрафы за высокие значения параметров модели. Тогда алгоритм обучения, который минимизирует именно функцию ошибки будет не так сильно отдавать сложным переобученным моделям. Небольшие ошибки модели могут быть скомпенсированы тем, что модель становится более простой.

Существует несколько конкретных реализаций регуляризаций. Регуляризация обычно не затрагивает свободный коэффициент, так как он не влияет на сложность модели.

В этом и состоит идея регуляризации - модификация функции ошибки таким образом, чтобы штрафовать сложные модели в пользу более простых.

Можно заметить, что регуляризация - это способ искусственно понизить сложность модели. То есть при использовании регуляризации берут модель более сложную, которая в "чистом" виде будет явно переобучаться на имеющихся данных. Но за счет этих дополнительных штрафов, ее сложность принижают. Причем, самое удобное в регуляризации то, что она параметрическая. То есть соотношением "силы” штрафа за ошибки и штрафа за сложность легко управлять, введя множитель - так называемый параметр регуляризации. Чем он больше, тем сильнее штрафуются сложные модели, то есть даже большие ошибки модели могут быть скомпенсированы небольшим понижением сложности.

Регуляризация настолько удобна и универсальна, что большинство библиотечных реализаций моделей уже реализуют встроенную регуляризацию. Причем это относится равно как к моделям классификации, так и к моделям регрессии, это прием не зависит от типа задачи обучения с учителем. За счет способности уменьшить сложность любой модели регуляризация является одним их основных способов борьбы с переобучением.

Обратите внимание, что мы в основном говорим именно о борьбе с переобучением. Борьба с недообучением в основном сводится к использованию более сложной модели. В настоящее время разработаны настолько сложные модели, что узким местом современного машинного обучения становятся вычислительные мощности и доступность данных.

Выводы:

1. Регуляризация - это способ искусственно ограничить вариативность моделей.

2. При использовании регуляризации можно применять более сложные модели и снижать склонность к переобучению.

3. Регуляризация модифицирует функцию ошибки модели, добавляя в нее штрафы за повышение сложности.

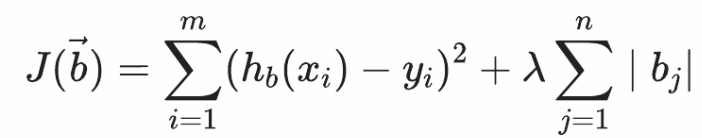
4. Основная идея регуляризации - отдавать предпочтение низким значениям параметров в модели.

5. Регуляризация обычно не затрагивает свободный коэффициент bо.

6. Регуляризация обычно параметрическая, можно управлять ее степенью.

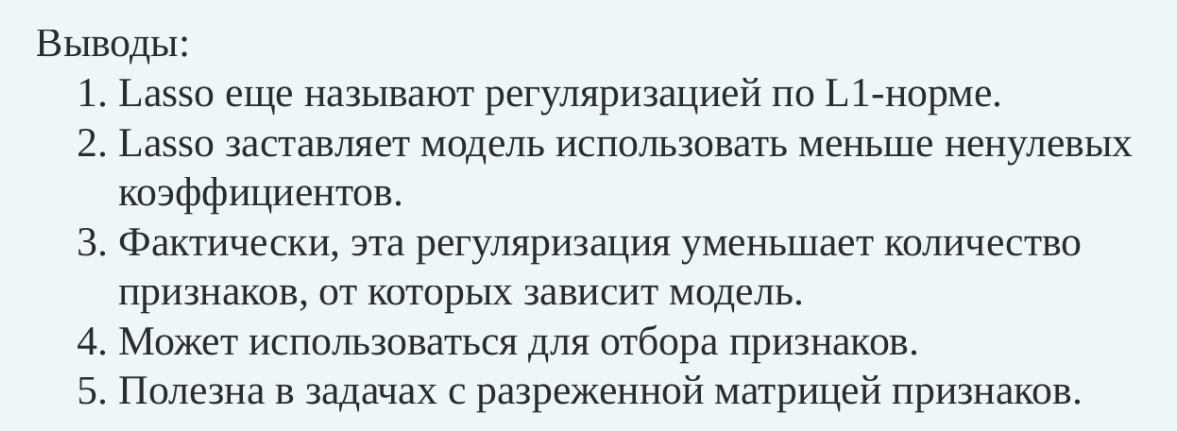
Существует несколько основных видов регуляризации:

1. L1 регуляризация (Lasso): добавляет к функции потерь сумму абсолютных значений весов модели, что приводит к разреженным решениям и отбору признаков.

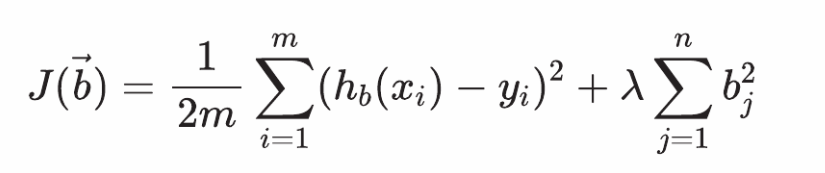


Из-за своей специфической формализации, модель лассо стремиться обратить в ноль как можно больше параметров модели. Это значит, что если какой-то признак не оказывает сильного влияния на целевую переменную, то модель с L1-регуляризацией с большой долей вероятности "занулит" этот признак. Это бывает очень полезно

в задачах, где в данных присутствует больше количество ненужной информации. Такие задачи еще часто называют разреженными. К ним часто относятся, например, задачи анализа текстов.

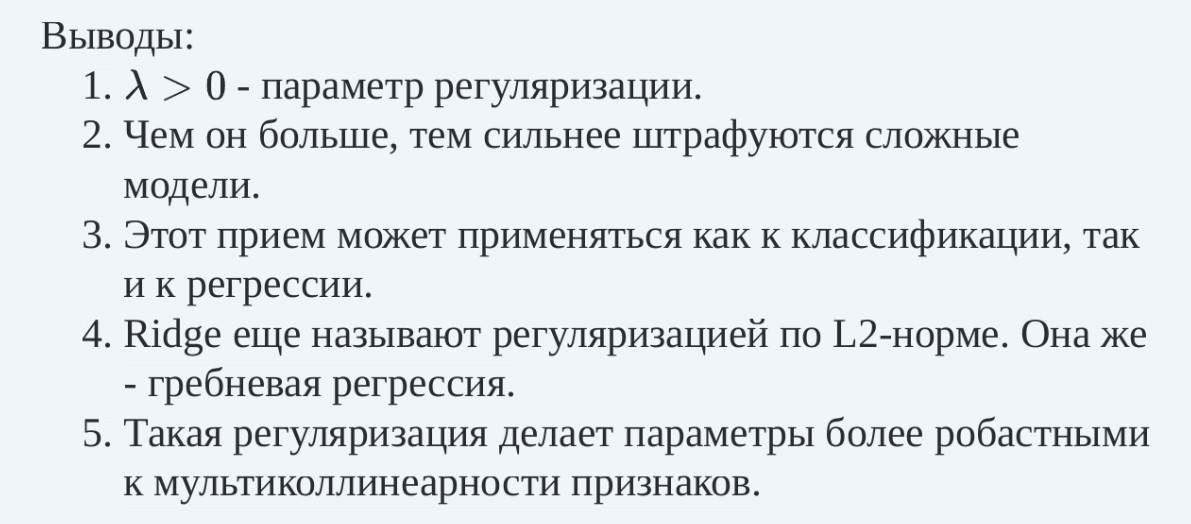


2. L2 регуляризация (Ridge): добавляет к функции потерь сумму квадратов весов модели, что уменьшает переобучение, но не приводит к разреженным решениям.

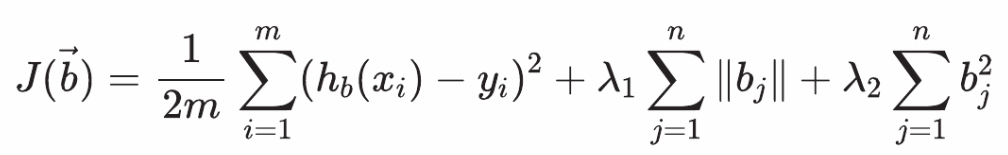


Как мы можем видеть, что функция ошибки аналогична обычной модели регрессии, но имеет одно дополнительное слагаемое - регуляризационный член. Это сумма квадратов значений параметров модели (начиная с первого). Эта сумма умножается на специальный параметр - параметр регуляризации . Он как раз отвечает

за "силу” регуляризации модели. В предельном случае, когда Х = 0, мы имеем обычную нерегуляризованную регрессию. Чем больше этот параметр, тем сравнительно больший вклад в ошибку дают отклонения параметров модели.

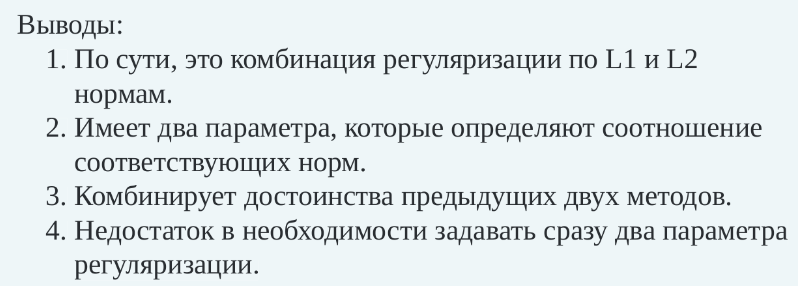


3. Elastic Net: линейная комбинация L1 и L2 регуляризации, позволяющая сочетать их преимущества.



Как мы видим, функция ошибки такой модели комбинирует подходы гребневой и лассо-регрессии. То есть включает в себя регуляризацию и по L1 и по L2 нормам. Поэтому в этой модели присутствует целых два параметра регуляризации, причем они независимы друг от друга и задают не только соотношение регуляризации

того или иного типа и классической функции ошибки, но и соотношение силы двух типов регуляризации между собой.



# **Проблема сбора и интеграции данных для машинного обучения.**

**Проблема сбора и интеграции** данных для машинного обучения заключается в том, что данные не всегда представлены в удобном для анализа и обучения виде. Они могут быть разрозненными, неструктурированными, содержать ошибки, пропуски или быть недоступными из-за ограничений доступа.

Для решения этой проблемы необходимо провести процесс сбора, очистки, обработки и интеграции данных.

Проблемы с которыми можно столкнутся при сборе данных для машинного обучения:

* Качество данных: Недостаточное качество данных может привести к неаккуратным или неполным данным, что может негативно повлиять на производительность модели
* Скудость данных: Ограничение или нехватка данных может привести к смещенным моделям, перегрузкам или недостаточным моделям
* Смещение данных: Проблемы с распределением данных
* Фрагментация данных: Данные могут быть разбросаны по нескольким источникам, что делает их менее доступными

**Проблема интеграции данных** - ситуация, когда необходимо объединить данные из различных источников, форматов и структур в единое целое. Это может быть проблематично как минимум по следующим причинам:

* Разный формат данных
* Различные структуры данных

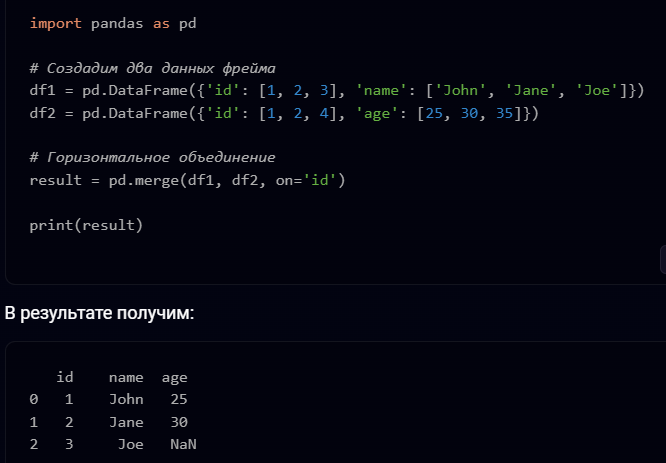
Загрузка данных чаще всего выглядит так:

data = pd.read\_csv('data.csv')

data = pd.read\_excel('data.xlsx')

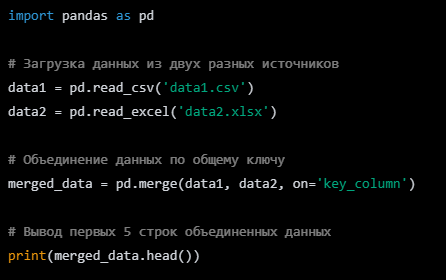
* для различных объединений

Горизонатльное объединение:



Вертикальное объединение





# **Понятие чистых данных и требования к данным.**

Чистые данные — это данные, которые были обработаны, очищены и структурированы таким образом, чтобы максимально уменьшить возможные искажения при их анализе и использовании в машинном обучении и других данных-интенсивных приложениях. Чистые данные критически важны для достижения надежных и точных результатов в аналитических проектах.

### **Требования к чистым данным**

Чистота данных может быть охарактеризована следующими аспектами:

1. **Полнота**: Все необходимые данные доступны, пропущенные значения обработаны должным образом.
2. **Консистентность**: Данные свободны от внутренних противоречий; данные, представленные в разных местах, согласованы между собой.
3. **Точность**: Данные точно отражают реальные значения. Ошибки и неточности должны быть исправлены.
4. **Актуальность**: Данные актуальны и релевантны на момент их использования. Устаревшие данные должны быть обновлены или отмечены как устаревшие.
5. **Релевантность**: Данные должны быть соответствующими и подходящими для анализа или задачи, для которой они используются.
6. **Уникальность**: В данных не должно быть ненужных дубликатов. Повторяющиеся записи должны быть идентифицированы и удалены, если это необходимо.
7. **Интерпретируемость**: Данные должны быть понятны и читаемы. Это включает в себя правильное и ясное определение всех переменных и единиц измерений.

### **Процессы очистки данных**

Для достижения чистоты данных используются различные методы и процессы:

**Обработка пропущенных значений**: Замена, удаление или восстановление пропущенных значений в зависимости от контекста и важности данных.

**Корректировка ошибок**: Исправление очевидных ошибок или неточностей, например, опечаток, некорректных форматов данных или неправильных значений.

**Удаление дубликатов**: Идентификация и удаление повторяющихся записей для предотвращения искажения анализа.

**Валидация и проверка**: Проверка данных на соответствие заданным правилам или стандартам, например, проверка кодов почтовых индексов или форматов телефонных номеров.

**Нормализация и стандартизация**: Приведение данных к общему формату, единицам измерения или шкале для обеспечения согласованности.

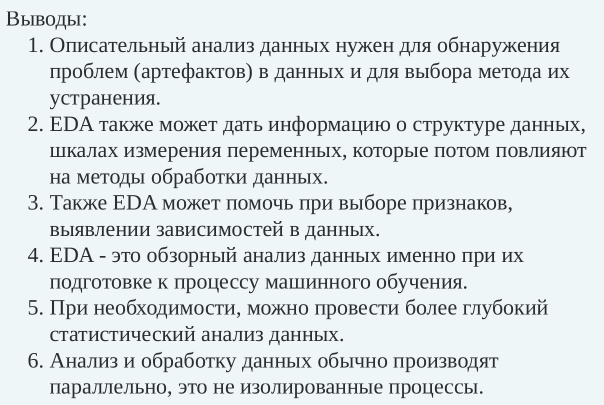
**Обогащение данных**: Дополнение данных информацией из других источников для повышения их полноты и релевантности.

# **Основные задачи описательного анализа данных.**

EDA - exploratory data analysis

Основные задачи описательного анализа данных:

* Числовое описание основных характеристик выборки, таких как средние значения, медианы, стандартные отклонения и т.д.Обеспечивает количественное описание набора данных.
* Выявление закономерностей и связей между переменными в данных. Помогает понять, как связаны переменные в данных.
* Обнаружение ошибок, выбросов и аномалий в данных. Позволяет выявить исключения и ошибки в данных.
* Подготовка данных для дальнейшего более глубокого статистического анализа. Служит предварительным шагом перед более сложным анализом, обеспечивая основу для дальнейшего исследования.
* Визуализация данных с помощью таблиц, диаграмм и графиков. Включает в себя числа, таблицы, диаграммы и графики для наглядного представления данных.



# **Полиномиальные модели машинного обучения.**

**Полиномиальные модели** машинного обучения представляют собой гибкий инструмент, который позволяет моделировать нелинейные зависимости между переменными. Они широко используются в задачах, где линейные модели не способны адекватно описать данные. Вот ключевые аспекты полиномиальных моделей машинного обучения, основанные на предоставленных источниках:

**Принцип работы:** Полиномиальная регрессия позволяет обучать линейную модель на нелинейных данных, добавляя дополнительные функции к данным для учета нелинейности. Это позволяет модели адаптироваться к сложным зависимостям и заполнять пробелы, которые не удается охватить линейной моделью.

**Преимущества:** Полиномиальные модели могут быть эффективны в случаях, когда данные имеют сильную нелинейность, и линейные модели не могут обеспечить достаточную точность. Они предоставляют большую свободу при работе с различными наборами данных и могут быть использованы там, где линейные модели недостаточны.

**Риск переобучения:** Гибкость полиномиальных моделей может привести к риску переобучения, особенно при недостаточном контроле за сложностью модели. Поэтому важно балансировать гибкость модели с риском переобучения, особенно при работе с необученными данными.

**Применение:** Полиномиальные модели машинного обучения широко используются в задачах, где ожидается нелинейная связь между переменными, таких как анализ временных рядов, прогнозирование, и другие задачи, где линейные модели оказываются недостаточными.

*Выводы:*

1. Добавление полиномиальных признаков возможно как к

регрессионным, так и к классификационным моделям.

2. Полиномиальная регрессия позволяет охватывать

нелинейные зависимости атрибутов и целевой

переменной.

3. Полиномиальная классификация позволяет очерчивать

нелинейные границы принятия решений.

4. Здесь и далее: атрибуты - характеристики объектов,

данные в датасете; признаки - компоненты вектора,

подающегося на вход модели машинного обучения.

5. Полиномиальные модели универсальны, но очень дороги

при высоких порядках полинома.

# **Основные виды преобразования данных для подготовки к машинному обучению.**

* Первичный анализ данных (Описательная статистика):

Обзор размера набора данных

Анализ типов переменных

Выявление пропущенных значений

Обнаружение дубликатов

Изучение основных статистических характеристик (среднее, медиана, стандартное отклонение и т.д.)

* Предобработка данных:

Очистка данных (удаление ошибок, дубликатов, выбросов)

Обработка пропущенных значений (заполнение, удаление)

Преобразование типов данных

Нормализация и стандартизация данных

* Разведывательный анализ данных (Exploratory Data Analysis):

Визуализация данных (гистограммы, графики рассеивания, корреляционные матрицы)

Выявление закономерностей и аномалий в данных

Проверка гипотез о взаимосвязях между признаками

* Создание признаков (Feature Engineering):

Генерация новых признаков на основе существующих

Преобразование признаков (категориальные в числовые, логарифмирование и т.д.)

Отбор наиболее информативных признаков

* Отбор признаков (Feature selection):

Фильтрационные методы (корреляция, ANOVA, mutual information и т.д.)

Обёрточные методы (recursive feature elimination, sequential feature selection и т.д.)

Встроенные методы (regularization, tree-based feature importance и т.д.)

* Подготовка данных (Data Preparation):

Нормализация данных (min-max scaling, z-score scaling и т.д.)

Обработка категориальных признаков (one-hot encoding, label encoding и т.д.)

Балансировка классов (oversampling, undersampling, SMOTE и т.д.)

Разделение данных на обучающую, валидационную и тестовую выборки

# **Задача выбора признаков в машинном обучении.**

Часто наборы данных, с которыми приходится работать, содержат большое количество признаков, число которых может достигать нескольких сотен и даже тысяч. При построении модели машинного обучения не всегда понятно, какие из признаков действительно для неё важны (т.е. имеют связь с целевой переменной), а какие являются избыточными (или шумовыми). Удаление избыточных признаков позволяет лучше понять данные, а также сократить время настройки модели, улучшить её точность и облегчить интерпретируемость. Иногда эта задача и вовсе может быть самой значимой, например, нахождение оптимального набора признаков может помочь расшифровать механизмы, лежащие в основе исследуемой проблемы. Это может быть полезным для разработки различных методик, например, банковского скоринга, поиска фрода или медицинских диагностических тестов. Методы отбора признаков обычно делят на 3 категории: фильтры (filter methods), встроенные методы (embedded methods) и обёртки (wrapper methods).

# 1. Фильтры

# Методы фильтрации применяются до обучения модели и, как правило, имеют низкую стоимость вычислений. К ним можно отнести визуальный анализ (например, удаление признака, у которого только одно значение, или большинство значений пропущено), оценку признаков с помощью какого-нибудь статистического критерия (дисперсии, корреляции, X2 и др.) и экспертную оценку (удаление признаков, которые не подходят по смыслу, или признаков с некорректными значениями).

2. Встроенные методы

Встроенные методы выполняют отбор признаков во время обучения модели, оптимизируя их набор для достижения лучшей точности. К этим методам можно отнести регуляризацию в линейных моделях (обычно L1) и расчёт важности признаков в алгоритмах с деревьями. Отметим, что для линейных моделей требуется масштабирование и нормализация данных.

# модели машинного обучения. Виды ансамблирования.

# Конвейеризация моделей машинного обучения.

# Методы векторизации текстов для задач машинного обучения.

# Представление графической информации в моделях машинного обучения.

# Задачи без учителя. Кластеризация. Метод k средних.

# Задачи без учителя. Обнаружение аномалий.

# Задачи без учителя. Понижение размерности. Метод PCA.

# Воспроизводимость алгоритма преобразования данных в машинном обучении.

# Случайный лес как ансамблевая модель машинного обучения.

# Частичное обучение с учителем.

# 