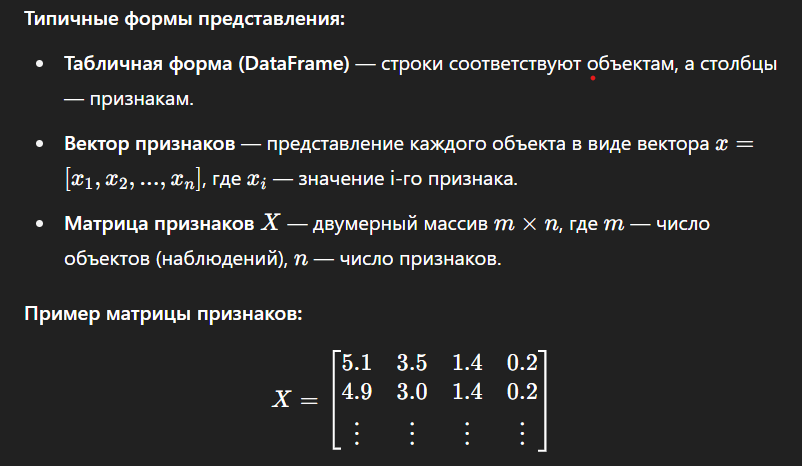
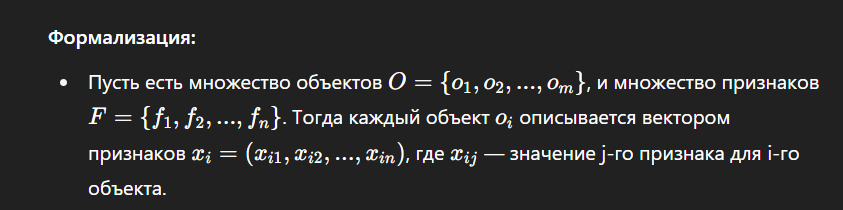
1. **Представление данных**

Представление данных (data representation) — это процесс организации и структурирования данных в формате, который подходит для анализа, обработки и использования алгоритмами машинного обучения и статистики. Данные могут быть представлены в виде таблиц, матриц, списков, графов и других структур.

****

**Типы признаков:**

* **Числовые признаки (numeric features):** непрерывные (рост, вес) или дискретные (количество детей).
* **Категориальные признаки (categorical features):** пол, страна, цвет (могут быть преобразованы в числовые через one-hot encoding или label encoding).
* **Бинарные признаки:** два возможных значения (0 или 1).
* **Текстовые данные:** строки, которые нужно преобразовать в числовой вид (например, с помощью Bag of Words, TF-IDF и т.д.).

**Зачем это нужно:**  
Правильное представление данных влияет на:

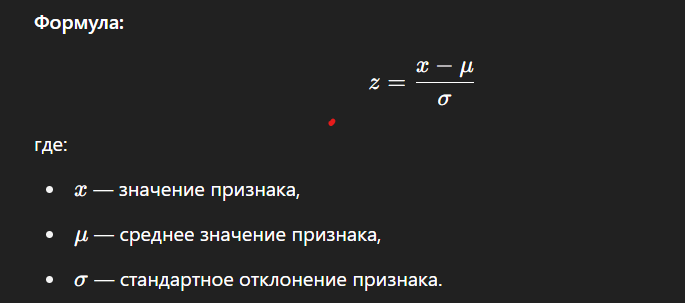
* эффективность алгоритма;
* возможность обработки (например, SVM требует числовые данные);
* точность модели.

**2. Предобработка данных. Стандартизация и нормализация**

**Определение:**  
Предобработка данных — это этап подготовки данных к обучению моделей. Он включает очистку, преобразование и масштабирование признаков.

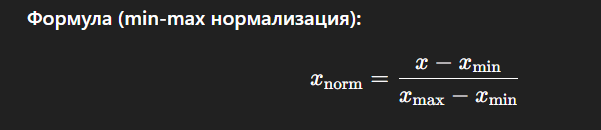
**Зачем нужна:**  
Большинство алгоритмов чувствительны к масштабу данных, пропускам и выбросам. Без предобработки результат может быть искажен.

**Стандартизация (Standardization):**  
Преобразование признаков так, чтобы они имели **среднее 0 и стандартное отклонение 1**.

**Алгоритм стандартизации:**

1. Вычислить среднее и стандартное отклонение признака.
2. Преобразовать каждое значение по формуле выше.

**Нормализация (Normalization):**  
Приведение значений признаков к диапазону [0, 1].

****

**Алгоритм нормализации:**

1. Найти минимум и максимум признака.
2. Применить формулу.

**Термины:**

* **Скалирование (scaling)** — изменение масштаба данных.
* **Центрирование (centering)** — вычитание среднего.

**3. Основные типы данных**

**Определение:**  
Типы данных описывают природу признаков. Корректное понимание типов важно для выбора алгоритма.

**Классификация:**

* **Числовые (quantitative):**
  + *Непрерывные* (float): рост, вес, температура.
  + *Дискретные* (int): количество детей, количество товаров.
* **Категориальные (qualitative):**
  + *Номинальные*: цвета, профессии (не имеют порядка).
  + *Порядковые*: уровень образования, размер одежды (имеют порядок).
* **Бинарные**: да/нет, 0/1.
* **Текстовые (строки)**: имена, адреса (нуждаются в преобразовании).

**Форматы в pandas:**

* int64, float64, object, category, bool.

**Алгоритмы преобразования:**

* **Label Encoding** — замена категорий числами.
* **One-Hot Encoding** — создание бинарных признаков.

**4. Визуализация данных**

**Определение:**  
Визуализация данных — это процесс представления информации в графической форме для выявления закономерностей, выбросов и структуры данных.

**Зачем нужна:**  
Позволяет «увидеть» данные, что помогает в их понимании и анализе.

**Основные методы:**

* **Гистограммы (histograms)** — распределение признаков.
* **Диаграммы рассеяния (scatter plots)** — взаимосвязь между признаками.
* **Ящики с усами (box plots)** — выявление выбросов.
* **Корреляционная матрица (heatmap)** — визуализация корреляций между признаками.

**Алгоритмы:**

* Построить гистограмму для оценки распределения.
* Построить диаграмму рассеяния для анализа связи между двумя признаками.
* Построить boxplot для выявления выбросов.

**Инструменты:**

* matplotlib, seaborn, pandas.plot().

**5. Функционал библиотеки Scikit-learn**

**Определение:**  
Scikit-learn — это популярная библиотека Python для машинного обучения. Она предоставляет готовые реализации алгоритмов, функций предобработки данных и инструментов оценки моделей.

**Основной функционал:**

1. **Предобработка данных (preprocessing)**:
   * StandardScaler, MinMaxScaler, LabelEncoder, OneHotEncoder.
   * Работа с пропущенными значениями: SimpleImputer, KNNImputer.
2. **Алгоритмы машинного обучения:**
   * Классификация: KNeighborsClassifier, SVC, DecisionTreeClassifier, LogisticRegression, GaussianNB.
   * Регрессия: LinearRegression, Ridge, Lasso.
   * Кластеризация: KMeans, DBSCAN.
3. **Разделение выборки:**
   * train\_test\_split — делит данные на обучающую и тестовую выборку.
4. **Оценка модели:**
   * Метрики: accuracy\_score, precision\_score, recall\_score, f1\_score, mean\_squared\_error.
   * Кросс-валидация: cross\_val\_score.
5. **Pipeline (конвейеры):**
   * Объединение шагов (масштабирование + обучение) в одну цепочку с помощью Pipeline.

**Алгоритм работы с моделью в Scikit-learn:**

1. Импорт библиотеки.
2. Разделение данных на X и y.
3. Разделение на обучающую и тестовую выборки.
4. Обучение модели: model.fit(X\_train, y\_train).
5. Предсказание: model.predict(X\_test).
6. Оценка модели: accuracy\_score(y\_test, y\_pred).

**6. Функционал библиотеки Pandas**

**Определение:**  
Pandas — это библиотека Python для работы с табличными данными. Она удобна для анализа, очистки и трансформации данных.

**Основные объекты:**

* Series — одномерный массив с индексами.
* DataFrame — двумерная таблица, где строки — объекты, а столбцы — признаки.

**Функционал:**

1. **Загрузка и сохранение данных:**
   * read\_csv(), read\_excel(), to\_csv(), to\_excel().
2. **Фильтрация и выборка:**
   * df['column'], df.loc[], df.iloc[].
3. **Обработка пропусков:**
   * df.isnull(), df.dropna(), df.fillna().
4. **Группировка и агрегация:**
   * df.groupby('column').mean(), sum(), count().
5. **Описательная статистика:**
   * df.describe(), df.info(), df.shape, df.dtypes.
6. **Объединение таблиц:**
   * merge(), concat(), join().
7. **Работа с признаками:**
   * Добавление/удаление: df['new'] = ..., df.drop().
   * Преобразование: apply(), map().

**7. Основные проблемы исходных данных и способы их решения (пропуски, выбросы, несовместимые типы данных)**

**1. Пропущенные значения (NaN):**

* **Причины:** ошибки сбора, отсутствие данных.
* **Опасность:** многие модели не могут обучаться с NaN.

**Методы обработки:**

* Удаление: df.dropna().
* Замена средним/медианой/модой: df.fillna(df.mean()).
* Восстановление по модели: KNNImputer из Scikit-learn.

**2. Выбросы (outliers):**

* Значения, сильно отличающиеся от остальных.
* Могут искажать средние и обучение модели.

**Методы выявления:**

* Boxplot, IQR (межквартильный размах), z-оценка.

**Методы обработки:**

* Удаление выбросов.
* Замена на медиану.
* Логарифмирование или нормализация.

**3. Несовместимые типы данных:**

* Категориальные признаки в виде строк мешают работе моделей.

**Решения:**

* **Преобразование в числа:** с помощью LabelEncoder или OneHotEncoder.
* Приведение типов: df['column'] = df['column'].astype(int).

**8. Восстановление пропущенных данных**

**Определение:**  
Это процесс замены отсутствующих значений в наборе данных подходящими приближенными значениями.

**Методы восстановления:**

1. **Статистические подходы:**
   * Среднее: df.fillna(df.mean()).
   * Медиана: df.fillna(df.median()).
   * Мода: df.fillna(df.mode().iloc[0]).
2. **Модели на основе ближайших соседей:**
   * KNNImputer — восстанавливает значения по аналогичным объектам.

**Алгоритм KNNImputer:**

1. Для каждого NaN ищутся k ближайших соседей (по евклидовому расстоянию).
2. Значение восстанавливается как среднее среди этих соседей.
3. **Регрессионные модели:** обучаются на непустых данных и предсказывают пропущенные значения.

**9. Смарт-формула**

**Определение:**  
СМАРТ — это аббревиатура, которая используется для формулировки целей. В контексте машинного обучения может применяться для постановки задач анализа.

**Расшифровка:**

* **S** (Specific) — конкретная.
* **M** (Measurable) — измеримая.
* **A** (Achievable) — достижимая.
* **R** (Relevant) — актуальная.
* **T** (Time-bound) — ограниченная по времени.

**Пример:** «Увеличить точность модели классификации до 90% на тестовых данных в течение 2 недель.»

**Алгоритм применения:**

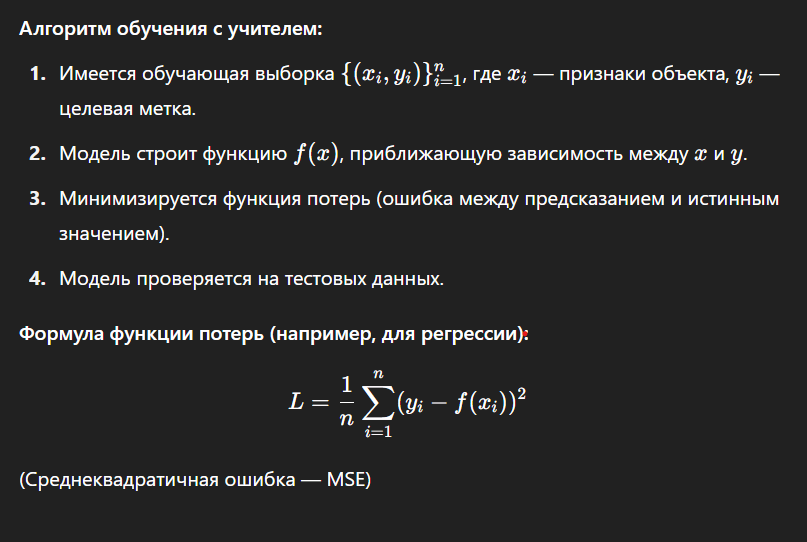
1. Определить, чего достичь (S).
2. Установить количественный критерий (M).
3. Оценить ресурсы (A).
4. Проверить соответствие общей цели (R).
5. Установить срок (T).

**10. Методы обучения с учителем**

**Определение:**  
Обучение с учителем — это класс методов машинного обучения, при котором модель обучается на размеченных данных, то есть для каждого объекта известна целевая переменная (метка).

**Задачи обучения с учителем:**

* **Классификация:** предсказать категорию (например, спам или не спам).
* **Регрессия:** предсказать числовое значение (например, цену дома).

****

**11. Методы классификации**

**Определение:**  
Методы классификации — это алгоритмы обучения с учителем, которые назначают объектам дискретные метки.

**Классы методов:**

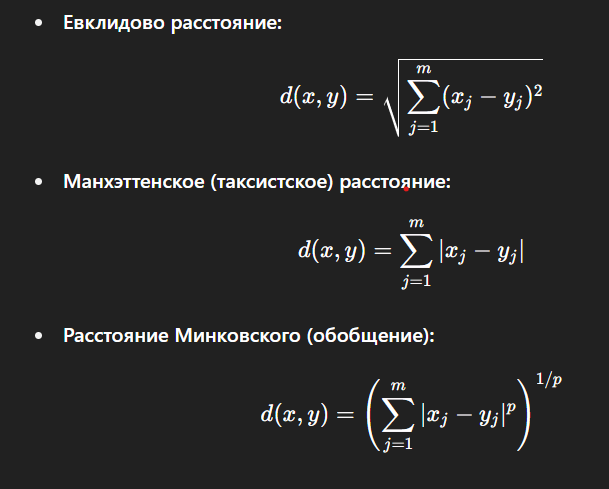
* **Метрические классификаторы** — основаны на расстояниях между объектами.
* **Пороговые методы** — например, логистическая регрессия.
* **Деревья решений.**
* **Наивный Байес.**

**12. Метрические классификаторы**

**Определение:**  
Классификаторы, использующие метрику (расстояние) для определения принадлежности объекта к классу.

**Основной принцип:**  
Объект относится к тому классу, к которому принадлежат ближайшие по расстоянию объекты.

**Основные метрики расстояния:**

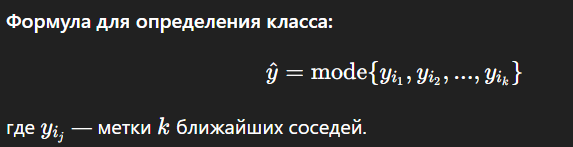
****

**13. Метод KNN (k-ближайших соседей)**

**Определение:**  
Метод KNN — простой метрический классификатор, который для классификации объекта смотрит на k ближайших соседей и выбирает наиболее частый класс среди них.

**Алгоритм KNN:**

1. Задать параметр k.
2. Для нового объекта вычислить расстояния до всех объектов обучающей выборки.
3. Отобрать k ближайших объектов.
4. Отнести объект к классу, наиболее часто встречающемуся среди соседей.

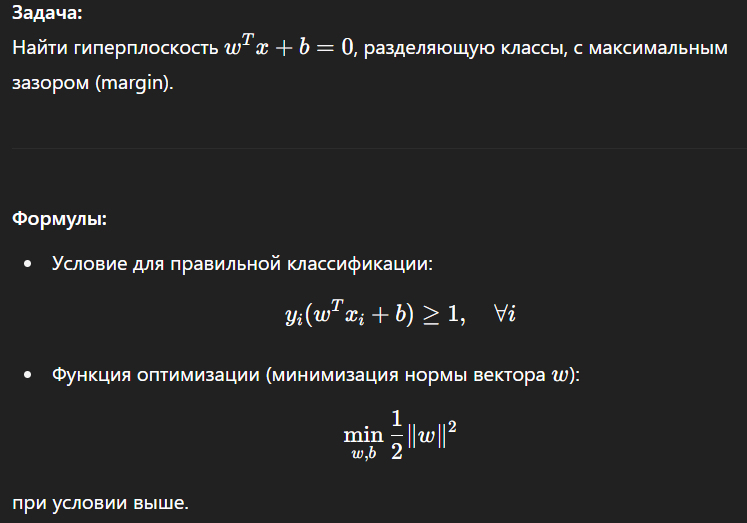
****

**Особенности:**

* Для регрессии — среднее значение по соседям.
* Требует нормализации признаков.
* Алгоритм «ленивый» — не строит модель заранее.

**14. Метод SVM (линейный)**

**Определение:**  
Метод опорных векторов (SVM) — алгоритм, который строит гиперплоскость, максимально разделяющую классы.

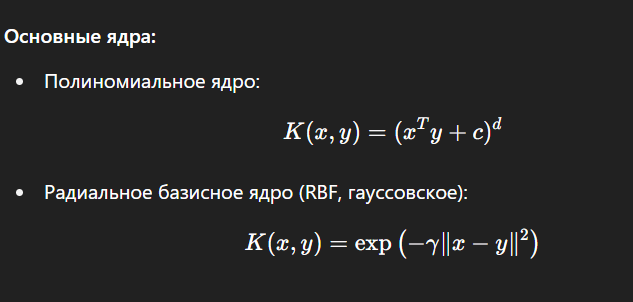
****

**Алгоритм:**

1. Выбрать гиперплоскость, которая максимизирует минимальное расстояние от объектов классов (margin).
2. Найти векторы, находящиеся на границе (опорные векторы).
3. Решить задачу оптимизации (например, с помощью метода множителей Лагранжа).

**15. Метод SVM (нелинейный)**

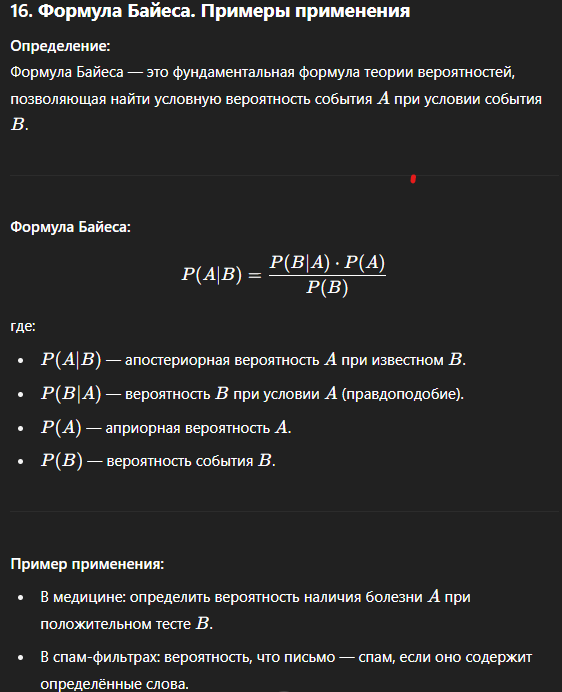
**Определение:**  
Если данные неразделимы линейно, используется **ядро (kernel)** — функция, которая преобразует признаки в более высокоразмерное пространство, где данные могут стать линейно разделимыми.

****

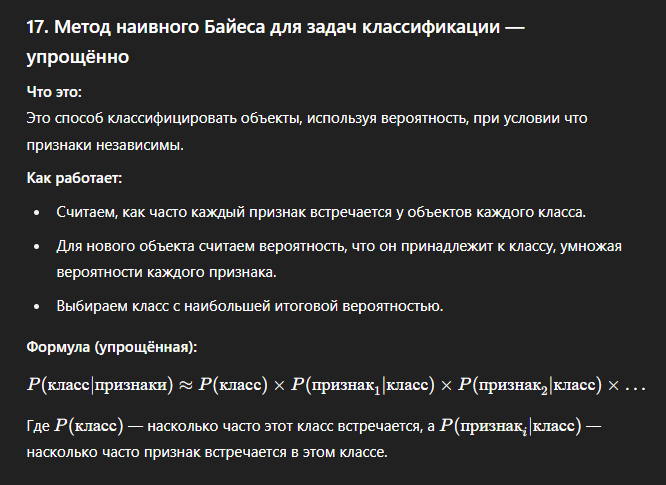
**Алгоритм:**

1. Задать ядро K.
2. Построить оптимальную гиперплоскость в новом пространстве.
3. Использовать ядро для вычисления скалярных произведений без явного перехода в пространство высокой размерности (триккер ядра).

16. Формула Байеса. Примеры применения



**17. Метод наивного Байеса для задач классификации — упрощённо**

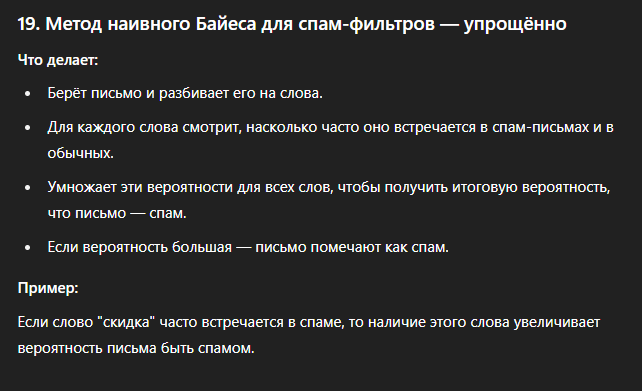


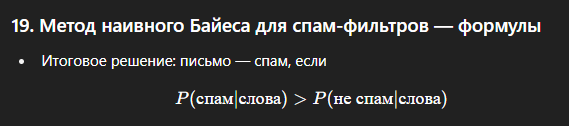
**18. Спам-фильтры**

**Определение:**  
Спам-фильтр — это система, которая автоматически классифицирует электронные письма на «спам» и «не спам».

**Принцип работы:**

* Использует набор слов-признаков (часто встречающихся в спаме).
* Для письма вычисляется вероятность принадлежности к спаму с помощью наивного Байеса.
* Если вероятность превышает порог — письмо помечается как спам.



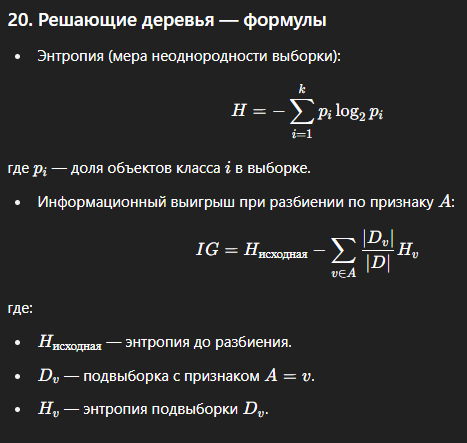
****

**20. Решающие деревья**

**Определение:**  
Решающие деревья — алгоритмы, строящие модель в виде дерева, где узлы — признаки, а листья — классы.

**Алгоритм построения:**

1. Выбрать признак, который максимизирует разделение данных (например, по критерию энтропии).
2. Разделить выборку по значению признака.
3. Рекурсивно повторять шаги для каждой подвыборки.
4. Остановить, когда все объекты одного класса или критерий остановки достигнут.



**21. Переобучение, его возможные причины и способы минимизации**

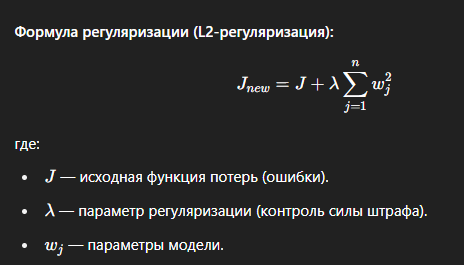
**Определение:**  
Переобучение (overfitting) — это ситуация, когда модель слишком точно запомнила обучающие данные, включая шум и случайные ошибки, и плохо работает на новых данных.

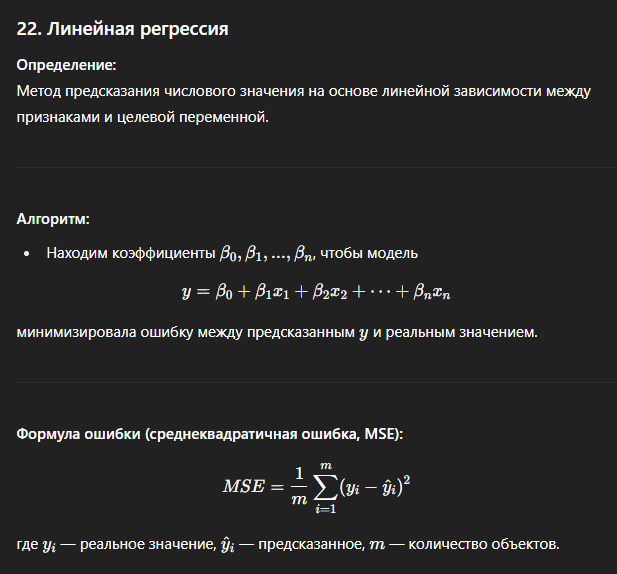
**Причины переобучения:**

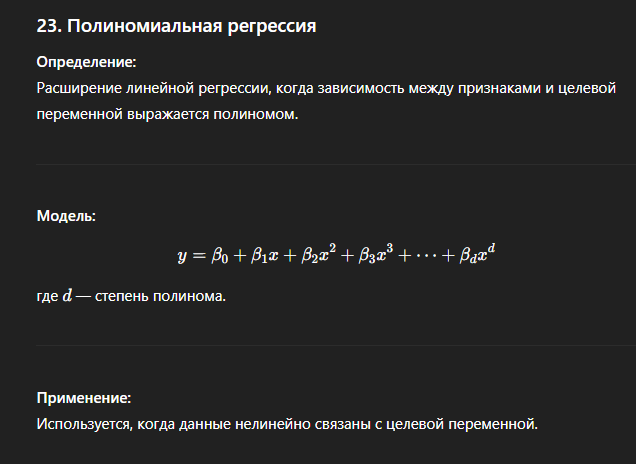
* Слишком сложная модель (много параметров).
* Недостаточно обучающих данных.
* Шум и выбросы в данных.

**Как минимизировать:**

* Использовать более простую модель.
* Регуляризация (например, L1 или L2, см. ниже).
* Увеличить объем обучающих данных.
* Применять методы кросс-валидации.
* Раннее прекращение обучения (early stopping).

****

****

****

**24. Понятие обучения без учителя. Методы обучения без учителя. Примеры алгоритмов**

Определение:  
Обучение без учителя — это метод машинного обучения, когда модель обучается на неразмеченных данных, то есть без меток классов или целевых значений.

**Цель:**  
Найти скрытые структуры, закономерности или группы в данных.

**Основные методы:**

* Кластеризация — группировка данных по сходству.
* Снижение размерности — упрощение данных при сохранении важных характеристик.

**Примеры алгоритмов:**

* k-Means (кластеризация).
* Иерархическая кластеризация.
* Метод главных компонент (PCA) — для снижения размерности.

**25. Методы кластеризации**

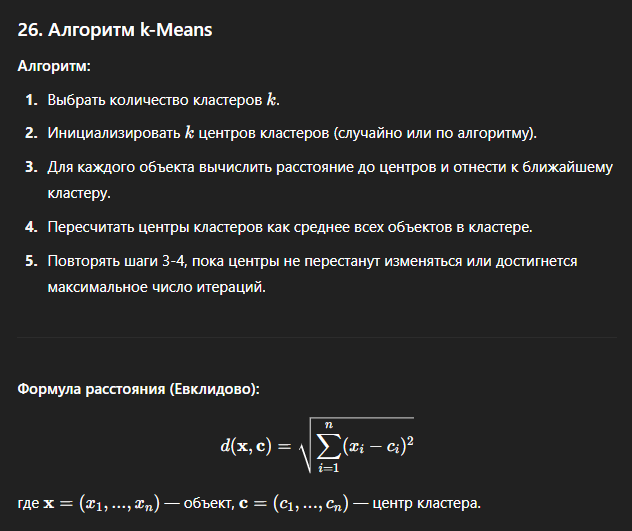
**Определение:**  
Кластеризация — разделение данных на группы (кластеры) так, чтобы объекты в одном кластере были похожи друг на друга, а между кластерами — различны.

**Ключевые идеи:**

* Использование меры расстояния (например, Евклидова).
* Поиск центроидов или плотных областей.

**Примеры методов:**

* k-Means.
* Иерархическая кластеризация.
* DBSCAN (кластеризация плотности).

****

**27. Графический метод кластеризации**

**Определение:**  
Графический метод кластеризации — это способ группировки данных, который использует представление объектов в виде графа, где объекты — это вершины, а рёбра показывают связи или сходство между ними.

**Идея:**

* Построить граф, в котором похожие объекты связаны рёбрами.
* Найти в этом графе сообщества (кластеры) — группы вершин, тесно связанных между собой.

**Основные шаги алгоритма:**

1. **Построение графа:**  
   Создаем вершины для каждого объекта. Соединяем рёбрами те объекты, которые похожи (например, расстояние между ними меньше порога).
2. **Анализ связности:**  
   Выделяем в графе подгруппы вершин, которые связаны друг с другом плотнее, чем с остальными. Такие подгруппы называются кластерами.
3. **Результат:**  
   Каждая выделенная группа — это кластер.

**Пример:**

* Представьте, что у вас есть точки на плоскости.
* Соединяем точки, которые находятся близко друг к другу.
* Получаем группы точек, которые связаны между собой — это и есть кластеры.

**Преимущества:**

* Позволяет найти кластеры любой формы.
* Хорош для сложных структур данных.

**28. Выбросы**

**Определение:**  
Выбросы — это отдельные данные, которые заметно отличаются от других значений в наборе данных. Они могут быть слишком большими, слишком маленькими или просто не соответствовать общей тенденции.

**Почему важны выбросы:**

* Они могут влиять на результаты анализа, делая модели менее точными.
* Иногда выбросы — это ошибки измерений, а иногда — важные редкие события.
* Нужно уметь их находить и решать, что с ними делать (удалять, исправлять или изучать).

**Пример:**  
Если в измерениях роста людей 170, 165, 172, 168, а один результат — 250 см, то 250 — это выброс.

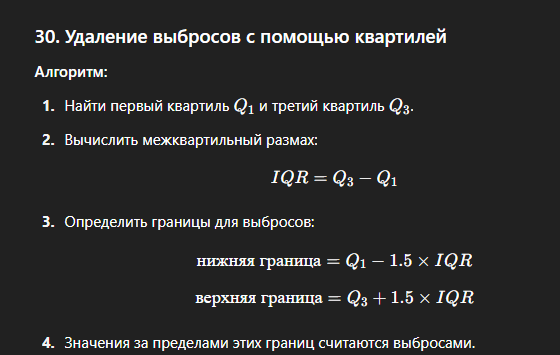
**Основные причины появления выбросов:**

* Ошибки измерений или ввода данных.
* Естественное разнообразие данных.
* Специфические редкие случаи.

**29. Методы удаления выбросов**

**Основные методы:**

* **Правило 3 сигм:** считать выбросами точки, выходящие за 3 стандартных отклонения от среднего.
* **Метод межквартильного размаха (IQR):** использовать квартильные границы.
* **Визуальные методы:** коробчатые диаграммы, scatter plot.



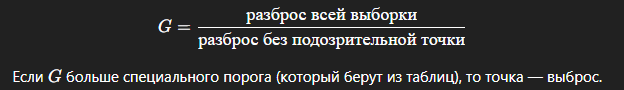
**31. Критерий Шовене — упрощённое объяснение**

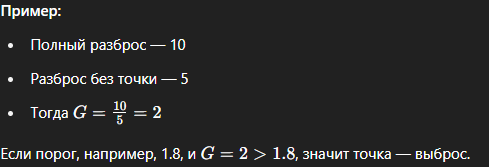
**Что это?**  
Критерий Шовене помогает понять, является ли одна точка в данных выбросом, сравнивая разброс данных с этой точкой и без неё.

**Как работает?**

1. Посчитай разброс (дисперсию) всех данных — это показывает, насколько данные «разбросаны» вокруг среднего.
2. Убери подозрительную точку из данных и посчитай разброс снова.
3. Если разброс значительно уменьшился без этой точки, значит она сильно влияет на данные и, скорее всего, выброс.

**Формула:**



****