Бакалавариат. Дисциплина «Методы искусственного интеллекта»

Вопросы к экзамену+варианты ответов

1. Задачи решаемые методами ИИ (классификация, кластеризация, регрессии и т.д.)
2. Алгоритм k ближайших соседей (KNN)
3. Алгоритм кластеризации k-средних (k-means)
4. Метод деревьев решений
5. Модель перцептрона, функции активации.
6. Модели нейронных сетей
7. Байесовский классификатор
8. Метрики в задачах машинного обучения (accuracy, precision, recall, F-мера и др.)
9. Модели знаний (продукционные, семантические сети, фреймы, логические)
10. Средства разработки интеллектуальных ИС
11. Особенности проектирования, разработки интеллектуальных систем
12. Методы извлечения знаний когнитологом (пассивные, активные)
13. Структура экспертной системы
14. Интеллектуальный анализ данных, общие сведения и терминология
15. Машинное обучение, общие сведения и терминология
16. Метод Бустинг (ансамблевые методы обучения)
17. Метод Бэггинг (ансамблевые методы обучения)
18. Сравнительная характеристика методов «деревья решений» и «искусственные нейронные сети»
19. Нормализация данных
20. Очистка данных

1. **Задачи, решаемые методами ИС.**

1. **Классификация** – это отнесение объектов (наблюдений, событий) к одному из заранее известных классов.
2. **Кластеризация** – это группировка объектов (наблюдений, событий) на основе данных (свойств), описывающих сущность этих объектов. Объекты внутри кластера должны быть "похожими" друг на друга и отличаться от объектов, вошедших в другие кластеры. Чем больше похожи объекты внутри кластера и чем больше отличий между кластерами, тем точнее кластеризация.
3. **Регрессия**, в том числе задачи [прогнозирования](http://www.basegroup.ru/tasks/forecast.htm). Установление зависимости непрерывных выходных от входных переменных.
4. **Ассоциация** – выявление закономерностей между связанными событиями. Примером такой закономерности служит правило, указывающее, что из события X следует событие Y. Такие правила называются ассоциативными. Впервые эта задача была предложена для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее еще называют анализом рыночной корзины (**market basket analysis**).
5. **Последовательные шаблоны** – установление закономерностей между связанными во времени событиями, т.е. обнаружение зависимости, что если произойдет событие X, то спустя заданное время произойдет событие Y.
6. **Анализ отклонений** – выявление наиболее нехарактерных шаблонов.

**2. Алгоритма k-ближайших соседей.**

В основе алгоритма k-ближайших соседей (k-nearest neighbor algorithm, KNN) лежит сходство объектов. Алгоритм способен выделить среди всех наблюдений k известных объектов (k-ближайших соседей), похожих на новый неизвестный ранее объект. На основе классов ближайших соседей выносится решение касательно нового объекта. Важной задачей данного алгоритма является подбор коэффициента k – количество записей, которые будут считаться похожими. Алгоритм KNN широко применяется в Data Mining.

Рассмотрим задачу классификации.

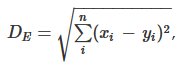
Пусть имеется **n** наблюдений, каждому из которых соответствует запись в таблице. Все записи принадлежат какому-либо классу. Необходимо определить класс для новой записи.

На первом шаге алгоритма следует задать число k – количество ближайших соседей. Если принять k = 1, то алгоритм потеряет обобщающую способность (то есть способность выдавать правильный результат для данных, не встречавшихся ранее в алгоритме) так как новой записи будет присвоен класс самой близкой к ней. Если установить слишком большое значение, то многие локальные особенности не будут выявлены.

На втором шаге находятся k записей с минимальным расстоянием до вектора признаков нового объекта (поиск соседей). Функция для расчета расстояния должна отвечать следующим правилам:

1. d(x,y) ≥ 0, d(x,y) = 0 тогда и только тогда, когда x = y;
2. d(x,y) = d(y,x);
3. d(x,z) ≤ d(x,y) + d(y,z), при условии, что точки x, y, z не лежат на одной прямой.

Для упорядоченных значений атрибутов находится Евклидово расстояние:

**, где **n** – количество атрибутов.

Для строковых переменных, которые не могут быть упорядочены, может быть применена функция отличия, которая задается следующим образом:

**

Часто перед расчетом расстояния необходима нормализация. Приведем некоторые полезные формулы.

Минимаксная нормализация:

**.

Нормализация с помощью стандартного отклонения:

,

где *σx* – стандартное отклонение, Xср – среднее значение.

При нахождении расстояния иногда учитывают значимость атрибутов. Она определяется экспертом или аналитиком субъективно, полагаясь на собственный опыт. В таком случае при нахождении расстояния каждый **i**-ый квадрат разности в сумме умножается на коэффициент Zi. Например, если атрибут A в три раза важнее атрибута **B** (**ZA = 3, ZB = 1**), то расстояние будет находиться следующим образом:

**

Подобный прием называют растяжением осей (stretching the axes), что позволяет снизить ошибку классификации.

На следующем шаге, когда найдены записи, наиболее похожие на новую, необходимо решить, как они влияют на класс новой записи. Для этого используется функция сочетания (combination function). Одним из основных вариантов такой функции является простое невзвешенное голосование (simple unweighted voting).

### Простое невзвешенное голосование

Вначале, найдя число k, мы узнали, сколько записей будет иметь право голоса при определении класса. Затем мы выявили эти записи, расстояние от которых до новой оказалось минимальным. Теперь можно приступить к простому невзвешенному голосованию.

Расстояние от каждой записи при голосовании здесь больше не играет роли. Все имеют равные права в определении класса. Запись отдает предпочтение тому классу, к которому она принадлежит, а тот, который наберет наибольшее количество голосов, присваивается новой записи.

Но что делать в случае, если несколько классов набрали равное количество голосов? Эту проблему снимает взвешенное голосование (weighted voting).

### Взвешенное голосование

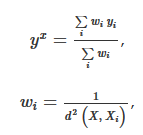
В такой ситуации учитывается также и расстояние до новой записи. Чем меньше расстояние, тем более значимый вклад вносит голос. Голоса за класс находятся по следующей формуле:

**,

где **d2(X, Yi)** – квадрат расстояния от известной записи Yi до новой X, n – количество известных записей класса, для которого рассчитываются голоса, **class** - наименование класса.

Класс, набравший наибольшее количество голосов, присуждается новой записи. При этом вероятность того, что несколько классов наберут одинаковые голоса, гораздо ниже. Совершенно очевидно, что при **k** = 1 новой записи присваивается класс самого ближайшего соседа.

Но что делать, если стоит задача прогнозирования непрерывной переменной? В этом случае вместо функции сочетания используют следующие формулы:

**

где – *d*2(*X*,*Xi*) квадрат расстояния от записи **Xi** до новой записи **X**, *y*x  – значение выходной непрерывной переменной для новой записи.

## Примеры работы алгоритма KNN

Решим задачу классификации для ирисов (используя данные об ирисах Фишера, которые доступны по [ссылке](http://archive.ics.uci.edu/ml/datasets/Iris)). Имеется набор данных, собранных Р. Фишером, о 150 цветках трех классов: Iris Setosa, Iris Versicolour, Iris Virginica, по 50 записей для каждого. Известна длина и ширина чашелистика и лепестка всех этих ирисов.

Для простоты рассмотрим только два входных поля: длина чашелистика и длина лепестка, а также выходное – класс цветка.

Необходимо определить класс двух цветков со следующими значениями длины чашелистика и лепестка в сантиметрах: 5,3 и 1,6 (первый), 6,1 и 4,8 (второй).

Рассмотрим первый цветок. Установим значение k = 3. Таким образом, необходимо найти трех ближайших соседей. Для первого цветка это будут цветки со следующими значениями длины чашелистика и лепестка: A (5,3; 1,5), В(5,2; 1,5) и С(5,2; 1,5). Покажем, чему равны расстояния до первого цветка (оба атрибута равнозначны):

**==0,1**

**==0,141421356**

**==0,141421356**

Полученные сведения объединим в таблицу 1.

Таблица 1 – 3 ближайших соседа для цветка 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Запись** | **Длина чашелистика** | **Длина лепестка** | **Расстояние** | **Класс** |
| Цветок 1 | 5,3 | 1,6 | – | – |
| A | 5,3 | 1,5 | 0,1 | Iris-Setosa |
| B | 5,2 | 1,5 | 0,14 | Iris-Setosa |
| C | 5,2 | 1,5 | 0,14 | Iris-Setosa |

Теперь определим класс нового цветка. Воспользуемся простым невзвешенным голосованием. Так как все три цветка A, B и C принадлежат к классу Iris-Setosa, то он получает 100% голосов, и первому цветку присваиваем этот класс.

Теперь рассмотрим более сложный случай cо вторым цветком.

Зададим k = 3 и предположим, что длина лепестка вдвое важнее длины чашелистика. Ближайшими тремя соседями будут следующие цветки:

A(6,1; 4,7), B(6; 4,8), C(6,2; 4,8)

Покажем, чему для них равны расстояния:

**==0,141421356**

**==0,1**

**==0,1**

**В** и **С** – это Iris Virginica, а **A** – Iris Versicolour.

Объединим полученные сведения в таблицу 2.

Таблица 2 – Три ближайших соседа для цветка 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Запись** | **Длина чашелистика** | **Длина лепестка** | **Расстояние** | **Класс** |
| Цветок 2 | 6,1 | 4,8 | – | – |
| A | 6,1 | 4,7 | 0,14 | Iris Versicolour |
| B | 6 | 4,8 | 0,1 | Iris Virginica |
| C | 6,2 | 4,8 | 0,1 | Iris Virginica |

Рассмотрим взвешенное голосование.

*Votes*(*IrisVersicolour*)= =1/0,02=50

*Votes*(*IrisVirginica*)=1/0,12+1/0,12=2/0,01=200

Так как 200>50, то второй цветок классифицируется как Iris Virginica.

**Достоинства и недостатки алгоритма KNN.**

Перечислим положительные особенности.

1. Алгоритм устойчив к аномальным выбросам, так как вероятность попадания такой записи в число k-ближайших соседей мала. Если же это произошло, то влияние на голосование (особенно взвешенное) (при k>2) также, скорее всего, будет незначительным, и, следовательно, малым будет и влияние на итог классификации.
2. Программная реализация алгоритма относительно проста.
3. Результат работы алгоритма легко поддаётся интерпретации. Экспертам в различных областях вполне понятна логика работы алгоритма, основанная на нахождении схожих объектов.
4. Возможность модификации алгоритма, путём использования наиболее подходящих функций сочетания и метрик позволяет подстроить алгоритм под конкретную задачу.

Алгоритм KNN обладает и рядом недостатков. Во-первых, набор данных, используемый для алгоритма, должен быть репрезентативным. Во-вторых, модель нельзя "отделить" от данных: для классификации нового примера нужно использовать все примеры. Эта особенность сильно ограничивает использование алгоритма

**3. Метод k-средних k-means**

**Метод к-средних** создает к-групп из набора объектов таким образом, чтобы члены группы были наиболее однородными. Один из распространённых алгоритмов кластерного анализа.

Напоминание.*Кластерный анализ* – это семейство алгоритмов, разработанных для формирования групп таким образом, чтобы члены группы были наиболее похожими друг на друга и не похожими на элементы, не выходящие в группу. Кластер и группа – это синонимы в мире кластерного анализа.

**Пример**. Предположим, что у нас есть данные о пациентах, например, возраст, пульс, кровяное давление, максимальное потребление кислорода, холестерин и так далее. Это вектор, представляющий пациента. В кластерном анализе это называется наблюдениями.

Задача: сгруппировать вместе пациентов по возрасту, пульсу, давлению с помощью этих векторов.

Вы говорите методу к-средних, сколько кластеров вам нужно, а он сделает все остальное.

**Как это происходит?** Метод к-средних имеет множество вариантов работы для различных типов данных.

В общем случае все они делают примерно следующее:

1. Метод к-средних выбирает точки многомерного пространства, которые будут представлять к-кластеры. Эти точки называются центрами тяжести.
2. Каждый пациент будет располагаться наиболее близко к одной из точек. Надеемся, что не все они будут стремиться к одному центру тяжести, поэтому образуется несколько кластеров.
3. Теперь у нас есть к-кластеров, и каждый пациент – это член какого-то из них.
4. Метод к-средних, учитывая положение членов кластера, находит центр каждого из к-кластеров (именно здесь используются векторы пациентов!).
5. Вычисленный центр становится новым центром тяжести кластера.
6. Поскольку центр тяжести переместился, пациенты могли оказаться ближе к другим центрам тяжести и перейти в другой кластер.
7. Шаги 2-6 повторяются до тех пор, пока центр тяжести не перестанут изменяться и членство не стабилизируется. Это называется сходимостью.

**Требует ли этот метод обучения или он самообучающийся?** Бывает по-разному. Но большинство расценивает метод к-средних как самообучающийся. Вместо того, чтобы уточнять количество кластеров, метод к-средних «изучает» кластеры самостоятельно, не требуя информации о том, к какому кластеру относятся данные наблюдения. Метод к-средних может быть полуобучаемым.

Основным достоинством алгоритма является его высокая скорость выполнения и эффективность по сравнению с другими алгоритмами, в особенности при работе с крупными наборами данных.

*Метод к-средних* может использоваться для предварительного разбиения на группы большого набора данных, после которого проводится более мощный кластерный анализ подкластеров. Метод к-средних может использоваться, чтобы «прикинуть» количество кластеров и проверить наличие неучтенных данных и связей в наборах.

Два основных недостатка метода к-средних заключаются в чувствительности к «выбросам» и начальному выбору центров тяжести.

**Описание 2**

Метод k-средних используется для кластеризации данных на основе алгоритма разбиения векторного пространства на заранее определенное число кластеров k. Алгоритм представляет собой итерационную процедуру, в которой выполняются следующие шаги:

1. Выбирается число кластеров k.
2. Из исходного множества данных случайным образом выбираются k наблюдений, которые будут служить начальными центрами кластеров.
3. Для каждого наблюдения исходного множества определяется ближайший к нему центр кластера (расстояния измеряются в метрике Евклида). При этом записи, «притянутые» определенным центром, образуют начальные кластеры.
4. Вычисляются **центроиды** — центры тяжести кластеров. Каждый центроид — это вектор, элементы которого представляют собой средние значения соответствующих признаков, вычисленные по всем записям кластера.
5. Центр кластера смещается в его центроид, после чего центроид становится центром нового кластера.
6. 3-й и 4-й шаги итеративно повторяются. Очевидно, что на каждой итерации происходит изменение границ кластеров и смещение их центров. В результате минимизируется расстояние между элементами внутри кластеров и увеличиваются меж-кластерные расстояния.

Остановка алгоритма производится тогда, когда границы кластеров и расположения центроидов не перестанут изменяться от итерации к итерации, т.е. на каждой итерации в каждом кластере будет оставаться один и тот же набор наблюдений. На практике алгоритм обычно находит набор стабильных кластеров за несколько десятков итераций.

**Преимуществом** алгоритма являются скорость и простота реализации. К **недостаткам** можно отнести неопределенность выбора начальных центров кластеров, а также то, что число кластеров должно быть задано изначально, что может потребовать некоторой априорной информации об исходных данных.

Существуют методы кластеризации, которые можно рассматривать как происходящие от k-средних. Например, в методе k-медиан (k-medians) для вычисления центроидов используется не среднее, а медиана, что делает алгоритм более устойчивым к аномальным значениям в данных.

Алгоритм g-средних (от gaussian) строит кластеры, распределение данных в которых стремится к нормальному (гауссовскому) и снимает неопределенность выбора начальных кластеров. Алгоритм C-средних использует элементы нечеткой логики, учитывая при вычислении центроидов не только расстояния, но и степень принадлежности наблюдения к множеству объектов в кластере. Также известен алгоритм Ллойда, который в качестве начального разбиения использует не множества векторов, а области векторного пространства.

Идея метода k-средних была одновременно сформулирована Гуго Штейнгаузом и Стюартом Ллойдом в 1957 г. Сам термин «k-средних» был впервые введен Дж. Маккуинном в 1967 г.

**4. Деревья решений**

**Метод «Деревьев решений» или правил типа «ЕСЛИ… ТО…».**

Данные метода апеллируют к информации, заключенной не только в отдельных признаках, но и в сочетаниях значений признаков. Они вычисляют частоты комбинаций простых логических событий в подгруппах данных. На основании анализа вычисленных частот делается заключение о полезности той или иной комбинации для установления ассоциаций в данных для процессов классификации или прогнозирования. Результаты работы данных методов оформляются в виде так называемых Деревьев решений или правил типа «ЕСЛИ… ТО…». Популярность данного подхода связана с наглядностью и интерпретируемость результатов. Проблемой логических методов обнаружения закономерностей является проблема перебора вариантов за приемлемое время и поиск оптимальной композиции предложенных правил.

Структура дерева представляет собой «листья» и «ветки». На рёбрах («ветках») дерева решения записаны признаки, от которых зависит целевая функция, в «листьях» записаны значения целевой функции, а в остальных узлах — признаки, по которым различаются случаи. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение. В качестве признаков могут использоваться различные критерии, наиболее популярными из которых стали информационный (метрика - информационная энтропия) и статистический (индекс Джини).

Деревья решений бывают двух основных типов:

* Дерево для классификации, когда предсказываемый результат является классом, к которому принадлежат данные;
* Дерево для регрессии, когда предсказываемый результат можно рассматривать как вещественное число (например, продолжительность болезни, цена акции, длительность засухи и т.д.).

**Терминология.** Введем в рассмотрение основные понятия, используемые в теории деревьев решений.

* Объект. Пример, шаблон, наблюдение
* Атрибут. Признак, независимая переменная, свойство
* Целевая переменная. Зависимая переменная, метка класса
* Узел. Внутренний узел дерева, узел проверки
* Корневой узел. Начальный узел дерева решений
* Лист. Конечный узел дерева, узел решения, терминальный узел
* Решающее правило. Условие в узле, проверка

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

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

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

* **ID3 (Iterative Dichotomizer 3)** — алгоритм позволяет работать только с дискретной целевой переменной, поэтому деревья решений, построенные с помощью данного алгоритма, являются классифицирующими. Число потомков в узле дерева не ограничено. Не может работать с пропущенными данными.
* **C4.5** — усовершенствованная версия алгоритма ID3, в которую добавлена возможность работы с пропущенными значениями атрибутов.
* **CART (Classification and Regression Tree)** — алгоритм обучения деревьев решений, позволяющий использовать как дискретную, так и непрерывную целевую переменную, то есть решать как задачи классификации, так и регрессии. Алгоритм строит деревья, которые в каждом узле имеют только два потомка.

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

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

**5. НЕЙРОННЫЕ СЕТИ (Перцептрон, виды ИНС)**

Особенностью интеллектуальных систем является способность решать слабоструктурированные и плохо формализованные задачи. Эта способность основана на применении различных методов моделирования рассуждений для обработки символьной информации. Традиционным подходом к построению механизмов рассуждения является использование дедуктивного логического вывода на правилах (rule-based reasoning), который применяется в экспертных системах продукционного и логического типа. При таком подходе необходимо заранее сформулировать весь набор закономерностей, описывающих предметную область. Альтернативный подход основан на концепции обучения по примерам (case-based reasoning). В этом случае при построении интеллектуальной системы не требуется заранее знать обо всех закономерностях исследуемой области, но необходимо располагать достаточным количеством примеров для настройки разрабатываемой адаптивной системы, которая после обучения будет способна получать требуемые результаты с определенной степенью достоверности. В качестве таких адаптивных систем применяются искусственные нейронные сети.

*Искусственная нейронная сеть* – это значительным образом упрощенная модель биологической нейронной сети (нервной ткани биологического мозга).

Естественная нервная клетка (нейрон) состоит из тела *(сомы),* содержащего ядро, отростков (*дендритов),* по которым в нейрон поступают входные сигналы и одного отростка (аксон) на конце сомы, который служит для передачи выходных сигналов нейрона другим нервным клеткам. Соединение аксона с дендритом другого нейрона называется *синапсом.* Нейрон возбуждается и передает сигнал через аксон, если число пришедших по дендритам возбуждающих сигналов больше, чем число тормозящих.

Из биологии заимствованы следующие основополагающие принципы:

* Нейрон – это переключатель, получающий и передающий импульсы, или сигналы. Если нейрон получает достаточно сильный импульс, то говорят, что нейрон активирован, то есть передает импульсы связанным с ним нейронам. Не активированный нейрон остается в состоянии покоя и не передает импульс.
* Нейрон состоит из нескольких компонентов: синапсов, соединяющих нейрон с другими нейронами и получающих импульсы от соседних нейронов, аксона, передающего импульс другим нейронам, и дендрита, получающего сигналы из различных источников, в т.ч. от синапсов.
* Когда нейрон получает импульс, превышающий определенный порог, он передает импульс последующим нейронам (активирует импульс).
* Синапс состоит из двух частей: пресинаптической, соединенной с аксоном передающей импульс клетки, и постсинаптической, соединенной с дендритом получающей импульс клетки. Обе части синапса соединяет синаптическая цепь.

Сеть ИНС представляет собой совокупность простых вычислительных элементов — искусственных нейронов, каждый из которых обладает определенным количеством входов (дендритов) и единственным выходом (аксоном), разветвления которого подходят к синапсам, связывающим его с другими нейронами. На входы нейрона поступает информация извне или от других нейронов. Каждый нейрон характеризуется функцией преобразования входных сигналов в выходной (функция возбуждения нейрона). Нейроны в сети могут иметь одинаковые или разные функции возбуждения. Сигналы, поступающие на вход нейрона, неравнозначны в том смысле, что информация из одного источника может быть более важной, чем из другого. Приоритеты входов задаются с помощью вектора весовых коэффициентов, моделирующих синаптическую силу биологических нейронов.

**5. Модель искусственного нейрона**

**Перцептро́н**, или **пер*с*ептрон** (англ. *perceptron* от лат. *perceptio* — восприятие; нем. *Perzeptron*) — математическая или компьютерная модель восприятия информации мозгом (кибернетическая *модель мозга*), предложенная Фрэнком Розенблаттом в 1958 году

Согласно современной терминологии, перцептроны могут быть классифицированы как искусственные нейронные сети:

1. с одним скрытым слоем;[]](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%86%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD#cite_note-3)
2. с пороговой передаточной функцией;
3. с прямым распространением сигнала.

**Модель искусственного нейрона** (рис. 3.1) представляет собой дискретно-непрерывный преобразователь информации. Информация, поступающая на вход нейрона, суммируется с учетом весовых коэффициентов **wi**сигналов **xi***,* i= 1,..., n,где n — размерность пространства входных сигналов. Потенциал нейрона определяется по формуле 

Взвешенная сумма поступивших сигналов (потенциал) преобразуется с помощью передаточной функции f(P) в выходной сигнал нейрона *Y,* который передается другим нейронам сети, т. е. *Y=f(P).* Вид передаточной (активационной) функции является важнейшей характеристикой нейрона. Существует множество различных функций, используемых в качестве передаточных. Выбор вида передаточной функции – это сложная задача. Как правило, к ним выдвигаются требования иметь область значения [0,1] или [-1,1] и быть возрастающими.



Рисунок 3.1. Схема математической модели нейрона.

Математическая модель нейрона строится следующим образом:

* Вход модели нейрона **X** – это вектор, состоящий из **n** компонент. Каждая из компонент входного вектора **Xi** – это один из импульсов, получаемых нейроном.
* Выход модели нейрона – это одно число **Y\***. Это означает, что внутри модели, входной вектор должен быть преобразован и агрегирован в скаляр. В дальнейшем этот импульс будет передан другим нейронам.
* Известно, что при получении импульса синапс нейрона изменяет его. Математически этот процесс изменения можно описать следующим образом: для каждой из компонент входа **xi** задают вес. Импульс, прошедший через синапс, принимает вид **wixi**. Заметим, что веса могут быть назначены при инициализации модели, а могут быть переменными и измеряться в ходе расчетов. Обучение сети – это в первую очередь нахождение весов синапса.
* Сложение полученных импульсов. Агрегирование полученных импульсов – это вычисление их суммы **∑wixi**.

Будет активирован нейрон или нет, определяется вычислением так называемой передаточной, или активационной, функции нейрона **f(∑wixi)**. Если значение функции превышает некоторый заранее определенный порог, то нейрон активирован и передает импульс следующим нейронам в сети. То есть передаточная функция должна моделировать скачок, резкий переход в состояние активации. В общем случае эта функция может быть ступенчатой (пороговой), линейной или нелинейной (рис. ).

**1.** **Пороговая функция.** Для простой передаточной функции нейросеть может выдавать 0 и 1, 1 и -1 или другие числовые комбинации. Передаточная функция в таких случаях является "жестким ограничителем" или пороговой функцией (рис.3.2). Пороговая функция **f(P)** пропускает информацию только в том случае, если алгебраическая сумма входных сигналов превышает некоторую постоянную величину **Р\****,* например: **Y=1** если **P>=P\*** и **Y= -1** при **P<P\***. На рисунке 3.2 **P\*=0**.

Рисунок 3.2. - Пороговая функция.

Пороговая функция не обеспечивает достаточной гибкости ИНС при обучении. Если значение вычисленного потенциала не достигает заданного порога, то выходной сигнал не формируется и нейрон «не срабатывает». Это приводит к снижению интенсивности выходного сигнала нейрона и, как следствие, к формированию невысокого значения потенциала взвешенных входов в следующем слое нейронов.

**2.** **Линейная функция.** Линейная функция **Y=kP**дифференцируема и легко вычисляется (рис.3.3), что в ряде случаев позволяет уменьшить ошибки выходных сигналов в сети, так как передаточная функция сети также является линейной. Однако она не универсальна и не обеспечивает решения многих задач.

**3. S-подобная функция**. Передаточная функция называется сигмоидой (рис.3.4 а), когда ее диапазон [0, 1], или гиперболическим тангенсом (рис.3.4 б), при диапазоне [-1,1]. Определенным компромиссом между линейной и ступенчатой функциями является сигмоидальная функция переноса **Y*=*1/(1+e-kP)**, которая удачно моделирует передаточную характеристику биологического нейрона (рис. 3.4 а)*.* Коэффициент **k**определяет крутизну нелинейной функции: чем больше **k***,* тем ближе сигмоидальная функция к пороговой; чем меньше **k***,* тем она ближе к линейной. Подобно ступенчатой функции она позволяет выделять в пространстве признаков множества сложной формы, в том числе невыпуклые и несвязные. Важной чертой **S**-кривых является непрерывность функций и их производных (в отличие от ступенчатой функции). **S**-подобная функция дифференцируема, как и линейная функция, и это качество можно использовать при поиске экстремума в пространстве параметров ИНС.

Рисунок 3.3. – Линейная функция.

Рисунок 3.4. - S-подобные передаточные функции.

Тип функции переноса выбирается с учетом конкретной задачи, решаемой с применением нейронных сетей.

Нейронная сеть представляет собой совокупность искусственных нейронов, организованных слоями. При этом выходы нейронов одного слоя соединяются с входами нейронов другого. В зависимости от топологии соединений нейронов ИНС подразделяются на одноуровневые и многоуровневые, с обратными связями и без них. Связи между слоями могут иметь различную структуру. В однолинейных сетях каждый нейрон (узел) нижнего слоя связан с одним нейроном верхнего слоя. Если каждый нейрон нижнего слоя соединен с несколькими нейронами следующего слоя, то получается пирамидальная сеть. Воронкообразная схема соединений предполагает связь каждого узла верхнего слоя со всеми узлами нижнего уровня. Существуют также древовидные и рекуррентные сети, содержащие обратные связи с произвольной структурой межнейронных соединений. Чтобы построить ИНС для решения конкретной задачи, нужно выбрать тип соединения нейронов, определить вид передаточных функций элементов и подобрать весовые коэффициенты межнейронных связей.

При всем многообразии возможных конфигураций ИНС на практике получили распространение лишь некоторые из них. Классические модели нейронных сетей рассмотрены ниже.

**6. Модели нейронных сетей.**

Теоретические основы нейронных сетей были заложены в начале 1940-х гг. Норбертом Винером в его работе «Кибернетика или управление и связь в животном и машине». Дж. Маккалохом и У. Питтс разработали собственную теорию деятельности головного мозга. К главным результатам их работы относятся следующие:

* предположение о том, что нейронная сеть способна обучаться и распознавать образы.
* модель нейрона в виде простейшего процессорного элемента, который вычисляет значение переходной функции от скалярного произведения вектора входных сигналов и вектора весовых коэффициентов;
* конструкция нейронной сети.

В формализме Дж. Маккалоха и У. Питтса нейроны имеют пороговую функцию перехода из состояния в состояние. Каждый нейрон в сети определяет взвешенную сумму состояний всех других нейронов и сравнивает ее с порогом, чтобы определить свое собственное состояние.

Серьезное развитие нейрокибернетика получила в трудах американского нейрофизиолога Ф. Розенблата, который предложил свою модель нейронной сети в 1958 г. и продемонстрировал созданное на ее основе электронное устройство, названное *перцептроном*. Ф. Розенблат ввел возможность модификации межнейронных связей, что сделало ИНС обучаемой.

Алгоритм обучения перцептрона включает следующие шаги.

1. Системе предъявляется эталонный образ.
2. Если результат распознавания совпадает с заданным, весовые коэффициенты связей не изменяются.
3. Если ИНС неправильно распознает результат, то весовым коэффициентам дается приращение в сторону повышения качества распознавания.

Теоретический анализ перцептрона, проведенный М. Минским и С. Пейпертом, показал его ограниченные возможности, поскольку не всегда существует такая комбинация весовых коэффициентов, при которой заданное множество образов будет распознаваться правильно. Причина этого недостатка состоит в том, что однослойный перцептрон реализует линейную поверхность, разделяющую пространство эталонов, вследствие чего происходит неверное распознавание образов в случаях, когда задача не является линейно сепарабельной. Для решения таких проблем предложены модели многослойных перцептронов, способные строить ломаную границу между распознаваемыми образами.

**Многослойные сети.** В многослойных сетях устанавливаются связи только между нейронами соседних слоев, как показано на рисунке 3.5.



Рисунок 3.5. - Схема многослойного перцептрона.

Каждый элемент может быть соединен модифицируемой связью с любым нейроном соседних слоев, но между элементами одного слоя связей нет. Каждый нейрон может посылать выходной сигнал только в вышележащий слой и принимать входные сигналы только с нижерасположенного слоя. Входные сигналы подаются на нижний слой, а выходной вектор сигналов определяется путем последовательного вычисления уровней активности элементов каждого слоя (снизу вверх) с использованием уже известных значений активности элементов предшествующих слоев. При распознавании образов входной вектор соответствует набору признаков, а выходной — распознаваемым образам. Скрытый слой (один или несколько) предназначен для отражения специфики знаний. В таких сетях обычно используются передаточные сигмоидальные функции.

Структура нейронной сети определяется типом, например 30—15—8, т.е. тридцать узлов находится в первом слое, пятнадцать узлов — в скрытом слое и восемь - в выходном. Определение числа скрытых слоев и числа нейронов в каждом слое для конкретной задачи является неформальной проблемой, при решении которой можно использовать эвристическое правило: «число нейронов в следующем слое в два раза меньше, чем в предыдущем» [2].

Выше отмечалось, что простой перцептрон с одним слоем обучаемых связей формирует границы областей решений в виде гиперплоскостей. Двухслойный перцептрон может выполнять операцию «логического и» ∧ над полупространствами, образованными гиперплоскостями первого слоя весов. Это позволяет формировать любые выпуклые области в пространстве входных сигналов. С помощью трехслойного перцептрона, используя «логическое или» ∨ для комбинирования выпуклых областей, можно получить области решений произвольной формы и сложности, в том числе невыпуклые и несвязные. То, что многослойные перцептроны с достаточным множеством внутренних нейроподобных элементов и соответствующей матрицей связей в принципе способны осуществлять любое отображение вход-выход, отмечали еще М. Минский и С. Пейперт, однако они сомневались, что для таких процедур можно открыть мощный аналог процедуры обучения простого перцептрона. В настоящее время в результате возрождения интереса к многослойным сетям предложено несколько таких процедур. Одной из них является алгоритм обратного распространения ошибки.

**Рекуррентные сети.** Они содержат обратные связи, благодаря которым становится возможным получение отличающихся значений выходов при одних и тех же входных данных. Наличие рекуррентных нейронов позволяет ИНС накапливать знания в процессе обучения.

Рекуррентные сети (рис. 3.6) являются развитием модели Хопфилда на основе применения новых алгоритмов обучения, исключающих попадание системы в локальные минимумы на поверхности энергетических состояний. Важной особенностью рекуррентных сетей является их способность предсказывать существование новых классов объектов.



Рисунок 3.6. Структура рекуррентной нейронной сети.

Модель Хопфилда Американскому биофизику Дж. Хопфилду удалось привлечь к анализу нейросетевых моделей математический аппарат статистической физики. В результате была сформулирована модель ассоциативной памяти на нейронной сети с использованием правила Д. Хебба для модификации весовых коэффициентов. Это правило основано на простом предположении: если два нейрона возбуждаются вместе, то сила связи между ними возрастает; если они возбуждаются порознь, то сила связи между ними уменьшается.

Сеть Хопфилда использует три слоя: *входной*, *слой Хопфилда* и *выходной слой*. Каждый слой имеет одинаковое количество нейронов. Входы слоя Хопфилда подсоединены к выходам соответствующих нейронов входного слоя через изменяющиеся *веса соединений*. Выходы слоя Хопфилда подсоединяются ко входам всех нейронов слоя Хопфилда, за исключением самого себя, а также к соответствующим элементам в выходном слое. В режиме функционирования, сеть направляет данные из входного слоя через фиксированные веса соединений к слою Хопфилда.

Сеть Хопфилда строится с учетом следующих условий:

* все элементы связаны со всеми;
* **wji=wij** - прямые и обратные связи симметричны;
* **wii = 0** - диагональные элементы матрицы связей равны нулю, т. е. исключаются обратные связи с выхода на вход одного нейрона.

Для однослойной нейронной сети со связями типа «все ко всем» характерна сходимость к одной из конечного множества равновесных точек, которые являются локальными минимумами функции энергии, отражающей структуру всех связей в сети. Введенная Хопфилдом функция вычислительной энергии нейронной сети описывает поведение сети через стремление к минимуму энергии, который соответствует заданному набору образов. В связи с этим сети Хопфилда могут выполнять функции ассоциативной памяти, обеспечивая сходимость к тому образу, в область притяжения которого попадает начальный паттерн (образец) активности нейронов сети.

Этот подход привлекателен тем, что нейронная сеть для конкретной задачи может быть запрограммирована без обучающих итераций. Веса связей вычисляются на основе вида функции энергии, сконструированной для решаемой задачи.

Сети Хопфилда получили применение на практике в основном как реализации подсистем более сложных систем. Они имеют определенные недостатки, ограничивающие возможности их применения:

* Предположение о симметрии связей между элементами, без которой нельзя ввести понятие энергии.
* Нейронная сеть - это устройство для запоминания и обработки информации, а не устройство минимизации энергии. Экономия энергии играет в этих процессах вспомогательную роль.
* Сети Хопфилда поддерживают множество лишних, неэффективных, иногда дублирующих друг друга связей. В реальных нервных системах такие связи не поддерживаются, так как их реализация требует определенных затрат. В биологических нервных системах происходит освобождение от лишних связей за счет их структуризации. При этом вместо организации связей «всех ко всем» используется многослойная иерархическая система связей.

**Самоорганизующиеся сети Т. Кохонена**. Идея сетей с самоорганизацией на основе конкуренции между нейронами базируется на применении специальных алгоритмов самообучения ИНС. Сети Кохонена обычно содержат один (выходной) слой обрабатывающих элементов с пороговой передаточной функцией. Число нейронов в выходном слое соответствует количеству распознаваемых классов. Настройка параметров межнейронных соединений проводится автоматически на основе меры близости вектора весовых коэффициентов настраиваемых связей к вектору входных сигналов в эвклидовом пространстве. В конкурентной борьбе побеждает нейрон, имеющий значения весов, наиболее близкие к нормализованному вектору входных сигналов. Кроме того, в самоорганизующихся сетях возможна классификация входных образцов (паттернов). На практике идея Кохонена обычно используется в комбинации с другими нейросетевыми парадигмами.

**Свёрточная нейронная сеть** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) convolutional neural network, CNN) — специальная архитектура [искусственных нейронных сетей](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), предложенная [Яном Лекуном](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D1%83%D0%BD,_%D0%AF%D0%BD) в 1988 году и нацеленная на эффективное [распознавание образов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%80%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2), входит в состав технологий [глубокого обучения](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D1%83%D0%B1%D0%BE%D0%BA%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) deep learning). Идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) convolution layers) и субдискретизирующих слоёв ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) subsampling layers или [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) pooling layers, слоёв подвыборки). Структура сети — однонаправленная (без обратных связей), принципиально многослойная. Для обучения используются стандартные методы, чаще всего [метод обратного распространения ошибки](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BE%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B3%D0%BE_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F_%D0%BE%D1%88%D0%B8%D0%B1%D0%BA%D0%B8). [Функция активации нейронов](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%B0%D1%86%D0%B8%D0%B8) (передаточная функция) — любая, по выбору исследователя.

Название архитектура сети получила из-за наличия операции [свёртки](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%91%D1%80%D1%82%D0%BA%D0%B0_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7)), суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения.

**7. Байесовский классификатор (наивный Байес)**

Самые простые решения обычно оказываются самыми действенными, и в этом смысле показателен пример наивного байесовского алгоритма. Несмотря на большие успехи машинного обучения в последние годы, наивный байесовский алгоритм остаётся не только одним из простейших, но и одним из самых быстрых, точных и надежных. Он успешно используется для многих целей, но особенно хорошо работает с задачами обработки естественного языка.

Наивный байесовский алгоритм — это вероятностный алгоритм машинного обучения, основанный на применении **теоремы Байеса** и используемый в самых разных задачах классификации. В статье разберём все основные принципы и понятия, связанные с наивным байесовским алгоритмом, чтобы у вас не осталось проблем с его пониманием.

**Теорема Байеса**

Теорема Байеса — это простая математическая формула, используемая для вычисления условных вероятностей.

**Условная вероятность** — это вероятность наступления одного события при условии, что другое событие (по предположению, допущению, подтверждённому или неподтверждённому доказательством утверждению) уже произошло.

Формула для определения условной вероятности:

**P(A|B)= P(B|A)\* P(A) / P(B)**

Она показывает, как часто происходит событие A *при наступлении события B*, обозначается как **P(A|B)** и имеет второе название «апостериорная вероятность». При этом мы должны знать:

1. как часто происходит событие B *при наступлении события A*, что обозначается в формуле как **P(B|A)**;
2. какова вероятность того, что A не зависит от других событий, обозначаемая в формуле как **P(A)**;
3. какова вероятность того, что B не зависит от других событий. В формуле она обозначается как **P(B)**.

*Вкратце можно сказать, что теорема Байеса — это способ определения вероятности исходя из знания других вероятностей.*

**Допущения наивного байесовского алгоритма**

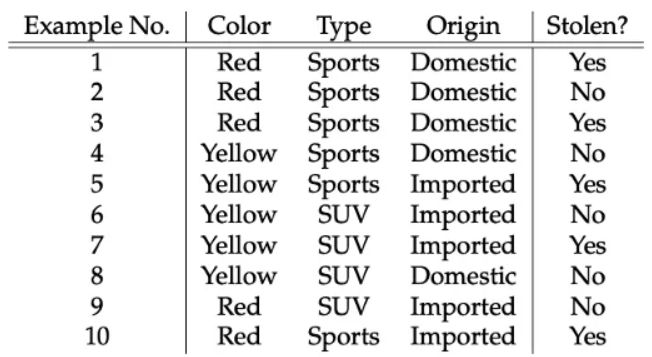
Основным допущением наивного байесовского алгоритма является то, что каждая характеристика вносит ***независимый*** и ***равный*** вклад в конечный результат.

Для лучшего понимания давайте рассмотрим пример. Это будет задача для автоугонщика.

Дано: параметры автомобиля (цвет, тип, страна производства).

Найти: угнана или нет (да или нет).

**Пример.** Все данные представлены в этой таблице:

****

Теперь мы можем оценить характер делаемых в рамках наивного байесовского алгоритма допущений с использованием данных нашего примера:

* предположим, что среди этих параметров нет зависимых друг от друга. Например, красный цвет не имеет никакого отношения к типу или стране производства автомобиля. Отсюда первое допущение, что эти параметры являются **независимыми**;
* второе допущение: у каждого параметра одинаковое влияние (или важность). Например, зная только цвет и тип, нельзя предсказать результат. Ведь ни один из параметров не имеет большего или меньшего значения по сравнению с другими, и все вносят **равный** вклад в результат.

**Примечание:** допущения наивного байесовского алгоритма, как правило, некорректны в реальных ситуациях. Допущение о независимости всегда некорректно, но часто хорошо работает на практике. **Поэтому алгоритм и называется наивным.**

Вернёмся к таблице: **здесь нам надо определиться с тем, какие машины угоняются, а какие — нет, исходя из параметров автомобилей**. В столбцах у нас параметры машин, а в строках — варианты с разными сочетаниями этих параметров. Если мы возьмём первую строку, то увидим: цвет — красный, тип — спортивный, страна производства — местное производство. То есть угоняется спортивная машина красного цвета, произведённая в РФ. Ах да, это же наивный алгоритм! Теперь попробуем решить, угоняется ли красный отечественный внедорожник. Вы когда-нибудь видели этого зверя? Вот и в нашей таблице такого варианта нет.

Для целей взятого нами примера перепишем теорему Байеса. Просто A и B заменим на y и X.

**P(y|X)= P(X|y)\* P(y) / P(X)**

Переменная **y** — это переменная класса (последний столбик с ответом на вопрос, угоняется машина или нет при данных условиях). Переменной **X** обозначаются параметры/характеристики:

**X=(x1,x2,…,xn)**

Здесь эти параметры **x1,x2,….xn** могут представлять цвет, тип и страну производства. Подставляя **X** и используя цепное правило, получаем теорему Байеса вот в таком расширенном виде:

**P(y| x1,x2,…,xn) = P(x1|y)\*P(x2|y)…\*P(xn|y)\*P(y) / (P(x1)\* P(x2)…\* P(xn))**

Теперь можно получить значения для каждого из параметров: берём данные и подставляем их в уравнение. Для всех вариантов сочетаний этих параметров знаменатель не меняется, он остаётся статичным. Так что его можно отбросить, введя в уравнение пропорциональность:

**P(y| x1,x2,…,xn) ~ P(x1|y)\*P(x2|y)…\*P(xn|y)\*P(y)=P(y)\*Πni=1P(xi)**

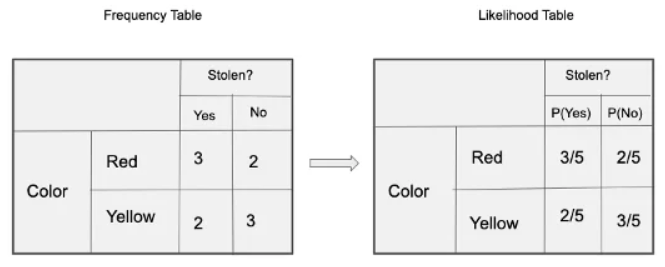
В нашем случае переменная класса (**y**) двумерная и имеет только два результата (класса): да или нет. Классификация может быть многомерной. Поэтому надо найти переменную класса (**y**) с максимальной вероятностью.

**y = argmaxy P(y)\*Πni=1P(xi)**

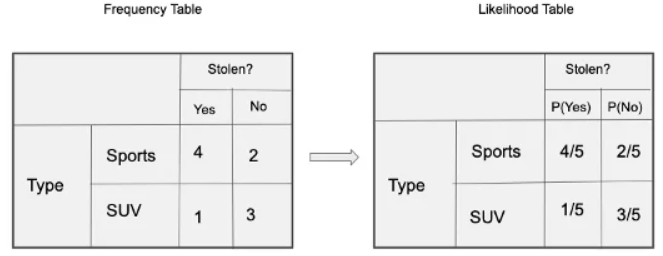
Используя эту функцию, можно получить класс, исходя из имеющихся предикторов/параметров.

Апостериорная вероятность **P(y|X)** высчитывается так: сначала создаётся **частотная таблица** для каждого параметра относительно искомого результата. Затем из частотных таблиц формируются **таблицы правдоподобия**, после чего с помощью уравнения Байеса высчитывается апостериорная вероятность для каждого класса. Класс с наибольшей апостериорной вероятностью и будет прогнозируемым результатом. Ниже приведены частотные таблицы и таблицы правдоподобия для всех трёх предикторов.

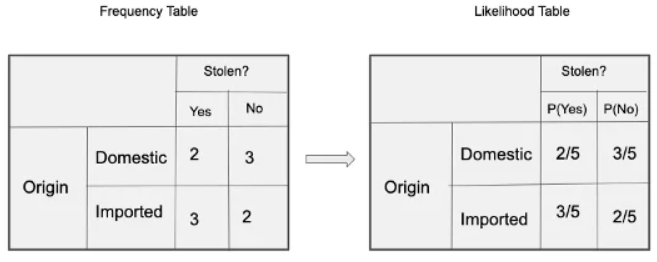
Частотная таблица и таблица правдоподобия для параметра «цвет»:



Частотная таблица и таблица правдоподобия для параметра «тип»:



Частотная таблица и таблица правдоподобия для параметра «страна производства»:



В нашем примере три предиктора **X**:

Подставляя значения из приведённых выше таблиц в уравнение, вычисляем апостериорную вероятность **P(Yes | X**):

**P(Yes|X)=P(Red|Yes)\*P(SUV|Yes)\*P(Domestic|Yes)\*P(Yes)**

**P(Yes|X)=3/5\*1/5\*2/5\*5/10=0,6\*0,2\*0,4\*0,5=0,024**

и

**P(No|X)=P(Red|No)\*P(SUV|No)\*P(Domestic|No)\*P(No)**

**P(No|X)=2/5\*3/5\*3/5\*5/10=0,4\*0,6\*0,6\*0,5=0.072**

Так как **0,072 > 0,024**, то для сочетания параметров RED SUV Domestic взятого нами примера (красного отечественного внедорожника) ответом на вопрос, угоняется ли машина, будет “нет”.

**Проблема нулевой частоты**

Одним из недостатков наивного байесовского алгоритма является то, что если класс и значение параметра не встречаются вместе, то оценка вероятности, высчитываемой с использованием частот, будет равна нулю. В итоге после перемножения всех вероятностей мы получим ноль.

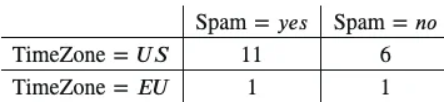
В байесовской среде было найдено решение этой проблемы нулевой частоты: к каждой комбинации класса и значения параметра добавлять единичку, когда значение параметра нулевое.

Предположим, у нас есть такой набор данных:

𝑃(TimeZone=𝑈𝑆|Spam=𝑦𝑒𝑠)=10/10=1

𝑃(TimeZone=𝐸𝑈|Spam=𝑦𝑒𝑠)=0/10=0

Появляется нулевое значение, поэтому при вычислении вероятностей добавляем единичку к каждому значению этой таблицы:



𝑃(TimeZone=𝑈𝑆|Spam=𝑦𝑒𝑠)=11/12

𝑃(TimeZone=𝐸𝑈|Spam=𝑦𝑒𝑠)=1/12

Вот так мы избавляемся от нулевой вероятности.

**Типы наивного байесовского классификатора:**

* **мультиномиальный**: здесь векторы признаков представляют собой значения частотности, то есть частоту, с которой генерируются те или иные события посредством **мультиномиального распределения**. Это модель событий, обычно используемая для классификации документов;
* **Бернулли**: в многомерной модели событий Бернулли характеристики являются независимыми логическими значениями (двоичными переменными), которыми описываются входные данные. Подобно мультиномиальной модели, эта модель широко применяется в задачах классификации документов, где используется не частотность термина (т. е. частота встречаемости слова в документе), а бинарные характеристики встречаемости терминов (т. е. встречается слово в документе или нет);
* **Гаусса:** предполагается, что непрерывные значения всех характеристик имеют **распределение Гаусса (**[нормальное распределение](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5)**)**. При нанесении на график получается колоколообразная кривая, симметричная относительно средней значений характеристик.

**8. Метрики в задачах машинного обучения (Accuracy, precision и recall)**

Перед переходом к самим метрикам необходимо ввести важную концепцию для описания этих метрик в терминах ошибок классификации — *confusion matrix* (матрица ошибок).

Допустим, что у нас есть два класса A, B и алгоритм, предсказывающий принадлежность каждого объекта одному из классов. Для каждого объекта в выборке возможно 4 ситуации:

* предсказали *положительную* метку и *угадали*. Такие объекты относятся к **true positive** (**TP**) группе (true – потому что предсказали правильно, а positive – потому что предсказали положительную метку);
* предсказали *положительную* метку, но *ошиблись* в своём предсказании – **false positive** (**FP**) (false, потому что предсказание было неправильным);
* предсказали *отрицательную* метку и *угадали* – **true negative** (**TN**);
* предсказали *отрицательную* метку, но *ошиблись* – **false negative** (**FN**).

Для удобства все эти 4 числа изображают в виде таблицы, которую называют **confusion matrix** (**матрицей ошибок**):

|  |  |  |
| --- | --- | --- |
|  | A | B |
| A\* | **TP** | **FN** |
| B\* | **FP** | **TN** |

Здесь A\*, B\* — это ответ алгоритма на объекте, а A,B — истинная метка класса на этом объекте.

Таким образом, ошибки классификации бывают двух видов: False Negative (FN) и False Positive (FP).

**Accuracy**

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

Accuracy= (TP+TN)/(TP+TN+FP+FN)

Эта метрика бесполезна в задачах с неравными классами, и это легко показать на примере.

Допустим, мы хотим оценить работу спам-фильтра почты. У нас есть 110 писем: 100 не-спам писем, 90 из которых наш классификатор определил правильно (True Negative = 90, False Positive = 10), и 10 спам-писем, 5 из которых классификатор также определил правильно (True Positive = 5, False Negative = 5).

Тогда accuracy:

Если мы просто будем предсказывать все письма как спам, то получим accuracy:

Accuracy=(5+90)/(5+90+10+5)=86,4

Если мы будем предсказывать все письма как не-спам, то получим более высокую accuracy:

Accuracy=(0+100)/(0+100+0+10)=90,9

При этом, наша модель совершенно не обладает никакой предсказательной силой, так как изначально мы хотели определять письма со спамом. Преодолеть это нам поможет переход с общей для всех классов метрики к отдельным показателям качества классов.

**Precision, recall и F-мера**

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

precision=TP/(TP+FP)

recall=TP/(TP+FN)

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

Именно введение precision не позволяет нам записывать все объекты в один класс, так как в этом случае мы получаем рост уровня False Positive. Recall демонстрирует способность алгоритма обнаруживать данный класс вообще, а precision — способность отличать этот класс от других классов.

Как мы отмечали ранее, ошибки классификации бывают двух видов: False Positive и False Negative. В статистике первый вид ошибок называют ошибкой I-го рода, а второй — ошибкой II-го рода. В нашей задаче по определению оттока абонентов, ошибкой первого рода будет принятие лояльного абонента за уходящего, так как наша нулевая гипотеза состоит в том, что никто из абонентов не уходит, а мы эту гипотезу отвергаем. Соответственно, ошибкой второго рода будет являться "пропуск" уходящего абонента и ошибочное принятие нулевой гипотезы.

Precision и recall не зависят, в отличие от accuracy, от соотношения классов и потому применимы в условиях несбалансированных выборок.

Часто в реальной практике стоит задача найти оптимальный (для заказчика) баланс между этими двумя метриками. Классическим примером является задача определения оттока клиентов.

Очевидно, что мы не можем находить **всех** уходящих в отток клиентов и **только** их. Но, определив стратегию и ресурс для удержания клиентов, мы можем подобрать нужные пороги по precision и recall. Например, можно сосредоточиться на удержании только высокодоходных клиентов или тех, кто уйдет с большей вероятностью, так как мы ограничены в ресурсах колл-центра.

Обычно при оптимизации гиперпараметров алгоритма используется одна метрика, улучшение которой мы и ожидаем увидеть на тестовой выборке.

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

Fβ=(1+β2)\* (precision \* recall)/(( β2 \* precision)+ recall)

β в данном случае определяет вес точности в метрике, и при β=1 это среднее гармоническое (с множителем 2, чтобы в случае precision = 1 и recall = 1 иметь F1=1)

F1=2\*(precision \* recall)/(precision+ recall)

F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

Пример. Три класса текстов

Таблица сопряжённости тестирования:

+ ---------- + ---------- + ---------- + ---------- +

| БЕЗ ЭМОЦИЙ | 17 | 0 | 1 |

+ ---------- + ---------- + ---------- + ---------- +

| УЖАСЫ | 1 | 10 | 7 |

+ ---------- + ---------- + ---------- + ---------- +

| ЮМОР | 1 | 2 | 16 |

+ ---------- + ---------- + ---------- + ---------- +

| | БЕЗ ЭМОЦИЙ | УЖАСЫ | ЮМОР |

+ ---------- + ---------- + ---------- + ---------- +

Обобщённый отчёт тестирования:

precision recall f1-score support

БЕЗ ЭМОЦИЙ 0.89 0.94 0.92 18

УЖАСЫ 0.83 0.56 0.67 18

ЮМОР 0.67 0.84 0.74 19

Accuracy: 0.7818181818181819

«БЕЗ ЭМОЦИЙ»

TP=17; FP=1+1=2; FN=1

precision=17/(17+2)=17/19=0,8947

recall=17/(17+1)=17/18=0,9444

f1=2\*(0,8947\*0,9444)/ (0,8947+0,9444)=2\*0,84495468/1,8391=0,9188

**9. Формальные модели данных и знаний**

**1. Знание**

Остановимся на следующем определении «знания».

***Знание*** – информация, выраженная, зафиксированная и функционирующая в особых символических знаковых системах – языках. Традиционная гносеология установила, что знание возникает в результате интерференции и суперпозиции ряда сложных процессов. Изучение этих процессов привело к возникновению ряда дисциплин – нейрофизиологии, психологии, психолингвистики, логики и т.д. Существует проблема установления взаимоотношений между биологическими (мозг) и механическими (компьютер) устройствами с точки зрения их способности вырабатывать и представлять знания.

Укажем три типа отношений знаков и знаковых конструкций, фиксирующих и выражающих знания.

1. Семантические отношения - правила установления смысла и значения знака или знаковой конструкции. Знак и знаковая конструкция обозначают некую вещь, процесс, предмет, ситуацию, действие или систему действий, каждый знак или знаковая конструкция должны иметь смысл и значение, чтобы расцениваться как средство выражения знаний.
2. Синтаксические отношения – правила синтаксиса между знаковыми конструкциями. Они указывают, как из одних осмысленных выражений получить другие, тоже осмысленные, с более или менее определенным значением.
3. Прагматические отношения - правила описания наборов и последовательностей действий, которые должны или могут осуществляться на базе тех или иных знаний. Все системы действий, выполняемых человеком, опираются на знания. Они образуют информационную базу деятельности.

В информационных технологиях, связанных с разработкой ИС, представление знаний осуществляется с использованием формальных моделей, которые делятся на декларативные и процедурные.

**1.1 Декларативные модели представления данных и знаний**

Одной из проблем ИС является обеспечение пользователей компетентными заключениями, касающимися определенных предметных областей. В основу решения данной проблемы положим исследования в области моделей представления знаний о предметной области или моделей информационного фонда. *Декларативное* описание моделей с позиций разработки программных систем будем осуществлять с точки зрения парадигмы «структуры предметной области». Выделим следующие виды моделей.

1. Иерархическая структура. Предметная область представляется в виде частей **Qi**, каждая из которых описывается в виде дерева c ветвями **r**-ого ранга и **j**-ым количеством элементов на **r**-ом ранге.

**Qi={Qr,j}**, **P={Q1,…,Qn}**

2. Сетевая структура. Вся предметная область рассматривается как совокупность частей, связанных между собой бинарными отношениями. Отдельные части, поддающиеся разбиению, имеют такую же структуру.

Объект идентифицируется своим именем **xi**. Бинарные отношения описываются с помощью символов **Rк**. Тогда описание предметной области примет вид:

**Ci=(xj,Rt,xg)**

**P={C1,…,Cn}**

3. Релятивная структура. Предметная область рассматривается как совокупность нескольких множеств, между элементами которых существуют отношения.

4. Объектно-характеристическая структура. Предметная область представляется множеством объектов. Каждому объекту приписываются определенные характеристики. Объект идентифицируется по своему наименованию, а характеристики по своему наименованию и значению, свойственному данному объекту.

**P={X1…Xk}, Xi=(x, {(y1,z1),…( yj,zj)}), i=1..k**

где **Р** – предметная область, **Xi** – объект описания, **x** – имя объекта, **yj** – название характеристики, **zj** – значение характеристики.

Описание каждого объекта можно представлять (с избыточностью) как набор троек: (**xi,yj,zi,j**).

5. **Фреймовая структура.** Стремление разработать описание структуры предметной области, соединяющее и обобщающее достоинства различных моделей, привело к возникновению фреймового представления.

Предметная область представляется множеством объектов, между которыми существуют отношения. Каждый объект **Xi** являет собой множество слотов **Yi**. Структура слота – есть пара **(yi,si)**, где yi - имя слота, **si** - значение слота. В качестве значения слота **si** могут выступать не только пассивные значения данных **Zi⊆Yi** (как в объектно-характеристической структуре), но и активные – программы **Ai⊆Yi**, а также ссылки **Si⊆Yi** на другие объекты. Последние позволяют организовывать сложные взаимосвязи (отношения **Rs**) между объектами.

**P={X1…Xk}, Xi=(x, {(y1,s1),…( yj,sj)}), ∃Si | sj∈Si: (xg Rsj xm), i=1..k.**

где **Xi** – объект описания, **xi** – имя объекта, **Yi={(y1,s1),…( yj,sj}** – множество слотов описания объекта.

Описатель объекта по составу и типу слотов и есть декларативное понятие фрейма.

**Fi=(fi, [(y1,t1),…, (yk,tk)]n)**,

где **fi** – описатель фрейма (ключ, комментарий и пр.), **yi** – имя слота, **ti** – тип слота, **n** – список экземпляров (при n>1 получаем релятивную структуру).

Рассматриваемая фреймовая организация при определенных допущениях позволяет имитировать модели данных с 1-ой по 4-ую. Необходимо отметить, что концепция фреймов в практическом применении имеет инвариантную реализуемость, поэтому ее формула носит сугубо теоретическое, обобщенно-концептуальное значение.

Но проблема ИС состоит не только в том, как представлять, но и как использовать знания, которыми располагают эксперты. Модели «использования» знаний будем относить к *процедурным* моделям знаний (в отличие от декларативных моделей).

**1.2 Процедурные модели знаний**

В соответствии с введенными видами подходов, классифицируем модели представления знаний на логические (формальные) и эвристические (формализованные). В основе логических моделей представления знаний лежит понятие формальной системы (теории). В логических моделях, как правило, используется исчисление предикатов первого порядка, дополненное рядом эвристических стратегий, а отношения, существующие между отдельными единицами знаний, выражаются только с помощью синтаксических правил, используемых формальной системой.

**Модели представления знаний на базе логики предикатов.** В основе *логических моделей* лежит понятие формальной теории **Tl**, задаваемой четверкой:

**Tl = (V,L,F,R)**, (1.1)

где **V** - счетное множество базовых символов (алфавит); **L** – множество синтаксических правил, позволяющее строить из **V** синтаксически правильные выражения – формулы теории; **F** - выделенное множество формул, называемых аксиомами теории **T**, т.е. множество априорно истинных формул; **R** - конечное множество отношений между формулами, называемыми правилами вывода.

Наиболее распространенной формальной системой, используемой для представления знаний, является исчисление предикатов первого порядка. Алфавит исчисления предикатов состоит из набора символов, к которым относятся знаки пунктуации, логические связки, знаки-кванторы, символы формирования переменных и пр. Основной задачей, решаемой в рамках исчисления предикатов, является выяснение истинности или ложности заданной формулы на некоторой области интерпретации. При этом особая роль отводится общезначимым формулам, т.е. формулам, истинным при любой интерпретации, и невыполнимым формулам, т.е. формулам, ложным при любой интерпретации.

**Вывод в логических моделях**.Вывод в формальной логической системе является процедурой, которая из заданной группы выражений выводит отличное от заданных семантически правильное выражение. Эта процедура, представленная в определенной форме, и является правилом вывода. Если группа выражений, образующая посылку, является истинной, то должно гарантироваться, что применение правила вывода обеспечит получение истинного выражения в качестве заключения.

Наиболее часто используются два метода. Первый – метод правил вывода, или метод естественного (натурального) вывода, названный так потому, что используемый тип рассуждений в исчислении предикатов приближается к обычному человеческому рассуждению. Второй – метод резолюций. В его основе лежит исчисление резольвент.

**Метод правил вывода.** В разной литературе можно встретить разные названия метода правил вывода, например, правила дедуктивных выводов или более часто **modus ponens**. Принцип работы правил вывода хорошо иллюстрирует следующий пример: «Если высказывание *«А»* влечет (имплицирует) высказывание *«В»* и высказывание *«А»* истинно, то, следовательно, *«В»* истинно»

В логике предикатов имеются универсальные правила, оперирующие с формулами, содержащими свободные переменные. Решение задач (получение выводов) в логических моделях может основываться на применении подобных правил к исходной совокупности истинных предикатов как доказательство правильности какого-либо составного предиката. Поскольку ответ получается как заключение из комбинации уже существующих логических формул, то по аналогии с выводами в продукционных моделях его можно назвать прямым (обратным) выводом. Однако всегда следует учитывать, что в формальной логике причинно-следственные отношения игнорируются.

Суть процедуры вывода заключается в рекурсивном применении подстановки известных значений в составной предикат. При этом принципиально гарантируется, что доказательство истинности результата можно проверить формальной процедурой.

Если в формальной логической модели механизм логического вывода использует метод правил вывода, то есть основания эту модель отнести к продукционным или логико-лингвистическим.

**Пример:** вывод решения в логической модели на основе правила вывода – modus ponens.

**Даны утверждения:**

* «Карась – рыба»;
* «Рыбы – обитают в воде»;
* «Обитающие в воде – могут попасть в сети».

Требуется доказать утверждение **«Карась может попасть в сети»**.

**Решение:**

**Шаг 1.** Представим высказывания в предикатной форме:

Рыба(Карась)

∀(**X**)(Рыбы(**X**)→Обитают\_в\_воде(**X**))

∀(**X**)(Обитющие\_в\_воде(**X**) → Могут\_попасть\_в\_сети(**X**)

**Шаг 2.** На основе правила вывода во втором предикате получим утверждение: «Карась обитает в воде»

**Шаг 3.** На основе правила вывода в третьем предикате получим утверждение: «Карась может попасть в сети»

**Метод резолюций.** Метод резолюций опирается на исчисление резольвент. Существует теорема, утверждающая, что вопрос о доказуемости произвольной формулы в исчислении предикатов сводится к вопросу о доказуемости пустого списка в исчислении резольвент. Поэтому доказательство того, что список формул в исчислении резольвент пуст, эквивалентно доказательству ложности формулы в исчислении предикатов.

Метод резолюций основывается на доказательстве истинности или ложности выдвинутого предположения методом «от противного». Для этого в исходное множество предложений включают аксиомы формальной системы и отрицание доказываемой гипотезы. Если в процессе доказательства возникает противоречие между отрицанием гипотезы и аксиомами, выражающееся в нахождении пустого списка (дизъюнкта), то выдвинутая гипотеза правильна.

В методе резолюций множество предложений обычно рассматривается как составной предикат, содержащий несколько предикатов, соединенных логическими функциями и кванторами существования и общности. Так как одинаковые по смыслу предикаты могут иметь разный вид, то предложения преобразуются в клаузальную форму – разновидность конъюнктивной нормальной формы (КНФ), в которой удалены кванторы существования, всеобщности, символы импликации, равнозначности и др. Клаузальную форму называют сколемовской конъюнктивной формой.

Если требуется методом резолюций доказать истинность какого-либо логического утверждения, то отрицание этого утверждения преобразуется в клаузальную форму, по его предложениям выполняется поиск пустого предложения с использованием унификации и вывода резольвент. Невыполнимость отрицания подтверждает истинность рассматриваемого утверждения.

Метод резолюций получил широкое распространение из-за высокой эффективности машинной обработки. На его основе построен язык «Prolog».

Существенным недостатком метода резолюций является то, что он предназначен только для доказательства теорем. Он не пригоден для порождения новых предложений. К тому же, если предложение не является теоремой, резолюция может привести к построению бесконечного дерева решений.

**Решение задачи примера метода правил вывода.** Требуется доказать утверждение **«Карась может попасть в сети»**.

**Шаг 1.** Преобразуем высказывания в дизъюнктивную форму:

|  |  |
| --- | --- |
| Предикатная форма: | Дизъюнктивная форма: |
| Рыба(карась) | Рыба(Карась) |
| ∀(X)(Рыбы(X)→Обитают\_в\_воде(X)) | ¬(Рыбы(X) ∨ Обитают\_в\_воде(X)) |
| ∀(Y)(Обитающие\_в\_воде(Y) → Могут\_попасть\_в\_сети(Y) | ¬ (Обитающие\_в\_воде(Y) ∨ Могут\_попасть\_в\_сети(Y)) |

**Шаг 2.** Запишем отрицание целевого выражения (требуемого вывода):

¬ Могут\_попасть\_в\_сети(карась)

**Шаг 3.** Составим конъюнкцию всех дизъюнктов (т. е. построим КНФ), включив в нее отрицание целевого выражения:

(¬(Рыбы(X) ∨ Обитают\_в\_воде(X))) ∧ (¬ (Обитающие\_в\_воде(X) ∨ Могут\_попасть\_в\_сети(X)))∧ (Могут\_попасть\_в\_сети(карась)) ∧ (¬ Могут\_попасть\_в\_сети(карась))

**Шаг 4.** В цикле проведем операцию поиска резольвент над каждой парой дизъюнктов:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 Дизъюнкт | | | 2 Дизъюнкт |
| ¬(Рыбы(X) ∨ Обитают\_в\_воде(X)) | | | ¬ (Обитающие\_в\_воде(Y) ∨ Могут\_попасть\_в\_сети(Y)) |
| Резольвента: ¬ Рыбы(Y) ∨ Могут\_попасть\_в\_сети(Y) | | | |
| ¬ Рыбы(Y) ∨ Могут\_попасть\_в\_сети(Y) | | Рыбы(карась) | |
| Резольвента: Могут\_попасть\_в\_сети(карась) | | | |
| Могут\_попасть\_в\_сети(карась) | ¬ Могут\_попасть\_в\_сети(карась) | | |

Получение пустого дизъюнкта означает, что высказывание «карась не может попасть в сети» ложно, значит истинно высказывание «карась может попасть в сети».

В целом метод резолюций интересен благодаря простоте и системности, но применим только для ограниченного числа случаев (доказательство не должно иметь большую глубину, а число потенциальных резолюций не должно быть большим). Кроме метода резолюций и правил вывода существуют другие методы получения выводов в логике предикатов.

**Продукционные модели знаний.** Такие модели превосходят предикатные путем использования эвристик **F**, которые дают возможность, как адекватного представления проблемной среды, так и эффективного использования правил вывода **R**. К эвристическим моделям, используемым в экспертных системах, можно отнести сетевые, продукционные и объектно-ориентированные модели. Следует отметить, что продукционные модели, используемые для представления знаний в экспертных системах, отличаются от формальных продукционных систем тем, что они используют более сложные конструкции правил, а также содержат эвристическую информацию о специфике проблемной среды, выражаемую часто в виде семантических структур. Эвристики строятся в соответствии с набором правил конструирования **L** на базе алфавита **V**. Продукционные модели будем представлять формулой:

**Tp = (V,L,F,R)**, (1.2)

где **V** – набор символов для формирования правил; **L** – язык конструирования правил; **F** – эвристики в виде правил; **R** – правила вывода (отношения эвристик).

**Продукции** являются наиболее популярными средствами **представления знаний** в информационных системах. В общем виде под продукцией понимают выражение вида A ® B. Обычное прочтение продукции выглядит так: ЕСЛИ А, ТО B. Импликация может истолковываться в обычном логическом смысле, как знак логического следования B из истинного А. Возможны и другие интерпретации продукции, например, А описывает некоторое условие, необходимое, чтобы можно было совершить действие B.

**Продукционная модель** *или модель, основанная на правилах,* позволяет представить знания в виде предложений типа

«**Если** <условие>, **то** <действие>».

Под *условием* понимается некоторое предложение — образец, по которому осуществляется поиск в базе знаний, а под *действием* — действия, выполняемые при успешном исходе поиска (они могут быть промежуточными, выступающими далее как условия, и терминальными или целевыми, завершающими работу системы).

При использовании продукционной модели база знаний состоит из набора правил, Программа, управляющая перебором правил, называется *машиной вывода*. Чаще всего вывод бывает прямой (от данных к поиску цели) или обратный (от цели для ее подтверждения – к данным). *Данные* — это исходные факты, на основании которых запускается машина вывода.

Если в памяти системы хранится некоторый набор продукций, то они образуют систему продукций. В системе продукций должны быть заданы специальные процедуры управления продукциями, с помощью которых происходит актуализация продукций и выполнение той или иной продукции из числа актуализированных.

В состав системы продукций входит база правил (продукций), глобальная база данных и система управления. *База правил* – это область памяти, которая содержит совокупность знаний в форме правил вида ЕСЛИ – ТО.

Глобальная база данных — область памяти, содержащая фактические данные (факты). Система управления формирует заключения, используя базу правил и базу данных. Существуют следующие способы формирования заключений — прямые и обратные выводы.

Правила вывода бывает удобно представлять в виде дерева решений (теория графов).

В прямых выводах выбирается один из элементов данных, содержащихся в базе данных, и если при сопоставлении этот элемент согласуется с левой частью правила (посылкой), то из правила выводится соответствующее заключение и помещается в базу данных или исполняется действие, определяемое правилом, и соответствующим образом изменяется содержимое базы данных.

В обратных выводах процесс начинается от поставленной цели. Если эта цель согласуется с правой частью правила (заключением), то посылка правила принимается за подцель или гипотезу. Этот процесс повторяется до тех пор, пока не будет получено совпадение подцели с данными.

При большом числе продукций в продукционной модели усложняется проверка непротиворечивости системы продукций, то есть множества правил. Поэтому число продукций, с которыми работают современные системы искусственного интеллекта, как правило, не превышает тысячи.

**Продукционная модель** привлекает разработчиков своей наглядностью, высокой модульностью, легкостью внесения дополнений и изменений и простотой механизма логического вывода.

**Семантические модели знаний.** В основе этих моделей лежит понятие сети, образованной помеченными вершинами и дугами. Вершины сети представляют некоторые сущности **V={v1,…,vn}** (объекты, события, процессы, явления), а дуги - отношения между сущностями **Rv={rv1,…,rvm}**, которые они связывают **(vl, rvi, vj)**.

Наложив ограничения на описание вершин и дуг, можно получить сети различного вида. Если вершины не имеют собственной внутренней структуры, то соответствующие сети называют простыми сетями. Если вершины обладают некоторой структурой **si (vi→si(vi))**, то такие сети называют иерархическими сетями. В настоящее время в большинстве приложений, использующих семантические сети, они являются иерархическими.

Одно из основных отличий иерархических семантических сетей от простых семантических сетей состоит в возможности разделить сеть на подсети (пространства) **pi (P={p1,…,pg}**, **pi⊆V, ∅)** и устанавливать отношения не только между вершинами **Rv**, но и между пространствами **Rp={rp1,…,rpz} (pi, rpk, pj)**. Все вершины и дуги являются элементами, по крайней мере, одного пространства. Отметим, что понятие пространства аналогично понятию скобок в математической нотации. Различные пространства, существующие в сети, могут быть упорядочены в виде дерева пространств, вершинам которого соответствуют пространства, а дугам - отношения видимости (**Rp**). Например, из пространства пространство-потомок видимы все вершины и дуги, лежащие в пространствах-предках, а остальные пространства невидимы. Отношение видимости позволяет сгруппировать пространства в упорядоченные множества – *перспективы*. Перспектива обычно используется для ограничения сетевых сущностей, видимых некоторой процедурой, работающей с сетью, и *могут быть положены в основу визуализации моделей знаний по иерархическим уровням представления*. Обычно в перспективу включают не любые, а иерархические сгруппированные пространства. Исходя из изложенного, будем представлять семантические модели в виде четверки:

**Tc = (V,Rv,P,Rp)**, (1.3)

где **V** – сущности; **Rv** – отношения между сущностями; **P** – пространства; **Rp** – отношения видимости.

При необходимости в иерархических сетях можно представить любые логические связки и кванторы. Кроме представления логических связок и кванторов сеть может быть использована также для кодирования других структур высших порядков.

**1.3 МОДЕЛИ ЗНАНИЙ НА ПРИМЕРАХ**

Представление знаний зависит от того, какая модель для этого используется. К моделям предъявляют следующие требования:

* формальность представления, ясность и простота, которые влияют на сложность программной реализации модели;
* дедуктивные возможности (возможности получения новых знаний) определяют правильность и непротиворечивость базы знаний и время получения ответов на запросы;
* реализация механизмов абстракции важна для понимания больших объемов информации и объединения представлений разных пользователей;
* возможность преобразования на естественный язык, в графическую форму, в таблицу и наоборот, что имеет значение для создания удобного пользовательского интерфейса, который обеспечивает отсутствие искажений и потерь информации при преобразованиях ее формы.

Ранее мы рассмотрели неформальную модель представления знаний “сущность - связь”. Рассмотрим еще три модели: продукционную, фреймовую модель и семантическую сеть. Каждая из моделей имеет свои достоинства и недостатки.

**Продукционная модель**

На использовании данной модели основаны многие современные экспертные системы, например, справочные средства текстовых и графических редакторов.

Модель основана на логике предикатов или продукционных правилах, которые описывают знания в форме "если - то". Под продукцией понимают выражение следующего вида:

*Q (i); P, A=>B, N,*

где i - имя продукции, например, "продажа товара",

Р - логическое выражение, описывающее условие применимости продукции, например, условием продукции "продажа товара" может быть "Товар имеется на складе";

N - описание постусловий продукции, то есть действий, которые выполняются после реализации В.

A=>B - ядро продукции, где А называют посылкой, а В - выводом или целью. Правило А => В читается как "Если справедливо А, то верно В" или "если выполняется условие А, то выполнить действие В".

Посылка “ЕСЛИ” содержит обычно один или несколько фактов, объединенных логической операцией “И”; вывод “ТО” представляет собой новый факт.

Экспертные системы, которые базируются на продукциях, состоят из 3-х компонентов: базы данных, базы правил и интерпретатора:

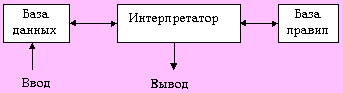


Рис. 5.1. Структура продукционной экспертной системы

БД хранит исходные факты и результаты выводов. Правила содержатся в базе правил. Интерпретатор (механизм логического вывода) соединяет факты и правила, чтобы получить новые факты; работа интерпретатора не зависит от того, что он интерпретирует, - этим логическое программирование принципиально отличается от обычного. Имеются языки логического программирования, например, ПРОЛОГ.

Различают прямой и обратный вывод.

В прямом выводе исходной точкой являются факты, которые содержатся в БД. Из БД выбирается факт и подставляется в очередное правило. Если посылка истинна, то получается новый факт, который заносится в БД либо выполняется какое-то действие. Процесс заканчивается, когда очередной факт подставлен во все правила. Не все правила приводят к появлению новых фактов.

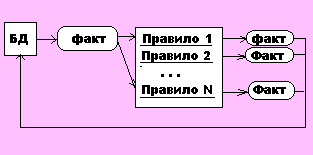


Рис.5 .2. Схема прямого вывода

Обратный вывод начинается от поставленной цели. Этот факт (гипотеза) подставляется в заключительную часть очередного правила, а посылка этого правила принимается за новую цель - гипотезу. Если посылка имеется в БД, то процесс заканчивается.

Нельзя сказать, какой из способов вывода лучше, так как это зависит от решаемой проблемы. В системах с обратным выводом можно часть правил исключить из перебора и тем самым сократить время вывода, поэтому такие системы более распространены.

**Пример.** Пусть имеются факты и правило:

Факт 1: Иванов - служащий

Факт 2 : Петров - начальник Иванова

Факт 3: Сидоров - начальник Петрова

Правило: Если X начальник для Y, то Х - служащий.

Прямой вывод: подставляем Факт 1 в Правило: Факт не подходит,

Факт 2: Х = Петров, Y= Иванов, получаем вывод: Петров - служащий,

Факт 3: аналогично получаем вывод: Сидоров - служащий.

Обратный вывод: Выдвигаем гипотезу: Сидоров - служащий и подставляем этот факт в правило:

Если *Сидоров начальник для Y*, то *Сидоров служащий*.

Ищем в БД левую часть правила - факт “*Сидоров начальник для Y”.* Путем перебора находим, что подходит факт 3 (при этом переменная Y принимает значение "Петров"). Значит, выдвинутая гипотеза верна.

Достоинства продукционной модели:

1. Простота создания и описания правил,
2. Простота механизмов выводов,
3. Возможность описания модели на языке логического программирования ПРОЛОГ.

Недостатки модели:

Правила могут быть противоречивы и эти противоречия может устранить только человек - разработчик модели, что затруднительно при решении больших задач.

**Семантические сети**

Идея модели состоит в том, чтобы представить в формальном виде предложения естественного языка. Семантическая сеть представляет собой ориентированный граф, в котором понятия предметной области моделируются вершинами графа, а связи между ними - дугами. Выделяют три вида вершин сети: конкретные значения свойств сущностей, понятия (объекты) предметной области, ситуации.

Группа взаимосвязанных понятий и утверждение об их истинности называется *ситуацией*. Ситуации являются основной категорией для описания предметной области. Связь между понятиями отражает простейший *факт.* Более сложные утверждения, которые в естественном языке составляют отдельное предложение, соответствуют *подграфу сети.* Дуги сети делятся на три типа: сущность - сущность, сущность - понятие, понятие - понятие.

В сети каждая связь (дуга) имеет имя, которое выражает *роль* понятия в данной ситуации. В отличие от других моделей, одна и та же сущность может быть связана ролевыми отношениями с несколькими понятиями. Например, сущность Иванов И.И. в разных ситуациях играет разные роли: Член профсоюза, Автор книги, Преподаватель, Садовод, Налогоплательщик, Автолюбитель и т.д.

Наиболее типичные роли:

* АГЕНТ - инициатор действия
* ОБЪЕКТ - кто (что) подвергается действию
* ИСТОЧНИК - местонахождение объекта до действия:
* ПРИЕМНИК - куда попал объект после совершения действия,
* ВРЕМЯ - когда происходит действие,
* МЕСТО - где происходит действие
* ЦЕЛЬ - зачем производится действие.

**Пример.** Ситуация описывается предложением: *Доцент Н.А.Шигина читает лекции по дисциплине ТЭИС для студентов потока 98Э в 5 семестре в аудитории 227.*

Построим семантическую сеть, описывающую данную ситуацию (рис.5.3).

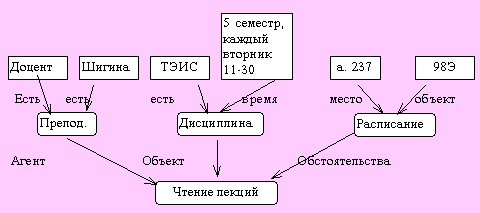


Рис. 5.3. Пример семантической сети

Все вершины в данной сети соответствуют конкретным понятиям.

В естественном языке часто встречаются предложения типа "Все волки серые". Таким связям между понятиями в математике соответствует квантор всеобщности. В семантической сети для отображения такой логической связи вводится вершина-переменная. Например, на рис. 5.4 представлен граф утверждения "В каждой группе есть староста и куратор".

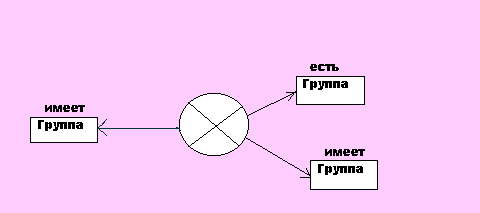


Рис.5.4. Сеть с переменной вершиной (обозначена кружком)

Для представления логических операций в семантических сетях используют рамки, охватывающие соответствующие дуги. Для операции "И" рамка не указывается, для операций "ИЛИ", "НЕ" и "Если -То" внутри рамки пишут название операции. Например, на рис. 5.5-а показана сеть для утверждения "Иванов учится в институте или работает на заводе, рис. 5.5-б изображает сеть для утверждения "Все студенты ПТИ специальности 0719 выпускаются кафедрой АИСТ".

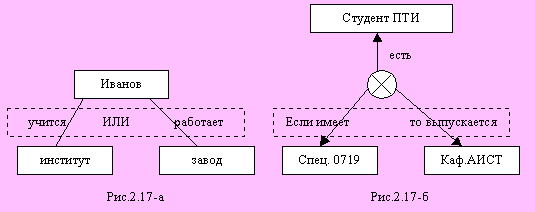


Рис. 5.5. Представление логических операций

Особенность семантической сети состоит в том, что эта модель не имеет механизма вывода новых фактов. Для поиска ответов в семантической сети используют следующие методы:

1. Установление соответствия сети фактов и целевой сети;
2. Наследование свойств тех множеств, в которые входит понятие.

С помощью этих методов можно определить свойства понятий или наоборот, найти объект по заданным свойствам.

**Идея метода А состоит в следующем.** Для каждого запроса к базе знаний строится своя семантическая сеть, в которой искомые данные моделируются пустыми вершинами сети. Построенная сеть сравнивается с общей моделью предметной области, хранящейся в БЗ. При этом “пустые вершины” принимают конкретные значения.

|  |  |
| --- | --- |
| Например, рассмотрим запрос: “Кто читает лекции по ТЭИС?” Сначала строим граф для заданного запроса (рис. 5.6), затем сравниваем этот граф с сетью (рис.5.3), начиная с вершин, моделирующих ситуацию (ситуация => понятия => сущность). В результате находим ответ: Шигина. | Рис.5.6. Граф для запроса |

Достоинства модели: наглядность и понятность, легкость отображения отношений абстракции и единство представления для всех видов знаний, возможность преобразования в естественный язык и графическую форму, наличие ассоциативных связей допускает целенаправленный поиск, в отличие от перебора в продукционной модели..

Недостатки модели: ограниченные возможности логического вывода; трудоемкость перебора вариантов для вывода при больших размерностях задач, отсутствие в настоящее время формального аппарата для описания и представления модели.

**Фреймовая модель**

Фреймовая модель была предложена М.Мински в 1975 году. *Фрейм* - это некоторая “пустая рамка”, структура для представления знаний, которая при заполнении конкретными значениями превращается в описание конкретного факта, события или ситуации. Центральная идея данного подхода состоит в отказе от формирования модели внешнего мира на основе разрозненных фактов и понятий. Фрейм должен строиться как модель стереотипной, часто повторяющейся ситуации или группы сходных ситуаций.

Модель знаний о предметной области представляет собой сеть фреймов. Основной тип связи между фреймами - связи по иерархии. Обычно верхний уровень иерархии (название ситуации, название действия) заполнен знаниями, а нижний уровень не полностью определен и уточняется, когда фрейм применяют к конкретной обстановке. Группа фреймов может объединяться в систему, в которой каждый фрейм играет свою роль. Систематизация знаний, то есть построение модели, возлагается на человека.

Основная структурная единица фрейма называется ***слотом***, который описывает некоторое действие, понятие или связь.

**Параметрами слота** являются уникальное имя, значение, тип данных и наследование.

*Наследование* определяет, какую информацию наследует данный слот из одноименного слота в фрейме верхнего уровня.

*Тип данных слота* может принимать следующие значения: указатель на фрейм верхнего уровня (для иерархической взаимосвязи фреймов), переменная, текст, список и процедура. Обратите внимание, что в отличие от других моделей, фрейм может включать в себя активные знания (процедуры).

**Пример.** Рассмотрим сеть фреймов, представляющую знания о студентах (рис.5.7).

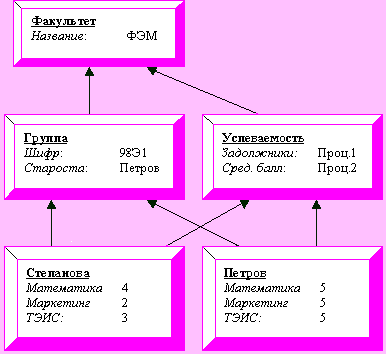


Рис.5.7. Пример сети фреймов

Фреймы "Степанова", "Петров" хранят уникальную информацию для каждого студента. Более общая информация о группе и факультете собрана в отдельных фреймах и может наследоваться. Процесс наследования состоит в получении данных при движении вверх по иерархии фреймов, например, следуя вверх по связи "Степанова" - "группа", можно узнать, что старостой группы, в которой учится Степанова, является Петров.

Между фреймами установлены связи типа "является частью", например, студенческая группа является частью факультета. Связь направлена к фрейму более высокого уровня.

Наряду с декларативными знаниями фрейм "Успеваемость" содержит процедурные знания - процедуры определения списка задолжников ("Проц.1") и расчета среднего балла ("Проц.2").

Различают два вида присоединенных процедур:

1. "демоны", которые автоматически активизируются при изменении данных в фрейме (например, как только Степанова сдает экзамен по маркетингу, она автоматически исключается из списка задолжников);
2. "слуги", которые активизируются по запросу пользователя (например, - "Проц.2" - расчет итогов в конце сессии).

Дополнительное новшество фреймовой модели по сравнению с другими моделями - наличие значений "по умолчанию", которые изначально приписываются слотам и могут в дальнейшем уточняться. Это помогает анализировать ситуации без их детального описания.

**Запросы к базе знаний** в данной модели реализуются следующим образом. Первоначально задаются значения слотов исходных данных, это может привести к активизации слотов-процедур, которые “генерируют” новые данные, в том числе могут задавать пользователю уточняющие вопросы, когда данных для ответа не хватает. В результате пользователю выдаются найденные значения искомых слотов.

Фреймовый подход к моделированию знаний реализован в языке ЛИСП.

Достоинства модели: естественность иерархического представления знаний для человека; легкость расширения и модификации знаний, комбинация декларативных и процедурных знаний, возможность преобразования знаний на естественный язык.

Недостатки модели: нет общепринятого формального аппарата, логический вывод целиком опирается на правила наследования, необходимость программирования процедур.

Таким образом, фреймовая модель удобна для представления знаний в тех областях, где четко выделена и систематизирована понятийная структура.

**5.4. Заключение**

Рассмотренные модели представления знаний имеют одинаковые возможности по структуризации знаний. Наиболее пригодны для реализации на ЭВМ продукционная и фреймовая модель, но последняя более трудоемка в программировании, а первая медленнее работает на больших задачах. Наиболее близка к естественному языку семантическая сеть.

**10. Средства разработки интеллектуальных ИС**

Можно говорить о трех поколениях ИС (или экспертных систем ЭС). К *первому поколению* следует относить статические поверхностные ЭС, ко *второму -* статические глубинные ИС (иногда ко второму поколению относят гибридные ИС), а к *третьему -* динамические ИС (они, как правило, будут глубинными и гибридными).

Инструментальные средства. Большинство *инструментальных средств (ИНС)* предназначено для создания прототипов ЭС, решающих статические задачи (обычно задачи расширения) в статических проблемных областях. По степени применимости ИНС выделяют следующие стадии существования: исследовательская, промышленная, коммерческая. Разделяют следующие типы ИНС:

* языки программирования;
* языки инженерии знаний;
* средства автоматизации разработки (проектирования) ЭС;
* оболочки ЭС.

С точки зрения потребителя, на выбор ИС влияют моменты:

* затраты труда на построение ЭС или ее прототипа с помощью ИНС;
* эффективность функционирования ЭС, построенной на основе выбранного ИНС;
* квалификация разработчика, необходимая для применения ИНС.

Оболочки ЭС ориентированы на работу с пользователем–непрофессионалом в области программирования. Основным свойством оболочек является то, что они содержат все компоненты ЭС в готовом виде и их использование не предполагает программирования, а сводится лишь к вводу в оболочку знаний о проблемной области. Каждая оболочка характеризуется фиксированным способом представления знаний и организации вывода и фиксирования компонентов, которые будут использоваться во всех положениях, где будет применяться оболочка.

Желание представить разработчику ЭС разнообразные средства для учета особенностей приложения, привело к объединению в рамках одной системы различных методов решения задач, представления и интерпретации знаний. В их состав могут входить средства модификации функционирования оболочки, набор компонентов, позволяющих конструировать собственные оболочки, средства комплексирования компонентов в виде языка высокого уровня, развитые интерактивные графические средства общения с пользователем. Подобные средства называют средствами автоматизации проектирования (разработки) ЭС.

Характеристика «*Универсальность»* определяет возможности ЭС в использовании различных способах представления знаний в рабочей памяти и базе знаний и различных парадигм функционирования системы. Наличие универсальности позволяет адекватно отображать в системе различные типы знаний о проблемной области. К настоящему времени в большинстве ЭС при представлении знаний используют фреймы и сети, а в качестве механизма функционирования, как правило, программирование, ориентированное на правила.

Средства приобретения знаний в существующих ЭС можно оценивать с точки зрения допустимых способов формирования БЗ. Выделяют следующие способы формирования БЗ:

* редакторы;
* средства отладки;
* средства индуктивного вывода новых знаний.

Редакторы позволяют отображать и модифицировать БЗ, возможно, в графическом виде, поддерживая ее целостность. Средства отладки обеспечивают анализ содержимого БЗ. переформирование и отображение его результатов пользователю. Средства индуктивного вывода осуществляют формирование новых знаний (правил) на основе вводимых пользователем примеров ситуаций с их решениями.

Класс ЭС. Как правило, выделяются два больших класса ЭС (существенно отличающихся по технологии их проектирования), которые условно можно назвать простыми и сложными ЭС. Простая ЭС может быть охарактеризована следующими значениями основных параметров: поверхностная ЭС; традиционная ЭС (реже гибридная). Сложная ЭС может быть охарактеризована следующими значениями параметров: глубинная ЭС; гибридная ЭС.

Следует отметить, что единую классификацию всех существующих на сегодня ЭС провести достаточно сложно, так как, с одной стороны, можно выделить большое количество специфических характеристик ЭС, а с другой стороны – у разных авторов существуют значительные различия в терминологии обозначения одних и тех же вещей.

Предложим классификацию ЭС на основе следующих базовых параметров:

* уровень используемого языка;
* машина вывода (решатель);
* методы описания ПО;
* способ представления знаний;
* парадигма программирования и др.

Примеры классификации ЭС по указанным параметрам:

1. Классификация ЭС по парадигмам программирования (механизм реализации исполняемых утверждений):

* процедурное программирование;
* объектно-ориентированное программирование;
* программирование, ориентированное на данные;
* программирование, ориентированное на правила.

2. Классификация ЭС по способу представления знаний (характеризующемуся моделью представления знаний):

* в виде правил (продукций);
* в виде фреймов или объектов;
* в виде семантических сетей;
* логические модели представления знаний (исчисление предикатов).

3. Классификация ЭС по реализации различных способов рассуждений, принятыми в конкретных предметных областях.

* дедуктивный способ рассуждений;
* индуктивный способ рассуждений;
* способ рассуждений по аналогам или на основе прецедентов;
* способ рассуждений посредством выдвижения гипотез.

4. Классификация ЭС по уровню используемого языка

* традиционные (в том числе объектно-ориентированные) языки программирования (С, С++ и др.);
* символьные языки программирования (LISP, Prolog и их разновидности);
* инструментарий, содержащий часть компонент ЭС (OPS-5, ИЛИС и др.);
* оболочки и среды разработки общего назначения, содержащие все компоненты ЭС (EMMYCIN, Leonardo, ЭКО, GURU, Nexpert Object, ProKappa, и др)
* проблемно-специализированные средства (ориентированные на некоторый класс решаемых задач);
* предметно-ориентированные средства (включающие знания о некоторых типах предметных областей).

5. Классификация ЭС по методам описания проблемных областей

* *Подход на базе поверхностных знаний* заключается в извлечении из эксперта фрагментов эвристических знаний о данной ПО, которые релевантные решаемой задаче, причем не предпринимается никаких попыток глубинного изучения области, что предопределяет использование поиска в пространстве состояний в качестве универсального механизма вывода. Как правило, этот подход применяется к задачам, которые не могут быть точно описаны, и в качестве способа представления знаний выбираются правила. Если же задача может быть заранее структурирована или при ее решении можно воспользоваться некоторой моделью, то такой подход неэффективен.
* *Структурированный подход* используется в качестве развития поверхностного подхода в том случае, если применение поверхностного не обеспечивает решения задачи. Используя декомпозицию задачи на подзадачи (дерево подзадач), можно затем решать каждую задачу на основе поверхностного или глубинного подхода, а возможно, и их комбинации.
* *Глубинный подход*. При использовании глубинного подхода к решению задачи качество и компетентность ЭС будут зависеть от модели ПО, причем эта модель может быть определена различными способами (декларативно, процедурно). При глубинном подходе используются ЭС с мощными моделирующими возможностями, а именно: объекты (фреймы) с присоединенными процедурами, иерархическое наследование свойств, активные объекты, механизмы передачи сообщений объектам и др. Если сравнить, описанные выше подходы, с типами ПО то можно более детально классифицировать и оценить конкретные ЭС по данному параметру.

Подавляющее большинство совместимых статических ЭС ориентированы на реализацию дедуктивного способа рассуждений, причем акценты делаются на такие параметры логического вывода, как:

* структура процесса получения решения;
* методы поиска решения;
* стратегии разрешения конфликтов;
* управление достоверностью и др.

Конкретные значения этих параметров могут выступать как некоторые критерии оценки машины вывода.

**11. Особенности проектирования, разработки интеллектуальных систем**

Технология разработки ЭС. Разработка (проектирование) ЭС существенно отличается от разработки обычного программного продукта. Опыт разработки ранних ЭС показал, что использование при разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату. Дело в том, что неформализованность задач, решаемых ЭС, отсутствие завершенной теории ЭС и методологии их разработки приводит к необходимости модифицировать их принципы и способы построения ЭС в ходе процесса разработки по мере того, как увеличивается знание разработчиков о проблемной области.

Перед тем, как приступить к разработке ЭС, инженер по знаниям должен рассматривать вопрос, следует ли разрабатывать ЭС для данного приложения. В обобщенном виде ответ может быть таким: использовать ЭС следует тогда, когда разработка ЭС: 1) *возможна*, 2) *оправдана* и 3) методы инженерии знаний *соответствуют* решаемой задаче.

1. Чтобы разработка ЭС была *возможна* (для данного приложения), необходимо одновременно выполнение по крайней мере следующих требований:

* существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;
* эксперты должны сходиться в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;
* эксперты должны уметь выразить на естественном языке (вербализовать) и объяснить используемые ими методы, в противном случае трудно рассчитывать на то, что знания экспертов будут "извлечены" и вложены в ЭС;
* задача, возложенная на ЭС, требует только рассуждений, а не действий (если требуется действие, то необходимо объединять ЭС с роботами);
* задача не должна быть слишком трудной, ее решение должно занимать у эксперта несколько часов, а не дней или недель;
* задача, хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно "понятной" и структурированной области, т.е. должны быть выделены основные понятия, отношения и известные (хотя бы эксперту) способы получения решения задач;
* решение задачи не должно в значительной степени использовать "здравый смысл'" (т.е. широкий спектр общих сведений о мире и о способе его функционирования, которые знает и умеет использовать любой нормальный человек), т.к. подобные знания пока не удается (в достаточном количестве) вложить в системы искусственного интеллекта.

Использование ЭС в данном приложении может быть возможно, но не оправдано.

2. Применение ЭС может быть *оправдано* одним из следующих факторов:

* решение задачи принесет значительных эффект, например, использование ЭС для поиска полезных ископаемых;
* использование человека-эксперта невозможно либо из-за недостаточного количества экспертов, либо из-за необходимости выполнять экспертизу одновременно в различных местах;
* при передаче информации эксперту происходит недопустимая потеря времени или информации;
* при необходимости решать задачу в окружении враждебном для человека.

3. Приложение *соответствует* методам ЭС, если решаемая задача обладает совокупностью следующих характеристик:

* может быть естественным образом решена посредством манипуляции символами (т.е. с помощью символьных рассуждений), а не с числами (как принято в математических методах и традиционных программах);
* должна иметь эвристическую (не алгоритмическую) природу, т.е. ее решение должно сводиться к применению эвристических правил; задачи, которые могут быть гарантировано решены (с соблюдением заданных ограничений) с помощью некоторых формальных процедур;
* должна быть достаточно сложной чтобы оправдать затраты на разработку ЭС, однако, не должна быть чрезмерно сложной (решение занимает у эксперта часы, а не недели), чтобы ЭС могла ее решить;
* должна быть достаточно узкой, чтобы решаться методами инженерии знаний, и практически значимой.

При разработке ЭС часто используется концепция "быстрого прототипа". Суть ее состоит в том, что разработчики не пытаются сразу создать конечный продукт. На начальном этапе они создают прототип (прототипные) ЭС, которые должны удовлетворять двум противоречивым требованиям: с одной стороны, решать типовые задачи конкретного приложения, а с другой - время и трудоемкость его разработки должны быть весьма незначительны, чтобы можно было максимально запараллелить процесс накопления и отладки знаний (осуществляемый экспертом) с процессом выбора (разработки) программных средств (осуществляемым инженером по знаниям и программистом). Прототип должен продемонстрировать пригодность методов инженерии знаний для данного приложения. В случае успеха эксперт с помощью инженера по знаниям расширяет знания прототипа о проблемной области. При неудаче может потребоваться разработка нового прототипа или разработчики могут прийти к выводу о непригодности методов инженерии знаний для данного приложения.

В ходе работ по созданию ЭС сложилась определенная технология их разработки, включающая шесть следующих этапов*: идентификация, концептуализация, формализация, выполнение, тестирование, опытная эксплуатация* (рис.2.10).

Рассмотрим этапы разработки ЭС:



Рис. 2.10. Технология разработки ЭС

**12. Методы извлечения знаний когнитологом**

Рассмотрим две разновидности коммуникативных методов: пассивные и активные.

**Пассивные методы.** Термин «пассивные» не должен вызывать иллюзий, поскольку он введен как противовес к «активным» методам. В реальности же пассивные методы требуют от инженера по знаниям не меньшей отдачи, чем такие активные методы, как игры и диалог

*Пассивные методы* извлечения знаний включают такие методы, где ведущая роль в процедуре извлечения фактически передается эксперту, а инженер по знаниям только фиксирует рассуждений эксперта во время работы по принятию решений.

Согласно классификации к этой группе относятся:

* наблюдения;
* анализ протоколов «мыслей вслух»;
* лекции.

**Наблюдения.** В процессе наблюдений инженер по знаниям находится непосредственно рядом с экспертом во время его профессиональной деятельности или имитации этой деятельности. При подготовке к сеансу извлечения эксперту необходимо объяснить цель наблюдений и попросить максимально комментировать свои действия. Во время сеанса аналитик записывает все действия эксперта, его реплики и объяснения. Может быть сделана и видеозапись в реальном масштабе времени. Непременное условие этого метода - невмешательство аналитика в работу эксперта хотя бы на первых порах. Именно метод наблюдений является единственно «чистым» методом, исключающим вмешательство инженера по знаниям и навязывание им каких-то своих структур представлений.

Существуют две основные разновидности проведения наблюдений:

* + - наблюдение за реальным процессом;
    - наблюдение за имитацией процесса.

Обычно используются обе разновидности. Сначала инженеру по знаниям полезно наблюдать за реальным процессом, чтобы глубже понять предметную область и отметить все внешние особенности процесса принятия решения. Это необходимо для проектирования эффективного интерфейса пользователя. Ведь будущая ЭС должна работать именно в контексте такого реального производственного процесса. Кроме того, только наблюдение позволит аналитику увидеть предметную область, а, как известно, «лучше один раз увидеть, чем сто раз услышать».

Наблюдение за имитацией процесса проводят обычно также за рабочим местом эксперта, но сам процесс деятельности запускается специально для аналитика. Преимущество этой разновидности в том, что эксперт менее напряжен, чем в первом варианте, когда он работает на «два фронта» — и ведет профессиональную деятельность, и демонстрирует ее. Недостаток совпадает с преимуществом - именно меньшая напряженность эксперта может повлиять на результат - раз работа ненастоящая, то и решение может отличаться от настоящего.

Наблюдения за имитацией проводят также и в тех случаях, когда наблюдения за реальным процессом по каким-либо причинам невозможны (например, профессиональная этика врача-психиатра может не допускать присутствия постороннего на приеме). Сеансы наблюдений могут потребовать от инженера по знаниям:

* + - овладения техникой стенографии для фиксации действий эксперта в реальном масштабе времени;
    - ознакомления с методиками хронометража для четкого структурирования производственного процесса во времени;
    - развития навыков «чтения по глазам», то есть наблюдательности к жестам, мимике и другим невербальным компонентам общения;
    - серьезного предварительного знакомства с предметной областью, так как из-за отсутствия «обратной связи» иногда многое непонятно в действиях экспертов.

Протоколы наблюдений после сеансов в ходе домашней работы тщательно расшифровываются, а затем обсуждаются с экспертом.

Таким образом, наблюдения - один из наиболее распространенных методов извлечения знаний на начальных этапах разработки. Обычно он применяется не самостоятельно, а в совокупности с другими методами.

**Анализ протоколов «мыслей вслух».** Протоколирование «мыслей вслух» отличается от наблюдений тем, что эксперта просят не просто прокомментировать свои действия и решения, но и объяснить, как это решение было найдено, то есть продемонстрировать всю цепочку своих рассуждений. Во время рассуждений эксперта все его слова, весь «поток сознания» протоколируется инженером по знаниям, при этом полезно отметить даже паузы и междометия. Иногда этот метод называют «вербальные отчеты».

Вопрос об использовании для этой цели магнитофонов и диктофонов является дискуссионным, поскольку магнитофон иногда парализующе действует на эксперта, разрушая атмосферу доверительности, которая может и должна возникать при непосредственном общении.

Основной трудностью при протоколировании «мыслей вслух» является принципиальная сложность для любого человека объяснить, как он думает. При этом существуют экспериментальные психологические доказательства того факта, что люди не всегда в состоянии достоверно описывать мыслительные процессы. Кроме того, часть знаний, хранящихся в невербальной форме (например, различные процедурные знания типа «как завязывать шнурки»), вообще слабо коррелируют с их словесным описанием. Автор теории фреймов М. Минский считает, что «только как исключение, а не как правило, человек может объяснить то, что он знает». Однако существуют люди, склонные к рефлексии, для которых эта работа является вполне доступной. Следовательно рефлексивность, является для эксперта более чем желательной.

Расшифровка полученных протоколов производится инженером по знаниям самостоятельно с коррекциями на следующих сеансах извлечения знаний. Удачно проведенное протоколирование «мыслей вслух» является одним из эффективных методов извлечения, поскольку в нем эксперт может проявить себя максимально ярко, он ничем не скован, никто ему не мешает, он парит в потоке собственных умозаключений и рассуждений. Он может блеснуть эрудицией, продемонстрировать глубину своих познаний, для большого числа экспертов это самый приятный и лестный способ извлечения знаний.

От инженера по знаниям метод «мысли вслух» требует тех же умений, что и метод наблюдений. Обычно «мысли вслух» дополняются потом одним из методов для реализации обратной связи между интерпретацией инженера по знаниям и представлениями эксперта.

**Лекции.** Лекция является самым старым способом передачи знаний. Нас сейчас интересует не способность эту лекцию слушать, конспектировать и усваивать, лось, что чаще всего экспертов не выбирают, и поэтому учить экс лекции инженер по знаниям не сможет. Но если эксперт имеет опыт преподавателя (например, профессор клиники иди опытный руководитель) то можно воспользоваться таким концентрированным фрагментом знаний, как лекция. В лекции эксперту также предоставлено много степеней свободы самовыражения; при этом необходимо сформулировать эксперту тему и задачу лекции. Например, тема цикла лекций «Постановка диагноза — воспаления легких», тема конкретной лекции «Рассуждения по анализу рентгенограмм», задача — научить слушателей по перечисленным экспертом признакам ставить диагноз воспаления легких и делать прогноз. При такой постановке от может заранее структурировать свои знания и ход рассуждений.

Опытный лектор знает, что все вопросы можно условно разбить на

* + - умные вопросы, углубляющие лекцию;
    - глупые вопросы или вопросы не по существу;
    - вопросы «на засыпку» или провокационные.

Если инженер по знаниям задает вопросы второго типа, то возможны две реакции. Вежливый эксперт будет разговаривать с таким аналитиком как с ребенком, который сейчас не понимает и все равно ничего уже не поймет. Заносчивый эксперт просто выйдет из контакта, не желая терять время. Если же инженер по знаниям захочет продемонстрировать свою эрудицию вопросами третьего типа, то ничего, кроме раздражения и отчуждения, он, по-видимому, в ответ не получит.

Продолжительность лекции рекомендуется стандартная — от 40 до 50 минут и через 5-10 минут — еще столько же. Курс обычно от двух до пяти лекций.

Метод извлечения знаний в форме лекций, как и вес пассивные методы, использут в начале разработки как эффективный способ быстрого погружения инженера по знаниям в предметную область.

В заключение несколько советов, как слушать лекции

1. К лекции подготовьтесь, то есть познакомьтесь с предметной областью.
2. Слушайте с максимальным вниманием, для этого: устраните мешающие факторы (скрип двери, шорохи и т. д.); удобно устройтесь; поменьше двигайтесь.
3. Учитесь отдыхать во время слушания (например, когда лектор приводит цифры, которые можно взять из справочника).
4. Слушайте одновременно и лектора, и самого себя (параллельно *с* мыслями лектора по ассоциации возникают собственные мысли).
5. Слушайте и одновременно записывайте, но записывайте текст сокращенно, используя условные значки (для этого вовсе не следует непременно быть стенографом, достаточно только установить для себя ряд условных значков и ими неизменно пользоваться).
6. Расшифруйте записи лекции в тот же день.
7. Не спорьте с лектором в процессе лекции.
8. Рационально используйте перерывы в лекции для подведения итогов прослушанного.

**Активные методы.** Активные индивидуальные методы извлечения знаний на сегодняшний день наиболее распространенные. В той или иной степени к ним при работке практически любой ЭС. К основным активным методам можно отнести:

* + - анкетирование;
    - интервью;
    - свободный диалог;
    - игры с экспертом.

Во всех этих методах активную функцию выполняет инженер по знаниям, который пишет сценарий и режиссирует сеансы извлечения знании. Игры существенно отличаются от трех других методов. Их можно назвать вопросными методами поиска знаний.

**Анкетирование.** Анкетирование — наиболее жесткий метод, то есть наиболее стандартизированной. В этом случае инженер по знаниям заранее составляет вопросник или анкету, размножает ее и использует для опроса нескольких экспертов. Это основное преимущество анкетирования.

Сама процедура может проводиться двумя способами:

1. Аналитик вслух задает вопросы и сам заполняет анкету по ответам эксперта.
2. Эксперт самостоятельно заполняет анкету после предварительного инструктирования.

Выбор способа зависит от конкретных условий (например, от оформления анкеты, ее понятности, готовности эксперта). Второй способ нам кажется предпочтительнее, так как у эксперта появляется неограниченное время на обдумывание ответов.

Основными факторами, на которые можно существенно повлиять при анкетировании, являются средства общения (в данном случае это вопросник) и ситуация общения. Вопросник (анкета) заслуживает особого разговора. Существует несколько общих рекомендаций при составлении анкет. Эти рекомендации являются универсальными, то есть не зависят от предметной области.

Анкета не должна быть монотонной и однообразной, то есть вызывать скуку или усталость. Это достигается вариациями формы вопросов, сменой тематики, вставкой в о просо шуток и игровых вопросов. Анкета должна быть приспособлена к языку экспертов. Следует учитывать, что вопросы влияют друг на друга и поэтому последовательность вопросов должна быть строго продумана.

Желательно стремиться к оптимальной избыточности. Известно, что в анкете всегда много лишних вопросов, часть из них необходима — это так называемые контрольные вопросы, а другая часть должна быть минимизирована.

Лишние вопросы появляются, например, в таких ситуациях. Фрагмент анкеты:

*12. Считаете ли вы, что для лечения ангины эффективен эритромицин?*

*13. Какие дозы эритромицина вы обычна рекомендуете?*

При отрицательном ответе на 12-й вопрос 13-й является лишним. Его можно избежать усложнив вопрос:

*12. Применяете ли вы эритромицин для лечения ангины и если да, то в каких дозах?*

Анкета должна иметь «хорошие манеры», то есть ее язык должен быть ясным понятным, предельно вежливым. Методическим мастерством составления анкеты можно овладеть только на практике.

**Интервью.** Под *интервью* будем понимать специфическую форму общения инженера по знаниям и эксперта, в которой инженер по знаниям задает эксперту серию заранее подготовленных вопросов с целью извлечения знаний о предметной области Наибольший опыт в проведении интервью накоплен, наверное, в журналистике и социологии. Большинство специалистов этих областей отмечают тем не мене крайнюю недостаточность теоретических и методических исследований по тематике интервьюирования.

Интервью очень близко тому способу анкетирования, когда аналитик сам заполняет анкету, занося туда ответы эксперта. Основное отличие интервью в том, что оно позволяет аналитику опускать ряд вопросов в зависимости от ситуации вставлять новые вопросы в анкету, изменять темп, разнообразить ситуацию общения. Кроме этого, у аналитика появляется возможность «взять в плен» эксперта своим обаянием, заинтересовать его самой процедурой и тем самым увеличит эффективность сеанса извлечения.

Вопросы для интервью

Теперь несколько подробнее о центральном эвене активных индивидуальных методов — о вопросах. Инженеры по знаниям редко сомневаются в своей способности задавать вопросы. В то время как и в философии и в математике эта проблем обсуждается с давних лет. Существует даже специальная ветвь математической логики — *эротетическая* логика (логика вопросов).

Все вопросительные предложения можно разбить на два типа:

* + 1. Вопросы *с неопределенностью,* относящейся ко всему предложению («Действительно, введение больших доз антибиотиков может вызвать анафилактический шок?).
    2. Вопросы *с неполной информацией* («При каких условиях необходимо включать кнопку?»), часто начинающиеся со слов «кто», «что», «где», «когда» и т. д.

*Открытый* вопрос называет тему или предмет, оставляя полную свободу эксперту по форме и содержанию ответа («Не могли бы вы рассказать, как лучше сбить высокую температуру у больного с воспалением легких?»).

*В закрытом* вопросе эксперт выбирает ответ из набора предложенных («Укажите пожалуйста, что вы рекомендуете при ангине: а) антибиотики, б) полоскание, в)компрессы, г) ингаляции»). Закрытые вопросы легче обрабатывать при последующем анализе, но они более опасны, так как «закрывают ход рассуждений эксперта и «программируют» его ответ в определенном направлении. При составлении сценария интервью полезно чередовать открытые и закрытые вопросы, особенно тщательно продумывать закрытые, поскольку для их составления требуется определенная эрудиция в предметной области.

*Личный* вопрос касается непосредственно личного индивидуального опыта эксперта («Скажите, пожалуйста, Иван Данилович, в вашей практике вы применяете вулнузан при фурункулезах?»). Личные вопросы обычно активизируют мышление эксперта, «играют» на его самолюбии, они всегда украшают интервью.

*Безличный* вопрос направлен на выявление наиболее распространенных и общепринятых закономерностей предметной области («Что влияет на скорость процесса ферментации лизина?»).

При составлении вопросов следует учитывать, что языковые способности эксперта, как правило, ограничены и вследствие скованности, замкнутости, робости он не может сразу высказать свое мнение и предоставить знания, которые от него требуются (даже если предположить, что он их четко для себя формулирует). Поэтому часто при «зажатости» эксперта используют не *прямые* вопросы, которые непосредственно указывают на предмет или тему («Как вы относитесь к методике доктора Сухарева?»), а *косвенные,* которые лишь косвенно указывают на интересующий предмет («Применяете ли вы методику доктора Сухарева? Опишите, пожалуйста, результаты лечения»). Иногда приходится задавать несколько десятков косвенных вопросов вместо одного прямого.

*Вербальные* вопросы — это традиционные устные вопросы. *Вопросы с использованием наглядною материала* разнообразят интервью и снижают утомляемость эксперта. В этих вопросах используют фотографии, рисунки и карточки. Например, эксперту предлагаются цветные картонные карточки, на которых выписаны признаки заболевания. Затем аналитик просит разложить эти карточки в порядке убывания значимости для постановки диагноза.

Деление вопросов по функции на основные, зондирующие, контрольные связано с тем, что часто *основные* вопросы интервью, направленные на выявление знаний не срабатывают — эксперт но каким-то причинам уходит в сторону от вопросов отвечает нечетко.

Тогда аналитик использует *зондирующие* вопросы, которые направляют рассуждения эксперта в нужную сторону. Например, если не сработал основной вопрос «Какие параметры определяют момент окончания процесса ферментации лизина?» — аналитик начинает задавать зондирующие вопросы: «Всегда ли процесс ферментации длится 72 часа? А если он заканчивается раньше, как это узнать? Если он продлится больше, то что заставит микробиолога не закончить процесс на 72-м часу?» и т. д.

*Контрольные* вопросы применяют для проверки достоверности и объективности информации, полученной в интервью ранее («Скажите, пожалуйста, а московская школа психологов так же как Вы трактует шкалу К опросника ММРI?» или «Рекомендуете ли вы инъекции АТФ?» (АТФ — препарат, снятый с производства). Контрольные вопросы должны быть «хитро» составлены, чтобы не обидеть эксперта недоверием (для этого используют повторение вопросов в другой форме, уточнения, ссылки па другие источники). «Лучше два раза спросить, чем один раз напутать».

О нейтральных и наводящих вопросах. В принципе интервьюеру (в нашем случае инженеру по знаниям) рекомендуют быть беспристрастным, отсюда и вопросы его должны носить *нейтральный* характер, то есть не должны указывать на отношение интервьюера к данной теме. Напротив, *наводящие* вопрос заставляют респондента (в данном случае эксперта) прислушаться или даже принять во внимание позицию интервьюера. Нейтральный вопрос: «Совпадают ли симптомы кровоизлияния в мозг и сотрясения мозга?» Наводящий вопрос: «Не правда ли, очень трудно дифференцировать симптомы кровоизлияния в мозг?»

Кроме вопросов, перечисленных выше, полезно различать и включать в интервью следующие вопросы:

* + - контактные («ломающие лед» между аналитиком и экспертом);
    - буферные (для разграничения отдельных тем интервью);
    - оживляющие память экспертов (для реконструкции отдельных случаев из практики);
    - «провоцирующие» (для получения спонтанных, неподготовленных ответов).

В заключение описания интервью укажем три основные характеристики вопросов, которые влияют на качество интервью:

* + - язык вопроса (понятность, лаконичность, терминология);
    - порядок вопросов (логическая последовательность и немонотонность);
    - уместность вопросов (этика, вежливость).

Вопрос в интервью — это не просто средство общения, но и способ передачи мыслей и позиции аналитика.

«Вопрос представляет собой форму движения мысли, в нем ярко выражен момент перехода от незнания к знанию, от неполного, неточного знания к более полному и более точному». Отсюда необходимость в протоколах фиксировать не только ответы, но и вопросы, предварительно тщательно отрабатывая их форму и содержание.

Очевидно, что любой вопрос имеет смысл только в контексте. Поэтому вопросы может готовить инженер по знаниям, уже овладевший ключевым набором знаний.

Вопросы имеют для эксперта диагностическое значение — несколько откровенных «глупых» вопросов могут полностью разочаровать эксперта и отбить у него охоту к дальнейшему сотрудничеству.

**Свободный диалог.** *Свободный диалог* — это метод извлечения знаний в форме беседы инженера по знаниям и эксперта, в которой нет жесткого регламентированного плана и вопросника. Это определение не означает, что к свободному диалогу не надо готовиться. Напротив, внешне свободная и легкая форма этого метода требует высочайшей профессиональной и психологической подготовки. Подготовка к свободному диалогу практически может совпадать с предлагаемой в работе подготовкой к журналистскому интервью. Подготовка занимает разное время в зависимости от степени профессионализма аналитика, но в любом случае она необходима, так как несколько уменьшает вероятность самого нерационального метода— метода проб и ошибок.

Квалифицированная подготовка к диалогу помогает аналитику стать драматургом или сценаристом будущих сеансов, то есть запланировать течение процедуры извлечения: от приятного впечатления в начале беседы к профессиональному контакту через пробуждение интереса и доверия эксперта.

Так, в одном из исследований по технике ведения профессиональных журналистских диалогов было экспериментально доказано, что одобрительное и поощрительное «хмыканье» интервьюера увеличивает длину ответов респондента. При этом одобрение должно быть искренним.

**13. Структура ЭС**

Знания, которыми обладает специалист в какой-либо области (дисциплине), можно разделить на формализованные (точные) и неформализованные (неточные). *Формализованные знания* формируются в книгах и руководствах в виде общих и строгих суждений (законов, формул, моделей, алгоритмов и т.п.), отражающих универсальные знания. *Неформализованные знания,* как правило, не попадают в книги и руководства в связи с их конкретностью, субъективностью и приблизительностью. Знания этого рода являются результатом обобщения многолетнего опыта работы и интуиции специалистов. Они обычно представляют многообразие эмпирических (эвристических) приемов и правил.

В зависимости от того, какие знания преобладают в той или иной области (дисциплине), ее относят к формализованным (если преобладают точные знания) или к неформализованным (если преобладают неточные знания) описательным областям. Задачи, решаемые на основе точных знаний, называют формализованными, а задачи, решаемые с помощью неточных знаний - неформализованными. (Речь идет не о неформализуемых, а о неформализованных задачах, т.е. о задачах, которые, возможно, и формализуемы, но эта формализация пока неизвестна).

Традиционное программирование в качестве основы для разработки программы использует алгоритм, т.е. формализованное значение. Поэтому до недавнего времени считалось, что ЭВМ не приспособлены для решения неформализованных задач. Расширение сферы использования ЭВМ показало, что неформализованные задачи составляют очень важный класс задач, вероятно, значительно больший, чем класс формализованных задач. Неумение решать неформализованные задачи сдерживает внедрение ЭВМ в описательные науки. По мнению авторитетов, основной задачей информатики является внедрение ее методов в *описательные* науки и дисциплины. На основании этого можно утверждать, что исследования в области ЭС занимают значительное место в информатике.

К неформализованным задачам относятся те, которые обладают одной или несколькими из следующих особенностей:

* Алгоритмическое решение задачи неизвестно (хотя, возможно, и существует) или не может быть использовано из-за ограниченности ресурсов ЭВМ (времени, памяти).
* Задача не может быть определена в числовой форме (требуется символьное представление).
* Цели задачи не могут быть выражены в терминах точно определенной целевой функции.

Как правило, неформализованные задачи обладают неполнотой, ошибочностью, неоднозначностью и/или противоречивостью знаний (как данных, так и используемых правил преобразования).

Экспертные системы не отвергают и не заменяют традиционного подхода к автоматизированным информационным системам, они отличаются от традиционных программных систем тем, что ориентированы на решение неформализованных задач и обладают следующими особенностями:

* Алгоритм решения не известен заранее, а строится самой ЭС с помощью символических рассуждений, базирующихся на эвристических приемах.
* Ясность полученных решений, т.е. система "осознает" в терминах пользователя, как она получила решение.
* Способность анализа и объяснения своих действий и знаний.
* Способность приобретения новых знаний от пользователя-эксперта, не знающего программирования, и изменения в соответствии с ними своего поведения (открытая система).
* Обеспечение «дружественного», как правило, естественно-языкового (ЕЯ) интерфейса с пользователем.

Обычно к ЭС относят *системы, основанные на знании,* т.е. системы, вычислительная возможность которых является в первую очередь следствием их наращиваемой базы знаний (БЗ) и только во вторую очередь определяется используемыми методами. *Методы инженерии знаний* (методы ЭС) в значительной степени инвариантны тому, в каких областях они могут применяться. В настоящее время ЭС используются при решении задач следующих типов: принятие решений в условиях неопределенности (неполноты информации), интерпретации символов и сигналов, предсказание, диагностика, конструирование, планирование, управление, контроль и др.

Типичная ЭС состоит из следующих основных компонентов (рис. 2.1): решателя (интерпретатора), рабочей памяти (РП), называемой также базой данных (БД), базы знаний (БЗ), компонентов приобретения знаний, объяснительного и диалогового.



Рисунок 2.1 - Схема обобщенной экспертной системы.

*База данных* предназначена для хранения исходных и промежуточных данных решаемой в текущей момент задачи.

*База знаний* в ЭС предназначена для хранения долгосрочных данных (фактов), описывающих рассматриваемую область, и правил, описывающие преобразования (выводы) этой области.

*Решатель (интерпретатор)* используя исходные данные из БД и знания из БЗ, формирует такую последовательность правил, которые, будучи примененными к исходным данным, приводят к решению задачи.

*Компонента приобретения знаний* автоматизирует процесс наполнения ЭС знаниями, осуществляемый пользователем-экспертом.

*Объяснительная компонента* объясняет, как система получила решение задачи (или почему она не получила решения) и какие знания при этом она использовала, что облегчает эксперту тестирование системы и повышает доверие пользователя к полученному результату.

*Интерфейс* необходим дляформирования понятийного, человеко-ориентированного взаимодействия (диалога) со всеми категориями пользователей, как в ходе решения задач, так и приобретения знаний, объяснения результатов работы.

В разработке ЭС участвуют представители следующих специальностей:

* эксперт в той проблемной области, задачи которой будет решать ЭС;
* инженер по знаниям (когнитолог) - специалист по разработке ЭС;
* программист - специалист по разработке инструментальных средств.

**Эксперт**определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введения в ЭС знаний.

**Инженер по знаниям (когнитолог)**помогает эксперту выявить и структурировать знания, необходимые для работы ЭС. Он выбирает тот инструментарий, который наиболее подходит для данной проблемной области, и определяет способ представления знаний в нем, выделяет и программирует (традиционными средствами) стандартные функции (типичные для данной проблемной области), которые будут использоваться в правилах, вводимых экспертом.

Уже при разработке первых ЭС стало очевидно, что наиболее ответственным этапом является построение БЗ, для чего в общем случае и необходим посредник – так называемый *инженер по знаниям*, который должен обеспечить проведение до машинных этапов разработки СОЗ, заключающихся, как правило, в анализе предметной области, извлечении знаний из эксперта и их структурировании.

Эти процедуры оказались самыми тяжелыми поскольку, с одной стороны, чрезвычайно высок уровень требований, предъявляемых к личности инженера по знаниям (высококвалифицированный специалист в вычислительной науке, обладающий способностями к контакту с экспертами, умеющий побудить эксперта поставлять нужную информацию, умеющий отделять главное от второстепенного и т.д.), а с другой – стали наблюдаться трудности с поисками собственно экспертов (например, эксперт испытывает затруднения с четкой формулировкой своих знаний, не всегда расположен полностью делиться знаниями и др).

Поэтому почти одновременно с появлением индустрии знаний стали разрабатываться автономные системы, автоматизирующие процессы получения необходимой информации от экспертов. Подобные программные средства получили название «оболочек приобретения».

**Программист**разрабатывает инструментальное средство, содержащее в пределе все основные компоненты ЭС, осуществляет сопряжение с той средой, в которой оно может быть использовано.

**Системы поддержки знаний.**Если рассматривать ЭС с информационной точки зрения, то ее структуру можно представить состоящую из следующих компонент:

**База знаний.** БЗ фактов и правил вывода, является центральной частью ЭС. В отличие от обычной информационной БД в ней хранятся не только факты, но и правила, позволяющие вывести новые факты.

**Операторы.** Три класса операторов ЭС:

* *пользователи* обращаются к системе за советом по специальным проблемам в узкой области, представляя ей специфические факты и свои гипотезы о следствиях или целях;
* *эксперты* обращаются к системе, чтобы передать ей свои знания по частной проблеме, а также общепринятые факты и процедуры вывода;
* *инженеры* действуют как промежуточные звенья между экспертом и системой, помогая первому зондировать свои знания и проверяя работу законченной ЭС.

**Оболочка ЭС -** представляет собой программную систему, используемую для приложения БЗ с целью выработки решения задачи:

* прикладная система проверяет гипотезы пользователя или ищет пути достижения поставленных целей путем вывода последовательности фактов о конкретной ситуации со слов пользователя, а также используя общие факты и правила вывода, представленные экспертом и введенные в ЭС инженером по знаниям;
* система обоснования отвечает на вопросы о том, каким способом были выведены факты, на основе какой информации и каких правил вывода;
* система приобретения служит для опроса эксперта с целью получения его словаря, общих фактов и правил вывода;
* система отображения обеспечивает необходимыми средствами для представления БЗ в понятной форме, а также связей между фактами и правилами вывода, содержащимися в БЗ;
* система редактирования содержит инструментарий для компоновки и редактирования БЗ, сохраняя при этом ее целостность при помощи словаря, переменных и применяемых операций;
* система проверки содержит инструментарий для испытаний и предохранения БЗ от специфических историй с известными последствиями.

Информационные оболочки большинства современных ЭС:

* предлагают определенную БЗ, представленную посредством правил, продукций и фреймов;
* имеют некоторые прикладные возможности, ограниченные в выводе следствий и испытании утверждений;
* дают простые объяснения посредством используемых фактов и правил;
* не имеют способностей интегрированного сбора знаний;
* имеют ограниченные возможности по отображению БЗ, в частности, в графической форме;
* имеют такой интерфейс пользователя с редактором БЗ, который требует понимания внутренних структур данных;
* представляют ограниченные контролирующие возможности и не имеют объединенной БД, фактических историй.

Однако в последние годы происходят быстрые положительные изменения в сторону улучшения методов представления знания, получения выводов, их обоснования, отображения и обучения ЭС и все это благодаря интенсивному развитию инженерии знаний.

Проблемы *инженерии знаний* можно сформулировать следующим образом. Приобретение знания является узким местом в построении ЭС. Инженер по знаниям имеет меньшее знание области, чем эксперт, возникающие проблемы связи препятствуют процессу превращения экспертизы в программу. Словарь, используемый экспертом при разговоре с новичком о предметной области, часто не годится для решения проблемы; таким образом, инженер по знаниям и эксперт должны работать вместе, чтобы обновить и расширить его. Один из наиболее трудных аспектов деятельности инженера по знаниям заключается в оказании помощи эксперту в разработке структуры знания предметной области, в определении и формализации основных концепций этой области.

Это узкое место является главным препятствием на пути применения ЭС и реализации их полного рабочего потенциала. Автоматизация приобретения и передачи знания стала главной целью многих исследований. Уменьшение стоимости аппаратных и программных средств поддержки ЭС вывело рассматриваемую технологию инженерии знаний на массовый спрос гораздо раньше, чем ожидалось.

В то же время движение в направлении интенсификации человеческого труда, которое наблюдается в инженерии знаний, противоречит основным тенденциям современной промышленности.

Все это усилило интерес к разработкам систем поддержки знания, способным автоматизировать инженерию знания как процесс прямого воздействия между экспертами и ЭВМ.

Очевидным подходом является анализ беседы, который и был использован во многих исследованиях. Анализ текстов из руководства оператора представляет собой другой важный источник информации и его поведении.

Очевидно, что эти методы являются только частью всей иерархии методологии передачи знания, определяя лишь те области приложения инженерии знания, которые апробированы.

**База знаний.** В центре, как и прежде, находятся факты и правила вывода.

**Оболочка ЭС -** операционная система для приложения БЗ к поддержке знания:

* система вывода определяет следствия фактов по конкретной ситуации;
* планирующая система выбирает оптимальный способ использования системы выводов для достижения поставленных целей;
* поясняющая система дает сведение о том, на какой основе сделаны выводы.

**Система сбора данных.** Процессы передачи знания, которые могут применяться при формировании базы знаний:

* система генерирования знания реализует сбор знания способом грубой индукции моделей, без культурной поддержки;
* система моделирования экспертизы осуществляет приобретение знания методом имитирования поведения эксперта;
* система пополнения характеристик принимает знание по линиям обратной связи;
* система извлечения знания накапливает знание в процессе диалога с экспертом;
* система структурирования знания получает новые знания, основываясь на аналогиях;
* система основных законов осуществляет выбор знаний путем построения моделей, как это делается в моделирующих языках;
* система систематических принципов берет знания путем их вывода из абстрактных принципов.

Раньше методы получения, интерпретации, представления знания трактовались как совершенно раздельные области исследования.

**Структура систем поддержки знания (СПЗ)**. Перечислим вначале общие требования для СПЗ, назначение которых заключается в сборе знаний для ЭС:

* СПЗ должны быть независимыми от предметной области;
* эксперты должны пользоваться СПЗ непосредственно, без промежуточного звена;
* СПЗ должны обладать способностью воспринимать знание из разнообразных источников, включая тексты, беседы с экспертами и наблюдения за поведением экспертов:
* СПЗ должна охватывать многообразие перспектив, включая частичные или противоположные входные данные от различных экспертов,
* система должна уметь охватывать разнообразные формы знания и взаимоотношений между знаниями;
* система должна обладать способностью представлять знание из разнородных источников с достаточной ясностью в отношении его получения, следствий и структурных связей;
* пользователям СПЗ должна быть представлена возможность приложить знание во многих предметных областях, а также возможность свободно экспериментировать с его применением;
* система должна иметь средства для изучения их обоснованности;
* работа системы в максимальной степени должна основываться на хорошо разработанных и понятных теориях сбора знаний и их представления.

1. **Интеллектуальный анализ данных, общие сведения и терминология**
2. **Машинное обучение, общие сведения и терминология**

**Интеллектуальный анализ данных**

Колоссальные информационные ресурсы породили проблему их эффективного использования – возникла потребность в развитии прогнозных и экспертно-ориентированных систем с элементами искусственного интеллекта, в основу которых легли бы современные методы «превращения» накопленных разнородных данных в полезные знания.

Начали развиваться технологии, направленные на качественный анализ информационных ресурсов. Такие технологии выводят на иной – революционный уровень применение вычислительной техники, переводя ее использование с рельсов математического прагматизма в сферу технических и гуманитарных исследований и превращая информационную парадигму из простого информационного ресурса «хранилища данных» в интеллектуального помощника анализа данных на базе «компетенции знаний». Однако стремление усовершенствовать процессы принятия решений нередко наталкивается на большие объемы и сложную структуру накапливаемых данных. Указанные обстоятельства стимулировали развитие технологий «обнаружения знаний», технологий интеллектуального анализа данных, предназначенных для автоматического поиска в разнотипных и разнородных данных скрытых закономерностей, раскрывающих взаимосвязи в той или иной предметной области.

**14. Общие сведения, терминология, сфера применения.**

На протяжении десятков лет активно развивается область компьютерных информационных технологий «обнаружение знаний в базах данных» (knowledge discovery in databases). Рядом с этим названием нередко также звучат термины «раскопка данных» (data mining), машинного обучения (machine learning) и «интеллектуальный анализ данных». Все эти термины можно считать синонимами. Их возникновение связано с новым витком в развитии средств и методов обработки различной информации, и такими фундаментальными проблемами искусственного интеллекта как распознавание и прогнозирование. Этот виток обязан пришедшему пониманию, что в накопленной информации содержатся скрытые знания, которые можно извлечь и воспользоваться в практических целях.

На сегодняшний день известны десятки методологий и сотни алгоритмов интеллектуального анализа данных (статистические, регрессионные, эвристические и т.д.), развита индустрия программного обеспечения, аналитические пакеты успешно используются в различных областях науки, бизнеса, промышленности, инноваций, защиты и безопасности. Большинство современных программных приложений, не зависимо от функционального назначения, снабжено встроенными модулями анализа, основанными на алгоритмах машинного обучения - инструментарии классификации, распознавания, прогнозирования и др.

Направление ИАД родилось как ответ на сложившуюся проблемную ситуацию. Исходное определение дал наш бывший соотечественник Григорий Пятецкий-Шапиро:

«Datamining– это процесс обнаружения в сырых данных ранее неизвестных нетривиальных практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности.» (G. Piatetsky‑Shapiro).

В настоящее время ИАД существует в двух ипостасях. Ряд специалистов делает акцент на обработке сверхбольших объемов данных. Здесь предъявляются повышенные требования к быстродействию алгоритмов, естественно, в ущерб оптимальности результатов. Подавляющее большинство классических процедур имеют время выполнения, квадратичное или даже кубическое по объёму исходных данных. При количестве объектов, превосходящем несколько десятков тысяч, они работают неприемлемо медленно даже на самых современных компьютерах.

**Сфера применения ИАД.** Сфера применения интеллектуального анализа данных ничем не ограничена – она везде, где имеются какие-либо данные. Data Mining представляют большую ценность для руководителей и аналитиков в их повседневной деятельности. Технологии интеллектуального анализа данных выводят отрасли производства на уровень экстенсивного развития. Области применения многочисленны и разнообразны, в том числе:

* геология – поиск полезных ископаемых, сейсмо-прогнозирование и т.д.;
* сельское хозяйство - прогнозирование урожая, борьбы с вредителями; и т.д.;
* административное управление - составление расписаний, оптимизация информационных потоков, мониторинг и контроль показателей деятельности;
* медицина – постановка диагноза, контроль хода лечения и т.д.;
* молекулярная генетика и генная инженерия - определение так называемых маркеров, контролирующих те или иные фенотипические признаки живого организма и т.д.;
* банковское дело - прогнозные модели ценности клиентов и услуг, выявление мошенничества по транзакциям и т.д.;
* прикладная химия - задачи выяснения особенностей, свойств химического строения тех или иных соединений и новых материалов.
* машиностроение, метеорология, связь и телекоммуникации, торговля и делопроизводство, важнейшее значение в военно-промышленном комплексе и т.д. и т.п.

Решаются различные задачи управления, контроля, оценки, в том числе: защиты и безопасности, машинного зрения (распознавание отпечатков пальцев, радужной оболочки глаза, номерных знаков машин, лиц и т.д.), распознавания текстов и перевода, дефектоскопия (металлопрокат, деревообработка и пр.), диагностика сложных технических систем (автомобиле-, корабле-, авиа- строение ), прогнозирование экономических, биржевых показателей и множество других примеров прикладного применения.

**Краткий обзор и современное состояние инструментария ИАД.** Интенсивно развивается индустрия программного обеспечения - аналитические пакеты успешно используются в различных областях науки, бизнеса, промышленности. Большинство современных программных приложений, не зависимо от функционального назначения, снабжено встроенными модулями анализа, основанными на алгоритмах машинного обучения (machine learning) - инструментарии классификации, распознавания, прогнозирования и др. Классические пакеты аналитики (SPSS, STATGRAPHICS, See5, WizWhy, Hugin и т.д. и т.п.) достигли высокого уровня по факту реализации аналитических методов и алгоритмов и используются не только в коммерческой эксплуатации но и в образовании.

Много платформ SaaS Business Intelegence (BI) или AaaS – Analysys as a Service отправились в облака, включая Teradata, IBM, Cloud9 Analytics, Cloudscale, In2Clouds, Vertica, Lucidera. Перечень компаний, которые занимаются аналитикой в облаках.

Основная политика разработчиков направлена на коммерческое использование и продажу готового продукта, с закрытым кодом и функциональностью, заточенной под ограниченный класс потребителей. В большинстве случаев аналитические модули носят вспомогательный характер и предоставляются в виде готовых библиотек для использования их в виде инструментария настроек к платформам, разворачиваемым у клиентов.

DataMining является мультидисциплинарной областью, возникшей и развивающейся на базе достижений прикладной статистики, распознавания образов, методов искусственного интеллекта, теории баз данных и др. Отсюда обилие методов и алгоритмов, реализованных в различных действующих системах DataMining. Многие из таких систем интегрируют в себе сразу несколько подходов. Тем не менее, как правило, в каждой системе имеется какая-то ключевая компонента, на которую делается главная ставка.

**15. Методы и алгоритмы ИАД**

За последние десятилетия значительные усилия в области ИАД были направлены на создание специализированных алгоритмов, способных выполнять те же задачи за линейное или даже логарифмическое время без существенной потери точности.

Другая группа специалистов концентрирует внимание на глубине раскопки данных. В понимании второй группы основные отличия технологии ИАД следующие:

* ИАД– это всегда сугубо многомерные задачи – поиск связи между значением целевого показателя и набором значений группы других показателей базы данных.
* Технология ИАД способна обрабатывать разнородную информацию. То есть, поля могут быть представлены количественными, качественными и текстовыми переменными.
* Технология ИАД в отличие от традиционных статистических методов не претендует на поиск взаимосвязей, характерных для полного объема данных (всей выборки). Ищутся правила, связывающие значения показателей, для подвыборок данных. При этом правила всегда высокоточные, а не «размытые» по всей выборке, общие и неточные статистические тенденции.
* Алгоритмы ИАД производят поиск указанных выше подвыборок данных и точных взаимосвязей для этих подвыборок в автоматическом режиме.

Таким образом, ключевые слова ИАД– точность, многомерность, разнотипность данных, автоматический поиск. Важное требование интерпретируемости получаемого результата.

Методы ИАД имеют много общего с методами решения задач классификации, диагностики и распознавания образов. Но их одной из главных отличительных черт, как отмечалось выше, является функция интерпретации закономерностей, лежащая в основе правил вхождения объектов в классы эквивалентности. Поэтому здесь большое распространение получили логические методы. Есть еще одна важная причина, обусловившая приоритет логических методов. Она заключается в сложной системной организации областей, составляющих предмет приложения современных информационных технологий. Эти области относятся, как правило, к надкибернетическому уровню организации систем, закономерности которого не могут быть достаточно точно описаны на языке статистических или иных аналитических математических моделей. Гибкость и многообразие логических конструкций индуктивного вывода позволяют нередко добиваться успешных результатов при описании таких сложных систем.

Другие методы ИАД для построения диагностических и прогностических моделей имеют менее прозрачную интерпретацию. Сюда относятся байесовские классификаторы, дискриминантный анализ, нейросетевой подход, метод ближайших соседей, метод опорных векторов, генетические алгоритмы и др. Как показала практика последнего десятилетия, в ряде задач (особенно в бизнес-приложениях, где требуется анализировать огромные базы данных), требование интерпретируемости результатов стало отступать на задний план. Акцент здесь стал делаться на стабильности получаемых решений. Более того, на передний план начали выходить методы работы с комитетами, содержащими сотни и тысячи методов и алгоритмов. Как выяснилось, подобные комитеты, состоящие даже из «слабых» алгоритмов, способны превосходить по точности изолированные «сильные» алгоритмы, нацеленные на поиск глубоких закономерностей в массивах данных. Эта тенденция современного ИАД нуждается в самостоятельном рассмотрении. Здесь наблюдается явное отступление от изначальных идеалов ИАД, связанных с попытками извлечения знаний из данных, а не построением моделей в виде «черных ящиков».

**16, 17. Ансамблевые методы, бустинг, бэггинг**

<https://habr.com/ru/company/ods/blog/324402/>

Хорошим примером ансамблей считается теорема Кондорсе «о жюри присяжных» (1784). Если каждый член жюри присяжных имеет независимое мнение, и если вероятность правильного решения члена жюри больше 0.5, то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри и стремится к единице. Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.

*N* — количество присяжных

*p* — вероятность правильного решения присяжного

*μ* — вероятность правильного решения всего жюри

*m* — минимальное большинство членов жюри,

*CiN*— число [сочетаний](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%87%D0%B5%D1%82%D0%B0%D0%BD%D0%B8%D0%B5) из *N* по *i*



Давайте рассмотрим ещё один пример ансамблей — "Мудрость толпы". Фрэнсис Гальтон в 1906 году посетил рынок, где проводилась некая лотерея для крестьян.

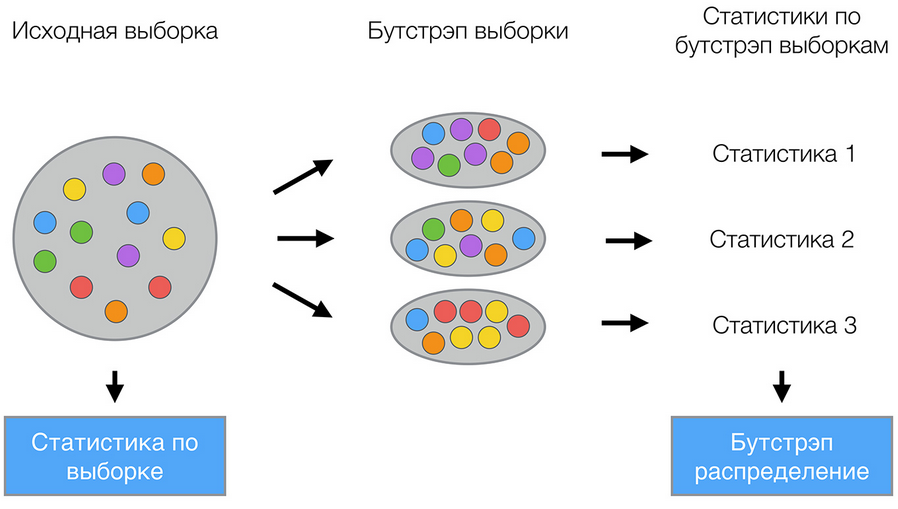
Их собралось около 800 человек, и они пытались угадать вес быка, который стоял перед ними. Бык весил 1198 фунтов. Ни один крестьянин не угадал точный вес быка, но если посчитать среднее от их предсказаний, то получим 1197 фунтов.

Эту идею уменьшения ошибки применили и в машинном обучении.

**Бутстрэп**

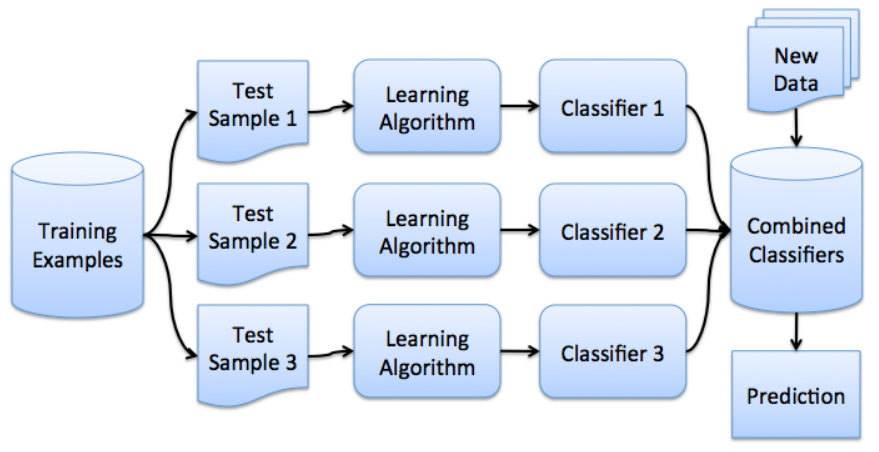
Bagging (от Bootstrap aggregation) — это один из первых и самых простых видов ансамблей. Он был придуман [Ле́о Бре́йманом](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B5%D0%B9%D0%BC%D0%B0%D0%BD,_%D0%9B%D0%B5%D0%BE) в 1994 году. Бэггинг основан на статистическом методе бутстрэпа, который позволяет оценивать многие статистики сложных распределений.

Метод бутстрэпа заключается в следующем. Пусть имеется выборка ***X*** размера ***N***. Равномерно возьмем из выборки ***N*** объектов с возвращением. Это означает, что мы будем ***N*** раз выбирать произвольный объект выборки (считаем, что каждый объект «достается» с одинаковой вероятностью **1/*N***), причем каждый раз мы выбираем из всех исходных ***N*** объектов. Можно представить себе мешок, из которого достают шарики: выбранный на каком-то шаге шарик возвращается обратно в мешок, и следующий выбор опять делается равновероятно из того же числа шариков. Отметим, что из-за возвращения среди них окажутся повторы. Обозначим новую выборку через ***X1***. Повторяя процедуру ***M*** раз, сгенерируем ***M*** подвыборок ***X1,…,XM***. Теперь мы имеем достаточно большое число выборок и можем оценивать различные статистики исходного распределения.



**Бэггинг**

Теперь вы имеете представление о бустрэпе, и мы можем перейти непосредственно к бэггингу. Пусть имеется обучающая выборка ***X***. С помощью бутстрэпа сгенерируем из неё выборки ***X1,…,XM***. Теперь на каждой выборке обучим свой классификатор ***ai(x)***. Итоговый классификатор будет усреднять ответы всех этих алгоритмов (в случае классификации это соответствует голосованию): . Эту схему можно представить картинкой ниже.



Бэггинг эффективен на малых выборках, когда исключение даже малой части обучающих объектов приводит к построению существенно различных базовых классификаторов. В случае больших выборок обычно генерируют подвыборки существенно меньшей длины.

Бэггинг позволяет снизить дисперсию обучаемого классификатора, уменьшая величину, на сколько ошибка будет отличаться, если обучать модель на разных наборах данных, или другими словами, предотвращает переобучение. Эффективность бэггинга достигается благодаря тому, что базовые алгоритмы, обученные по различным подвыборкам, получаются достаточно различными, и их ошибки взаимно компенсируются при голосовании, а также за счёт того, что объекты-выбросы могут не попадать в некоторые обучающие подвыборки.

**Бустинг — ещё один способ машинного обучения**

**(** [**https://thecode.media/boosting/**](https://thecode.media/boosting/) **)**

Как с помощью слабых алгоритмов сделать сильный.

Суть метода: строим серию не особо точных алгоритмов и обучаем их на ошибках друг друга.

**Дерево решений**

Чтобы понять бустинг, нужно сначала понять дерево решений. Вот это — очень простое дерево:

[](https://thecode.media/wp-content/uploads/2020/08/1.jpg)

Сейчас это дерево решений, но может быть деревом предсказаний. Представьте, что в заголовке дерева написано «Выйдет ли Юзернейм гулять?» — и вы получите машину предсказаний, которая на основе данных о Юзернейме и погоде будет строить точные предсказания о Юзернейме — при условии, что мы задаём правильные вопросы.

**Машина предсказания**

Теперь пример сложнее. Допустим, у нас есть данные по миллиону музыкальных клипов на Ютубе. По каждому есть 100 критериев, например:

* Длится ли клип дольше трёх минут.
* Есть ли там прямая бочка.
* Этот трек в жанре «хип-хоп» или нет.
* Выпустил ли клип популярный лейбл.
* Записан ли клип во дворе на мобильный телефон.
* …

Также у нас есть данные о том, набрал ли клип больше миллиона просмотров. Мы хотим научиться предсказывать этот критерий — назовём его популярностью. То есть мы хотим получить некий алгоритм, которому на вход подаёшь 100 критериев клипа в формате да/нет, а на выходе он тебе говорит: «Этому клипу суждено стать популярным».

Миллион клипов, сто критериев — обучающая выборка для алгоритма

**Первая проблема**

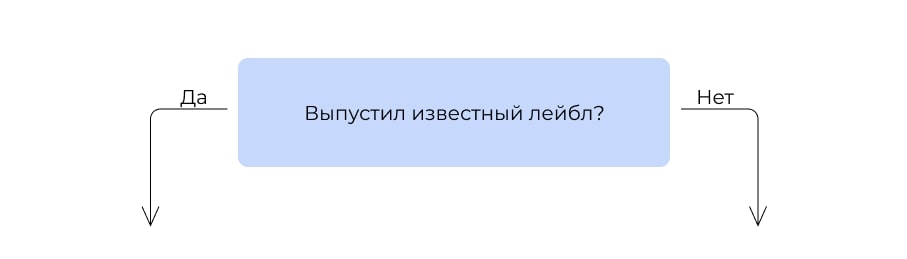
Если мы захотим построить дерево предсказаний для этой задачи, мы столкнёмся с проблемой: мы не знаем, какие критерии ставить в начало, а какие в конец, какие запихивать в какие ветки, а какие вообще не нужны и потому не влияют.

**Первый шаг решения**

Чтобы построить дерево для такой задачи, мы должны будем сначала посчитать, насколько каждый критерий связан с желаемым результатом:

Среди миллиона клипов в обучающей выборке 300 000 выпустили известные лейблы. 120 000 из них стали популярны. Этот критерий лучше других предсказывает популярность

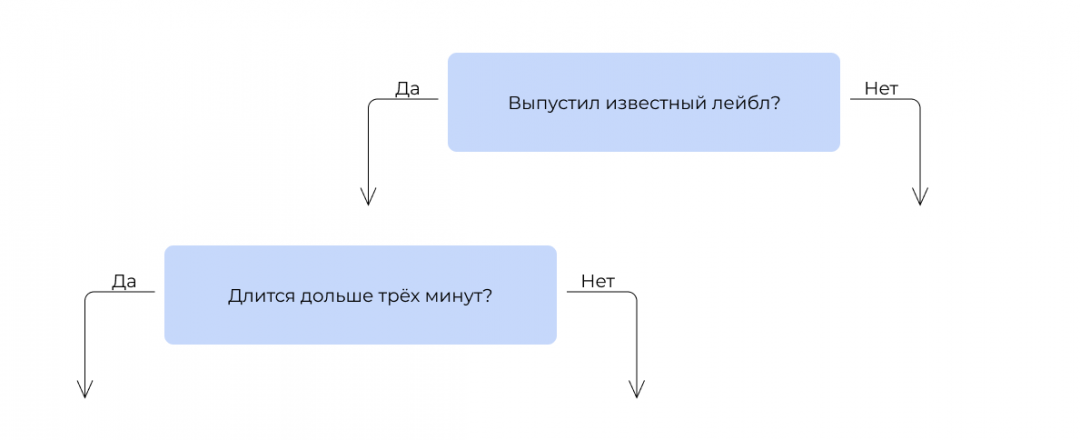
В итоге мы видим, что самая сильная связь с итоговой популярностью у критерия «Выпустил ли клип популярный лейбл». Получается, если клип выпустил лейбл, это влияет на успех сильнее, чем бочка или автотюн. Этот критерий встаёт на вершину дерева.

[](https://thecode.media/wp-content/uploads/2020/08/5.jpg)

Затем мы смотрим, какой критерий поставить следующим. Берём те 300 000 клипов, которые выпустили лейблы, и прогоняем их по остальным критериям. Ищем тот, который даёт самую высокую итоговую точность предсказания.

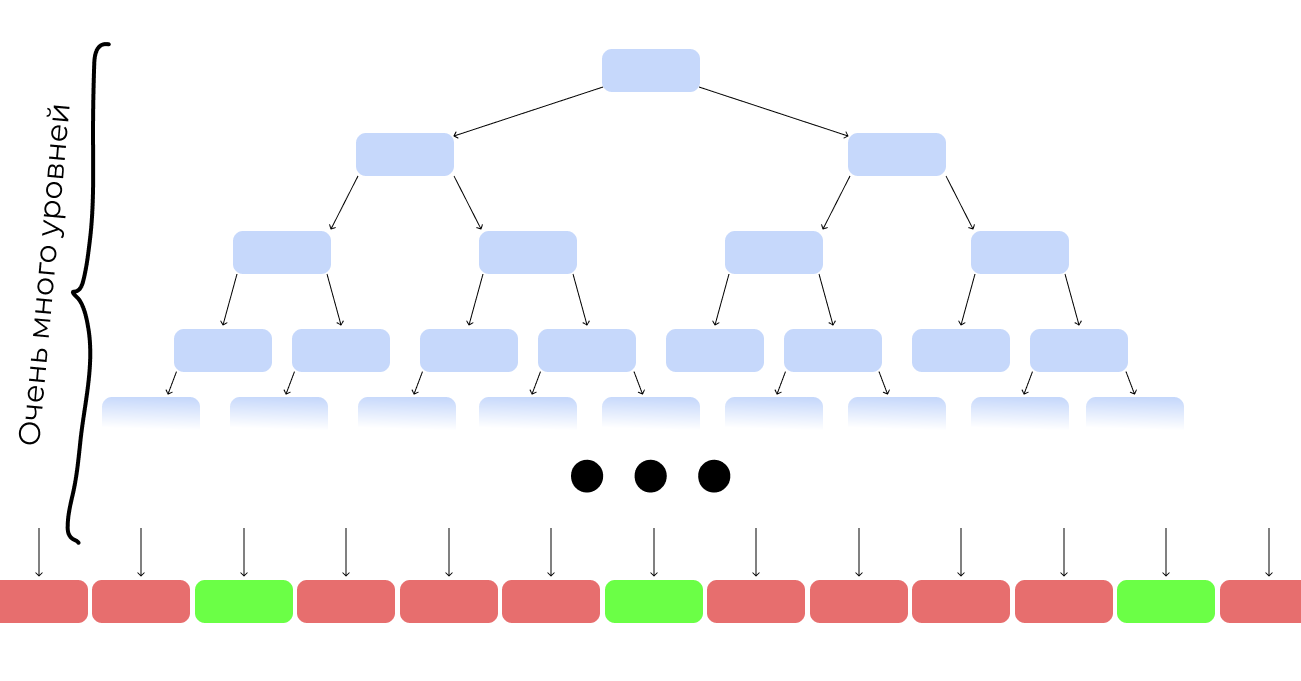
Среди 300 000 клипов, которые выпустили лейблы, 250 000 клипов идут дольше 3 минут и 90 000 из них набрали больше миллиона просмотров. Этот критерий лучше других предсказывает популярность клипов лейбла

Ставим его на второе место.

[](https://thecode.media/wp-content/uploads/2020/08/7.png)

То же самое делаем для другой ветки. И так строим последовательность из остальных критериев. На практике вручную этим не занимаются: есть специальные алгоритмы, которые делают это автоматически.

В итоге, у нас появилось дерево решений:

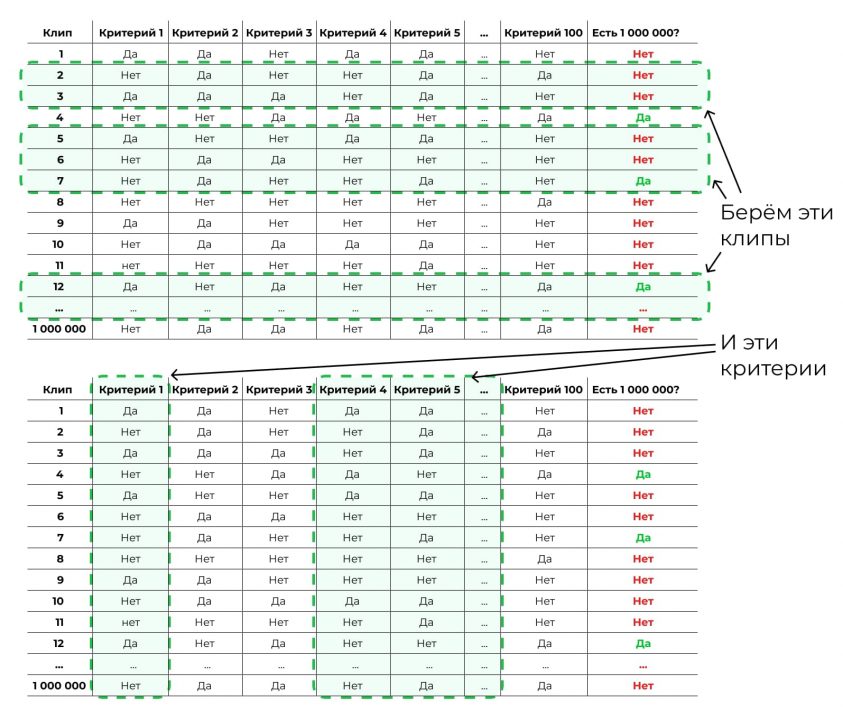


Большинство клипов не взлетит, но какие-то станут популярными

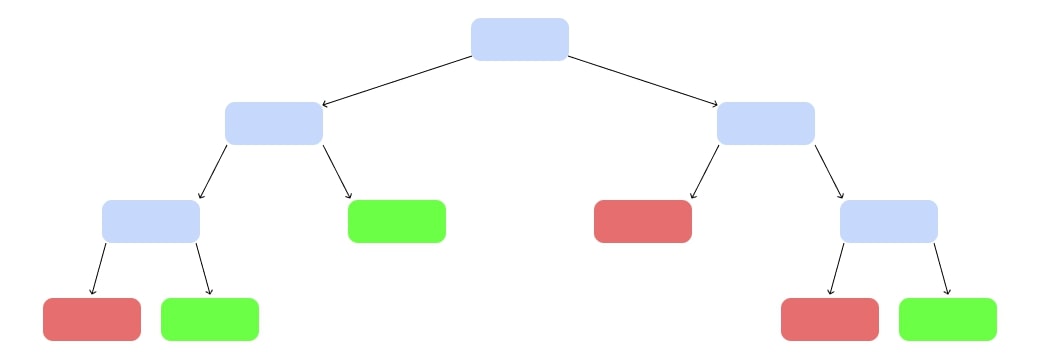
**Случайный лес**

Есть проблема: построенное таким образом дерево очень сложное и, вероятно, не очень точное. Попробуем сделать не одно огромное дерево, а несколько небольших.

Возьмём случайную выборку из наших исходных данных. Не миллион клипов, а 10 000. К ним — случайный набор критериев, не все 100, а 5:

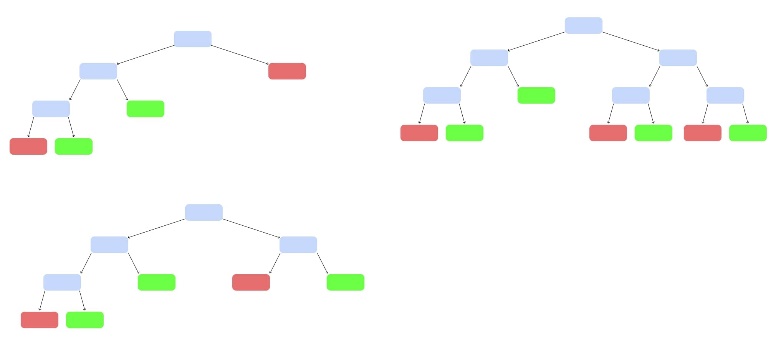
[](https://thecode.media/wp-content/uploads/2020/08/9.jpg)

И построим дерево попроще:



Было много уровней, стало меньше

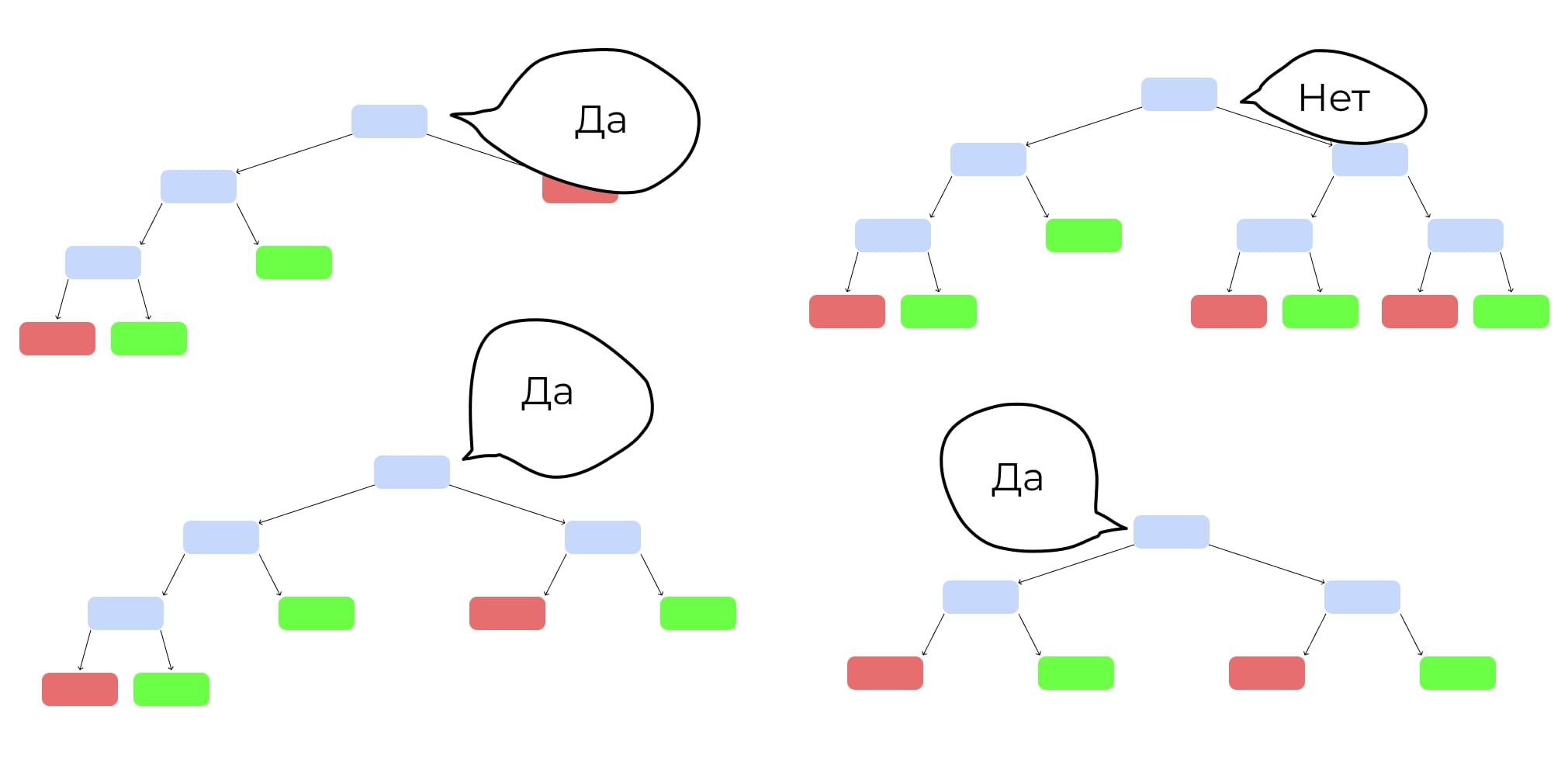
Так построим ещё несколько деревьев, каждое — на своём наборе данных и своём наборе критериев:



**Лес алгоритмов.**

У нас появился случайный лес. Случайный — потому что мы каждый раз брали рандомный набор данных и критериев. Лес — потому что много деревьев.

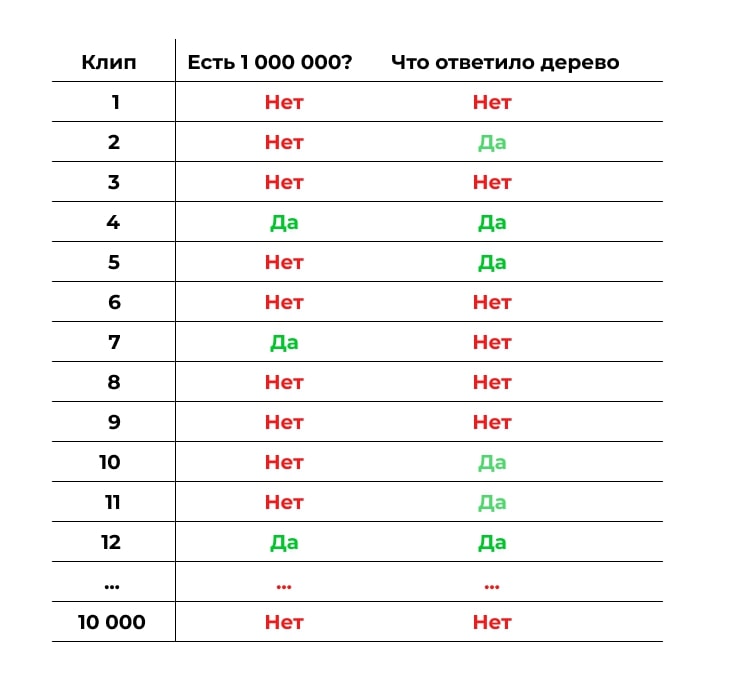
Теперь запустим клип, которого не было в обучающей выборке. Каждое дерево выдаст свой вердикт, станет ли он популярным — «да» или «нет». Как голосование на выборах. Выбираем вариант, который получит больше всего голосов.



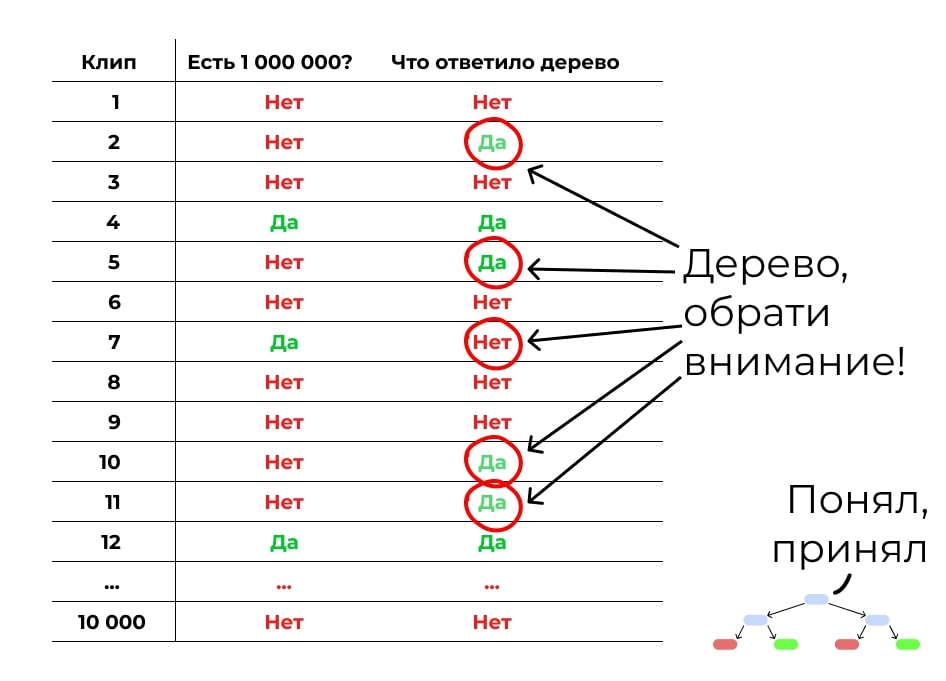
Три за, один против — клип ждёт успех. Наверное.

**Неслучайный лес — бустинг**

Теперь построим похожий лес, но набор данных будет неслучайным. Первое дерево мы построим так же, как и раньше, на случайных данных и случайных критериях. А потом прогоним через это дерево контрольную выборку: другие клипы, по которым у нас есть все данные, но которые не участвуют в обучении. Посмотрим, где дерево ошиблось:



Первое дерево может давать много ошибок. Теперь делаем следующее дерево. Обратим внимание на места, где первое дерево ошиблось. Дадим этим ошибкам больший вес при подборе данных и критериев для обучения. Задача — сделать дерево, которое исправит ошибки предыдущего.



Учим дерево исправлять ошибки предшественника.

Но второе дерево наделает своих ошибок. Делаем третье, которое их исправит. Потом четвёртое. Потом пятое. Вы поняли принцип.

Делаем такие деревья, пока не достигнем желаемой точности или пока точность не начнёт падать из-за переобучения. Получается, у нас много деревьев, каждое из которых не очень сильное. Но вместе они складываются в лес, который даёт хорошую точность. Бустинг!

**И где это используют?**

Бустинг часто используют в задачах, когда нейронные сети не очень подходят.

Нейросети хорошо справляются с однородными данными, например изображениями или голосом. А вот если данные разного характера и с разной структурой, то бустинг справляется лучше. Например, чтобы предсказать погоду, надо учесть температуру и влажность, данные с радаров, наблюдения пользователей и другие параметры. Бустинг с этим справляется отлично.

Вот что про бустинг рассказывает Роман Халкечев, руководитель отдела аналитики в Яндекс.Еде:

«В Яндексе повсеместно используем библиотеку CatBoost. Это внутренняя разработка, которую в 2017 году выложили в open source. Она помогает решать много задач, например, ранжирование в Поиске, предсказание в Погоде, рекомендации в Музыке.

**18. Сравнительная характеристика методов «деревья решений» и «искусственные нейронные сети»**

**Самостоятельно (см. искусственные нейронные сети и деревья решений)**

**19. Нормализация данных**

Нормализация — это метод, который часто применяется как часть подготовки данных для машинного обучения. Цель нормализации — изменить значения числовых столбцов в наборе данных для использования общей шкалы без искажения различий в диапазонах значений или потери информации. Нормализация также требуется для некоторых алгоритмов для правильного моделирования данных.

Например, предположим, что ваш входной набор данных содержит один столбец со значениями от 0 до 1 и другой столбец со значениями от 10 000 до 100 000. Большая разница в *масштабе* чисел может вызвать проблемы, когда вы попытаетесь объединить значения как функции во время моделирования.

*Нормализация* позволяет избежать этих проблем, создавая новые значения, которые поддерживают общее распределение и соотношения в исходных данных, сохраняя при этом значения в пределах шкалы, применяемой ко всем числовым столбцам, используемым в модели.

Этот компонент предлагает несколько вариантов преобразования числовых данных:

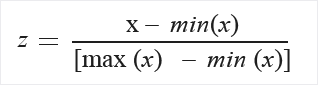
* Вы можете изменить все значения на шкалу 0–1 или преобразовать значения, представив их как процентильные ранги, а не абсолютные значения.
* Нормализацию можно применить к одному столбцу или к нескольким столбцам в одном наборе данных.
* Если вам нужно повторить конвейер или применить те же шаги нормализации к другим данным, вы можете сохранить эти шаги как преобразование нормализации и применить его к другим наборам данных с такой же схемой.

**Метод преобразования**.

* **MinMax**: нормализатор min-max линейно изменяет масштаб каждой функции до интервала [0,1].

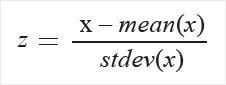
Масштабирование в интервале [0,1] осуществляется путем сдвига значений каждого компонента таким образом, чтобы минимальное значение было равно 0, а затем деления на новое максимальное значение (которое представляет собой разницу между первоначальными максимальными и минимальными значениями).

Значения в столбце преобразуются по следующей формуле:



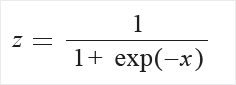
* **Zscore**: преобразует все значения в z-оценку.

Значения в столбце преобразуются по следующей формуле:



Среднее и стандартное отклонения вычисляются для каждого столбца отдельно. Используется стандартное отклонение совокупности.

* **Логистика**: значения в столбце преобразуются по следующей формуле:



**20. Очистка данных для Data Mining**

**Очистка данных –** процесс обработки выборки для интеллектуального анализа информации (**Data Mining**) с помощью алгоритмов машинного обучения (**Machine Learning**). Этот этап, на котором выполняется выявление и удаление ошибок и несоответствий в данных с целью улучшения качества датасета, также называется data cleaning, data cleansing или scrubbing. Некорректная, дублирующая или утраченная информация может стать причиной неадекватной статистикии неверных выводов в контексте поставленной задачи. Поэтому очистка данных является обязательной процедурой Data Preparation.

**От чего надо чистить большие данные**

Существуют 2 вида проблем с данными, от которых избавляет процедура их очистки:

* **проблемы с признаками** – значениями переменных, столбцами в табличном представлении датасета;
* **проблемы с записями** – объектами, которые являются строками датасета и описываются значениями признаков.

**На уровне признаков** выделяют 6 основные проблемы:

1. **недопустимые значения**, которые лежат вне нужного диапазона, например, цифра 7 в поле для школьных оценок по пятибалльной шкале;
2. **отсутствующие значения**, которые не введены, бессмысленны или не определены, к примеру, число 000-0000-0000 в качестве телефонного номера;
3. **орфографические ошибки** – неверное написание слов: «водитл» вместо «водитель» или «Омск» вместо «Томск», что искажает первичный смысл переменной, подставляя вместо одного города другой;
4. **многозначность**: использование разных слов для описания одного и того же по смыслу значения, например, «водитель» и «шофёр» или применение одной аббревиатуры для разных по смыслу значений, к примеру, «БД» может быть сокращением для словосочетания «[большие данные](https://www.bigdataschool.ru/wiki/%d0%b1%d0%be%d0%bb%d1%8c%d1%88%d0%b8%d0%b5-%d0%b4%d0%b0%d0%bd%d0%bd%d1%8b%d0%b5)» или «база данных»;
5. **перестановка слов**, обычно встречается в текстовых полях свободного формата;
6. **вложенные значения** – несколько значений в одном признаке, например, в поле свободного формата.

**На уровне записей** выделяют 4 основные проблемы:

* **нарушение уникальности**, например, паспортного номера или другого идентификатора;
* **дублирование записей**, когда один и тот же объект описан дважды;
* **противоречивость записей**, когда один и тот же объект описан различными значениями признаков;
* **неверные ссылки —** нарушение логических связей между признаками.

**Детали фазы Data Preparation**

**Метод очистки** данных должен удовлетворять ряду критериев:

* быть способным выявлять и удалять все основные ошибки и несоответствия, как в отдельных источниках данных, так и при интеграции нескольких источников;
* поддерживаться определенными инструментами, чтобы сократить объемы ручной проверки и программирования;
* быть гибким в плане работы с дополнительными источниками.

На практике исследователи данных применяют **2 способа** для Data Cleaning:

* **Автоматизированная очистка данных** с помощью встроенных средств СУБД или интегрированных систем для статистического анализа;
* **Очистка данных собственными силами**, когда аналитик самостоятельно ищет готовые или разрабатывает свои скрипты для исправления опечаток в текстовых полях.

Используя 1 из вышеуказанных способов или сразу оба, data scientist также конвертирует типы данных, агрегирует признаки, заполняет отсутствующие значения, а также избавляется от шумов и выбросов.