ГОСЫ

1. Создайте CSS-файл, устанавливающий стиль шрифта: его цвет, размер и гарнитуру таким образом, чтобы иметь возможность применения данного стиля к различным тегам. Укажите, каким образом созданный CSS-файл может быть применен к web-документу

```css
// style.css
.common-font {
  color: #333333;
  font-size: 18px;
  font-family: sans-serif;
}
```

- `font-family` указывает шрифт, также через запятую можно указывать фолбеки, например: `Arial, sans-serif`.
- `font-size` задаёт кегль; абсолютные единицы вроде px чаще всего предсказуемы в браузерах.

Применяем стиль к разным тегам

```html
<h1 class="common-font">Заголовок сайта</h1>
<p class="common-font">Основной абзац текста.</p>
<li class="common-font">Пункт в списке.</li>
```

Одна и та же CSS-класс `.common-font` может быть назначена любому элементу через атрибут `class`.

Подключение CSS к web-документу

```html
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Пример страницы</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1 class="common-font">Заголовок сайта</h1>
  <p class="common-font">Основной абзац текста.</p>
  <li class="common-font">Пункт в списке.</li>
</body>
</html>
```

Тег `<link rel="stylesheet" href="style.css">` в заголовке (`<head>`) подключает внешний файл и делает его правила доступными всему документу.

Как еще можно указывать стили

- Внутренний блок: `<style> …CSS… </style>` в `<head>`;
- Инлайн: `<p style="color:#333;">`;
- Директива @import: `@import url("theme.css");` внутри CSS для модульности.

Кому еще можно указывать стили

- тегам, например `p {...}`
- классам, например `.class {...}`
- индентификаторам, например `#id {...}`
- группам/вложенности селекторов, например `p, h1 {...} | p h1 {...}`
- псевдоклассам, например `button:hover {...}`
- псевдоэлементам, например `blockquote::before { content: "“"; font-size: 2em; }`

3. Создайте CSS-файл, обеспечивающий изменение цвета, а также отмену подчеркивания текстовых гиперссылок при наведении на них курсора

```css
a {
    color: #0056b3;          /* Стандартный цвет ссылки */
    text-decoration: underline; /* Подчеркивание по умолчанию */
    transition: color 0.3s ease, text-decoration 0.3s ease;
}

a:hover {
    color: #e63946;            /* Новый цвет при наведении */
    text-decoration: none;     /* Убираем подчеркивание */
}
```

4. Создайте web-документ с маркированным списком и для него CSS-файл, задающий тип маркера и обтекание маркера текстом

```html
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Маркированный список с обтеканием</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Пример маркированного списка с обтеканием</h1>
    <ul class="custom-list">
        <li>Первый элемент списка с достаточно длинным текстом, чтобы показать обтекание маркера текстом</li>
        <li>Второй элемент списка, тоже с длинным текстом для демонстрации работы CSS-свойств обтекания текста вокруг маркера</li>
        <li>Третий элемент списка, который продолжает демонстрировать возможности оформления маркированных списков</li>
    </ul>
</body>
</html>
```

```css
.custom-list {
    list-style-type: square; /* Тип маркера – квадрат */
    list-style-position: inside; /* Маркер внутри блока с текстом */
    width: 300px; /* Ограничиваем ширину для демонстрации обтекания */
}
```

5. Перечислите абсолютные и относительные единицы измерения, с помощью которых можно задать размер шрифта, используя свойство font-size. Приведите пример описания стиля для создания верхнего индекса в тексте

Набор констант (xx-small, x-small, small, medium, large, x-large, xx-large) задает размер, который называется абсолютным. По правде говоря, они не совсем абсолютны, поскольку зависят от настроек браузера и операционной системы.

Другой набор констант (larger, smaller) устанавливает относительные размеры шрифта. Поскольку размер унаследован от родительского элемента, эти относительные размеры применяются к родительскому элементу, чтобы определить размер шрифта текущего элемента.

Верхний индекс:

```
<p>Квадратное уравнение: y=ax<sup>2</sup>+bc+c</p><br>
```

```
<p>Формула воды – H<sup>2</sup>O</p>
```

Нижний индекс:

```
<p>Формула дофамина: C<sub>8</sub>H<sub>11</sub>NO <sub>2</sub></p>
```

6. Создайте CSS-файл и web-документ с блоком, содержащим изображение и обтекающий его справа текст. Рамка блока должна иметь скругленные уголки

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Обтекание текста</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="block">
        <img src="image.jpg" alt="Пример изображения">
        <p>Здесь находится текст, который обтекает изображение справа. Этот блок имеет
скруглённые уголки.</p>
    </div>
</body>
</html>
```

```
.block {
    border: 2px solid #333;
    border-radius: 15px; /* Скруглённые уголки */
    padding: 20px;
    overflow: hidden; /* Чтобы рамка охватывала floated элементы */
}

.block img {
    float: left; /* Обтекание текста справа */
    margin-right: 20px;
    width: 150px;
    height: auto;
}
```

7. Создайте web-документ, содержащий блочный элемент контейнер. Создайте CSS-файл, задающий для этого блока его размер, расположение в окне браузера (сверху посередине),

цвет фона, размер полей и цвет шрифта

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Контейнер с CSS-стилями</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        Содержимое контейнера
    </div>
</body>
</html>
```

```
.container {
    /* Размеры */
    width: 80%;
    min-height: 200px;

    /* Расположение */
    margin: 0 auto; /* Центрирование по горизонтали */
    margin-top: 30px; /* Отступ сверху */

    /* Цвета */
    background-color: #f0f8ff; /* Светло-голубой фон */
    color: #2e4053; /* Темно-синий цвет текста */

    /* Поля */
    padding: 25px;
}
```

8. Создайте web-документ и для него CSS-файл, с помощью которого фоновое изображение web-страницы будет занимать всю доступную площадь окна браузера

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <title>Задание</title>
        <link rel="stylesheet" href="Style.css" />
    </head>
    <body>
        <br/>
    </body>
</html>
```

```
 Style.css
html {
    height: 100%;
    width: 100%;
    }
    body {
    background-image:
    url(https://pg12.ru/userfiles/picfullsize/img-45237-15581813813568.jpg);
    background-size: cover;
```

```
    background-repeat: no-repeat;
}
```

9. Создайте web-документ с таблицей и соответствующий CSS-файл, обеспечивающий задание ширины таблицы и ячеек, а также выравнивание текста в ячейке таблицы по вертикали

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <table>
        <tr>
            <th>Заголовок 1</th>
            <th>Заголовок 2</th>
            <th>Заголовок 3</th>
        </tr>
        <tr>
            <td>Ячейка 1</td>
            <td>Ячейка 2</td>
            <td>Ячейка 3</td>
        </tr>
        <tr>
            <td>Ячейка 4</td>
            <td>Ячейка 5</td>
            <td>Ячейка 6</td>
        </tr>
    </table>
</body>
</html>
```

```
table {
    width: 100%;
    border-collapse: collapse; /* Убирает двойные границы */
}
th, td {
    width: 33.33%; /* Ширина ячеек */
    padding: 8px; /* Отступы в ячейках */
    text-align: center; /* Выравнивание текста по горизонтали */
    vertical-align: middle; /* Выравнивание по вертикали */
    border: 1px solid #000; /* Границы */
}
th {
    background: #f0f0f0; /* Фон заголовков */
}
```

10. Создайте web-документ, содержащий многострочную таблицу. Создайте CSS-файл, обеспечивающий выделение всех нечетных строк таблицы одним цветом, а четных - другим цветом.

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Таблица с чередующимися строками</title>
    <link rel="stylesheet" href="styles.css">   <!-- Подключение внешнего стиля -->
```

```
</head>
<body>
    <h1>Пример таблицы</h1>
    <table>
        <thead>     <!-- Первая строка таблицы – обозначаения столбцов -->
            <tr>
                <th>№</th>  <!-- Разделение по столбцам + <h> – стиль заголовка -->
                <th>Имя</th>
                <th>Возраст</th>
                <th>Город</th>
            </tr>
        </thead>
        <tbody>     <!-- Основная таблица -->
            <tr>    <!-- Разделение по срокам -->
                <td>1</td>  <!-- Разделение по столбцам -->
                <td>Анна</td>
                <td>28</td>
                <td>Москва</td>
            </tr>
            <tr>
                <td>2</td>
                <td>Иван</td>
                <td>34</td>
                <td>Санкт-Петербург</td>
            </tr>
            <tr>
                <td>3</td>
                <td>Мария</td>
                <td>22</td>
                <td>Новосибирск</td>
            </tr>
            <tr>
                <td>4</td>
                <td>Пётр</td>
                <td>41</td>
                <td>Екатеринбург</td>
            </tr>
            <tr>
                <td>5</td>
                <td>Ольга</td>
                <td>30</td>
                <td>Казань</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

11. Создайте web-документ, содержащий форму с двумя группами кнопок-переключателей и кнопкой отправки

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Форма с кнопками-переключателями</title>
</head>
<body>
  <h2>Выберите параметры</h2>
  <form action="#">
    <!-- Первая группа переключателей -->
```

```
    <fieldset>
      <legend>Группа 1</legend>
      <label><input type="radio" name="group1" required> Выбор 1</label><br>
      <label><input type="radio" name="group1"> Выбор 2</label><br>
      <label><input type="radio" name="group1"> Выбор 3</label>
    </fieldset>

    <!-- Вторая группа переключателей -->
    <fieldset>
      <legend>Группа 2</legend>
      <label><input type="radio" name="group2" required> Выбор 1</label><br>
      <label><input type="radio" name="group2"> Выбор 2</label><br>
      <label><input type="radio" name="group2"> Выбор 3</label>
    </fieldset>

    <!-- Кнопка отправки -->
    <br>
    <button type="submit">Отправить</button>
  </form>
</body>
</html>
```

12. Создайте web-документ, содержащий форму с текстовым полем ввода, полем пароля и кнопкой отправки.

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Форма входа</title>
</head>
<body>
    <h2>Вход на сайт</h2>
    <form action="/login" method="post">
        <label for="username">Имя пользователя:</label><br>
        <input type="text" id="username" name="username" required><br><br>

        <label for="password">Пароль:</label><br>
        <input type="password" id="password" name="password" required><br><br>

        <button type="submit">Войти</button>
    </form>
</body>
</html>
```

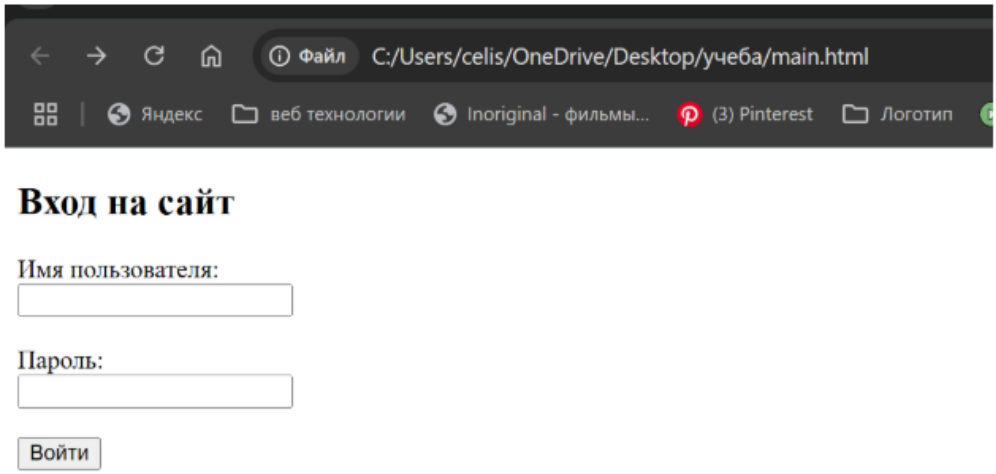
Пояснение

```
<!DOCTYPE html> <!-- Объявление типа документа, указывает, что используется HTML5 -->
<html lang="ru"> <!-- Открывающий тег HTML, язык страницы — русский -->
<head> <!-- Начало секции с метаданными -->
    <meta charset="UTF-8"> <!-- Установка кодировки для поддержки русских букв -->
    <title>Форма входа</title> <!-- Заголовок страницы, отображается на вкладке браузера -->
</head>
<body> <!-- Начало основного содержимого страницы -->
    <h2>Вход на сайт</h2> <!-- Заголовок второго уровня на странице -->
    <form action="/login" method="post"> <!-- Форма. action: куда отправлять данные, method: способ отправки -->
        <label for="username">Имя пользователя:</label><br> <!-- Подпись к полю ввода, связывается с id "username" -->
        <input type="text" id="username" name="username" required><br><br> <!-- Текстовое поле для ввода имени пользователя. required — обязательное для заполнения -->

        <label for="password">Пароль:</label><br> <!-- Подпись к полю пароля, связывается с id "password" -->
        <input type="password" id="password" name="password" required><br><br> <!-- Поле для ввода пароля, символы скрываются. required — обязательное поле -->

        <button type="submit">Войти</button> <!-- Кнопка отправки формы -->
    </form>
</body>
</html>
```

Результат:



13. Создайте web-документ, содержащий форму с текстовой областью, полем для выбора даты и кнопкой отправки.

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Форма с датой и текстом</title>
</head>
<body>
  <h1>Оставьте ваш комментарий</h1>
  <form action="/submit" method="post">
    <label for="comment">Комментарий:</label><br>
    <textarea id="comment" name="comment" rows="5" cols="40" placeholder="Введите ваш текст здесь..."></textarea><br><br>
```

```
    <label for="date">Выберите дату:</label><br>
    <input type="date" id="date" name="date"><br><br>

    <button type="submit">Отправить</button>
  </form>
</body>
</html>
```

form action="/submit" – указывает куда отправлять данные формы после нажатия кнопки отправить.

method="post" определяет метод отправки данные. (get – данные передаются в URL, post – данные отправляются в теле запроса)

input type="date" определяет, что это поле выбора даты, браузер отобразит календарь

button type="submit" определяет, что это кнопка отправки формы

14. Создайте web-документ с горизонтальным меню, содержащим три гиперссылки, по щелчку на каждой из которых будет выполнен переход к одному из расположенных ниже параграфов с текстом. В качестве текста можно использовать любой подстановочный текст

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Мини-меню</title>
</head>
<body>

  <nav>
    <a href="#para1">Пункт 1</a>
    <a href="#para2">Пункт 2</a>
    <a href="#para3">Пункт 3</a>
  </nav>

  <p id="para1">Lorem ipsum dolor sit amet, consectetur adipiscing elit…</p>
  <p id="para2">Sed ut perspiciatis unde omnis iste natus error sit…</p>
  <p id="para3">At vero eos et accusamus et iusto odio dignissimos…</p>

</body>
</html>
```

Внутренние ссылки. Атрибут `href="#para1"` указывает браузеру прокрутить страницу к элементу с `id="para1"`. Такая «якорная» навигация описана в спецификации HTML и остаётся самым простым способом переходов внутри документа.

16. На базе технологии CSS Grid создайте web-страницу с шестью блоками, расположенными в 2 ряда путем размещения элементов web- страницы в ячейках сетки. При ширине экрана до величины меньшей 700 px все элементы страницы должны выстроиться в одну колонку друг под другом

```
!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>CSS Grid Пример</title>
    <link rel="stylesheet" href="style16.css">
</head>
<body>
    <div class="grid-container">
        <div class="grid-item">Блок 1</div>
        <div class="grid-item">Блок 2</div>
        <div class="grid-item">Блок 3</div>
        <div class="grid-item">Блок 4</div>
        <div class="grid-item">Блок 5</div>
        <div class="grid-item">Блок 6</div>
    </div>
</body>
</html>
```

```
body {
    margin: 0;
    padding: 0;
    font-family: sans-serif;
}

.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 16px;
    padding: 20px;
}

.grid-item {
    background-color: #4a90e2;
    color: white;
    padding: 40px 0;
    text-align: center;
    font-size: 18px;
    border-radius: 8px;
}

/* Адаптация под экраны меньше 700px */
@media (max-width: 699px) {
    .grid-container {
        grid-template-columns: 1fr;
    }
}
```

17. Напишите возможные способы выборки первого параграфа веб-страницы, имеющего класс class="bluetext"

1. С помощью querySelector. querySelector возвращает первый элемент, который соответствует CSS-селектору p.bluetext.

```
<script>
  const firstBlueText = document.querySelector("p.bluetext");
  console.log(firstBlueText.textContent);
</script>
```

2. Через getElementsByClassName + фильтрацию по тегу. Этот способ вручную проверяет, что элемент именно `<p>`.

```
<script>
  const bluetexts = document.getElementsByClassName("bluetext");
  for (let el of bluetexts) {
    if (el.tagName.toLowerCase() === "p") {
      console.log(el.textContent);
      break;
    }
  }
</script>
```

3. jQuery

```
<script src="https://code.jquery.com/jquery-3.7.0.min.js"></script>
<script>
  const firstP = $("p.bluetext").first();
  console.log(firstP.text());
</script>
```

18. Напишите возможные способы выборки элемента веб-страницы, имеющего идентификатор id="element1"

Через getElementById

```
const el = document.getElementById("element1");
```

- Быстро и эффективно.
- Работает только с id.

Через querySelector

```
const el = document.querySelector("#element1");
```

- Универсальный способ - можно использовать любой CSS-селектор.
- Возвращает первый найденный элемент.

Через querySelectorAll (если нужно получить NodeList, например для перебора)

```
const elList = document.querySelectorAll("#element1");
// elList[0] — это нужный элемент
Через getElementsByName (если элемент имеет соответствующий name)
const elList = document.getElementsByName("element1");
```

- id и name - разные атрибуты. Этот способ сработает, только если у элемента есть name="element1".

Через getElementsByTagName или getElementsByClassName в сочетании с фильтрацией по id

```
const divs = document.getElementsByTagName("div");
const el = Array.from(divs).find(div => div.id === "element1");
```

- Неэффективно, если известен id - используется редко.

Через jQuery (если библиотека подключена)

```
const el = $("#element1");
```

19. Пусть имеется веб-документ, в котором имеются заголовок, изображение и кнопка с надписью «Моя мечта». По клику на кнопке в конце веб-документа должен появиться текст «Хочу стать хорошим программистом».

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Моя мечта</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Моя страница мечты</h1>
    <img src="img.jpg" alt="Пример изображения">
    <button id="dream-button">Моя мечта</button>
    <div id="dream-text"></div>

    <script src="script.js"></script>
</body>
</html>
```

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    margin: 20px;
}

img {
    max-width: 300px;
    margin: 20px 0;
}

button {
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
}

#dream-text {
    margin-top: 30px;
    font-size: 18px;
    font-weight: bold;
    color: green;
    min-height: 24px;
}
```

```
document.getElementById('dream-button').addEventListener('click', function() {
    document.getElementById('dream-text').textContent = 'Хочу стать хорошим программистом';
});
```

20. Пусть имеется контейнер с текстом «Я учусь в университете» и кнопка с надписью «Мой университет». По клику на кнопке под ней должен появиться текст «Мой университет – ГУАП»

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Университет</title>
    <style>
        #university-info {
            display: none;
        }
    </style>
</head>
<body>
    <div>
        <div>Я учусь в университете</div>
        <button id="university-btn">Мой университет</button>
        <div id="university-info">Мой университет – ГУАП</div>
    </div>

    <script>
        document.getElementById('university-btn').addEventListener('click', function() {
            document.getElementById('university-info').style.display = 'block';
        });
    </script>
</body>
</html>
```

21. Создайте web-документ, содержащий блок с текстом, напишите сценарий на языке JavaScript, обеспечивающий по клику на блоке изменение его текстового содержимого и цвета фона

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="UTF-8" />
        <title>Изменение блока по щелчку</title>
        <link rel="stylesheet" href="style.css" />
    </head>
    <body>
        <div id="myBlock">Нажмите на этот блок</div>

        <script src="script.js"></script>
    </body>
</html>
```

```
#myBlock {
    width: 500px;
    padding: 20px;
    margin: 50px auto;
    text-align: center;
    background-color: lightblue;
    border: 2px solid #333;
    border-radius: 10px;
    cursor: pointer;
    font-size: 18px;
```

```
    transition: background-color 0.3s;
}
```

```
const block = document.getElementById('myBlock');


// Сохраняем начальные значения
const originalText = 'Нажмите на этот блок';
const newText = 'Текст и фон изменены!';
const originalColor = 'lightblue';
const newColor = 'lightgreen';


// Состояние переключателя
let toggled = false;


block.addEventListener('click', () => {
    if (!toggled) {
        block.textContent = newText;
        block.style.backgroundColor = newColor;
    } else {
        block.textContent = originalText;
        block.style.backgroundColor = originalColor;
    }
    toggled = !toggled;
});
```

22. Создайте web-документ, содержащий слайд-шоу, разработанное на языке JavaScript, количество изображений равно трем, интервал смены изображений 1 секунда

```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript Slideshow Example</title>
        <style>
            #slideshow-container {
                position: relative;
                height: 300px;
                overflow: hidden;
            }

            .slideshow-img {
                position: absolute;
                top: 0;
                left: 0;
                width: 100%;
                height: 100%;
                object-fit: cover;
                transition: opacity 0.5s ease-in-out;
            }
        </style>
    </head>
    <body>
        <div id="slideshow-container">
            <img class="slideshow-img" src="image1.jpg" alt="Image 1" />
            <img class="slideshow-img" src="image2.jpg" alt="Image 2" />
            <img class="slideshow-img" src="image3.jpg" alt="Image 3" />
        </div>
        <script>
```

```
            var images = document.querySelectorAll(".slideshow-img");
            var currentImageIndex = 0;
            var slideshowInterval = setInterval(nextImage, 1000);

            function nextImage() {
                images[currentImageIndex].style.opacity = 0;
                currentImageIndex = (currentImageIndex + 1) % images.length;
                images[currentImageIndex].style.opacity = 1;
            }
        </script>
    </body>
</html>
```

23. Создайте web-документ, в котором имеется кнопка, по клику на которой под кнопкой должно появиться какое-либо изображение. Требуется для назначения событию обработчика использовать свойство DOM-элемента on{событие}

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="UTF-8" />
        <title>Показ изображения по клику</title>
        <style>
            body {
                font-family: Arial, sans-serif;
                padding: 20px;
            }
            #image-container img {
                margin-top: 20px;
                max-width: 100%;
                height: auto;
            }
        </style>
    </head>
    <body>
        <button id="showBtn">Показать изображение</button>
        <div id="image-container"></div>

        <script>
            // Назначение обработчика с использованием свойства DOM-элемента
            document.getElementById("showBtn").onclick = function () {
                const container = document.getElementById("image-container");

                // Проверяем, не добавлено ли уже изображение
                if (container.children.length === 0) {
                    const img = document.createElement("img");
                    img.src = "image.jpg";
                    img.alt = "Пример изображения";
                    container.appendChild(img);
                }
            };
        </script>
    </body>
</html>
```

24. Создайте web-документ, в котором имеется кнопка, по клику на которой под кнопкой должно появиться какое-либо изображение. Требуется для назначения событию обработчика использовать метод addEventListener()

```html
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Показать изображение</title>
</head>
<body>
  <button id="showImageBtn">Показать изображение</button>
  <div id="imageContainer"></div>

  <script>
    document.getElementById("showImageBtn").addEventListener("click", function() {
      const container = document.getElementById("imageContainer");

      // Проверка, если изображение уже добавлено — не добавлять снова
      if (container.querySelector("img")) return;

      const img = document.createElement("img");
      img.src = "image.jpg"; // можно заменить на любое изображение
      img.alt = "Пример изображения";
      container.appendChild(img);
    });
  </script>

</body>
</html>
```

25. Задачи решаемые методами ИИ (классификация, кластеризация, регрессии и т.д.)

- **Классификация** – это отнесение объектов (наблюдений, событий) к одному из заранее известных классов. Пример: Определение электронной почты как «спам» или «не спам»; классификация изображений на «кошек» и «собак»; диагностика заболеваний по симптомам.
- **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. Пример: Сегментация покупателей по стилю покупок; группировка новостных статей по темам; выделение сообществ в социальных сетях.
- **Регрессия**, в том числе задачи прогнозирования. Установление зависимости непрерывных выходных от входных переменных. Пример: Прогнозирование цены квартиры по её характеристикам; предсказание температуры воздуха; оценка уровня продаж в будущем.
- **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). Пример: Если покупатель приобретает хлеб и масло, он часто покупает и молоко; анализ совместных покупок товаров в супермаркете.
- **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. Пример: Анализ покупательского поведения по времени (после покупки телефона через месяц приобретается чехол); выявление закономерностей в действиях пользователей на сайте.
- **Анализ отклонений** – выявление наиболее нехарактерных шаблонов. Пример: Обнаружение мошеннических транзакций в банковских операциях; выявление сбоев в работе оборудования по данным сенсоров; поиск аномалий в медицинских анализах.

26. Алгоритм k-ближайших соседей (KNN)

KNN (k-nearest neighbors) это алгоритм машинного обучения, который используется для классификации и регрессии. Он относится к методам обучения без учителя и основан на сравнении объектов по расстоянию. Метод относится к методам обучения с учителем, но при этом называется ленивым (lazy learning) — он не строит явную модель заранее, а просто запоминает данные и применяет логику при каждом новом запросе.

Суть алгоритма:

1. Получение обучающей выборки: У нас есть данные, где каждый объект описан признаками и меткой (класс или значение).
2. Выбор параметра k — количество ближайших соседей, которые будут учитываться.
3. Поступает новый объект (без метки) — надо его классифицировать (или предсказать значение).
4. Вычисление расстояний от нового объекта до всех объектов обучающей выборки. Самое популярное: Евклидово расстояние

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

5. Сортировка объектов по расстоянию — от ближайшего к самому далекому.
6. Выбор k ближайших объектов (с минимальным расстоянием).
7. Анализ меток соседей:
   - Для классификации: выбирается класс, который встречается чаще всего среди k соседей (например, из 5 соседей — трое принадлежат к классу A → выбрать A).
   - Для регрессии: берётся среднее значение по k соседям.

27. Алгоритм кластеризации k-средних (k-means)

Описание 1 (упрощенный)

Метод к-средних создает k-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 входящие 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 k-средних, сколько кластеров вам нужно, а он сделает всё остальное. Как это происходит? Метод k-средних имеет множество вариантов работы для различных типов данных. В общем случае все они делают примерно следующее:

1. Метод k-средних выбирает точки многомерного пространства, которые будут представлять k-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 k-кластеров, и каждый пациент – это член какого-то из них.
4. Метод k-средних, учитывая положение членов кластера, находит центр каждого из k-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 центры 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 k-средних как самообучающийся. Вместо того чтобы уточнять количество кластеров, метод k-средних «изучает» кластеры самостоятельно, не требуя информации о том, к какому кластеру относятся данные наблюдения. Метод k-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 k-средних может использоваться для предварительного разбиения на группы большого набора данных, после которого проводится более мощный кластерный анализ подкластеров. Метод k-средних может использоваться, чтобы «прикинуть» количество кластеров и проверить наличие неучтённых данных и связей в наборах.

Два основных недостатка метода k-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 определённое число кластеров k. Алгоритм представляет собой итерационную процедуру, в которой выполняются 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» определённым 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. Шаги 3 и 4 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 межкластерные расстояния.

Остановка алгоритма производится тогда, когда границы кластеров и расположения центроидов 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 неопределённость 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 неопределённость 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 введён Дж. Маккуинном в 1967 г.

29. Модель перцептрона, функции активации.

Перцептро́н, или **персептрон** — математическая или компьютерная модель восприятия информации мозгом (кибернетическая модель мозга).

Модель искусственного нейрона (рис. 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:

$$P = \sum_{i=1}^{n} w_i x_i$$

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).

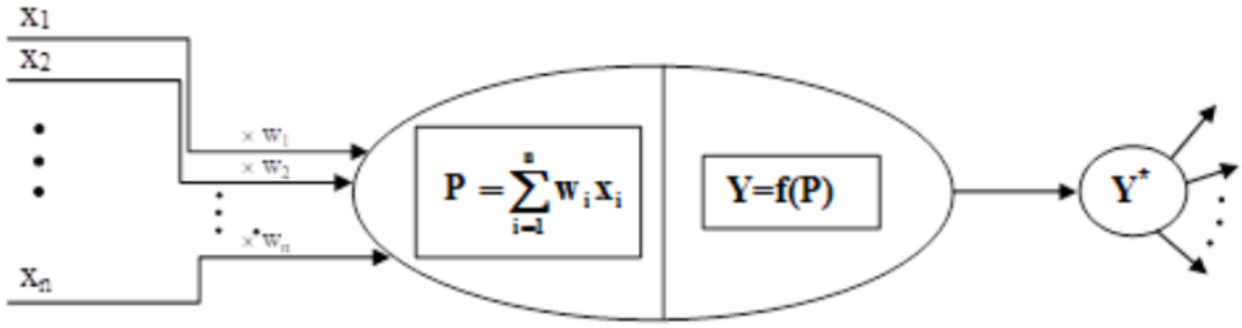
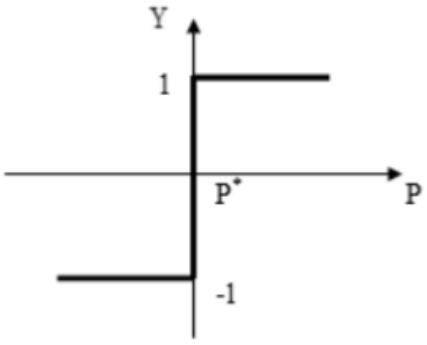


Рисунок 1. Схема математической модели нейрона.

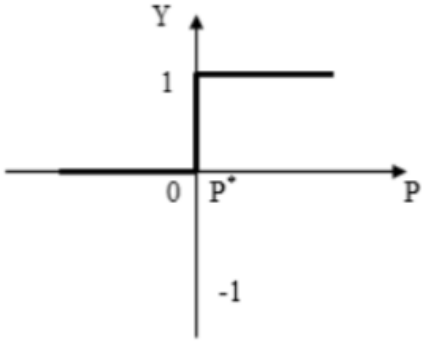
Будет активирован нейрон или нет, определяется вычислением так называемой передаточной, или активационной, функции нейрона f(∑wixi). Если значение функции превышает некоторый заранее определенный порог, то нейрон активирован и передает импульс следующим нейронам в сети.

В общем случае эта функция может быть **ступенчатой** (пороговой), **линейной** или **нелинейной** (S-подобной):

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. 2). Пороговая функция f(P) пропускает информацию только в том случае, если алгебраическая сумма входных сигналов превышает некоторую постоянную величину P*



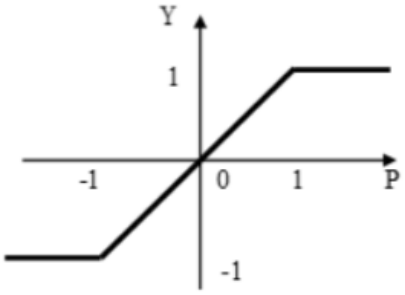
а) Y=1 если P>=P* и Y= -1 при P<P*



б) Y=1 если P>=P* и Y= 0 при P<P*

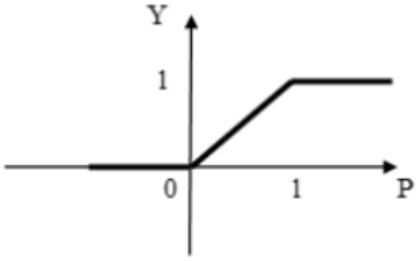
Пороговая функция не обеспечивает достаточной гибкости ИНС при обучении. Если значение вычисленного потенциала не достигает заданного порога, то выходной сигнал не формируется и нейрон «не срабатывает».

2. Линейная функция. Линейная функция Y=kP дифференцируема и легко вычисляется (рис.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.



Y= -1 если P< -1,
Y= P если -1≤P≤1,
Y= 1 если P>=1

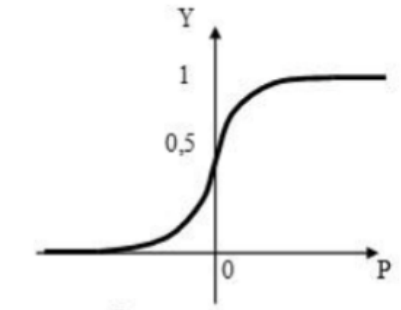
а)



Y= 0 если P< 0,
Y= P если 0≤P≤1,
Y= 1 если P>=1

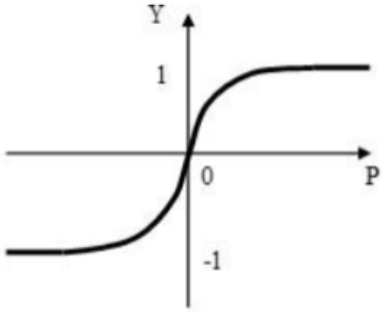
б)

3. S-подобная функция. Передаточная функция называется сигмоидой (рис. 4 а), когда ее диапазон [0, 1], или гиперболическим тангенсом (рис. 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. 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.



$Y=1/(1+e^{-kP})$

а)



$Y=1-1/(1+P)$ если $P\geq 0$

$Y=-1+1/(1-P)$ если $P\leq 0$

б)

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

30. Модели нейронных сетей

Исторические основы

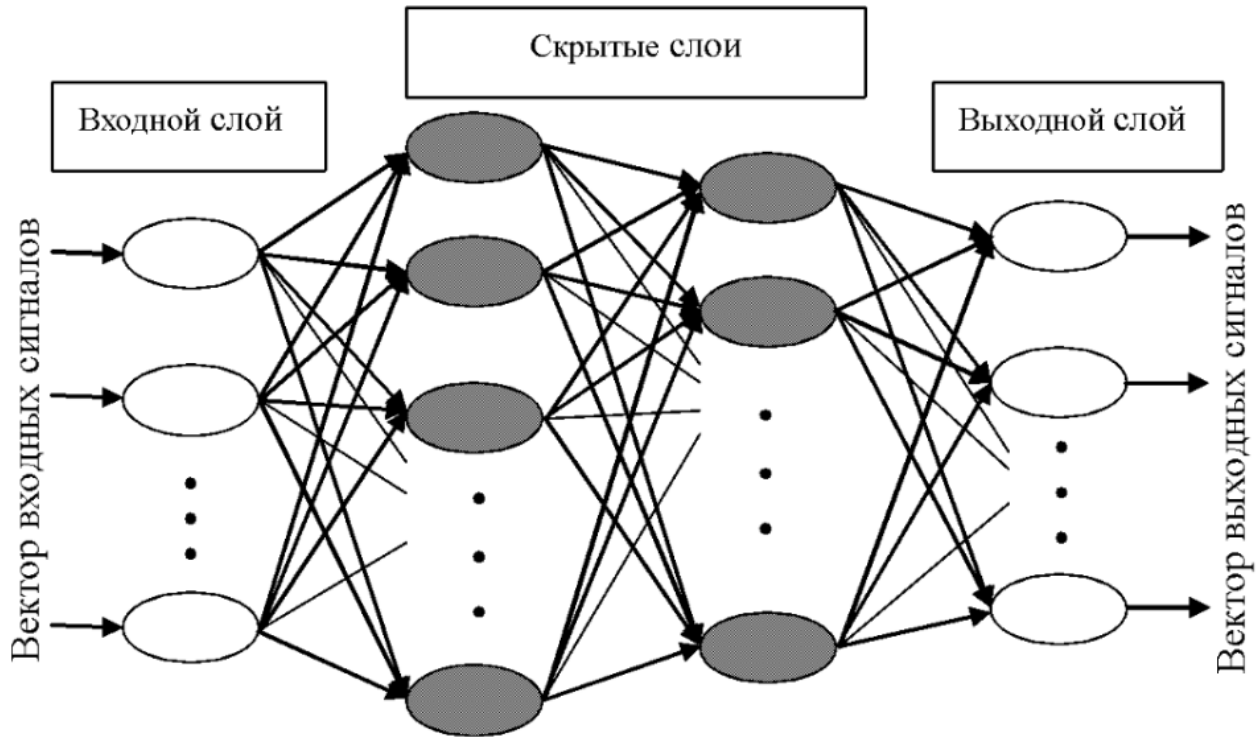
- Маккалох и Питтс (1943): первая модель нейрона — простейший процессор, вычисляющий функцию от суммы входов.
- Розенблатт (1958): предложил перцептрон — обучаемую сеть, корректирующую веса при ошибках распознавания.

Однослойный перцептрон

- Работает только для линейно сепарабельных задач.
- Ограничен в распознавании сложных образов.

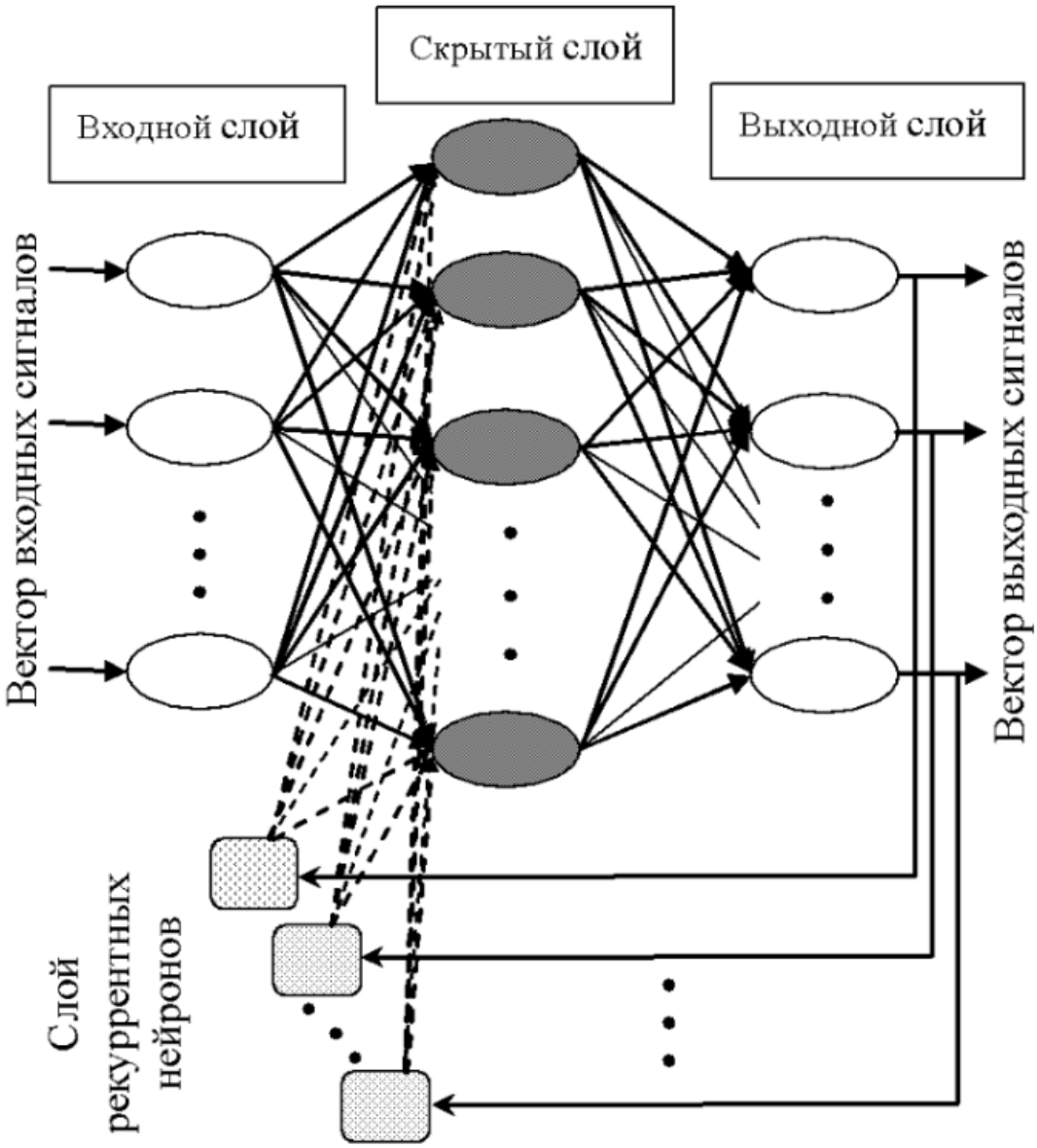
Многослойный перцептрон (MLP)

- Позволяет строить нелинейные границы между классами.
- Использует скрытые слои, сигмоидальные функции и обратное распространение ошибки.
- Пример структуры: 30–15–8 (вход–скрытый–выход).
- Способен аппроксимировать любые функции, при наличии достаточного числа нейронов.



Рекуррентные нейронные сети (RNN):

- Имеют обратные связи: состояние зависит от предыдущих входов.
- Позволяют моделировать память, предсказывать последовательности.
- Модель Хопфилда: ассоциативная память, сведение задачи к минимизации энергии.



Сети Хопфилда

- Все связи симметричны, нет самосвязей.
- Реализуют ассоциативную память.
- Недостатки: переизбыточность связей, симметричность, ограниченная биологическая реалистичность.

Самоорганизующиеся карты Кохонена (SOM)

- Один слой, обучение без учителя, основано на конкуренции нейронов.
- Побеждает нейрон с наиболее близким к входу вектором весов.
- Используются для кластеризации и визуализации данных.

Свёрточные нейронные сети (CNN):

- Предложены Яном Лекуном (1988).
- Состоят из свёрточных и пуллинговых слоёв, обучаются по методу обратного распространения.
- Эффективны для распознавания изображений и применяются в глубоком обучении.

31. Байесовский классификатор

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:

как часто происходит событие B при наступлении события A, что обозначается в формуле как P(B|A);

- какова вероятность того, что A не зависит от других событий, обозначаемая в формуле как P(A);
- 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:

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

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»:

Frequency Table

		Stolen?	
		Yes	No
Color	Red	3	2
	Yellow	2	3

Likelihood Table

		Stolen?	
		P(Yes)	P(No)
Color	Red	3/5	2/5
	Yellow	2/5	3/5

Частотная таблица и таблица правдоподобия для параметра «тип»:

Frequency Table

		Stolen?	
		Yes	No
Type	Sports	4	2
	SUV	1	3

Likelihood Table

		Stolen?	
		P(Yes)	P(No)
Type	Sports	4/5	2/5
	SUV	1/5	3/5

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

Frequency Table

		Stolen?	
		Yes	No
Origin	Domestic	2	3
	Imported	3	2

Likelihood Table

		Stolen?	
		P(Yes)	P(No)
Origin	Domestic	2/5	3/5
	Imported	3/5	2/5

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). При нанесении на график получается колоколообразная кривая, симметричная относительно средней значений характеристик.

32. Метрики в задачах машинного обучения (accuracy, precision, recall, F-мера и др.)

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)

Метрики в задачах машинного обучения (accuracy, precision, recall, F-мера и др.)

	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 100 не-спам писем, 90 из которых наш классификатор определил верно (True Negative = 90, False Positive = 10), и 10 спам-писем, 5 из которых классификатор также определил верно (True Positive = 5, False Negative = 5).

Тогда accuracy:

Однако если мы просто будем предсказывать все письма как не-спам, то получим более высокую accuracy:
Accuracy=(5+90)/(5+90+10+5)=86,4

Однако если мы просто 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	УЖАСЫ	ЮМОР
БЕЗ ЭМОЦИЙ	17	0	1
УЖАСЫ	1	10	7
ЮМОР	1	2	16

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

33. 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 0 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:

$$z = \frac{\mathrm{x} - min(x)}{[\max(x) - min(x)]}$$

Zscore: преобразует все значения в z-оценку. Значения в столбце преобразуются по следующей формуле

$$z = \frac{\mathrm{x} - mean(x)}{stdev(x)}$$

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

$$z = \frac{1}{1 + \exp(-x)}$$

Код разной нормализации, вдруг пригодиться, Фомин как то просил

1. Минимаксная нормализация (Min-Max Scaling)

Приводит значения к диапазону [0, 1] или [a, b]:

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
```

Пример данных

```
data = np.array([[10, 2.5], [20, 3.1], [15, 2.8], [12, 2.3]])
```

Создание и применение MinMaxScaler

```
scaler = MinMaxScaler(feature_range=(0, 1))  # можно задать другой диапазон
normalized_data = scaler.fit_transform(data)

print("Исходные данные:\n", data)
print("Нормализованные данные:\n", normalized_data)
```

2. Z-нормализация (Standard Scaling)

Приводит данные к распределению с μ=0 и σ=1:

```
from sklearn.preprocessing import StandardScaler
```

Создание и применение StandardScaler

```
scaler = StandardScaler()
standardized_data = scaler.fit_transform(data)

print("Исходные данные:\n", data)
print("Стандартизированные данные:\n", standardized_data)
print("Средние значения:", scaler.mean_)
print("Стандартные отклонения:", scaler.scale_)
```

3. Нормализация по L2 (норма вектора)

```
from sklearn.preprocessing import Normalizer

# Нормализация по строкам (каждый образец как вектор)
normalizer = Normalizer(norm='l2')  # можно также 'l1' или 'max'
l2_normalized = normalizer.fit_transform(data)

print("L2-нормализованные данные:\n", l2_normalized)
```

4. Robust Scaling (устойчивый к выбросам)

```
from sklearn.preprocessing import RobustScaler

robust_scaler = RobustScaler()
robust_scaled = robust_scaler.fit_transform(data)

print("Robust-нормализованные данные:\n", robust_scaled)
```

34. Очистка данных

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 статистики и 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:

- недопустимые значения, которые лежат вне нужного диапазона, например, цифра 7 в поле для школьных оценок по пятибалльной шкале;
- отсутствующие значения, которые не введены, бессмысленны или не определены, к примеру, число 000-0000-0000 в качестве телефонного номера;
- орфографические ошибки – неверное написание слов: «водитл» вместо «водитель» или «Омск» вместо «Томск», что искажает первичный смысл переменной, подставляя вместо одного города другой;
- 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» или «база данных»;
- перестановка слов, обычно встречается в текстовых полях свободного формата;
- 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.

35. Метод Бустинг (ансамблевые методы обучения)

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;

C i N— число сочетаний из N по i.

$$\mu = \sum_{i=m}^{N} C_N^i p^i (1-p)^{N-i}$$

Если $p > 0.5$, то $\mu > p$
Если $N \to \infty$, то $\mu \to 1$

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

Бустинг

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:

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».

Клип	Критерий 1	Критерий 2	Критерий 3	Критерий 4	Критерий 5	...	Критерий 100	Есть 1 000 000?
1	Да	Да	Нет	Да	Да	...	Нет	Нет
2	Нет	Да	Нет	Нет	Да	...	Да	Нет
3	Да	Да	Да	Нет	Да	...	Нет	Нет
4	Нет	Нет	Да	Да	Нет	...	Да	Да
5	Да	Нет	Нет	Да	Да	...	Нет	Нет
6	Нет	Да	Да	Нет	Нет	...	Нет	Нет
7	Нет	Да	Нет	Нет	Да	...	Нет	Да
8	Нет	Нет	Нет	Нет	Нет	...	Да	Нет
9	Да	Да	Нет	Нет	Нет	...	Нет	Нет
10	Нет	Да	Да	Да	Да	...	Нет	Нет
11	нет	Нет	Нет	Нет	Да	...	Нет	Нет
12	Да	Нет	Да	Нет	Нет	...	Да	Да
...	...	...	...	...	...	...	...	...
1 000 000	Нет	Да	Да	Нет	Да	...	Да	Нет

Миллион клипов, сто критериев — обучающая выборка для алгоритма

Первая проблема

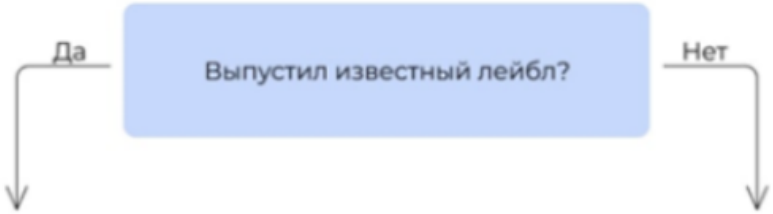
Если мы захотим построить дерево предсказаний для этой задачи, мы столкнёмся с проблемой: мы не знаем, какие критерии ставить в начало, а какие в конец, какие запихивать в какие ветки, а какие вообще не нужны и потому не влияют.

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.

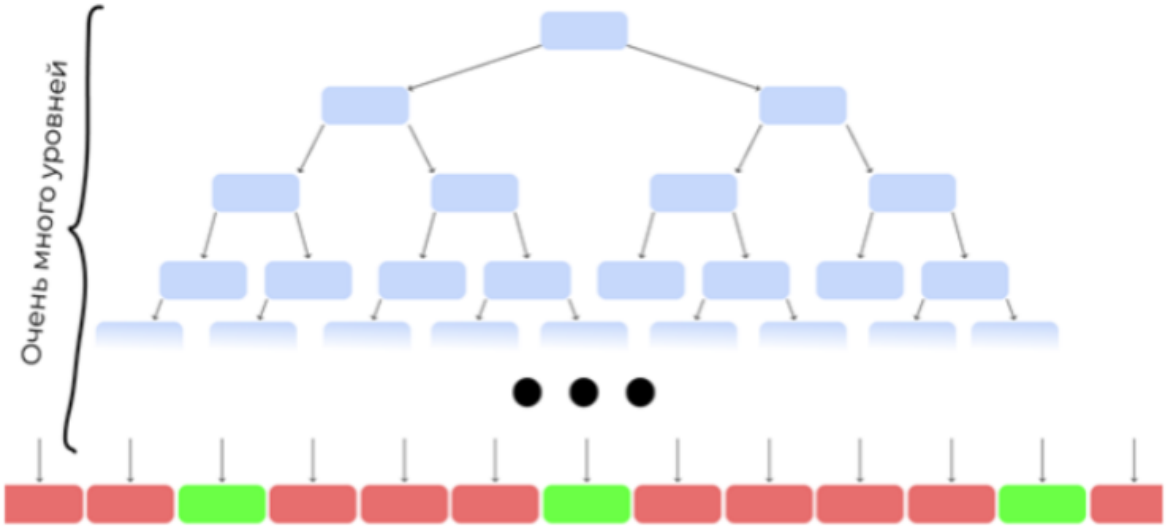
В итоге мы видим, что самая сильная связь с итоговой популярностью у критерия «Выпустил ли клип популярный лейбл». Получается, если клип выпустил лейбл, это влияет на успех сильнее, чем бочка или автотюн. Этот критерий встаёт на вершину дерева.



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.

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:

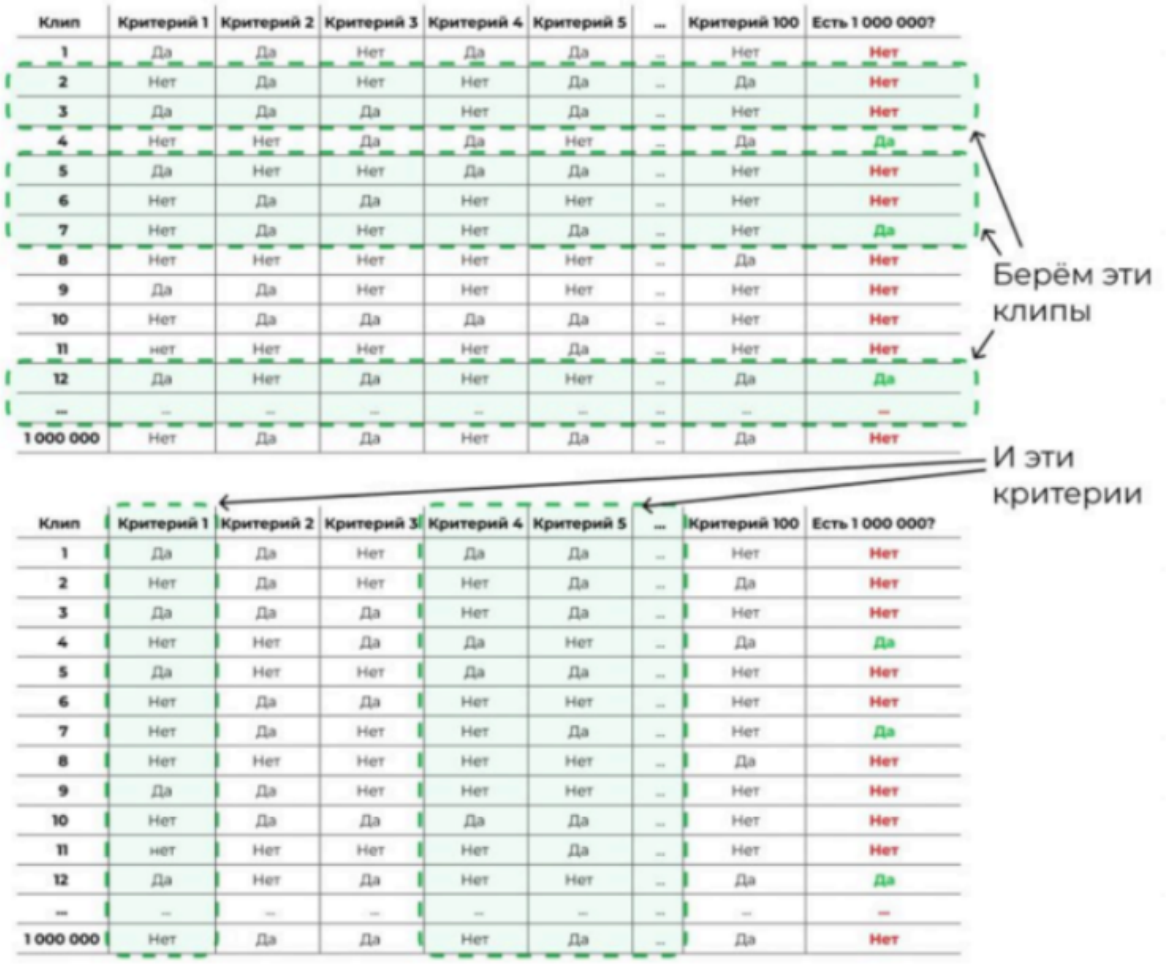


Большинство клипов не взлетит, но какие-то станут популярными

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:



Клип	Критерий 1	Критерий 2	Критерий 3	Критерий 4	Критерий 5	...	Критерий 100	Есть 1 000 000?
1	Да	Да	Нет	Да	Да	..	Нет	Нет
2	Нет	Да	Нет	Нет	Да	..	Да	Нет
3	Да	Да	Да	Нет	Да	..	Нет	Нет
4	Нет	Нет	Да	Да	Нет	..	Да	Да
5	Да	Нет	Нет	Да	Да	..	Нет	Нет
6	Нет	Да	Да	Нет	Нет	..	Нет	Нет
7	Нет	Да	Нет	Нет	Да	..	Нет	Да
8	Нет	Нет	Нет	Нет	Нет	..	Да	Нет
9	Да	Да	Нет	Нет	Нет	..	Нет	Нет
10	Нет	Да	Да	Да	Да	..	Нет	Нет
11	нет	Нет	Нет	Нет	Да	..	Нет	Нет
12	Да	Нет	Да	Нет	Нет	..	Да	Да
...	..	..	..	..	..	..	..	..
1 000 000	Нет	Да	Да	Нет	Да	..	Да	Нет

Клип	Критерий 1	Критерий 2	Критерий 3	Критерий 4	Критерий 5	...	Критерий 100	Есть 1 000 000?
1	Да	Да	Нет	Да	Да	..	Нет	Нет
2	Нет	Да	Нет	Нет	Да	..	Да	Нет
3	Да	Да	Да	Нет	Да	..	Нет	Нет
4	Нет	Нет	Да	Да	Нет	..	Да	Да
5	Да	Нет	Нет	Да	Да	..	Нет	Нет
6	Нет	Да	Да	Нет	Нет	..	Нет	Нет
7	Нет	Да	Нет	Нет	Да	..	Нет	Да
8	Нет	Нет	Нет	Нет	Нет	..	Да	Нет
9	Да	Да	Нет	Нет	Нет	..	Нет	Нет
10	Нет	Да	Да	Да	Да	..	Нет	Нет
11	нет	Нет	Нет	Нет	Да	..	Нет	Нет
12	Да	Нет	Да	Нет	Нет	..	Да	Да
...	..	..	..	..	..	..	..	..
1 000 000	Нет	Да	Да	Нет	Да	..	Да	Нет

И построим дерево попроще:

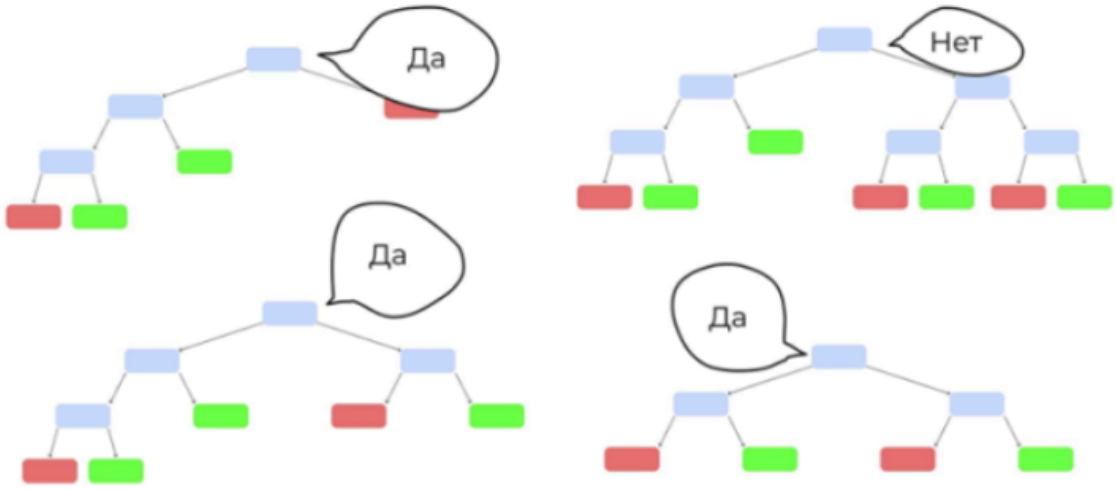
Было много уровней, стало меньше

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.



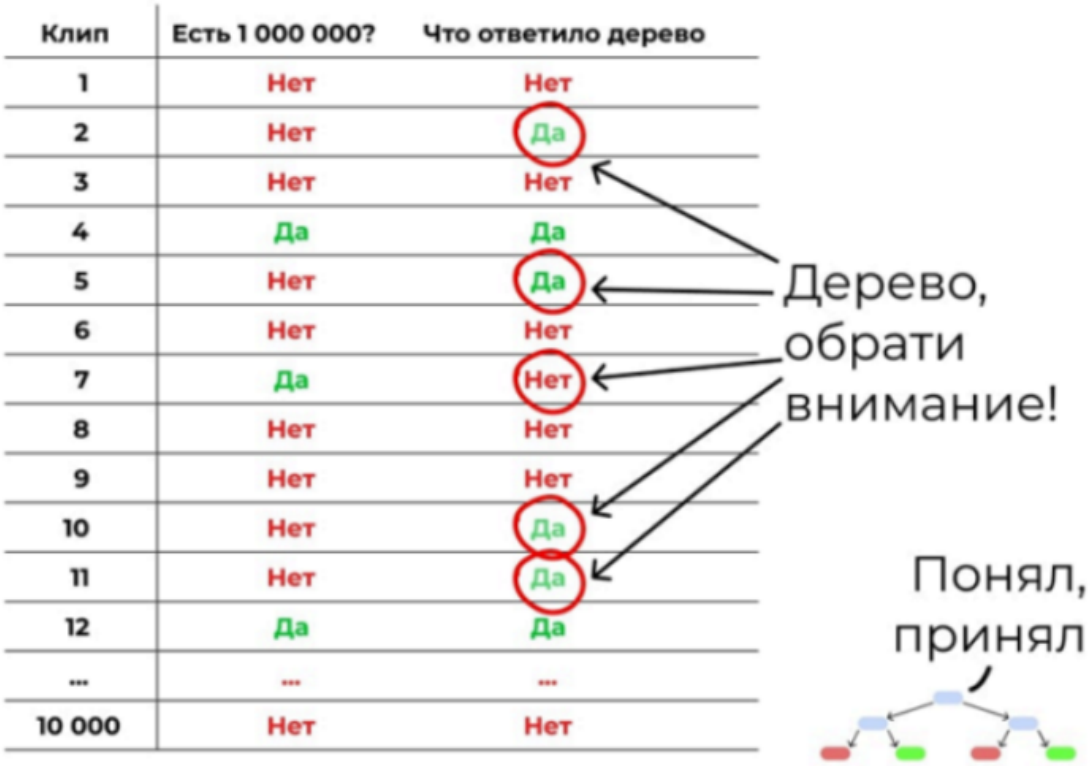
Три за, один против — клип ждёт успех. Наверное.

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:

Клип	Есть 1 000 000?	Что ответило дерево
1	Нет	Нет
2	Нет	Да
3	Нет	Нет
4	Да	Да
5	Нет	Да
6	Нет	Нет
7	Да	Нет
8	Нет	Нет
9	Нет	Нет
10	Нет	Да
11	Нет	Да
12	Да	Да
...	...	...
10 000	Нет	Нет

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.

36. Метод Бэггинг (ансамблевые методы обучения)

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 из N по i

$$\mu = \sum_{i=m}^{N} C_N^i p^i (1-p)^{N-i}$$

Если $p > 0.5$, то $\mu > p$
Если $N \to \infty$, то $\mu \to 1$

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

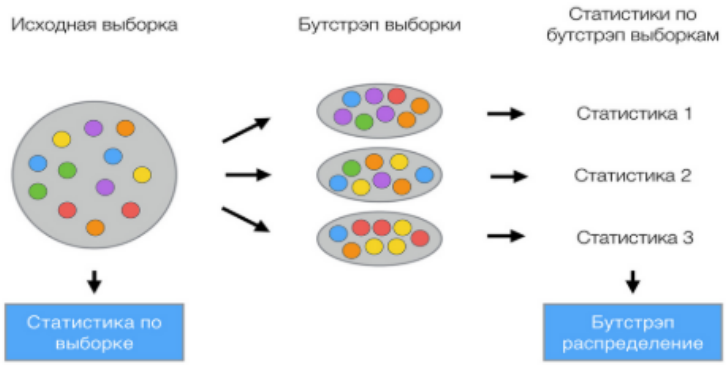
Их собралось около 800 человек, и они пытались угадать вес быка, который стоял перед ними. Бык весил 1198 фунтов. Ни один крестьянин не угадал точный вес быка, но если посчитать среднее от их предсказаний, то получим 1197 фунтов.

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 в 1994 году. Бэггинг основан на статистическом методе бутстрэпа, который позволяет оценивать многие статистики сложных распределений.

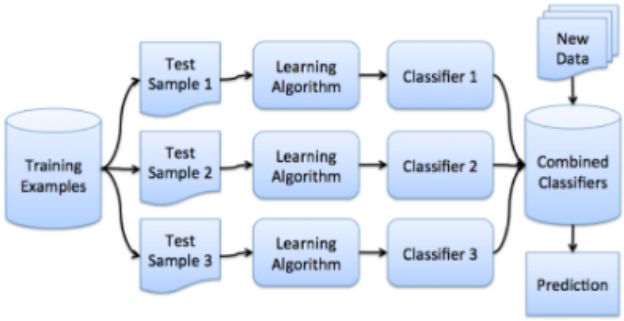
Метод бутстрэпа заключается в следующем. Пусть имеется выборка X размера N. Равномерно возьмем из выборки N объектов с возвращением. Это означает, что мы будем N раз выбирать произвольный объект выборки (считаем, что каждый объект «достается» с одинаковой вероятностью 1/N), причем каждый раз мы выбираем 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): $a(x) = \frac{1}{M}\sum_{i=1}^{M} a_i(x)$. Эту схему можно представить картинкой ниже.

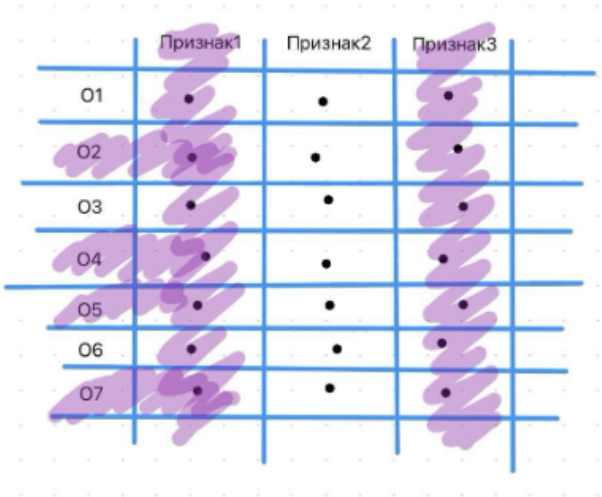


Бэггинг эффективен на малых выборках, когда исключение даже малой части обучающих объектов приводит к построению существенно различных базовых классификаторов. В случае больших выборок обычно генерируют подвыборки существенно меньшей длины.

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.

Случайный лес (не уверена, что нужно, но часто вставляется к пояснению к бэггингу)

Бэггинг направлен на уменьшение разброса (дисперсии) в данных, и зачастую данный прием предстает в виде алгоритма случайного леса, где слабые модели - это довольно глубокие случайные деревья. Однако, при построении случайного леса используется еще один прием, такой как метод случайных подпространств. Мало того, что благодаря бутсрэпу выбираются некоторые объекты нашего датасета, так еще и выбирается случайное подмножество признаков. В итоге, наша условная матрица признаков уменьшается как по строкам, так и столбцам (рис. 5). Это помогает действительно снизить корреляцию между слабыми учениками.



37. 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 (C, C++ 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.

38. Методы извлечения знаний когнитологом (пассивные, активные)

Извлечение знаний — это процесс получения информации о предметной области от эксперта с целью формализации и использования в экспертных системах. Методы делят на пассивные и активные в зависимости от роли инженера по знаниям.

1. Пассивные методы

Пассивные методы предполагают, что основную роль играет эксперт, а когнитолог преимущественно фиксирует получаемую информацию.

- Наблюдение: Когнитолог присутствует при работе эксперта и записывает его действия и комментарии.

  > Пример: Наблюдение за опытным врачом во время диагностики пациента.

- Анализ протоколов «мыслей вслух»: Эксперт озвучивает свои мысли и решения при выполнении задания, а инженер по знаниям их фиксирует.

  > Пример: Эксперт-программист вслух объясняет свои шаги при отладке программы.

- Лекция: Эксперт структурированно рассказывает о своей области, передаёт систематизированные знания.

  > Пример: Профессор читает лекцию о методах диагностики заболеваний.

2. Активные методы

Активные методы подразумевают, что инженер по знаниям сам управляет процессом получения информации, активно взаимодействуя с экспертом.

- Анкетирование: Эксперту предлагают заполнить анкету с вопросами по предметной области.

  > Пример: Врачам рассылают анкеты с вопросами по стандартам лечения.

- Интервью: Инженер по знаниям задаёт эксперту подготовленные или свободные вопросы, получая развернутые ответы.

  > Пример: Беседа с ведущим специалистом для уточнения особенностей производственного процесса.

- Свободный диалог: Обсуждение без строгой структуры для выявления неформализованных знаний.

  > Пример: Неофициальная беседа с экспертом о сложных случаях из практики.

- Игры и моделирование: Использование деловых или ролевых игр для выявления стратегии принятия решений экспертом.

  > Пример: Моделирование аварийной ситуации с последующим анализом действий эксперта.

> Для комплексного и эффективного извлечения знаний используют как пассивные, так и активные методы, выбирая подход в зависимости от задач, типа знаний и особенностей эксперта.

39. Структура экспертной системы

Экспертная система (ЭС) — интеллектуальная программная система, основанная на специализированной базе знаний и механизме логического вывода, которая имитирует рассуждения квалифицированного эксперта для решения неформализованных задач в узкой предметной области, объясняя ход получения результата и допускающая пополнение знаний без программирования.

Формализованные и неформализованные знания

- **Формализованные** — отражены в законах, формулах, алгоритмах; легко записываются, лежат в основе традиционного программирования.
- **Неформализованные** — эмпирические, субъективные, эвристические-приёмы специалистов; важны для описательных наук и составляют большинство реальных задач.
- **Формализованные / неформализованные области** — классифицируются по преобладающему типу знаний; соответственно различают **формализованные задачи** и **неформализованные задачи** (формализуемы, но формализация пока неизвестна).

Экспертные системы (ЭС)

Главное назначение — автоматизация **неформализованных** задач. Отличительные черты:

1. **Алгоритм** решения строится во время работы (символические рассуждения, эвристики).
2. **Прозрачность**: система объясняет ход вывода.
3. **Обучаемость**: пополняет знания от эксперта без программирования.
4. **Дружественный интерфейс** (часто естественный язык).

Базовая архитектура

Компонент	Роль
База данных (рабочая память)	Исходные и промежуточные факты задачи
База знаний (БЗ)	Долговременные факты и **правила вывода**
Решатель (интерпретатор)	Подбирает последовательность правил, выводит решение
Модуль приобретения знаний	Диалог с экспертом, пополнение БЗ
Объяснительная подсистема	Отвечает «почему?» и «как?»
Интерфейс	Человеко-ориентированный диалог

Участники разработки

- **Эксперт** — источник предметных знаний.
- **Инженер по знаниям (когнитолог)** — извлекает, структурирует и формализует знания, выбирает методы представления.
- **Программист** — создает инструментальную среду («оболочку»), интеграцию с внешними системами.

Основная проблема

*«Узкое место» — **приобретение знаний** (knowledge acquisition). Трудно — нужен высококвалифицированный инженер, эксперты редко формализуют опыт, возможен неполный или противоречивый ввод.*

Оболочки и системы поддержки знаний (СПЗ)

- **Оболочка ЭС** — универсальная программная рамка, куда подключают конкретную БЗ. Включает: систему вывода, обоснования, приобретения, отображения и редактирования знаний.
- **Современные СПЗ** стремятся:
  - быть независимыми от предметной области;
  - работать напрямую с экспертом (без «ручного» кодирования);
  - объединять текстовые источники, беседы, наблюдения;
  - поддерживать несколько экспертов и разные формы знания;
  - обеспечивать графическое отображение БЗ и контроль целостности;
  - базироваться на формальных теориях представления знания.

Классы задач, решаемых ЭС

Диагностика, прогнозирование, планирование, проектирование, управление, контроль, интерпретация сигналов, принятие решений в условиях **неопределённости**.

40. Интеллектуальный анализ данных, общие сведения и терминология

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 её 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 объёмы 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.

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), машинное обучение (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 использовать 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, независимо 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:

> Data Mining – 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 объёмов 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 представляет 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, сейсмопрогнозирование и т.д.;
- сельское хозяйство – прогнозирование урожая, борьба 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, военно-промышленный комплекс 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, дефектоскопии (металлопрокат, деревообработка и пр.), диагностики сложных технических систем (автомобилестроение, кораблестроение, авиастроение), прогнозирования экономических и 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, независимо 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.) 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 Intelligence (BI) или AaaS – Analytics as a Service – ушли 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 как инструментария настроек к платформам, развёртываемым у клиентов.

Data Mining 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 Data Mining. 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.

42. Сравнительная характеристика методов “деревья решений” и “искусственные нейронные сети”

Сравнительная таблица

Критерий	Деревья решений	Искусственные нейронные сети
Принцип работы	Последовательные логические разветвления	Распределённая нелинейная обработка через слои
Устойчивость к шуму	Низкая (может переобучаться)	Может быть выше при регуляризации
Работа с пропущенными данными	Может справляться	Обычно требует предварительной обработки
Интерпретируемость	Полная – можно проследить путь принятия решения	Низкая – «чёрный ящик», скрытые слои
Скорость обучения	Высокая	Долгое обучение, особенно при большом числе параметров
Требования к объёму данных	Низкие	Высокие
Склонность к переобучению	Высокая (контролируется глубиной)	Есть, но смягчается регуляризацией
Масштабируемость	Ограниченная	Высокая
Области применения	Простые задачи, табличные данные	Сложные задачи, неструктурированные данные (изображения, текст)

Деревья решений и искусственные нейронные сети — это два фундаментальных подхода, представляющих логическую и статистико-биологическую парадигмы машинного обучения. Первый метод более прост и прозрачен, подходит для интерпретируемых моделей и табличных данных. Второй — более мощный и универсальный, позволяющий решать задачи любой сложности, особенно на больших и неструктурированных данных, но при этом требует больших вычислительных ресурсов, и объяснить его поведение гораздо труднее. Грамотный выбор между этими методами должен основываться на характере задачи, объёме и типе данных, требуемой интерпретируемости и вычислительных возможностях.

★★★

Дерево решений

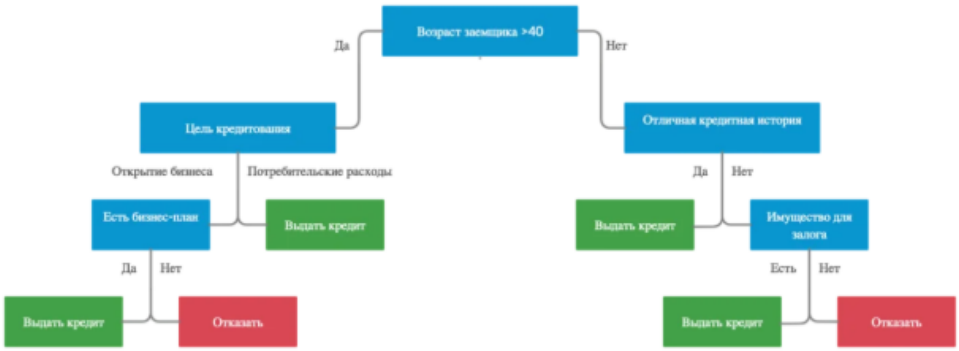
Дерево решений — это иерархическая структура, в которой каждый внутренний узел представляет собой логическое условие по одному из признаков, а листья — итоговые предсказания (например, классы). Алгоритм рекурсивно разбирает выборку, стремясь максимизировать «чистоту» классов в каждом подмножестве.
Представляет собой древовидную структуру «если …, то …», в которой правила генерируются автоматически в процессе обучения на основе обучающего множества.

Как работает алгоритм

1. Выбирается наилучший признак для разделения данных (например, по приросту информации или индексу Джини).
2. Множество данных делится на подмножества в зависимости от значений признака.
3. Процесс рекурсивно повторяется для каждого поддерева.
4. Остановка происходит при достижении заданной глубины, отсутствии улучшения разбиения или если в узле остаётся один класс.

Свойства

- Не требует нормализации или масштабирования признаков.
- Подходит для категориальных и числовых данных.
- Может быть легко визуализирован.



Искусственная нейронная сеть

Искусственная нейронная сеть — это вычислительная модель, вдохновлённая работой биологических нейронов. Она состоит из входного слоя, одного или нескольких скрытых слоёв и выходного слоя. Каждый «нейрон» производит линейное преобразование входных данных с последующей нелинейной функцией активации. Обучение заключается в минимизации ошибки между прогнозом и фактическими значениями с помощью алгоритма обратного распространения ошибок и градиентного спуска.

Структура

- **Входной слой** принимает исходные данные.
- **Скрытые слои** выполняют вычисления и извлекают зависимости.
- **Выходной слой** формирует результат (классификация или регрессия).

Алгоритм обучения

1. Данные подаются на вход.
2. Сигналы проходят через слои нейронов.
3. На выходе сравниваются предсказания и истинные значения.
4. Вычисляется ошибка.
5. Применяется backpropagation и градиентный спуск — веса корректируются для минимизации ошибки.
6. Процесс повторяется много раз (эпох), пока не достигнется нужная точность.

Пример
Картинка → [множество нейронов на скрытых слоях] → Класс: «кошка»

Свойства

- Требует нормализации входных данных.
- Эффективно работает с изображениями, звуком, текстом.
- Позволяет моделировать сложные нелинейные зависимости.