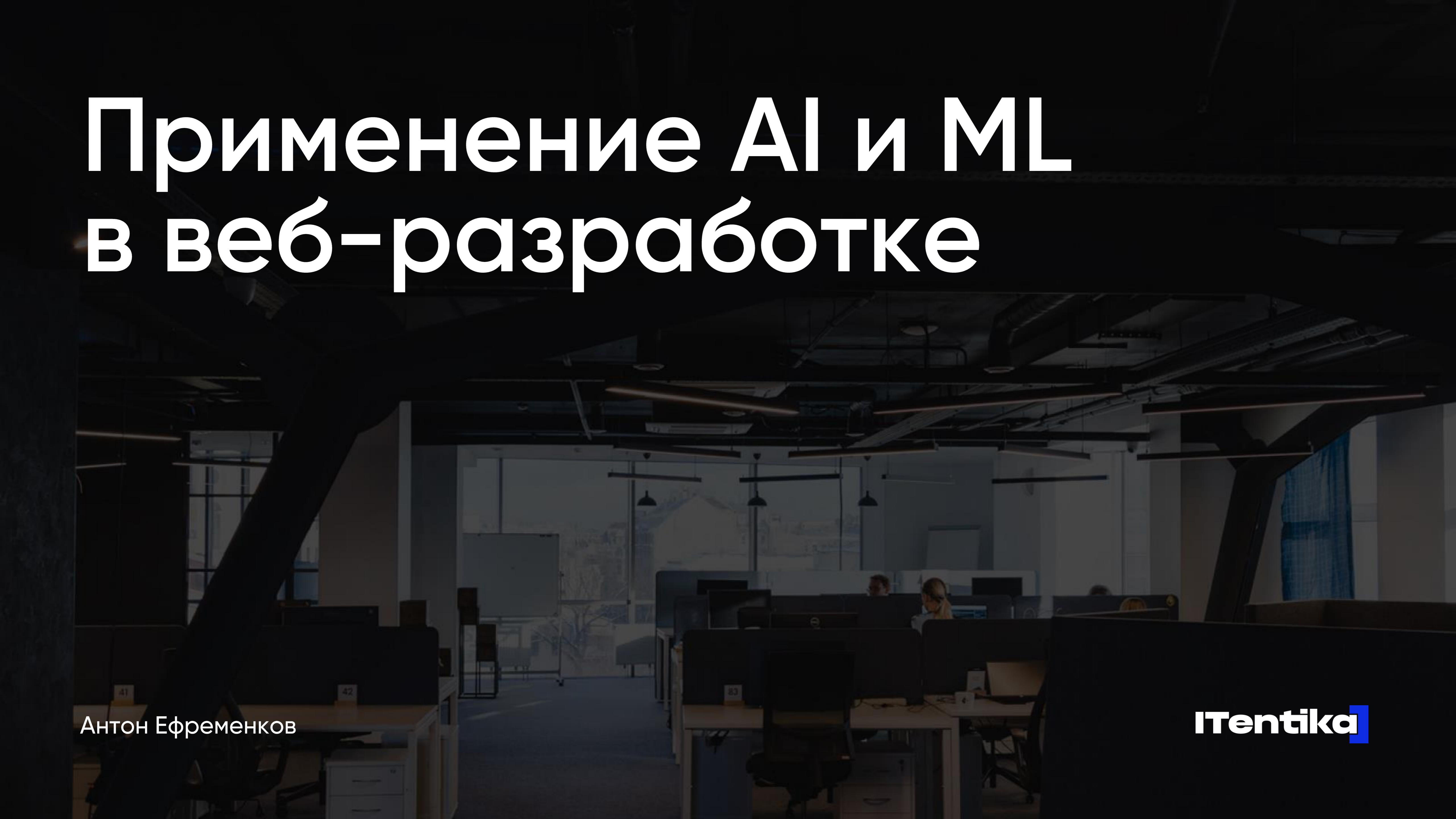


Применение AI и ML в веб-разработке

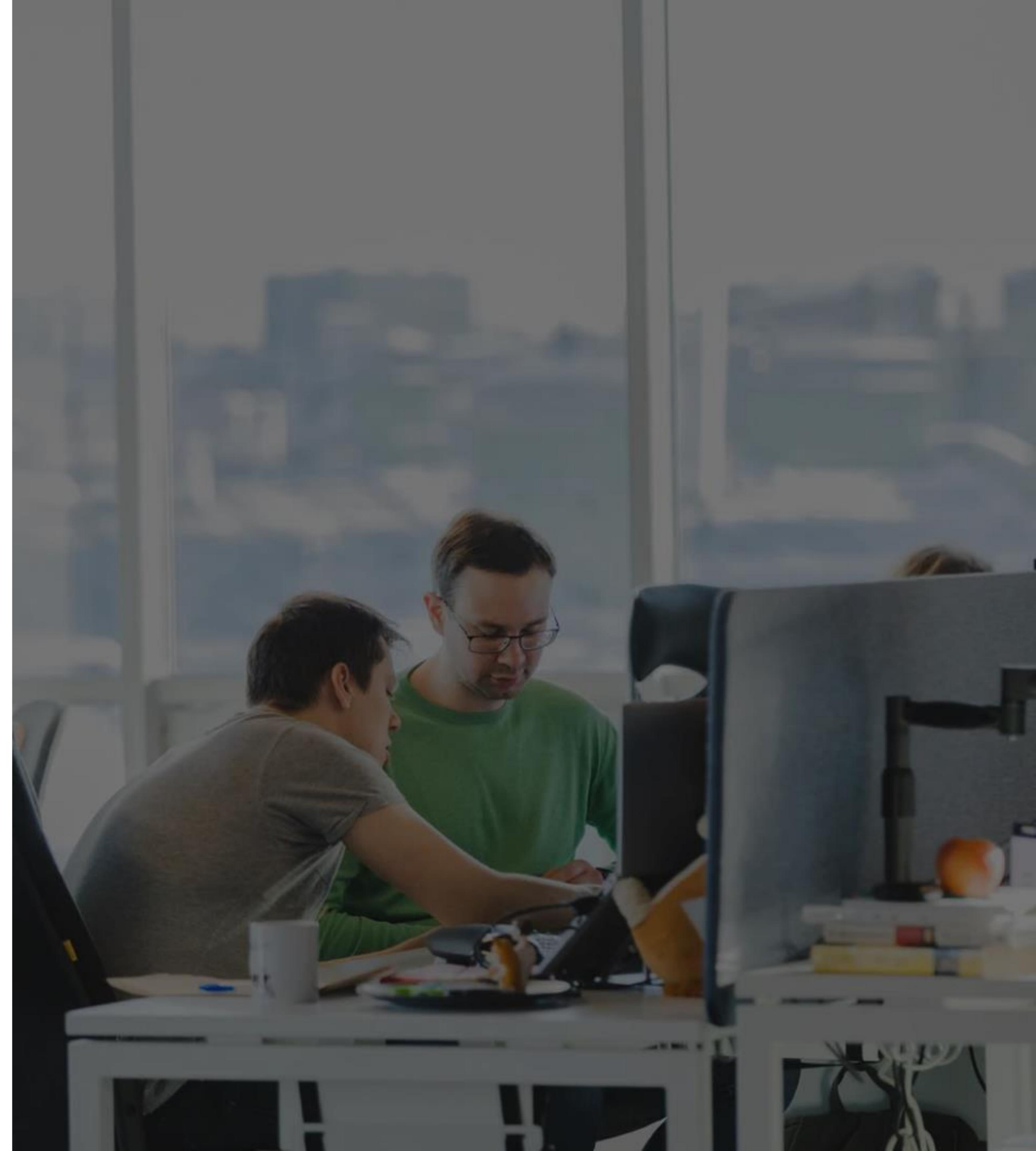


Антон Ефременков

ITentika

01

Немного теории о ML



Немного теории о ML

Machine Learning – это область искусственного интеллекта, которая изучает алгоритмы и модели, позволяющие компьютерам извлекать полезные знания из данных.



Немного теории о ML

Machine Learning – это область искусственного интеллекта, которая изучает алгоритмы и модели, позволяющие компьютерам извлекать полезные знания из данных.

Основные сферы применения:

Анализ данных

анализ настроений, прогнозирование продаж, рекомендательные системы, обнаружение мошенничества



Немного теории о ML

Machine Learning – это область искусственного интеллекта, которая изучает алгоритмы и модели, позволяющие компьютерам извлекать полезные знания из данных.

Основные сферы применения:

Анализ данных

анализ настроений, прогнозирование продаж, рекомендательные системы, обнаружение мошенничества

Компьютерное зрение

распознавание лиц, детекция объектов, реставрация изображений, понимание сцен



Немного теории о ML

Machine Learning – это область искусственного интеллекта, которая изучает алгоритмы и модели, позволяющие компьютерам извлекать полезные знания из данных.

Основные сферы применения:

Анализ данных

анализ настроений, прогнозирование продаж, рекомендательные системы, обнаружение мошенничества

Компьютерное зрение

распознавание лиц, детекция объектов, реставрация изображений, понимание сцен

Обработка естественного языка

чат-боты, автоматический перевод, анализ тональности



Немного теории о ML

Machine Learning – это область искусственного интеллекта, которая изучает алгоритмы и модели, позволяющие компьютерам извлекать полезные знания из данных.

Основные сферы применения:

Анализ данных

анализ настроений, прогнозирование продаж, рекомендательные системы, обнаружение мошенничества

Компьютерное зрение

распознавание лиц, детекция объектов, реставрация изображений, понимание сцен

Обработка естественного языка

чат-боты, автоматический перевод, анализ тональности

Медицинская диагностика

анализ медицинских изображений (например, МРТ/КТ), диагностика инфекционных заболеваний, прогнозирование исходов лечения



Немного теории о ML

Machine Learning – это область искусственного интеллекта, которая изучает алгоритмы и модели, позволяющие компьютерам извлекать полезные знания из данных.

Основные сферы применения:

Анализ данных

анализ настроений, прогнозирование продаж, рекомендательные системы, обнаружение мошенничества

Компьютерное зрение

распознавание лиц, детекция объектов, реставрация изображений, понимание сцен

Обработка естественного языка

чат-боты, автоматический перевод, анализ тональности

Медицинская диагностика

анализ медицинских изображений (например, МРТ/КТ), диагностика инфекционных заболеваний, прогнозирование исходов лечения

Финансы

оценка кредитного риска, прогнозирование фондового рынка, управление портфелем



Немного теории о ML

Machine Learning – это область искусственного интеллекта, которая изучает алгоритмы и модели, позволяющие компьютерам извлекать полезные знания из данных.

Основные сферы применения:

Анализ данных

анализ настроений, прогнозирование продаж, рекомендательные системы, обнаружение мошенничества

Компьютерное зрение

распознавание лиц, детекция объектов, реставрация изображений, понимание сцен

Обработка естественного языка

чат-боты, автоматический перевод, анализ тональности

Медицинская диагностика

анализ медицинских изображений (например, МРТ/КТ), диагностика инфекционных заболеваний, прогнозирование исходов лечения

Финансы

оценка кредитного риска, прогнозирование фондового рынка, управление портфелем

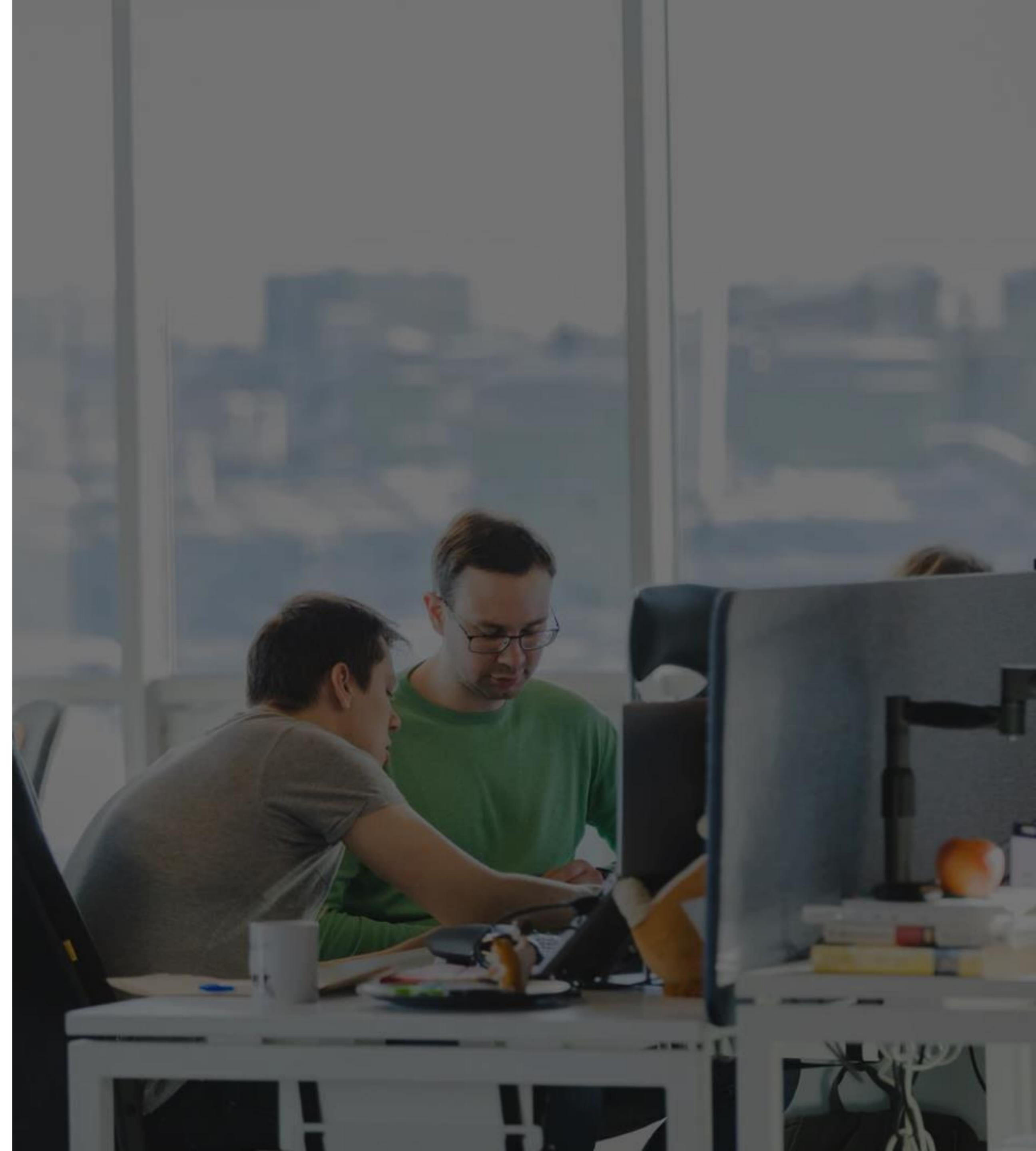
Автоматизация

Автоматическое управление складом, энергопотреблением, транспортом, производственными процессами



02

А как с этим связан JS?



А как с этим связан JS?

У нас есть целый арсенал специализированных библиотек для реализации алгоритмов машинного обучения

TensorFlow.js

Библиотека машинного обучения, созданная Google, позволяет разработчикам создавать и обучать нейронные сети в браузере или в Node.js.

Music Generation, Object Detection, Face Detection, Face Emotion Recognition, Speech Recognition, Image Classifier



А как с этим связан JS?

У нас есть целый арсенал специализированных библиотек для реализации алгоритмов машинного обучения

TensorFlow.js

Библиотека машинного обучения, созданная Google, позволяет разработчикам создавать и обучать нейронные сети в браузере или в Node.js.

Music Generation, Object Detection, Face Detection, Face Emotion Recognition, Speech Recognition, Image Classifier

Brain.js

Ещё одна популярная ML-библиотека, специализируется на нейронных сетях. Она предлагает простой интерфейс для создания и обучения нейронных сетей для различных задач.

Stock Market Prediction, Chatbot, Text Generation, Sentiment Analysis, Music Generation



А как с этим связан JS?

У нас есть целый арсенал специализированных библиотек для реализации алгоритмов машинного обучения

TensorFlow.js

Библиотека машинного обучения, созданная Google, позволяет разработчикам создавать и обучать нейронные сети в браузере или в Node.js.

Music Generation, Object Detection, Face Detection, Face Emotion Recognition, Speech Recognition, Image Classifier

Brain.js

Ещё одна популярная ML-библиотека, специализируется на нейронных сетях. Она предлагает простой интерфейс для создания и обучения нейронных сетей для различных задач.

Stock Market Prediction, Chatbot, Text Generation, Sentiment Analysis, Music Generation

OpenCV.js

Включает в себя коллекцию мощных инструментов и алгоритмов для обработки и анализа изображений и видео в браузере или в Node.js.



А как с этим связан JS?

У нас есть целый арсенал специализированных библиотек для реализации алгоритмов машинного обучения

TensorFlow.js

Библиотека машинного обучения, созданная Google, позволяет разработчикам создавать и обучать нейронные сети в браузере или в Node.js.

Music Generation, Object Detection, Face Detection, Face Emotion Recognition, Speech Recognition, Image Classifier

Brain.js

Ещё одна популярная ML-библиотека, специализируется на нейронных сетях. Она предлагает простой интерфейс для создания и обучения нейронных сетей для различных задач.

Stock Market Prediction, Chatbot, Text Generation, Sentiment Analysis, Music Generation

OpenCV.js

Включает в себя коллекцию мощных инструментов и алгоритмов для обработки и анализа изображений и видео в браузере или в Node.js.

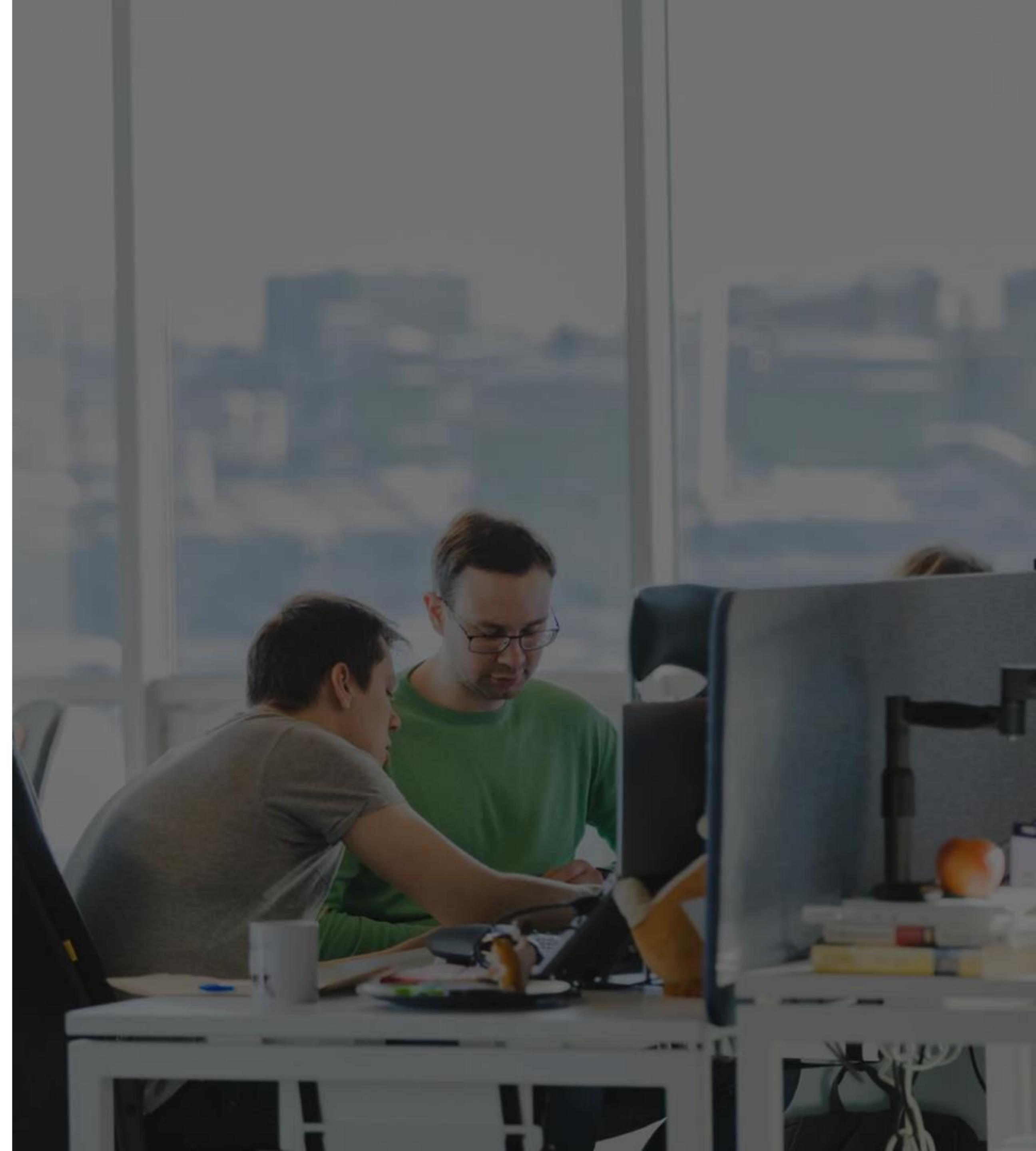
Neuro.js

Разработка AI-ассистентов и чат-ботов



03

А зачем это всё, если
есть Python?



А зачем это всё, если есть Python?

У ML на JS, безусловно, есть целый набор минусов по сравнению с ML на Python

Производительность

некоторые ML-библиотеки на JS используют WebGL для ускорения вычислений, они все равно могут уступать по производительности специализированным библиотекам на Python, (TensorFlow или PyTorch), особенно на больших объемах данных или сложных моделях



А зачем это всё, если есть Python?

У ML на JS, безусловно, есть целый набор минусов по сравнению с ML на Python

Производительность

некоторые ML-библиотеки на JS используют WebGL для ускорения вычислений, они все равно могут уступать по производительности специализированным библиотекам на Python, (TensorFlow или PyTorch), особенно на больших объемах данных или сложных моделях

Библиотеки и инструменты

Python имеет гораздо больше библиотек и инструментов для машинного обучения, чем JavaScript, что делает его более мощным и гибким для решения широкого спектра задач



А зачем это всё, если есть Python?

У ML на JS, безусловно, есть целый набор минусов по сравнению с ML на Python

Производительность

некоторые ML-библиотеки на JS используют WebGL для ускорения вычислений, они все равно могут уступать по производительности специализированным библиотекам на Python, (TensorFlow или PyTorch), особенно на больших объемах данных или сложных моделях

Библиотеки и инструменты

Python имеет гораздо больше библиотек и инструментов для машинного обучения, чем JavaScript, что делает его более мощным и гибким для решения широкого спектра задач

Объем поддержки

Python имеет более крупное и активное сообщество разработчиков, что обеспечивает большую поддержку и ресурсы для изучения и работы с машинным обучением



А зачем это всё, если есть Python?

У ML на JS, безусловно, есть целый набор минусов по сравнению с ML на Python

Производительность

некоторые ML-библиотеки на JS используют WebGL для ускорения вычислений, они все равно могут уступать по производительности специализированным библиотекам на Python, (TensorFlow или PyTorch), особенно на больших объемах данных или сложных моделях

Библиотеки и инструменты

Python имеет гораздо больше библиотек и инструментов для машинного обучения, чем JavaScript, что делает его более мощным и гибким для решения широкого спектра задач

Объем поддержки

Python имеет более крупное и активное сообщество разработчиков, что обеспечивает большую поддержку и ресурсы для изучения и работы с машинным обучением

Разработка и отладка

разработка и отладка ML-моделей на JavaScript могут быть более сложными по сравнению с Python из-за особенностей языка и инструментов.



А зачем это всё, если есть Python?

У ML на JS, безусловно, есть целый набор минусов по сравнению с ML на Python

Производительность

некоторые ML-библиотеки на JS используют WebGL для ускорения вычислений, они все равно могут уступать по производительности специализированным библиотекам на Python, (TensorFlow или PyTorch), особенно на больших объемах данных или сложных моделях

Разработка и отладка

разработка и отладка ML-моделей на JavaScript могут быть более сложными по сравнению с Python из-за особенностей языка и инструментов.

Библиотеки и инструменты

Python имеет гораздо больше библиотек и инструментов для машинного обучения, чем JavaScript, что делает его более мощным и гибким для решения широкого спектра задач

Ограниченност WebAssembly

Хотя WebAssembly позволяет запускать код на других языках в браузере, он все ещё не настолько популярный инструмент, и не все библиотеки и инструменты могут быть легко перенесены на JavaScript

Объем поддержки

Python имеет более крупное и активное сообщество разработчиков, что обеспечивает большую поддержку и ресурсы для изучения и работы с машинным обучением



А зачем это всё, если есть Python?

Но есть и целый набор плюсов!!

Портативность

JS – стандартный язык для веб-разработки.
Это значит, что машинное обучение на
JavaScript может быть легко внедрено в веб-
приложения, что делает его идеальным для
облачных сервисов, мобильных приложений
и веб-сайтов



А зачем это всё, если есть Python?

Но есть и целый набор плюсов!!

Портативность

JS – стандартный язык для веб-разработки. Это значит, что машинное обучение на JavaScript может быть легко внедрено в веб-приложения, что делает его идеальным для облачных сервисов, мобильных приложений и веб-сайтов

Удобство использования

Многие библиотеки машинного обучения на JavaScript предоставляют высокоуровневые абстракции, которые упрощают разработку и обучение моделей



А зачем это всё, если есть Python?

Но есть и целый набор плюсов!!

Портативность

JS – стандартный язык для веб-разработки. Это значит, что машинное обучение на JavaScript может быть легко внедрено в веб-приложения, что делает его идеальным для облачных сервисов, мобильных приложений и веб-сайтов

Удобство использования

Многие библиотеки машинного обучения на JavaScript предоставляют высокоуровневые абстракции, которые упрощают разработку и обучение моделей

Сообщество

JavaScript тоже имеет большое и активное сообщество разработчиков, что облегчает поиск помощи и ресурсов при работе с машинным обучением на этом языке



А зачем это всё, если есть Python?

Но есть и целый набор плюсов!!

Портативность

JS – стандартный язык для веб-разработки. Это значит, что машинное обучение на JavaScript может быть легко внедрено в веб-приложения, что делает его идеальным для облачных сервисов, мобильных приложений и веб-сайтов

Удобство использования

Многие библиотеки машинного обучения на JavaScript предоставляют высокоуровневые абстракции, которые упрощают разработку и обучение моделей

Сообщество

JavaScript тоже имеет большое и активное сообщество разработчиков, что облегчает поиск помощи и ресурсов при работе с машинным обучением на этом языке

Интерактивность

JS хорошо подходит для интерактивных приложений, что позволяет разработчикам создавать интерактивные интерфейсы для ML, позволяющие пользователям взаимодействовать с моделями в реальном времени.



А зачем это всё, если есть Python?

Но есть и целый набор плюсов!!

Портативность

JS – стандартный язык для веб-разработки. Это значит, что машинное обучение на JavaScript может быть легко внедрено в веб-приложения, что делает его идеальным для облачных сервисов, мобильных приложений и веб-сайтов

Интерактивность

JS хорошо подходит для интерактивных приложений, что позволяет разработчикам создавать интерактивные интерфейсы для ML, позволяющие пользователям взаимодействовать с моделями в реальном времени.

Удобство использования

Многие библиотеки машинного обучения на JavaScript предоставляют высокоуровневые абстракции, которые упрощают разработку и обучение моделей

Privacy

Если реализовывать ML в браузере, никакие данные клиента не будут уходить на сервер

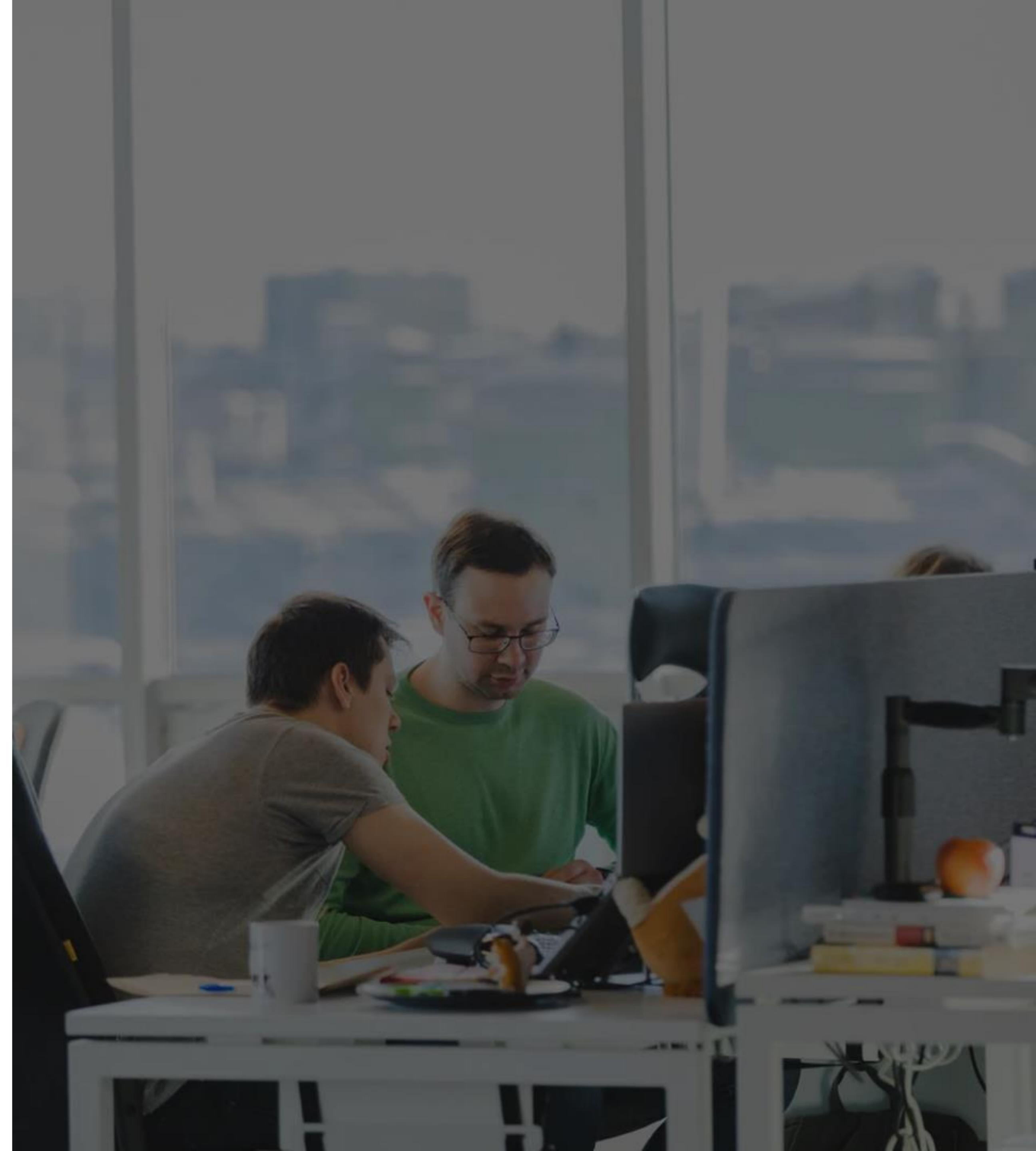
Сообщество

JavaScript тоже имеет большое и активное сообщество разработчиков, что облегчает поиск помощи и ресурсов при работе с машинным обучением на этом языке



04

Применение нейронных
сетей в веб-разработке



Обработка изображений на клиенте

Пожалуй, самый простой пример:
отражение изображения относительно
оси Y

```
import * as tf from '@tensorflow/tfjs';

// Get content image
let image = new Image(512, 512);
image.src = exampleImage;

// Convert image to tensor and add batch dimension
let tfTensor = tf.browser.fromPixels(image);

const flippedTensor = tf.reverse(tfTensor, 1)

// Prepare rendering of the result
await tf.browser.toPixels(flippedTensor, canvas);
```

Обработка изображений на клиенте

Подготовка к работе с изображением

```
import * as tf from '@tensorflow/tfjs';

// Get content image
let image = new Image(512, 512);
image.src = exampleImage;

// Convert image to tensor and add batch dimension
let tfTensor = tf.browser.fromPixels(image);

const flippedTensor = tf.reverse(tfTensor, 1)

// Prepare rendering of the result
await tf.browser.toPixels(flippedTensor, canvas);
```



Обработка изображений на клиенте

Формирование тензора из изображения

```
import * as tf from '@tensorflow/tfjs';

// Get content image
let image = new Image(512, 512);
image.src = exampleImage;

// Convert image to tensor and add batch dimension
let tfTensor = tf.browser.fromPixels(image);

const flippedTensor = tf.reverse(tfTensor, 1)

// Prepare rendering of the result
await tf.browser.toPixels(flippedTensor, canvas);
```



Обработка изображений на клиенте

Отражение и отображение полученного изображения

```
import * as tf from '@tensorflow/tfjs';

// Get content image
let image = new Image(512, 512);
image.src = exampleImage;

// Convert image to tensor and add batch dimension
let tfTensor = tf.browser.fromPixels(image);

const flippedTensor = tf.reverse(tfTensor, 1)

// Prepare rendering of the result
await tf.browser.toPixels(flippedTensor, canvas);
```

Обработка изображений на клиенте

Результат



Обработка изображений на клиенте

Распознавание изображений – изи!*

* Если есть модель, конечно =)

```
import * as cocoSsd from "@tensorflow-models/coco-ssd";
import "@tensorflow/tfjs";

const handleImageUpload = async (e) => {
  const file = e.target.files[0];

  if (file) {
    const imageElement = imageRef.current;
    imageElement.src = URL.createObjectURL(file);
    setLoading(true);
    const model = await cocoSsd.load();
    const predictions = await model.detect(imageElement);
    setPredictions(predictions);
    setLoading(false);
  }
};
```



Обработка изображений на клиенте

Вся логика - тут

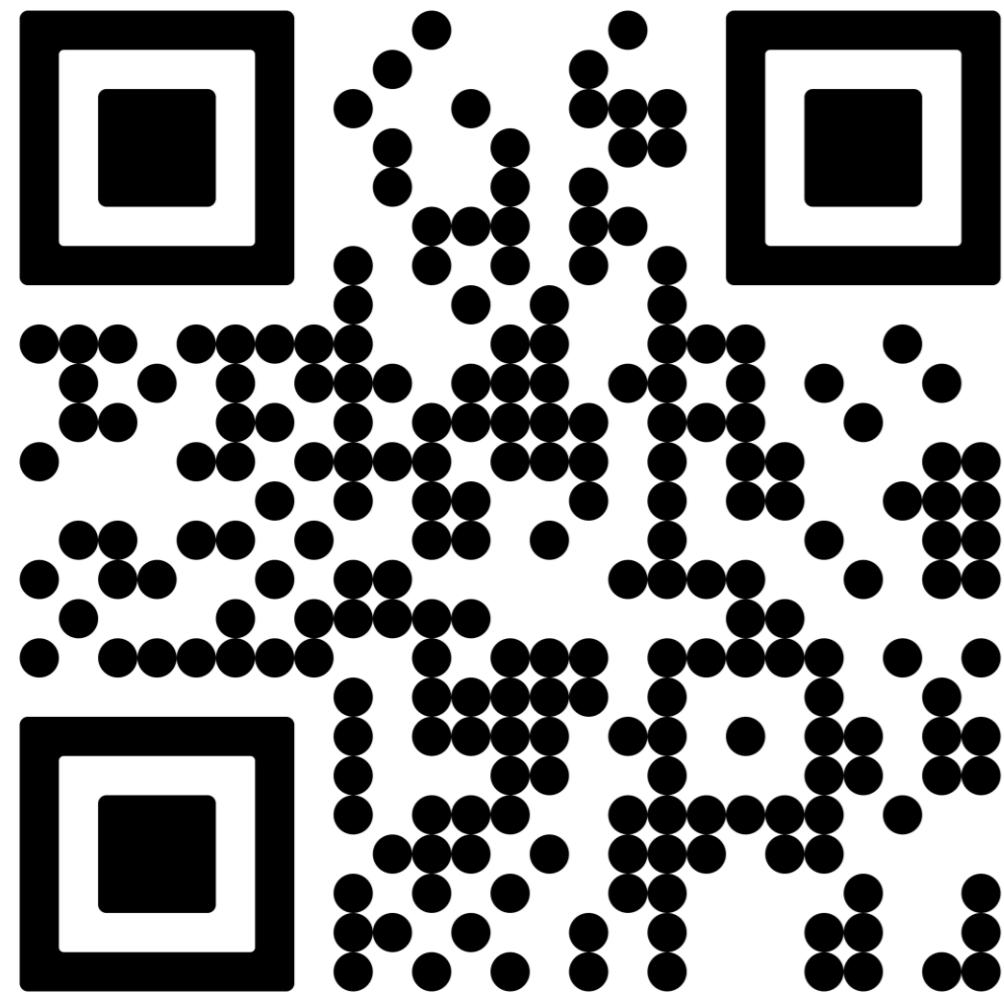
```
import * as cocoSsd from "@tensorflow-models/coco-ssd";
import "@tensorflow/tfjs";

const handleImageUpload = async (e) => {
    const file = e.target.files[0];

    if (file) {
        const imageElement = imageRef.current;
        imageElement.src = URL.createObjectURL(file);
        setLoading(true);
        const model = await cocoSsd.load();
        const predictions = await model.detect(imageElement);
        setPredictions(predictions);
        setLoading(false);
    }
};
```



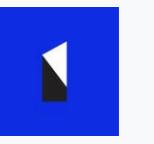
Обработка изображений на клиенте



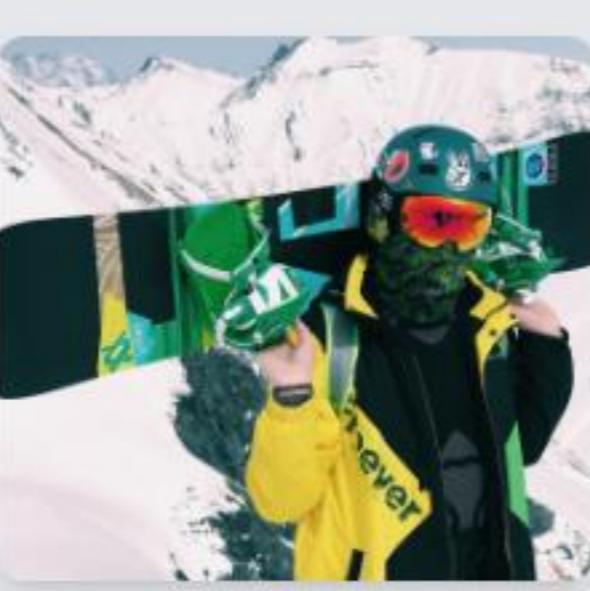
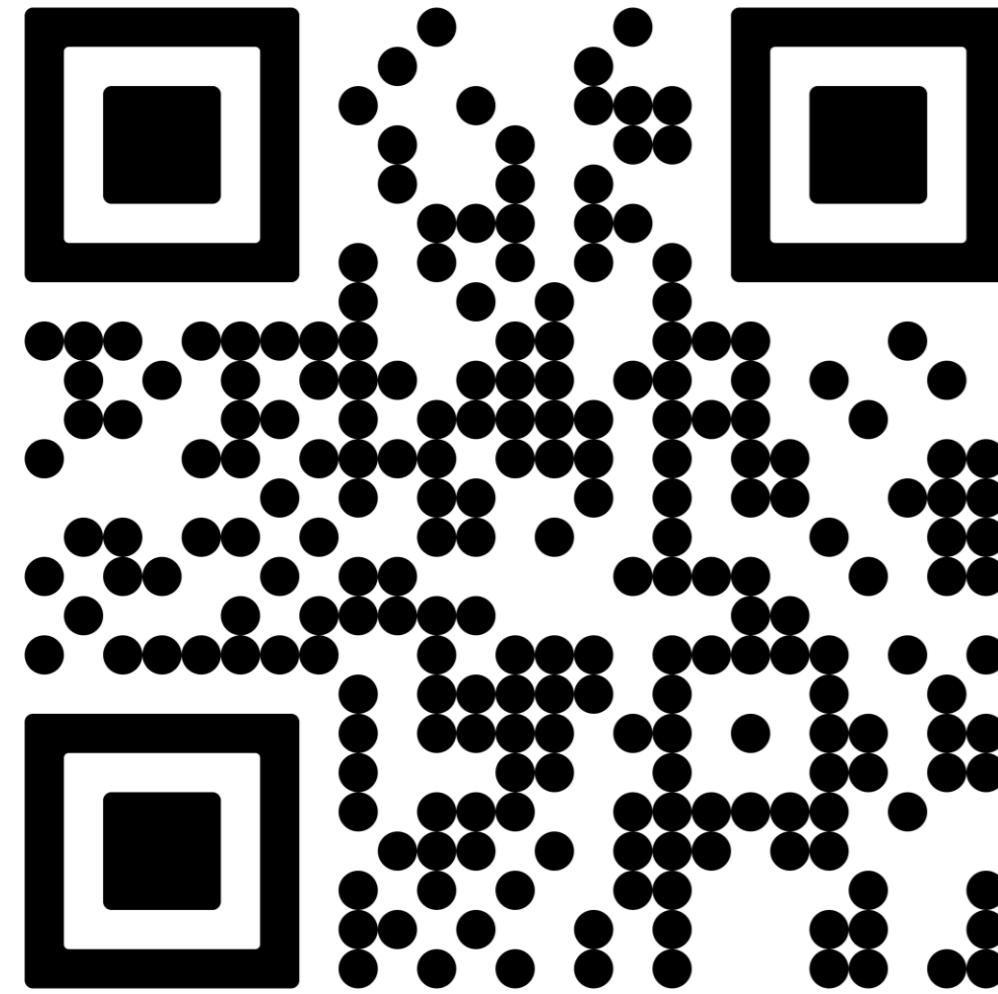


Predictions:

- snowboard - Confidence: 70.47%
- person - Confidence: 58.51%



Обработка изображений на клиенте



Predictions:

- snowboard - Confidence: 70.47%
- person - Confidence: 58.51%

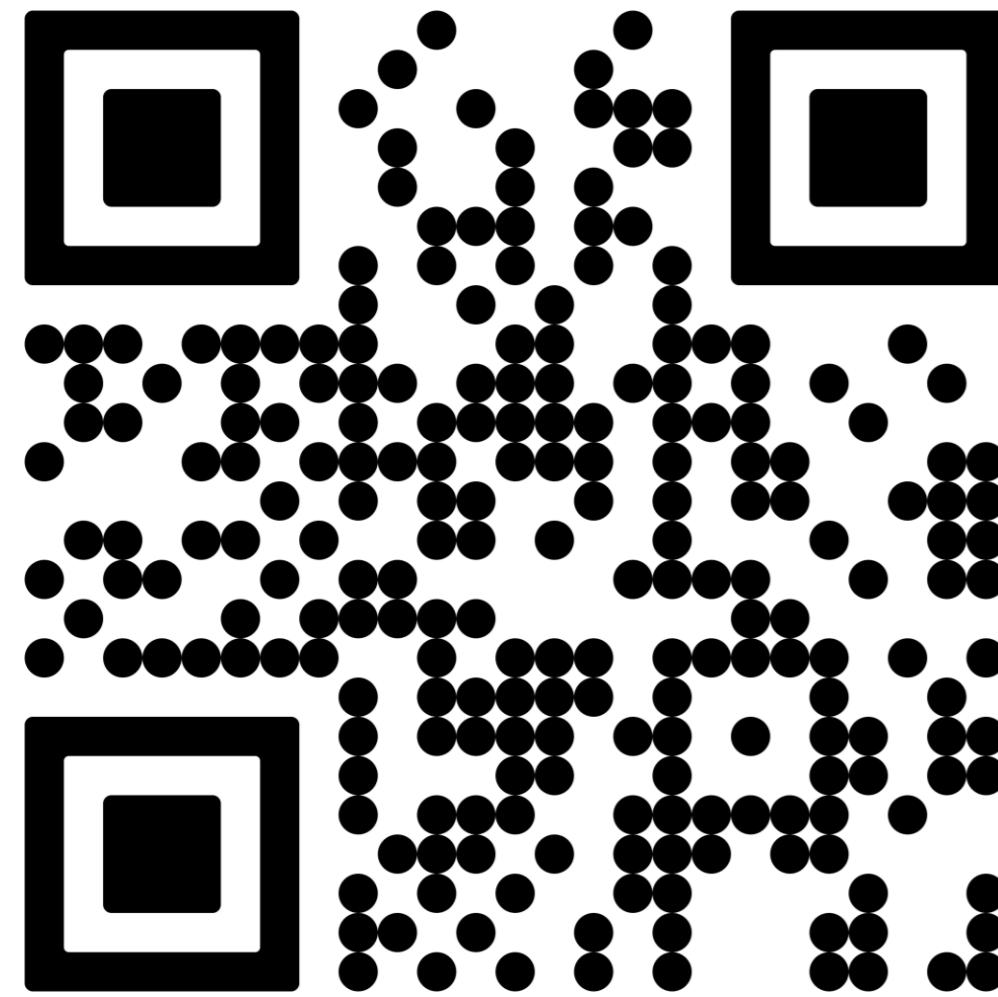


Predictions:

- bear - Confidence: 97.21%



Обработка изображений на клиенте



Predictions:

- snowboard - Confidence: 70.47%
- person - Confidence: 58.51%



Predictions:

- zebra - Confidence: 89.49%
- zebra - Confidence: 58.59%
- zebra - Confidence: 51.26%



Predictions:

- bear - Confidence: 97.21%

Анализ текста на клиенте

На примере анализа настроений

```
// Import the library
const brain = require('brain.js');

// Create a network
const net = new brain.recurrent.LSTM();

// Prepare the training data
const data = [
    { input: 'I love this library', output: 'positive' },
    { input: 'This is awesome', output: 'positive' },
    { input: 'I hate this library', output: 'negative' },
    { input: 'This is terrible', output: 'negative' },
    { input: 'This is okay', output: 'neutral' },
    { input: 'I don't care', output: 'neutral' },
];

// Train the network
net.train(data, {
    iterations: 200,
    log: true,
});

// Test the network
console.log(net.run('This is amazing')); // positive
console.log(net.run('This is boring')); // negative
console.log(net.run('This is fine')); // neutral
```



Анализ текста на клиенте

Импорт библиотеки, создание сети

```
// Import the library
const brain = require('brain.js');

// Create a network
const net = new brain.recurrent.LSTM();

// Prepare the training data
const data = [
    { input: 'I love this library', output: 'positive' },
    { input: 'This is awesome', output: 'positive' },
    { input: 'I hate this library', output: 'negative' },
    { input: 'This is terrible', output: 'negative' },
    { input: 'This is okay', output: 'neutral' },
    { input: 'I don't care', output: 'neutral' },
];

// Train the network
net.train(data, {
    iterations: 200,
    log: true,
});

// Test the network
console.log(net.run('This is amazing')); // positive
console.log(net.run('This is boring')); // negative
console.log(net.run('This is fine')); // neutral
```



Анализ текста на клиенте

Подготовка данных

```
// Import the library
const brain = require('brain.js');

// Create a network
const net = new brain.recurrent.LSTM();

// Prepare the training data
const data = [
    { input: 'I love this library', output: 'positive' },
    { input: 'This is awesome', output: 'positive' },
    { input: 'I hate this library', output: 'negative' },
    { input: 'This is terrible', output: 'negative' },
    { input: 'This is okay', output: 'neutral' },
    { input: 'I don't care', output: 'neutral' },
];

// Train the network
net.train(data, {
    iterations: 200,
    log: true,
});

// Test the network
console.log(net.run('This is amazing')) // positive
console.log(net.run('This is boring')) // negative
console.log(net.run('This is fine')) // neutral
```



Анализ текста на клиенте

Получение результатов

```
// Import the library
const brain = require('brain.js');

// Create a network
const net = new brain.recurrent.LSTM();

// Prepare the training data
const data = [
  { input: 'I love this library', output: 'positive' },
  { input: 'This is awesome', output: 'positive' },
  { input: 'I hate this library', output: 'negative' },
  { input: 'This is terrible', output: 'negative' },
  { input: 'This is okay', output: 'neutral' },
  { input: 'I don't care', output: 'neutral' },
];

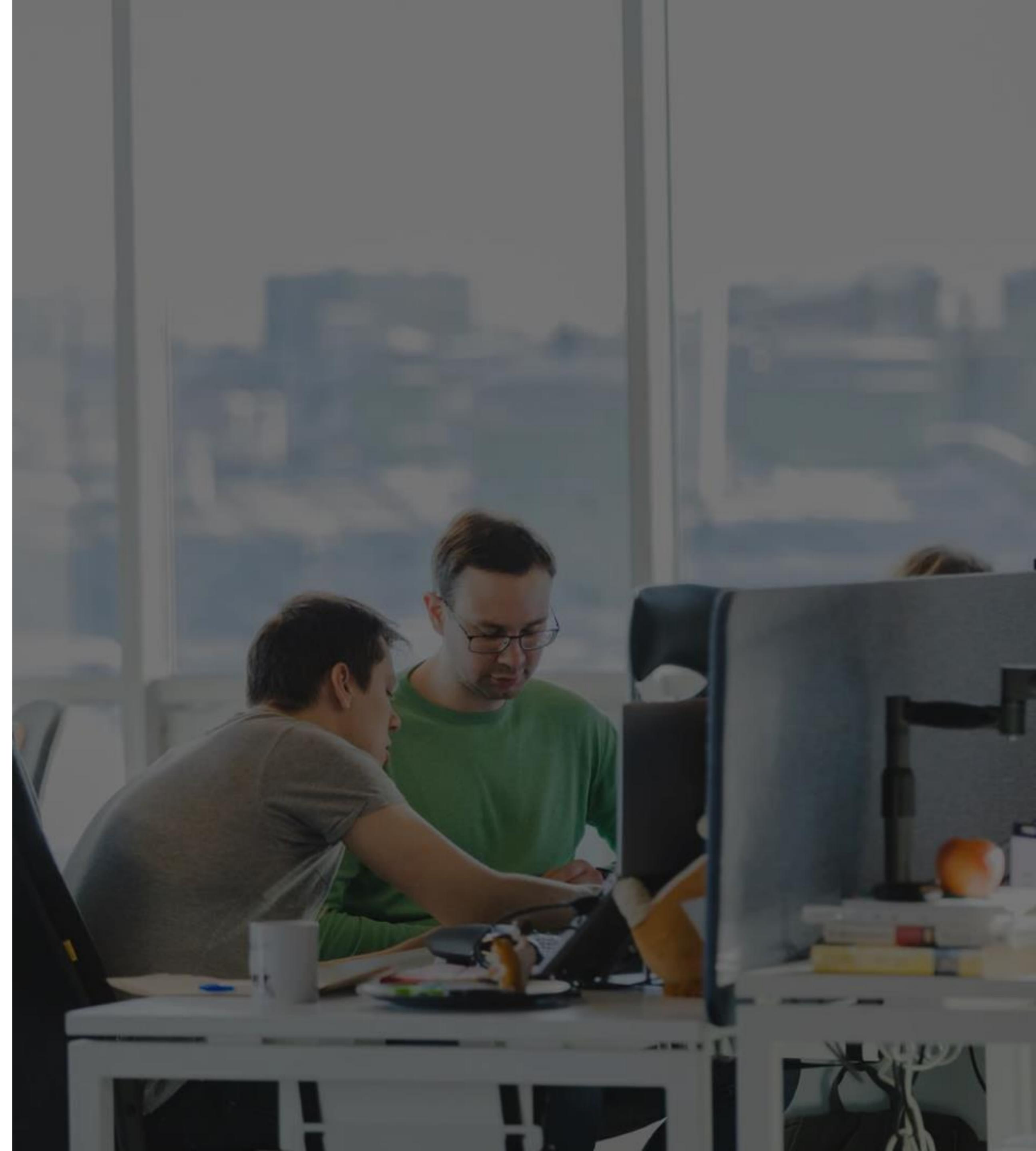
// Train the network
net.train(data, {
  iterations: 200,
  log: true,
});

// Test the network
console.log(net.run('This is amazing')); // positive
console.log(net.run('This is boring')); // negative
console.log(net.run('This is fine')); // neutral
```



05

Персонализированный
контент и рекомендации



Персонализированный контент и рекомендации

Создание алгоритмов рекомендаций на основе AI и данных пользователя может быть достаточно сложной задачей, но вот общий подход:



Персонализированный контент и рекомендации

Создание алгоритмов рекомендаций на основе AI и данных пользователя может быть достаточно сложной задачей, но вот общий подход:

Сбор данных

Сначала нужно собрать данные о взаимодействии пользователей с контентом.

Например: данные о просмотренных страницах, покупках, оценках, предпочтениях и т. д. Для этого может потребоваться создать БД или использовать существующие аналитические инструменты



Персонализированный контент и рекомендации

Создание алгоритмов рекомендаций на основе AI и данных пользователя может быть достаточно сложной задачей, но вот общий подход:

Сбор данных

Сначала нужно собрать данные о взаимодействии пользователей с контентом. Например: данные о просмотренных страницах, покупках, оценках, предпочтениях и т. д. Для этого может потребоваться создать БД или использовать существующие аналитические инструменты

Предварительная обработка данных

Перед тем как приступить к созданию алгоритмов, необходимо предварительно обработать собранные данные. Например: удалить выбросы, заполнить пропущенные значения, нормализовать данные и преобразовать в подходящий формат.



Персонализированный контент и рекомендации

Создание алгоритмов рекомендаций на основе AI и данных пользователя может быть достаточно сложной задачей, но вот общий подход:

Сбор данных

Сначала нужно собрать данные о взаимодействии пользователей с контентом. Например: данные о просмотренных страницах, покупках, оценках, предпочтениях и т. д. Для этого может потребоваться создать БД или использовать существующие аналитические инструменты

Предварительная обработка данных

Перед тем как приступить к созданию алгоритмов, необходимо предварительно обработать собранные данные. Например: удалить выбросы, заполнить пропущенные значения, нормализовать данные и преобразовать в подходящий формат.

Выбор модели рекомендаций

Есть несколько типов моделей рекомендаций (коллаборативная фильтрация, содержательная фильтрация, гибридные модели). Нужно выбрать модель, которая лучше всего подходит для имеющихся данных и целей



Персонализированный контент и рекомендации

Создание алгоритмов рекомендаций на основе AI и данных пользователя может быть достаточно сложной задачей, но вот общий подход:

Сбор данных

Сначала нужно собрать данные о взаимодействии пользователей с контентом. Например: данные о просмотренных страницах, покупках, оценках, предпочтениях и т. д. Для этого может потребоваться создать БД или использовать существующие аналитические инструменты

Предварительная обработка данных

Перед тем как приступить к созданию алгоритмов, необходимо предварительно обработать собранные данные. Например: удалить выбросы, заполнить пропущенные значения, нормализовать данные и преобразовать в подходящий формат.

Выбор модели рекомендаций

Есть несколько типов моделей рекомендаций (коллаборативная фильтрация, содержательная фильтрация, гибридные модели). Нужно выбрать модель, которая лучше всего подходит для имеющихся данных и целей

Обучение модели

Время использовать выбранную модель для обучения на имеющихся данных. Это может включать в себя использование методов ML - например, классификацию, кластеризацию или регрессию



Персонализированный контент и рекомендации

Создание алгоритмов рекомендаций на основе AI и данных пользователя может быть достаточно сложной задачей, но вот общий подход:

Сбор данных

Сначала нужно собрать данные о взаимодействии пользователей с контентом. Например: данные о просмотренных страницах, покупках, оценках, предпочтениях и т. д. Для этого может потребоваться создать БД или использовать существующие аналитические инструменты

Предварительная обработка данных

Перед тем как приступить к созданию алгоритмов, необходимо предварительно обработать собранные данные. Например: удалить выбросы, заполнить пропущенные значения, нормализовать данные и преобразовать в подходящий формат.

Выбор модели рекомендаций

Есть несколько типов моделей рекомендаций (коллаборативная фильтрация, содержательная фильтрация, гибридные модели). Нужно выбрать модель, которая лучше всего подходит для имеющихся данных и целей

Обучение модели

Время использовать выбранную модель для обучения на имеющихся данных. Это может включать в себя использование методов ML - например, классификацию, кластеризацию или регрессию

Интеграция с JavaScript

После того, как модель обучена, нужно интегрировать ее с веб-сайтом или приложением посредством JS



Персонализированный контент и рекомендации

Создание алгоритмов рекомендаций на основе AI и данных пользователя может быть достаточно сложной задачей, но вот общий подход:

Сбор данных

Сначала нужно собрать данные о взаимодействии пользователей с контентом. Например: данные о просмотренных страницах, покупках, оценках, предпочтениях и т. д. Для этого может потребоваться создать БД или использовать существующие аналитические инструменты

Обучение модели

Время использовать выбранную модель для обучения на имеющихся данных. Это может включать в себя использование методов ML - например, классификацию, кластеризацию или регрессию

Предварительная обработка данных

Перед тем как приступить к созданию алгоритмов, необходимо предварительно обработать собранные данные. Например: удалить выбросы, заполнить пропущенные значения, нормализовать данные и преобразовать в подходящий формат.

Интеграция с JavaScript

После того, как модель обучена, нужно интегрировать ее с веб-сайтом или приложением посредством JS

Выбор модели рекомендаций

Есть несколько типов моделей рекомендаций (коллаборативная фильтрация, содержательная фильтрация, гибридные модели).

Нужно выбрать модель, которая лучше всего подходит для имеющихся данных и целей

Отображение рекомендаций

Использовать полученные рекомендации для отображения персонализированного контента. После внедрения алгоритмов рекомендаций важно оценить их эффективность и производительность



Пример создания алгоритма рекомендаций

Так может выглядеть начальная база данных

```
const users = [
    {
        id: 1,
        name: 'Сара',
        ratings: {
            'Интерстеллар': 5,
            'Оппенгеймер': 4,
            'Железный человек': 5, // и так далее
        }
    },
    {
        id: 2,
        name: 'Ванесса',
        ratings: {
            'Интерстеллар': 4,
            'Оппенгеймер': 5,
            'Железный человек': 3, // и так далее
        }
    },
];
```



Пример создания алгоритма рекомендаций

Функция для предсказания оценки пользователя для фильма, использующая колаборативную фильтрацию на основе сходства между пользователями.

В данном примере используется **простая** метрика сходства, а именно косинусное сходство.

```
function predictRating(userId, movieTitle) {
  const currentUser = users.find(user => user.id === userId);
  const currentUserRatings = Object.values(currentUser.ratings);

  const otherUsers = users.filter(user => user.id !== userId);

  let [numerator, denominator1, denominator2] = [0, 0, 0];

  otherUsers.forEach(user => {
    const otherUserRatings = Object.values(user.ratings);
    const commonMovies = Object.keys(currentUser.ratings)
      .filter(movie => user.ratings.hasOwnProperty(movie));

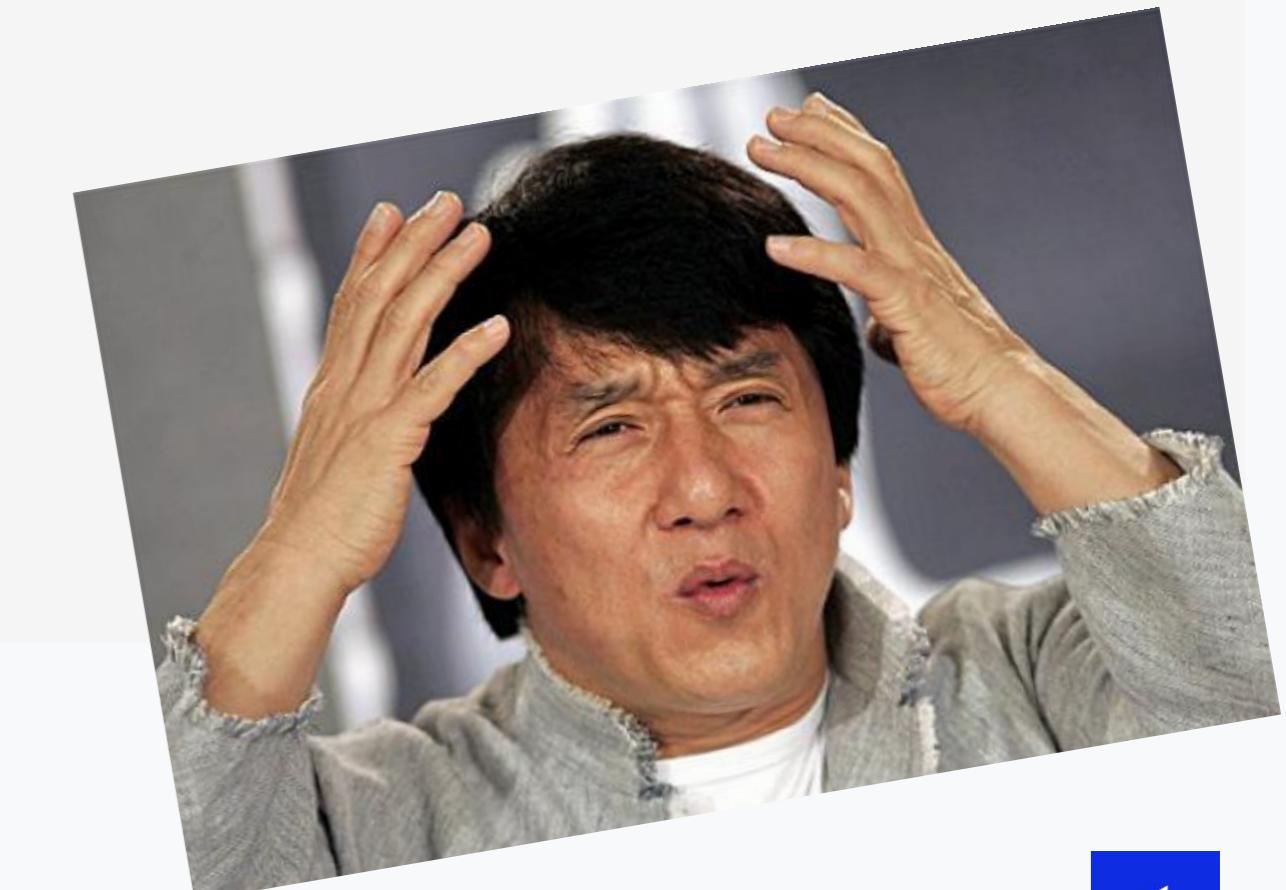
    const currentUserCommonRatings = commonMovies.map(movie => currentUser.ratings[movie]);
    const otherUserCommonRatings = commonMovies.map(movie => user.ratings[movie]);

    const dotProduct = currentUserCommonRatings.reduce((acc, rating, index) => {
      return acc + rating * otherUserCommonRatings[index];
    }, 0);

    const normCurrentUser = Math.sqrt(currentUserCommonRatings.reduce((acc, rating) => acc + rating ** 2, 0));
    const normOtherUser = Math.sqrt(otherUserCommonRatings.reduce((acc, rating) => acc + rating ** 2, 0));
    numerator += dotProduct;
    denominator1 += normCurrentUser;
    denominator2 += normOtherUser;
  });

  const similarity = numerator / (denominator1 * denominator2);
  const userRatingForMovie = currentUser.ratings[movieTitle];
  const predictedRating = similarity * userRatingForMovie;

  return predictedRating;
}
```



Пример создания алгоритма рекомендаций

- Базовый пример

```
function predictRating(userId, movieTitle) {
  const currentUser = users.find(user => user.id === userId);
  const currentUserRatings = Object.values(currentUser.ratings);

  const otherUsers = users.filter(user => user.id !== userId);

  let [numerator, denominator1, denominator2] = [0, 0, 0];

  otherUsers.forEach(user => {
    const otherUserRatings = Object.values(user.ratings);
    const commonMovies = Object.keys(currentUser.ratings)
      .filter(movie => user.ratings.hasOwnProperty(movie));

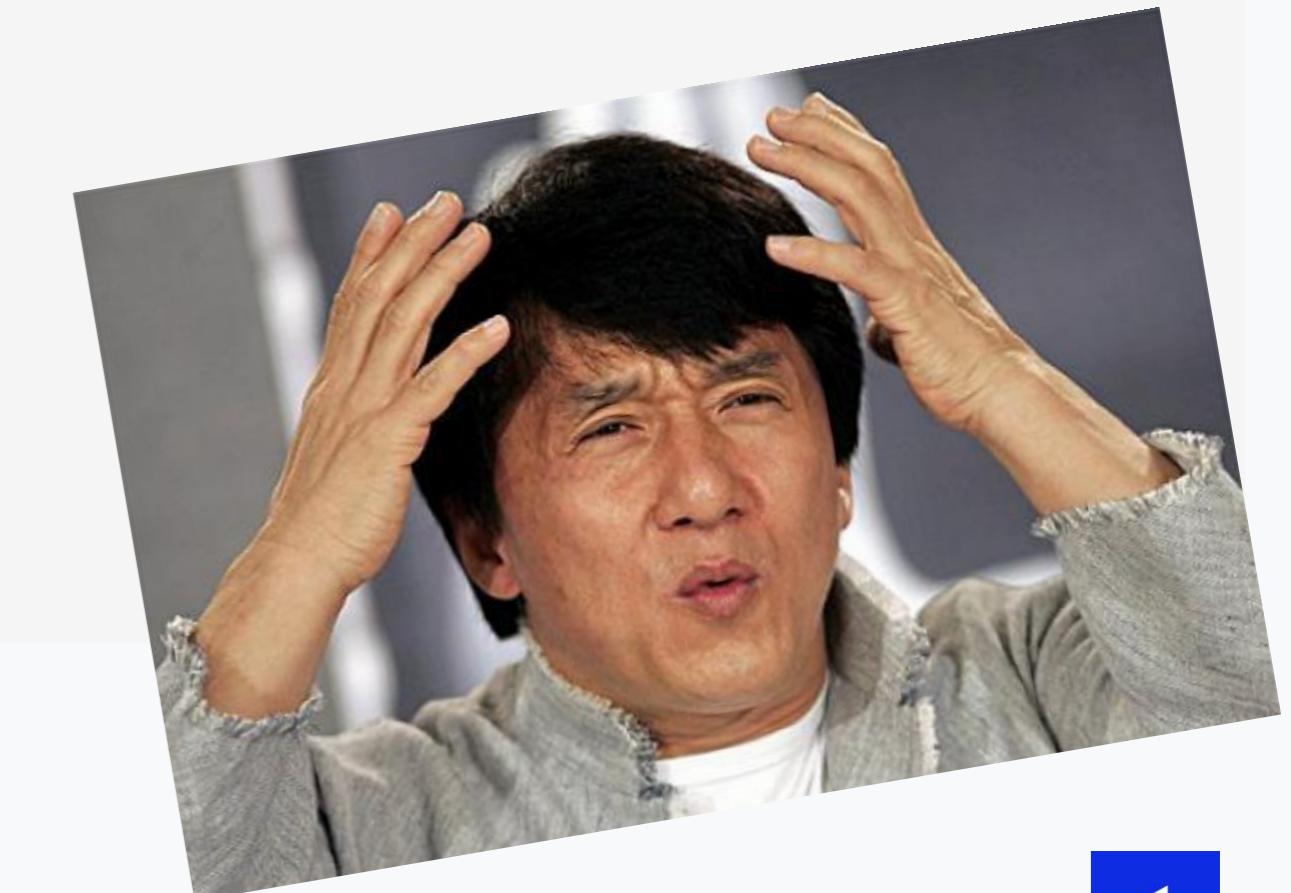
    const currentUserCommonRatings = commonMovies.map(movie => currentUser.ratings[movie]);
    const otherUserCommonRatings = commonMovies.map(movie => user.ratings[movie]);

    const dotProduct = currentUserCommonRatings.reduce((acc, rating, index) => {
      return acc + rating * otherUserCommonRatings[index];
    }, 0);

    const normCurrentUser = Math.sqrt(currentUserCommonRatings.reduce((acc, rating) => acc + rating ** 2, 0));
    const normOtherUser = Math.sqrt(otherUserCommonRatings.reduce((acc, rating) => acc + rating ** 2, 0));
    numerator += dotProduct;
    denominator1 += normCurrentUser;
    denominator2 += normOtherUser;
  });

  const similarity = numerator / (denominator1 * denominator2);
  const userRatingForMovie = currentUser.ratings[movieTitle];
  const predictedRating = similarity * userRatingForMovie;

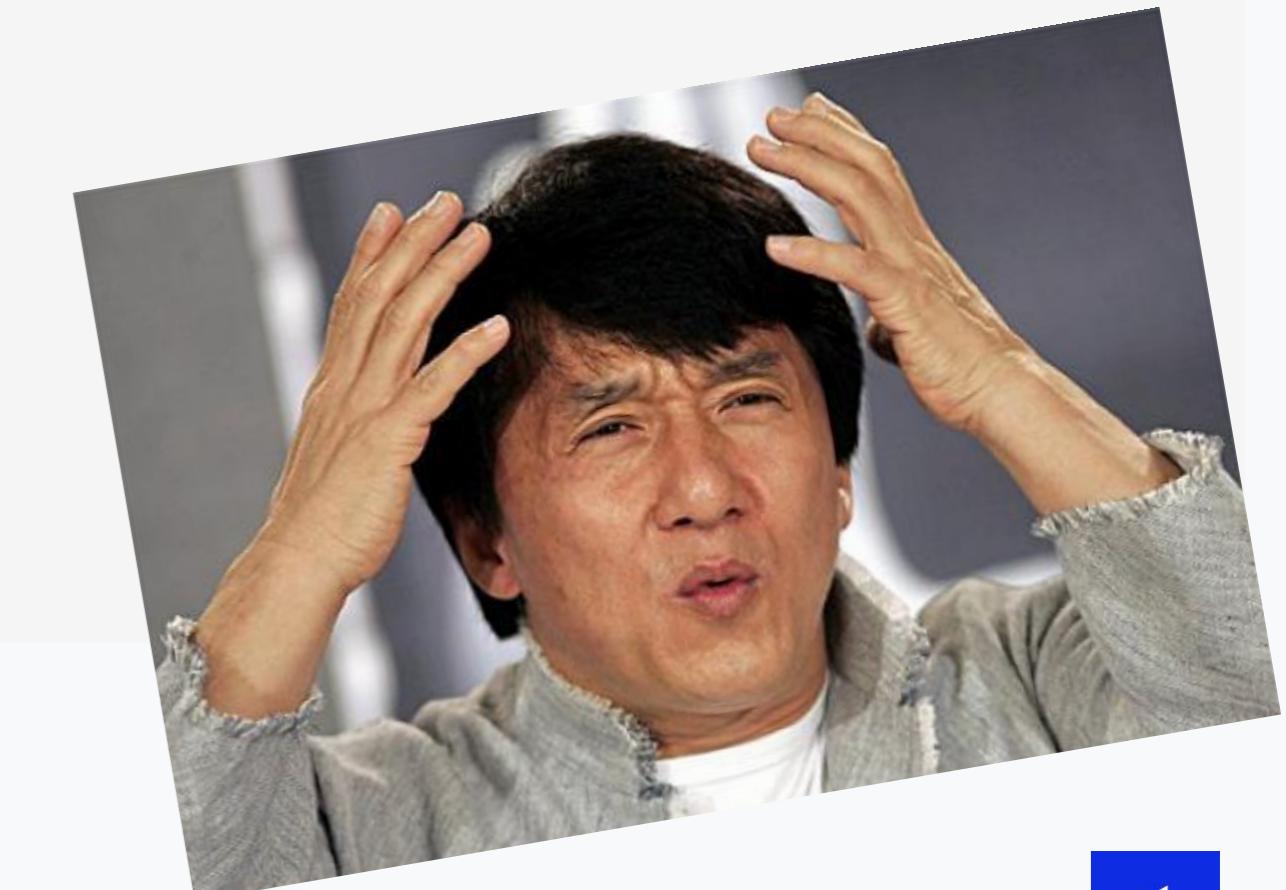
  return predictedRating;
}
```



Пример создания алгоритма рекомендаций

- Базовый пример
- Не учитывает многие факторы (нормализация рейтингов, обработка пропущенных значений и др.)

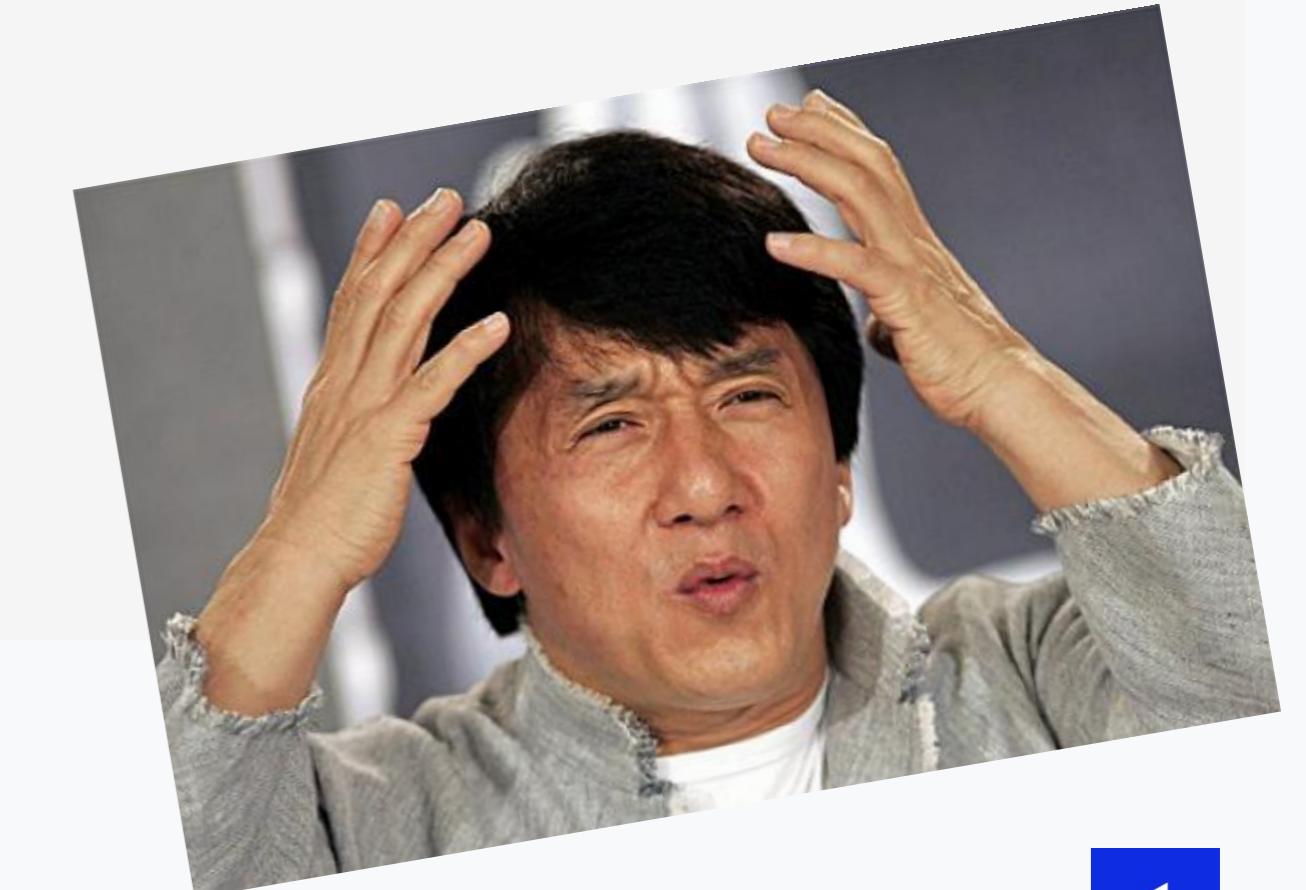
```
function predictRating(userId, movieTitle) {  
    const currentUser = users.find(user => user.id === userId);  
    const currentUserRatings = Object.values(currentUser.ratings);  
  
    const otherUsers = users.filter(user => user.id !== userId);  
  
    let [numerator, denominator1, denominator2] = [0, 0, 0];  
  
    otherUsers.forEach(user => {  
        const otherUserRatings = Object.values(user.ratings);  
        const commonMovies = Object.keys(currentUser.ratings)  
            .filter(movie => user.ratings.hasOwnProperty(movie));  
  
        const currentUserCommonRatings = commonMovies.map(movie => currentUser.ratings[movie]);  
        const otherUserCommonRatings = commonMovies.map(movie => user.ratings[movie]);  
  
        const dotProduct = currentUserCommonRatings.reduce((acc, rating, index) => {  
            return acc + rating * otherUserCommonRatings[index];  
        }, 0);  
  
        const normCurrentUser = Math.sqrt(currentUserCommonRatings.reduce((acc, rating) => acc + rating ** 2, 0));  
        const normOtherUser = Math.sqrt(otherUserCommonRatings.reduce((acc, rating) => acc + rating ** 2, 0));  
        numerator += dotProduct;  
        denominator1 += normCurrentUser;  
        denominator2 += normOtherUser;  
    });  
  
    const similarity = numerator / (denominator1 * denominator2);  
    const userRatingForMovie = currentUser.ratings[movieTitle];  
    const predictedRating = similarity * userRatingForMovie;  
  
    return predictedRating;  
}
```



Пример создания алгоритма рекомендаций

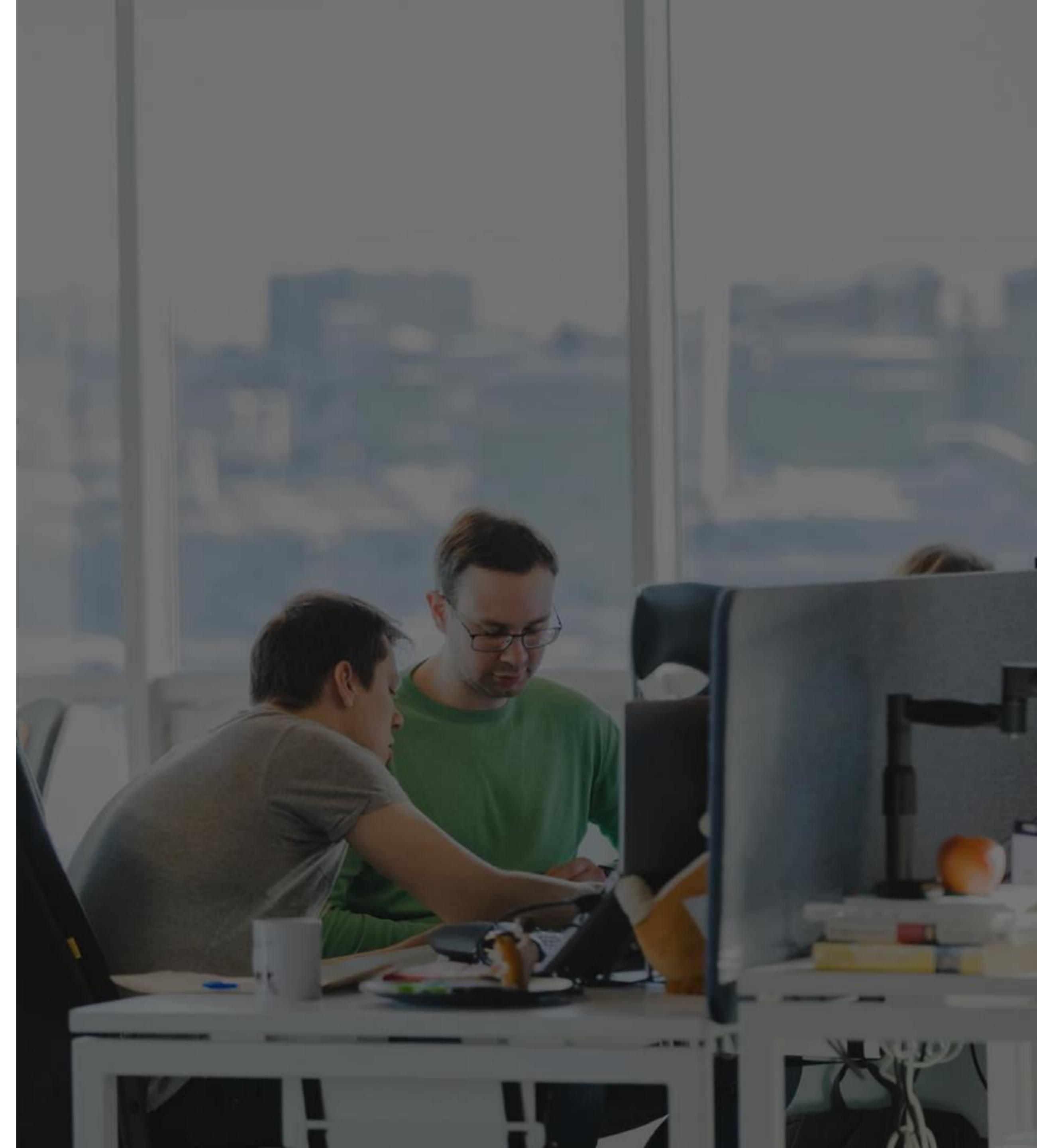
- Базовый пример
- Не учитывает многие факторы (нормализация рейтингов, обработка пропущенных значений и др.)
- Демонстрирует общий подход к созданию алгоритмов рекомендаций на основе JavaScript и данных пользователя

```
function predictRating(userId, movieTitle) {  
    const currentUser = users.find(user => user.id === userId);  
    const currentUserRatings = Object.values(currentUser.ratings);  
  
    const otherUsers = users.filter(user => user.id !== userId);  
  
    let [numerator, denominator1, denominator2] = [0, 0, 0];  
  
    otherUsers.forEach(user => {  
        const otherUserRatings = Object.values(user.ratings);  
        const commonMovies = Object.keys(currentUser.ratings)  
            .filter(movie => user.ratings.hasOwnProperty(movie));  
  
        const currentUserCommonRatings = commonMovies.map(movie => currentUser.ratings[movie]);  
        const otherUserCommonRatings = commonMovies.map(movie => user.ratings[movie]);  
  
        const dotProduct = currentUserCommonRatings.reduce((acc, rating, index) => {  
            return acc + rating * otherUserCommonRatings[index];  
        }, 0);  
  
        const normCurrentUser = Math.sqrt(currentUserCommonRatings.reduce((acc, rating) => acc + rating ** 2, 0));  
        const normOtherUser = Math.sqrt(otherUserCommonRatings.reduce((acc, rating) => acc + rating ** 2, 0));  
        numerator += dotProduct;  
        denominator1 += normCurrentUser;  
        denominator2 += normOtherUser;  
    });  
  
    const similarity = numerator / (denominator1 * denominator2);  
    const userRatingForMovie = currentUser.ratings[movieTitle];  
    const predictedRating = similarity * userRatingForMovie;  
  
    return predictedRating;  
}
```



06

Анализ данных и визуализация



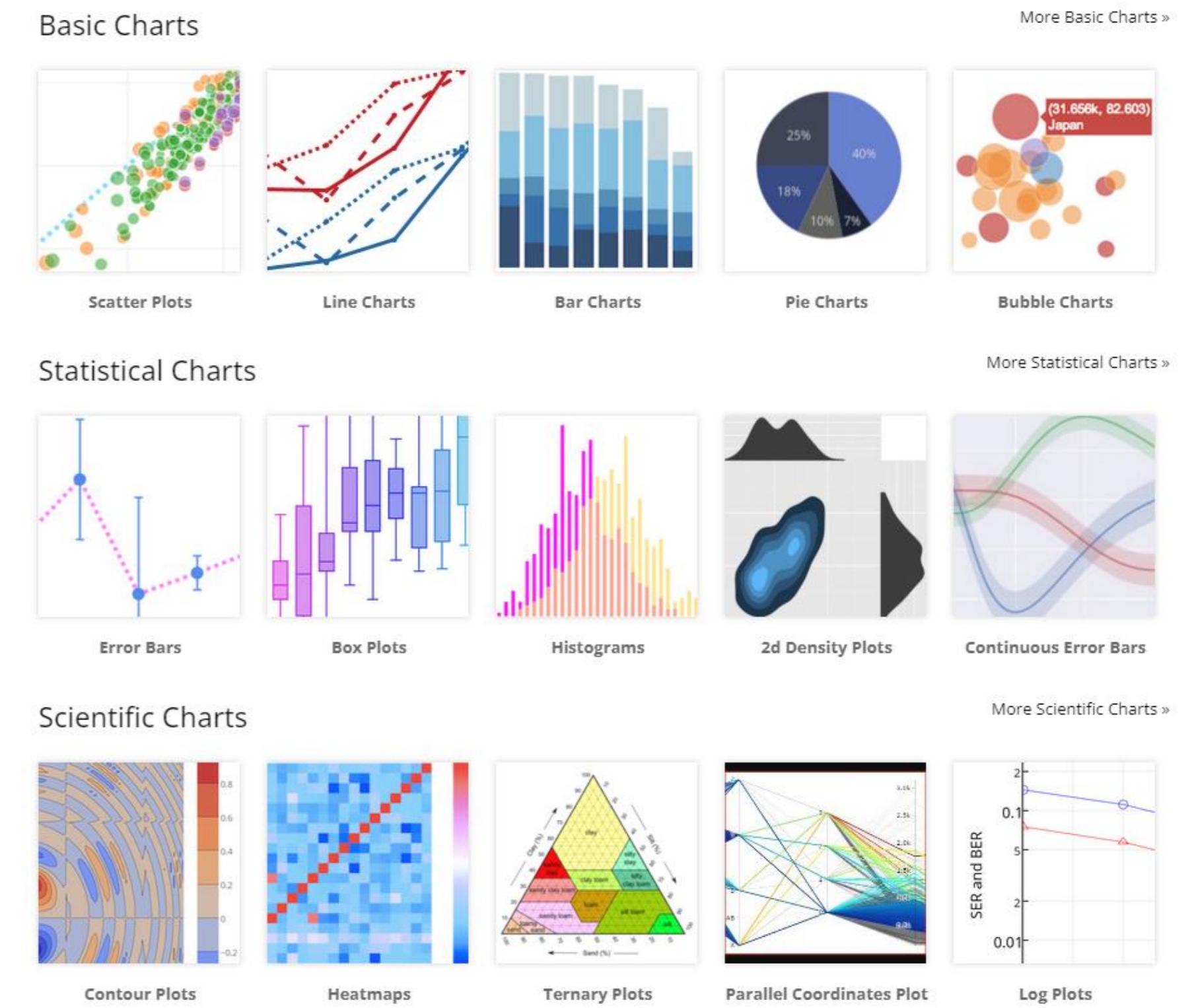
Анализ данных и визуализация

Наиболее часто задачи визуализации данных решаются с помощью d3.js и Plotly.js

d3.js

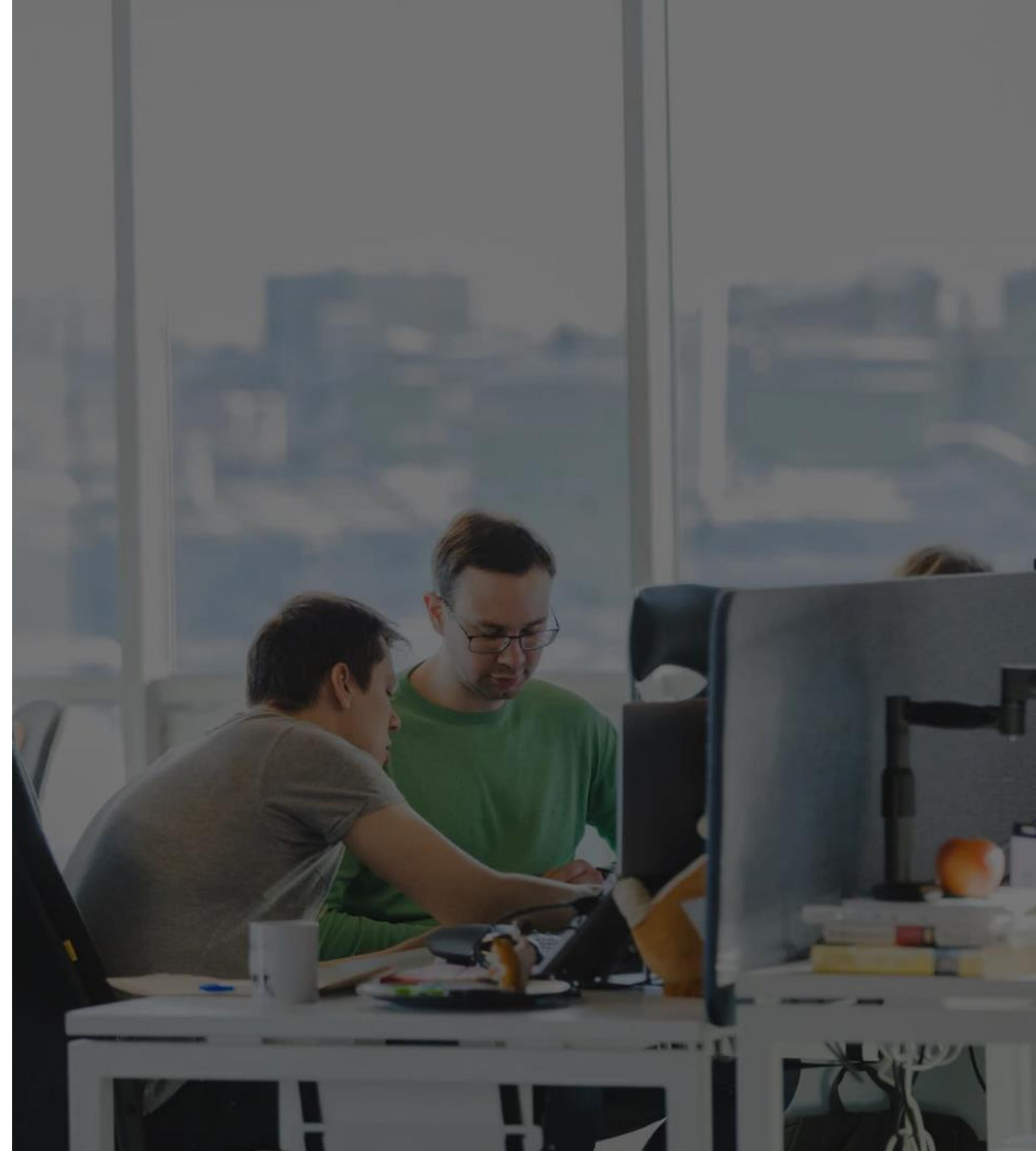


Plotly.js



07

Будущее машинного обучения в веб-разработке



Будущее машинного обучения в веб-разработке

Оно точно есть!)

- Генерация вёрстки по макету и когда уже нас заменят?!



Будущее машинного обучения в веб-разработке

Оно точно есть!)

- Генерация вёрстки по макету и когда уже нас заменят?!
- Анализ уязвимостей на сайтах



Будущее машинного обучения в веб-разработке

Оно точно есть!)

- Генерация вёрстки по макету и когда уже нас заменят?!
- Анализ уязвимостей на сайтах
- Copilot для private-проекта (когда никакой код не утекает на бэк)



Будущее машинного обучения в веб-разработке

Оно точно есть!)

- Генерация вёрстки по макету и когда уже нас заменят?!
- Анализ уязвимостей на сайтах
- Copilot для private-проекта (когда никакой код не утекает на бэк)
- Автогенерация кода интеграции по документации



Будущее машинного обучения в веб-разработке

Оно точно есть!)

- Генерация вёрстки по макету и когда уже нас заменят?!
- Анализ уязвимостей на сайтах
- Copilot для private-проекта (когда никакой код не утекает на бэк)
- Автогенерация кода интеграции по документации
- Анализ логов



Будущее машинного обучения в веб-разработке

Оно точно есть!)

- Генерация вёрстки по макету и когда уже нас заменят?!
- Анализ уязвимостей на сайтах
- Copilot для private-проекта (когда никакой код не утекает на бэк)
- Автогенерация кода интеграции по документации
- Анализ логов
- Обнаружение «поехавшей» вёрстки



Спасибо за внимание!

Вы супер! =)

ITentika