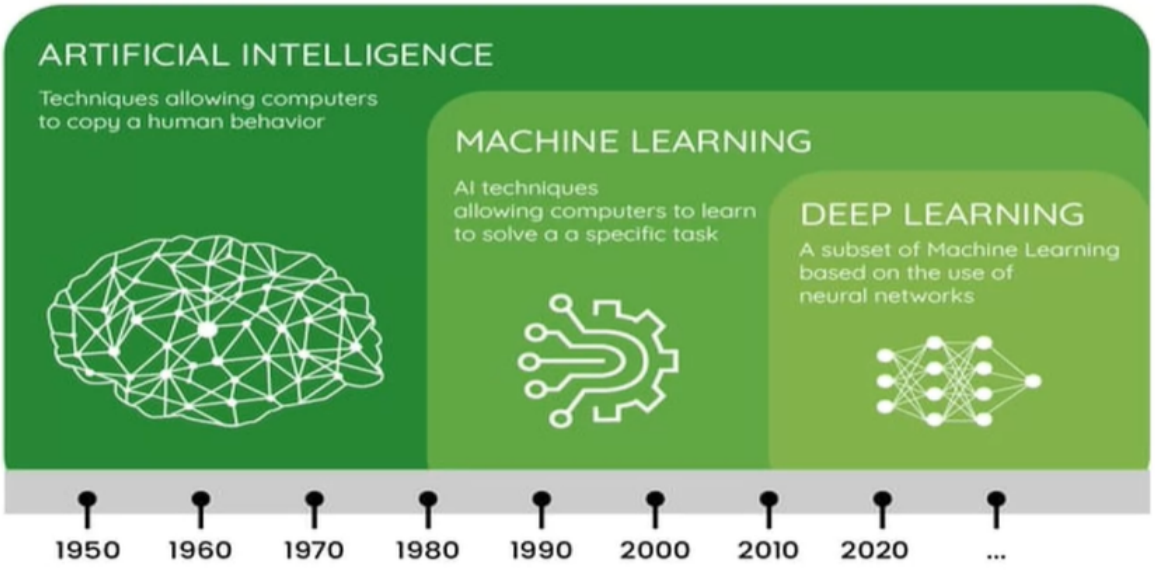
01 - Введение в искусственный интеллект и машинное обучение

Искусственный интеллект (ИИ): область компьютерных наук, направленных на создание систем, способных выполнить задачи, традиционно требующие человеческого интеллекта

Машинное обучение (ML): подмножество ИИ, фокусирующееся на разработке моделей для решения конкретной задачи, обучающихся на данных, без явного программирования решения



Принцип машинного обучения:

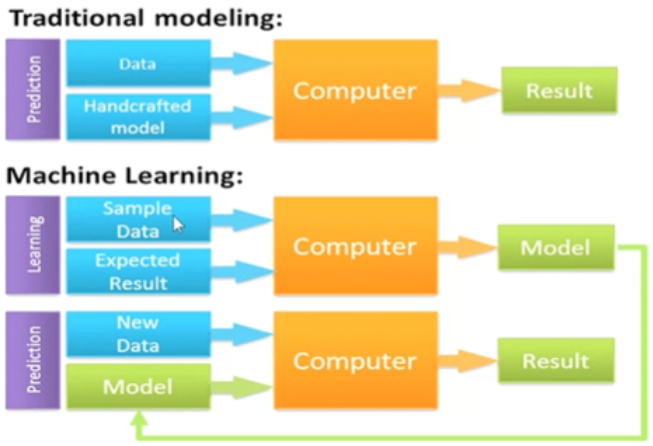
Традиционное моделирование (Traditional modeling):

1. На вход подаются **данные** (Data)
2. Создается **вручную разработанная модель** (Handcrafted model), часто это математические формулы или программный код
3. **Компьютер** выполняет обработку данных с помощью модели
4. На выходе получается **результат** (Result)

Машинное обучение (Machine Learning):

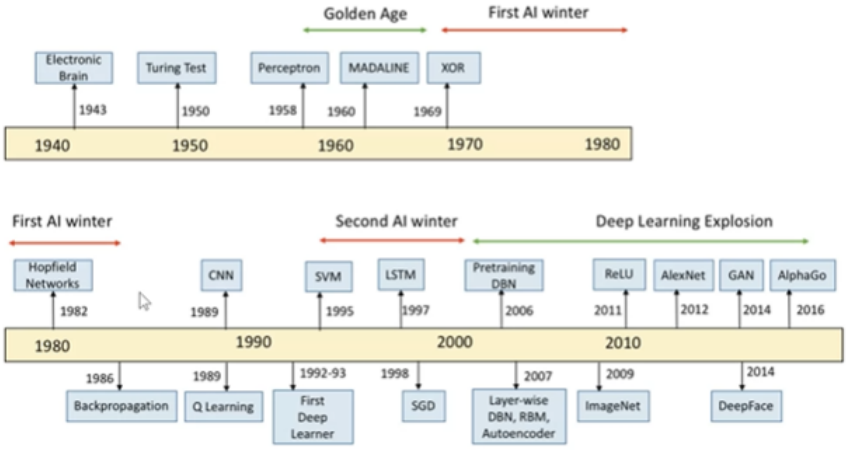
1. **Обучение (Learning):**
   - На вход подаются **обучающие данные** (Sample Data) и **ожидаемые результаты** (Expected Result)
   - Компьютер создает **модель** с коэффициентами, которые вычисляются автоматически
2. **Предсказание (Prediction):**
   - На вход подаются **новые данные** (New Data)
   - **Модель** применяет полученные параметры для обработки данных
3. Результат:
   - На выходе получается **предсказание** (Result)
   - Модель может обновляться, улучшая свои параметры на основе новых данных

Принцип машинного обучения



История развития ИИ:

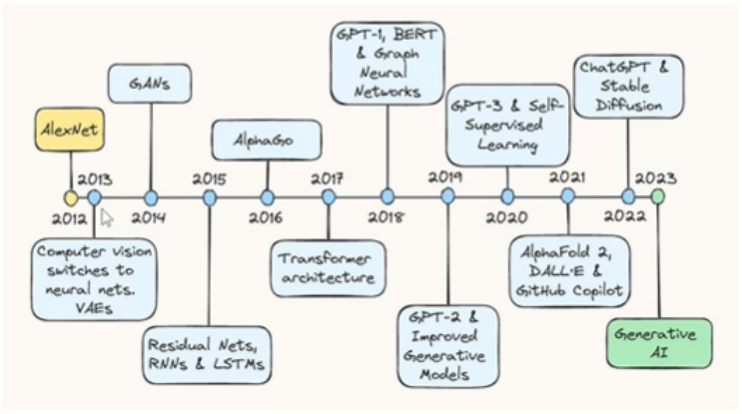
История развития ИИ (1)



1. **1943–1980 (Golden Age):**
   - 1943: Создан "Electronic Brain"
   - 1950: Тест Тьюринга
   - 1958–1969: Разработка перцептрона, MADALINE, решение задачи XOR
2. **1980 (First AI Winter):**
   - Снижение интереса к ИИ из-за ограниченных возможностей
3. **1982–1990:**
   - 1982: Hopfield Networks
   - 1986: Метод обратного распространения (Backpropagation)
   - 1989: Q-Learning, CNN
4. **1990–2000 (Second AI Winter):**
   - 1992–1993: Первая глубокая модель (First Deep Learner)
   - 1995: SVM
   - 1997: LSTM
5. **2000–2010 (Возрождение ИИ):**
   - 2006–2007: Pretraining DBN, RBM, Autoencoder
   - 2009: ImageNet
6. **2010–2016 (Deep Learning Explosion):**
   - 2011: ReLU
   - 2012: AlexNet

- 2014: GAN, DeepFace
- 2016: AlphaGo

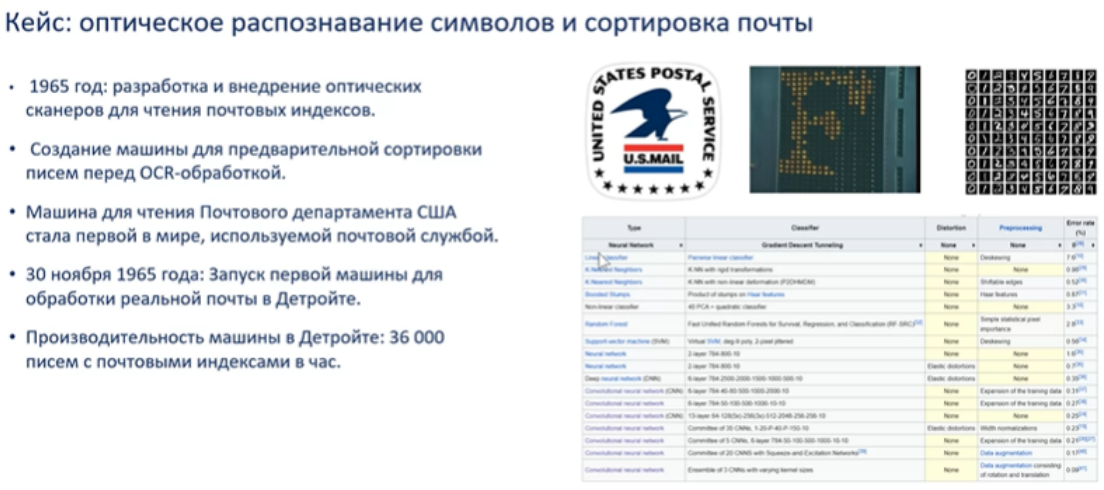
История развития ИИ (2)



7. **2012–2014:**
   - 2012: AlexNet – революция в компьютерном зрении с использованием нейронных сетей
   - Внедрение **VAE** (Вариационные автоэнкодеры)
   - Разработка **Residual Nets**, **RNNs**, и **LSTMs**
8. **2014–2016:**
   - 2014: **GANs** (Генеративно-состязательные сети)
   - 2016: **AlphaGo** – значительный успех в играх с искусственным интеллектом
9. **2017–2019:**
   - 2017: Разработка архитектуры **Transformer**
   - 2018: **GPT-1**, **BERT**, и графовые нейронные сети
   - 2019: **GPT-2** и улучшенные генеративные модели
10. **2020–2021:**
    - 2020: **GPT-3** и самообучение (Self-Supervised Learning)
    - 2021: **AlphaFold 2**, **DALL·E**, и **GitHub Copilot**
11. **2022–2023:**

- 2022: Появление **ChatGPT** и **Stable Diffusion**.
- 2023: Эпоха **Generative AI** – широкое применение генеративных моделей

Кейс: Оптическое распознавание символов и сортировка почты

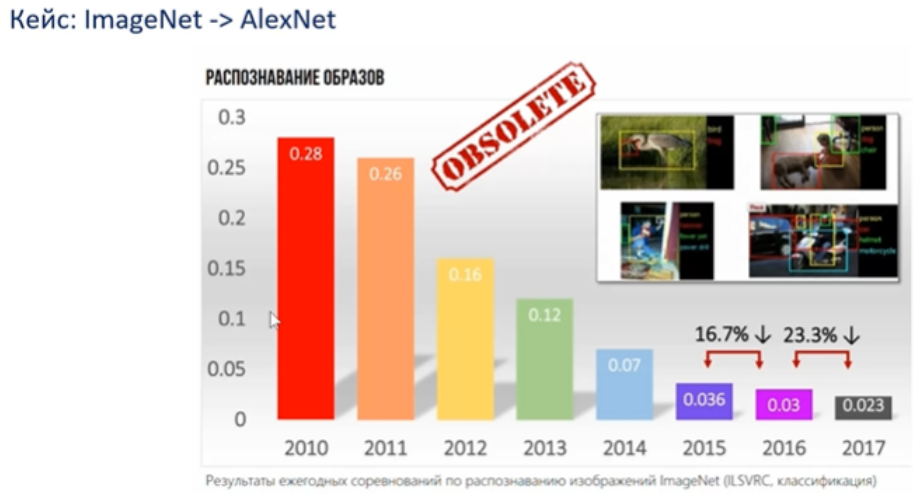


12. **1965 год:**
    - Разработка и внедрение оптических сканеров для чтения почтовых индексов
13. **Создание системы:**

- Машины для предварительной сортировки писем перед OCR-обработкой (распознавание символов)

14. **Первая машина:**
    - Машина для чтения, разработанная для Почтового департамента США, стала первой в мире, используемой почтовой службой
    - **30 ноября 1965 года**: Запуск первой машины для обработки реальной почты в Детройте
15. **Производительность:**
    - Производительность машины в Детройте достигала **36,000 писем с почтовыми индексами в час**

Кейс: ImageNet → AlexNet



16. **Соревнования по распознаванию образов:**
    - Проводятся ежегодно в рамках конкурса **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)**
    - Задача: улучшение алгоритмов классификации изображений
17. **Динамика ошибок классификации:**
    - **2010:** Ошибка 28% (ручные алгоритмы)
    - **2011:** Ошибка 26%
    - **2012:** Появление **AlexNet**, снижение ошибки до 16% — начало использования глубоких нейронных сетей
    - **2013–2014:** Постепенное снижение ошибок благодаря улучшенным архитектурам
    - **2015–2017:** Резкое снижение ошибок:
        - 2015: 3.6%.
        - 2016: 3.0%.
        - 2017: 2.3% — ниже человеческой точности
18. **Революция AlexNet (2012):**
    - AlexNet заложил основу глубокого обучения в задачах классификации
    - Благодаря архитектуре с использованием **сверточных нейронных сетей** (CNN), результаты конкурсов улучшились до уровня, который сделал традиционные методы **устаревшими (obsolete)**

Итог: ImageNet стал ключевым этапом в развитии глубокого обучения. Успех AlexNet и последующих моделей открыл путь для использования ИИ в широком спектре визуальных задач

Причины второй весны искусственного интеллекта и её интерпретация в бизнес-приложениях

1. **Создание новых моделей машинного обучения:**
   - Появление многослойных нейронных сетей, которые стали основой современных AI-технологий
2. **Быстрое накопление массивов данных:**
   - Интенсивное развитие интернета и информатизации позволило собирать и использовать огромные объемы данных для обучения моделей
3. **Рост производительности:**
   - Достижения в аппаратном обеспечении, включая Закон Мура и специализированные процессоры (например, GPU и TPU), обеспечили возможность обработки сложных моделей

Основой всех решений в ML является сбор и подготовка данных

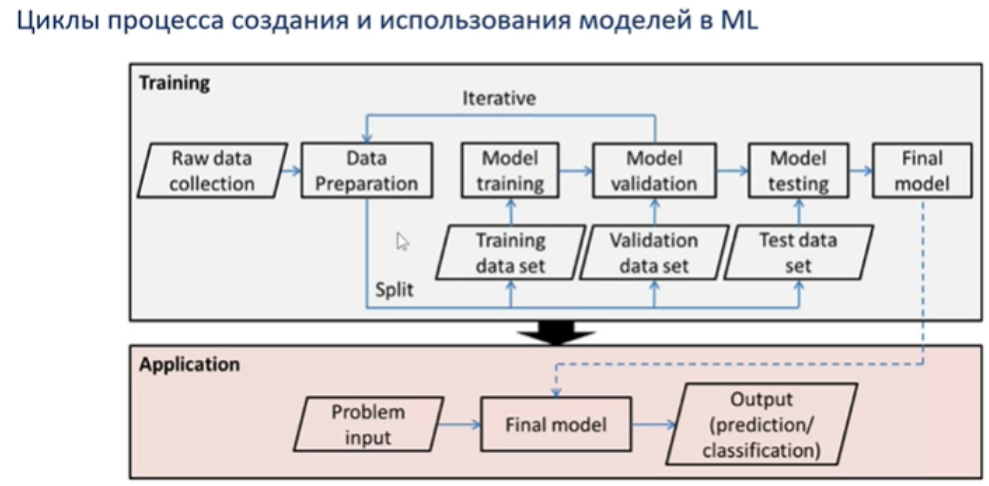
Иерархия потребностей Data Science (The Data Science Hierarchy of Needs):

1. **Collect (Сбор):**

- Сбор данных из различных источников: датчики, логирование, внешние данные, пользовательский контент

2. **Move/Store (Хранение и передача):**
   - Надежная инфраструктура для передачи данных, пайплайны, ETL-процессы, структурированные и неструктурированные хранилища данных
3. **Explore/Transform (Изучение и преобразование):**
   - Очистка данных, обнаружение аномалий, подготовка данных для анализа
4. **Aggregate/Label (Агрегация и маркировка):**
   - Анализ, метрики, сегментация, агрегирование, создание признаков, формирование обучающих выборок
5. **Learn/Optimize (Обучение и оптимизация):**
   - Применение простых алгоритмов ML, проведение A/B тестов, экспериментов.
6. **AI/Deep Learning (Искусственный интеллект и глубокое обучение):**
   - Использование продвинутых методов глубокого обучения и ИИ для решения сложных задач

Циклы процесса создания и использования моделей в ML



1. Training (Обучение модели):

- **Raw Data Collection (Сбор данных):**
  - Сбор необработанных данных, которые будут использоваться для обучения
- **Data Preparation (Подготовка данных):**
  - Очистка, нормализация и преобразование данных для использования в модели
- **Split (Разделение данных):**
  - Разделение данных на три подмножества:
    - **Training Data Set:** для обучения модели
    - **Validation Data Set:** для проверки качества и настройки модели
    - **Test Data Set:** для финального тестирования
- **Model Training (Обучение модели):**
  - Обучение модели на тренировочном наборе данных
- **Model Validation (Валидация модели):**
  - Оценка и оптимизация модели с использованием валидационного набора данных
- **Model Testing (Тестирование модели):**
  - Финальное тестирование модели для проверки её качества
- **Final Model (Финальная модель):**

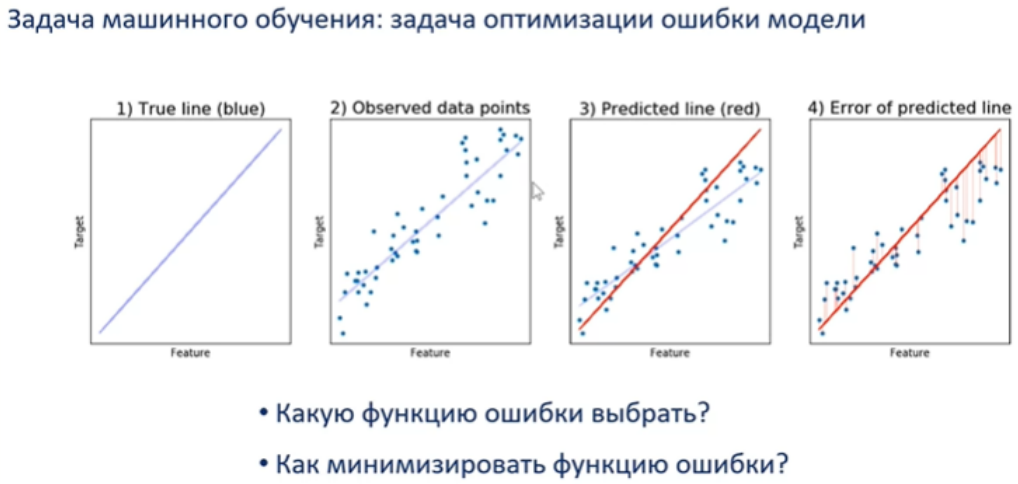
- Итоговая модель после всех циклов обучения, валидации и тестирования

2. Application (Использование модели):

- **Problem Input (Входные данные):**
  - Ввод реальных данных, на которых модель будет применять свои знания
- **Final Model (Финальная модель):**
  - Использование обученной и протестированной модели для обработки входных данных
- **Output (Выходные результаты):**
  - Генерация предсказаний или классификаций на основе входных данных

Задача машинного обучения: задача оптимизации ошибки модели

Описание графиков:



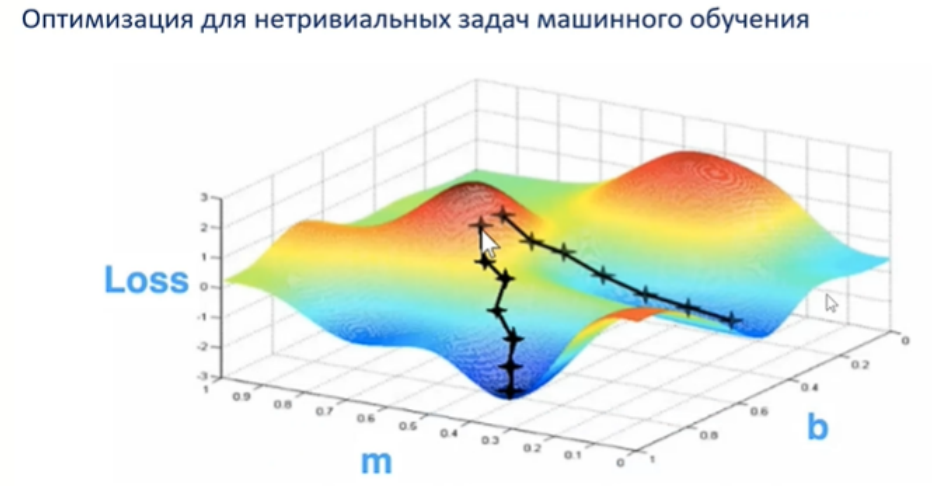
1. **True line (blue):**
   - Синяя линия представляет истинную зависимость между признаками (features) и целевой переменной (target)
2. **Observed data points:**
   - Наблюдаемые точки данных (синие точки) показывают реальные значения, которые содержат шум или отклонения от истинной линии
3. **Predicted line (red):**
   - Красная линия — это предсказанная моделью зависимость, которая пытается приблизиться к истинной линии
4. **Error of predicted line:**
   - Вертикальные линии между наблюдаемыми точками и предсказанной линией показывают величину ошибки предсказания модели

Основные вопросы:

1. **Какую функцию ошибки выбрать?**
   - Выбор функции ошибки зависит от задачи:
     - **MSE (Mean Squared Error):** для регрессии, штрафует большие отклонения
     - **MAE (Mean Absolute Error):** для устойчивости к выбросам
     - **Cross-Entropy Loss:** для классификации
     - **Hinge Loss:** для задач SVM
2. **Как минимизировать функцию ошибки?**

- Используются методы оптимизации:
  - **Градиентный спуск (Gradient Descent):** постепенное снижение значения функции ошибки
  - **Стохастический градиентный спуск (SGD):** обновление параметров модели на основе случайных подвыборок данных
  - **Адаптивные методы (Adam, RMSProp):** улучшенные алгоритмы для более быстрой сходимости

Оптимизация для нетривиальных задач машинного обучения



Описание графика:

- График показывает поверхность функции потерь (**Loss**) в зависимости от двух параметров модели (**m** и **b**)
- Цветовая шкала указывает значения функции потерь: чем светлее цвет, тем выше значение ошибки (Loss)
- Черные стрелки показывают путь оптимизации, который модель проходит для нахождения минимального значения ошибки

Ключевые аспекты оптимизации:

1. **Многомерная поверхность ошибки:**
   - Задачи машинного обучения имеют сложные, нелинейные функции потерь с множеством локальных минимумов
2. **Цель оптимизации:**
   - Найти глобальный минимум, где ошибка модели минимальна
3. **Методы решения:**
   - **Градиентный спуск (Gradient Descent):**
     - Итеративное обновление параметров модели в направлении уменьшения функции потерь
   - **Стохастический градиентный спуск (SGD):**
     - Быстрее для больших данных, обновляет параметры на основе случайных выборок
   - **Адаптивные алгоритмы (Adam, RMSProp):**
     - Автоматически регулируют шаг обучения для ускорения сходимости

Проблемы оптимизации:

- **Локальные минимумы:**

- Модель может застрять в точке, которая не является глобальным минимумом
- **Седловые точки:**
  - Области, где градиент близок к нулю, но это не минимум
- **Высокая размерность:**
  - Усложняет визуализацию и поиск оптимального решения

Гиперпараметры в моделях машинного обучения

Определение:

- **Гиперпараметры в ML** — настройки модели, которые определяют её общую структуру и способ обучения
- Устанавливаются **до начала обучения** и остаются неизменными в процессе обучения (в отличие от параметров модели, которые вычисляются во время оптимизации)

Пример гиперпараметра: Learning Rate (шаг обучения):

- Шаг обучения определяет, насколько сильно изменяются параметры модели на каждом этапе оптимизации (градиентного спуска)

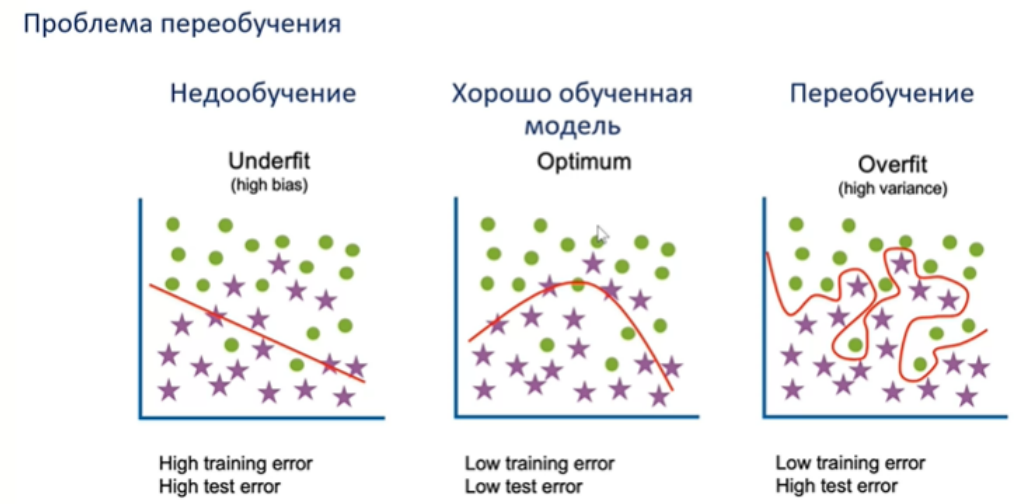
Сравнение:

1. **Big Learning Rate (Большой шаг обучения):**
   - Быстрое изменение параметров
   - Риск перескочить минимум и застрять в колебаниях
   - Менее точный результат
2. **Small Learning Rate (Малый шаг обучения):**
   - Медленное, плавное приближение к минимуму
   - Более высокая точность, но увеличивается время обучения

Ключевые аспекты:

1. **Почему важны гиперпараметры?**
   - Они сильно влияют на производительность модели, её точность и скорость обучения
2. **Как их выбирать?**
   - **Grid Search:** поиск комбинаций гиперпараметров по заданной сетке
   - **Random Search:** случайный выбор комбинаций
   - **Bayesian Optimization:** адаптивный поиск для уменьшения числа проб

Проблема переобучения



1. Недообучение (Underfitting):

- Характеристики:
  - Модель слишком проста, чтобы уловить зависимости в данных
  - Высокая ошибка на обучающей и тестовой выборках
  - **Причина:** модель имеет высокое смещение (*high bias*), что приводит к плохой производительности
- Пример:
  - Линейная зависимость для сложных данных

2. Хорошо обученная модель (Optimum):

- Характеристики:
  - Модель сбалансирована, хорошо описывает данные
  - Низкая ошибка на обучающей и тестовой выборках
  - Хорошо улавливает закономерности без перенасыщения деталями
- Пример:
  - Модель точно предсказывает целевые значения, сохраняя обобщающую способность

3. Переобучение (Overfitting):

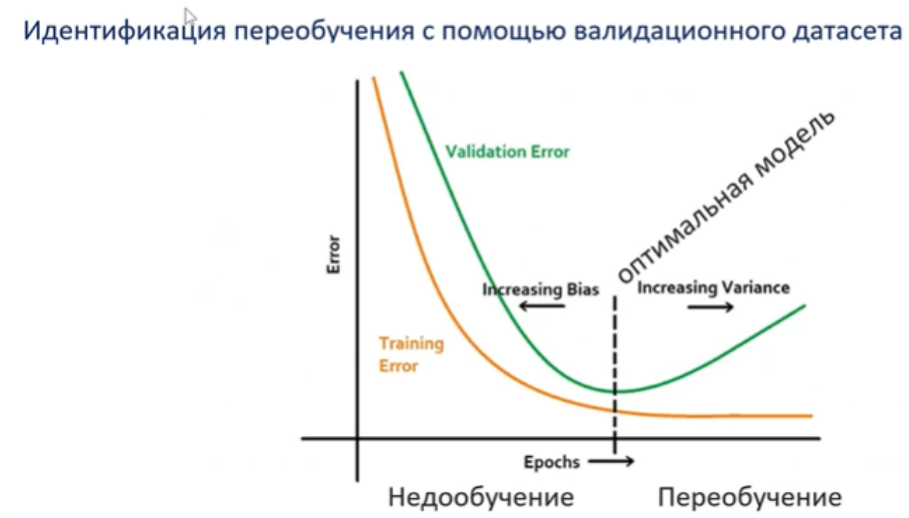
- Характеристики:
  - Модель слишком сложная, подстраивается под шум и незначительные детали
  - Низкая ошибка на обучающей выборке, но высокая на тестовой
  - **Причина:** модель имеет высокую дисперсию (*high variance*), что приводит к плохой генерализации
- Пример:
  - Избыточное совпадение с обучающими данными, потеря обобщающей способности

Решения:

1. **Для недообучения:**
   - Увеличить сложность модели (например, добавить больше признаков)
   - Больше эпох обучения
   - Уменьшить регуляризацию
2. **Для переобучения:**

- Применить регуляризацию (L1, L2).
- Уменьшить сложность модели
- Использовать больше данных для обучения
- Применить раннюю остановку (early stopping)

Идентификация переобучения с помощью валидационного датасета



Описание графика:

- **Training Error (Ошибка на обучающей выборке):**
  - Уменьшается по мере увеличения количества эпох обучения
  - Указывает на то, насколько модель подстраивается под обучающие данные
- **Validation Error (Ошибка на валидационной выборке):**
  - Сначала уменьшается, затем начинает расти после определённого количества эпох
  - Указывает на качество обобщения модели на новых данных

Фазы обучения:

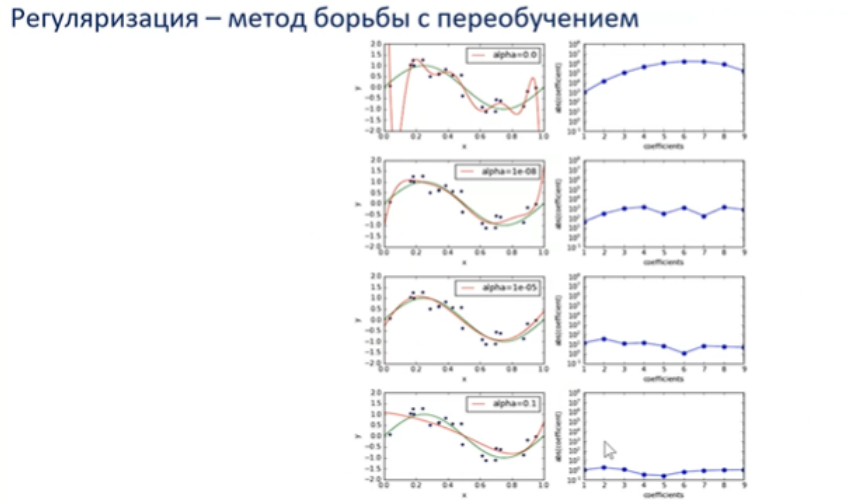
1. **Недообучение (Underfitting):**
   - Обе ошибки (обучающая и валидационная) высоки
   - Модель ещё не успела обучиться и уловить закономерности
   - **Решения:** увеличить сложность модели или число эпох обучения
2. **Оптимальная модель:**
   - Минимальная ошибка на валидационной выборке
   - Модель хорошо обобщает данные
   - **Цель:** остановить обучение на этой точке
3. **Переобучение (Overfitting):**
   - Ошибка на обучающей выборке продолжает уменьшаться, но ошибка на валидационной выборке растёт
   - Модель подстраивается под шум и специфические детали обучающей выборки, теряя обобщающую способность
   - **Решения:** использовать регуляризацию, раннюю остановку, или увеличить объём данных

Ключевые аспекты:

- **Increasing Bias (Возрастающее смещение):**
  - Характерно для недообучения.
  - Модель недостаточно сложная для адекватного представления данных.
- **Increasing Variance (Возрастающая дисперсия):**
  - Характерно для переобучения.
  - Модель слишком сложная и слишком подстраивается под данные

Итог: для предотвращения переобучения и выбора оптимальной модели важно отслеживать ошибки на валидационном наборе данных и останавливать обучение до момента, когда ошибка начинает расти

Регуляризация — метод борьбы с переобучением



Описание графика:

- Графики показывают влияние коэффициента регуляризации (**alpha**) на поведение модели:
  - **Слева:** Красная линия — предсказание модели, синяя линия — истинные данные, точки — обучающие данные
  - **Справа:** Величина коэффициентов модели при разном уровне регуляризации

Влияние регуляризации:

1. **Без регуляризации (alpha = 0):**
   - Модель слишком подстраивается под обучающие данные
   - Выраженное переобучение, так как модель пытается идеально описать шум
   - Коэффициенты модели имеют большие значения
2. **Слабая регуляризация (alpha = 1e-05):**
   - Модель становится более гладкой, уменьшается подстройка под шум
   - Коэффициенты уменьшаются, но всё ещё заметна тенденция к переобучению
3. **Средняя регуляризация (alpha = 1e-03):**
   - Оптимальный уровень регуляризации:
     - Модель хорошо описывает основные закономерности данных
     - Коэффициенты становятся более сбалансированными и стабильными
4. **Сильная регуляризация (alpha = 0.1):**
   - Модель становится слишком простой, начинает недообучаться
   - Коэффициенты стремятся к нулю, теряя способность описывать сложные зависимости

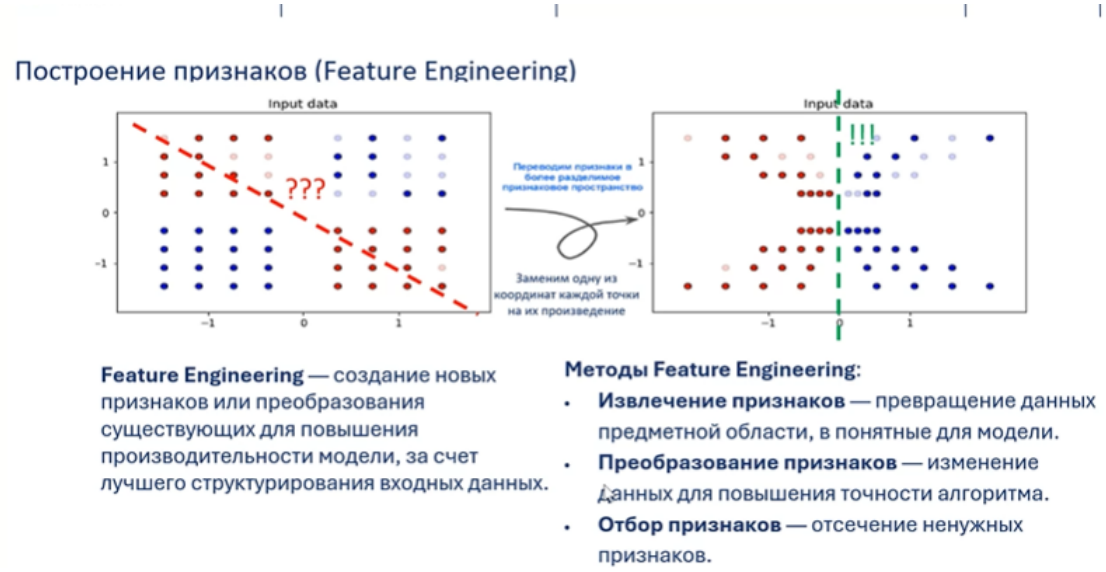
Методы регуляризации:

1. **L1-регуляризация (Lasso):**
   - Добавляет штраф за сумму абсолютных значений коэффициентов
   - Способствует занулению некоторых коэффициентов, что помогает в выборе признаков
2. **L2-регуляризация (Ridge):**
   - Добавляет штраф за сумму квадратов коэффициентов
   - Уменьшает значения коэффициентов, делая модель более гладкой
3. **Elastic Net:**
   - Комбинация L1 и L2-регуляризаций

Вывод:

Регуляризация помогает справиться с переобучением, контролируя сложность модели и предотвращая излишнюю подстройку под данные. Выбор оптимального значения **alpha** имеет решающее значение для достижения баланса между переобучением и недообучением

Построение признаков (Feature Engineering)



Что такое Feature Engineering?

- **Feature Engineering** — это процесс создания новых признаков или преобразования существующих для улучшения производительности модели.
- Цель: лучше структурировать входные данные, чтобы алгоритм смог более эффективно выявлять закономерности

Методы Feature Engineering:

1. **Извлечение признаков:**
   - Преобразование данных из исходного формата в вид, понятный модели.
   - Пример: извлечение временных признаков из даты (год, месяц, день)
2. **Преобразование признаков:**
   - Модификация данных для улучшения точности модели
   - Пример: логарифмирование для уменьшения влияния выбросов, масштабирование
3. **Отбор признаков:**
   - Исключение нерелевантных или избыточных признаков.
   - Пример: использование методов корреляции или алгоритмов отбора (Lasso, Recursive Feature Elimination)

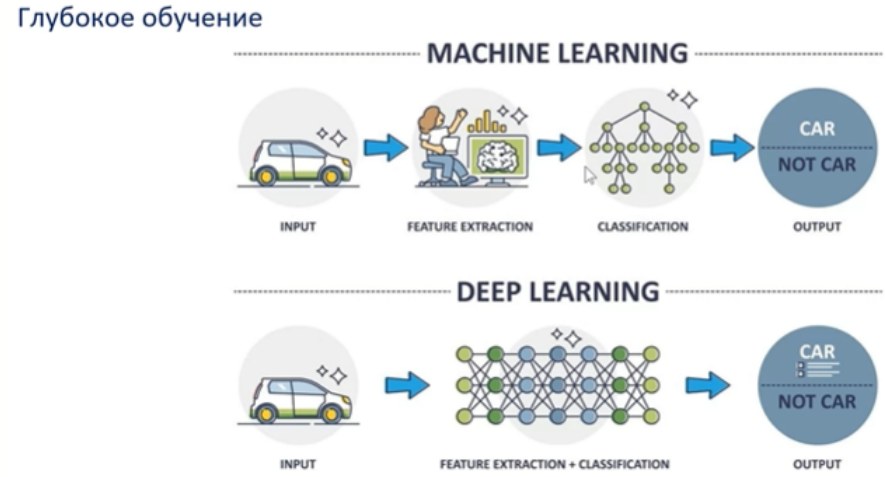
Пример на графике:

1. **Исходные данные (Input data):**
   - Два признака (x1, x2), представленные точками
   - Разделение классов плохо выражено в текущем пространстве признаков
2. **Создание новых признаков:**
   - Перевод данных в новое пространство с помощью комбинации признаков (например, произведение или квадраты координат)
   - Результат: более четкая разделимость классов, что повышает точность модели

Итог:

Feature Engineering является важным этапом машинного обучения, который позволяет преобразовать необработанные данные в вид, обеспечивающий максимальную производительность модели. Этот процесс требует как знания предметной области, так и владения инструментами анализа данных

Глубокое обучение (Deep Learning) и отличие от машинного обучения (Machine Learning)



Machine Learning (Машинное обучение):

1. **Input (Вход):**
   - На вход подаются данные, например, изображения автомобиля
2. **Feature Extraction (Извлечение признаков):**
   - Признаки (особенности изображения) извлекаются вручную человеком или с помощью алгоритмов
   - Этот этап требует знаний предметной области
3. **Classification (Классификация):**
   - Алгоритм машинного обучения использует извлечённые признаки для классификации объекта (например, "Машина" или "Не машина")
4. **Output (Выход):**
   - Результат классификации

Deep Learning (Глубокое обучение):

1. **Input (Вход):**
   - На вход также подаются данные, например, изображения автомобиля
2. **Feature Extraction + Classification (Извлечение признаков и классификация):**

- Эти этапы объединены
- Глубокая нейронная сеть автоматически извлекает признаки и выполняет классификацию в одном процессе
- Используются многослойные архитектуры, такие как сверточные нейронные сети (CNN)

3. **Output (Выход):**
   - Результат классификации ("Машина" или "Не машина")

Ключевые отличия:

- **Machine Learning:**
  - Требует ручного извлечения признаков.
  - Менее автоматизировано, больше зависит от человека.
- **Deep Learning:**
  - Автоматически выполняет извлечение признаков и классификацию.
  - Обрабатывает большие объемы данных и сложные задачи благодаря многослойным нейронным сетям.

Итог: глубокое обучение упрощает процесс обработки данных и может справляться с более сложными задачами без ручного этапа извлечения признаков, делая его мощным инструментом для современных приложений ИИ

Глубокая архитектура сверточной нейросети (CNN)

Принципы работы сверточной нейросети:

1. **Входные данные:**
   - На вход подается изображение, например, фотографии кота.
2. **Свертка (Convolution):**
   - Применяется **ядро свертки** (kernel) — небольшой фильтр, который проходит по изображению и выделяет ключевые особенности:
     - Линии.
     - Углы.
     - Текстуры.
3. **Иерархия признаков:**
   - На первом уровне сеть выделяет простые элементы (линии, углы).
   - На следующих слоях признаки становятся более сложными, например, форма глаз, ушей.
4. **Сверточные и объединяющие слои:**

- Свертки сочетаются с **пулингом** (Pooling) для уменьшения размерности и выделения наиболее значимых признаков.

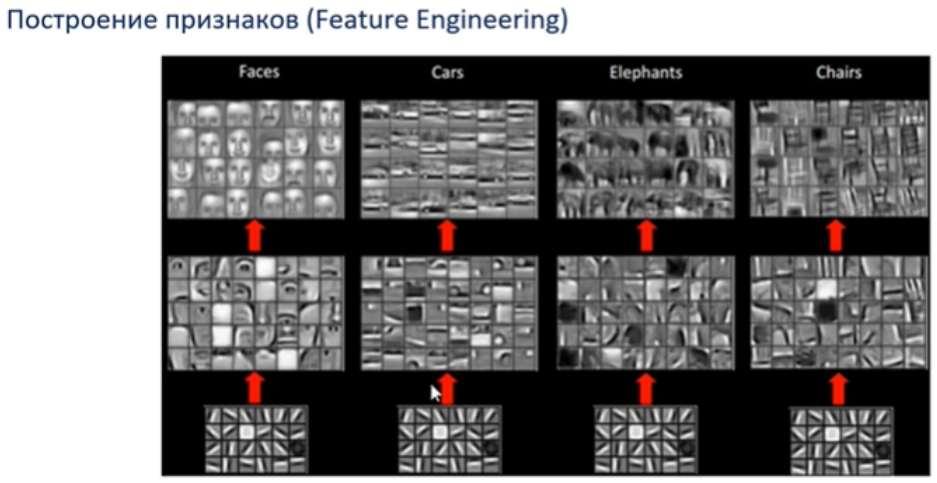
5. **Полносвязные слои:**
   - На выходе признаки преобразуются в одномерный вектор, который передается в **перцептрон** (полносвязную нейросеть) для классификации.
6. **Выход:**
   - Модель выдает вероятность того, что на изображении присутствует определенный объект (например, "Котик")

Ключевые особенности CNN:

- **Автоматическое извлечение признаков:**
  - Сеть сама учится находить значимые признаки изображения без ручного вмешательства
- **Обработка изображений:**
  - Особенно эффективна для работы с данными, представленными в виде матриц (изображения, видео)
- **Устойчивость к трансформациям:**
  - Модель устойчива к масштабированию, сдвигам и другим изменениям входных данных

Итог: сверточные нейросети (CNN) — основа современных методов работы с изображениями. Они позволяют выделять сложные иерархические признаки, обеспечивая высокую точность классификации

Построение признаков (Feature Engineering) в глубоких нейросетях



Описание визуализации:

1. **Классы объектов:**
   - Изображения из разных категорий: **лиц (Faces)**, **автомобилей (Cars)**, **слонов (Elephants)** и **стульев (Chairs)**
2. **Иерархия признаков:**
   - **Нижний уровень (нижний ряд):**
     - Простые геометрические элементы: линии, углы, текстуры.
     - Эти примитивы используются для построения более сложных признаков
   - **Средний уровень (средний ряд):**
     - Признаки средней сложности: контуры, формы, комбинации простых элементов
   - **Верхний уровень (верхний ряд):**

- Высокоуровневые признаки: части объектов (глаза, окна, уши, ножки стула)
- Используются для классификации целых объектов

Процесс построения признаков в глубоких сетях:

1. **Автоматическое извлечение признаков:**
   - Глубокие нейросети автоматически выделяют признаки из входных данных, начиная с простых и переходя к более сложным
2. **Иерархическая структура:**
   - Каждый последующий слой сети изучает иерархию признаков, постепенно усложняя их
3. **Специализация слоёв:**
   - Разные слои сети "отвечают" за разные уровни абстракции:
     - Первые слои: базовые элементы (линии, текстуры)
     - Средние слои: части объекта
     - Последние слои: полное представление объекта

Ключевые аспекты:

- **Особенность глубоких сетей:**
  - Они учатся на данных, избегая ручного извлечения признаков
- **Универсальность:**
  - Схожий подход используется для разных типов объектов, например, лиц, автомобилей, животных и мебели

Итог: глубокие нейросети обеспечивают мощное автоматическое извлечение признаков, что делает их незаменимыми для анализа сложных данных, таких как изображения и видео

Предобученные модели (Pretrained Models)

DL (Deep Learning):

1. **Описание процесса:**
   - Вход: Подразумевает подачу данных (например, изображения автомобиля)
   - Автоматическое извлечение признаков и классификация с помощью глубокой нейронной сети
   - Выход: Результат, например, классификация ("Машина" или "Не машина")

2. **Плюсы:**
    - **Высокая выразительность модели:** Подходит для сложных задач.
    - **Не требует участия эксперта:** Признаки выделяются автоматически
3. **Минус:**
    - Требует **большого объема данных** для обучения модели с нуля

DL: Pretrained Models (Предобученные модели):

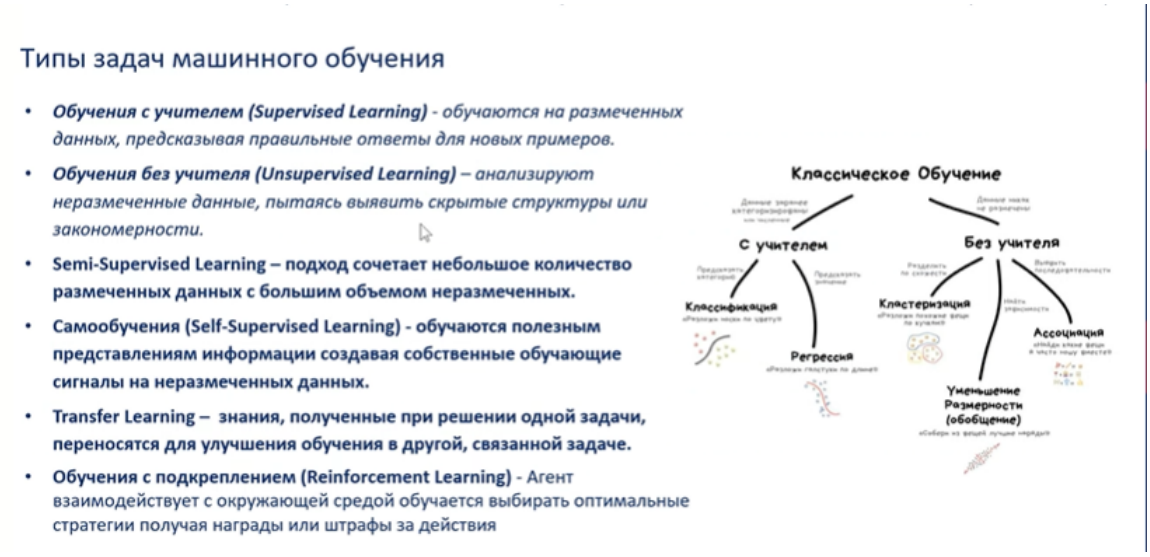
1. **Описание процесса:**
    - Используется предобученная нейронная сеть, обученная на большом наборе данных
    - Переход (Transfer Learning): Части модели замораживаются (**frozen**), а небольшая часть (последние слои) дообучается (**fine-tune**) на новой задаче с использованием небольшого датасета
2. **Пример этапов:**
    - Использование базовой сети (например, ResNet, VGG), предобученной на ImageNet
    - Заморозка сверточных слоев, чтобы сохранить ранее изученные признаки
    - Дообучение на небольшом специализированном наборе данных (например, классификация редких объектов)
3. **Плюсы:**
    - **Высокая выразительность модели:** Сохраняет преимущества глубокой сети
    - **Не требует эксперта:** Модель уже предобучена
    - **Экономия данных:** Нужен только небольшой датасет для дообучения

Ключевые аспекты:

- Предобученные модели сокращают время и ресурсы, необходимые для обучения
- Переход обучения позволяет адаптировать мощные глубокие сети к новым задачам, где объем данных ограничен

Итог: Использование предобученных моделей идеально подходит для задач, где доступ к большим датасетам ограничен. Это делает их популярным инструментом для широкого спектра прикладных задач в машинном обучении

Типы задач машинного обучения



1. Обучение с учителем (Supervised Learning):

- **Описание:**
  - Модель обучается на размеченных данных, где каждому входу соответствует известный выход
  - Используется для предсказания правильных ответов для новых примеров
- **Примеры задач:**
  - **Классификация:** Определение класса объекта (например, "Кот" или "Собака")
  - **Регрессия:** Предсказание числового значения (например, цены дома)

2. Обучение без учителя (Unsupervised Learning):

- **Описание:**
  - Модель анализирует неразмеченные данные, пытаясь выявить скрытые структуры или закономерности
- **Примеры задач:**
  - **Кластеризация:** Группировка данных (например, сегментация клиентов)
  - **Ассоциация:** Выявление зависимостей между переменными (например, "если покупают хлеб, то покупают молоко")

3. Полуобучение (Semi-Supervised Learning):

- **Описание:**
  - Используется небольшое количество размеченных данных и большой объем неразмеченных
  - Эффективно, когда разметка данных трудоемка

4. Самообучение (Self-Supervised Learning):

- **Описание:**
  - Модель обучается на основе собственных данных, создавая метки самостоятельно
  - Применяется в задачах обработки текста и изображений

5. Перенос обучения (Transfer Learning):

- **Описание:**
  - Знания, полученные в одной задаче, переносятся для решения другой задачи
  - Экономит время и ресурсы, особенно при работе с малыми данными

6. Обучение с подкреплением (Reinforcement Learning):

- **Описание:**
  - Агент взаимодействует с окружающей средой, учась выбирать оптимальные действия для максимизации награды
  - Применяется в играх, робототехнике, управлении

Схема:

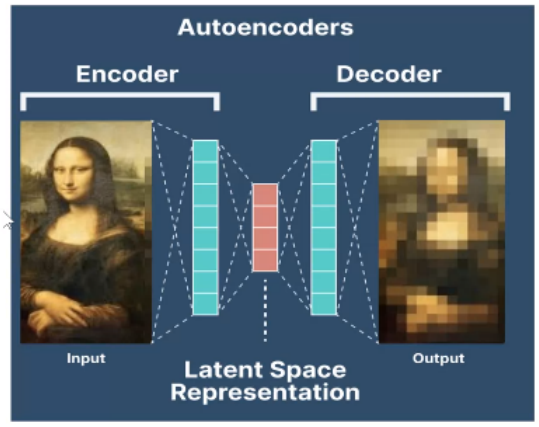
- **С учителем:**
  - Классификация, Регрессия
- **Без учителя:**
  - Кластеризация, Ассоциация

Итог: каждый тип обучения решает определённые задачи, а выбор подхода зависит от наличия данных, их разметки и цели анализа

Пример задачи самообучения: Autoencoders

Пример задачи самообучения

+ Очень высокая выразительность модели
+ Очень большой неразмеченный датасет
- Очень большие затраты на обучение



Autoencoders:

- Автоэнкодеры — это архитектура нейронной сети, которая обучается кодировать входные данные в компактное представление (**Latent Space Representation**) и затем восстанавливать их обратно
- Используются для обучения на неразмеченных данных

Процесс:

1. **Encoder (Энкодер):**
   - Кодирует входное изображение (например, "Мона Лиза") в сжатое латентное представление
   - Выявляет основные признаки, которые минимально описывают данные
2. **Latent Space Representation (Латентное пространство):**
   - Сжатое представление данных
   - Используется для анализа и хранения ключевых характеристик объекта
3. **Decoder (Декодер):**
   - Восстанавливает данные из латентного представления
   - Результат: изображение, максимально близкое к исходному

Преимущества:

Очень высокая выразительность модели

- Подходит для сложных данных
  Очень большой неразмеченный датасет
- Не требуется разметка данных, что экономит ресурсы

Недостатки:

Очень большие затраты на обучение

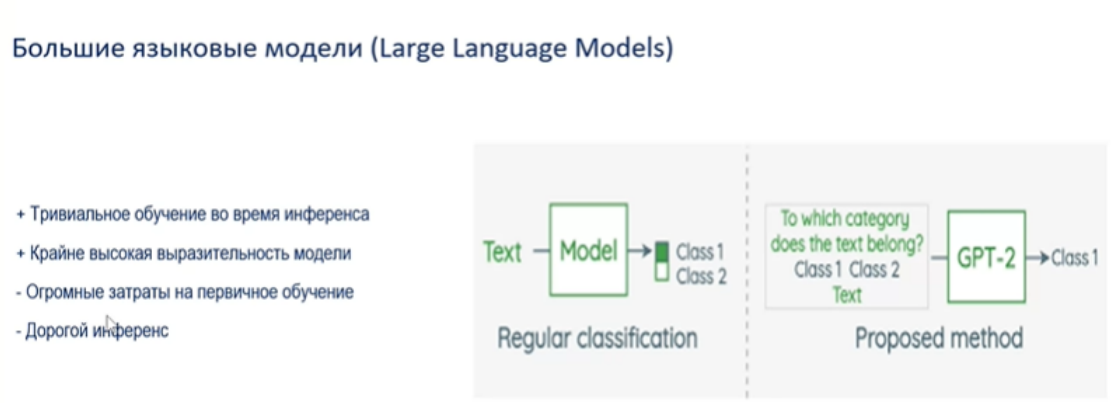
- Требуется значительное количество вычислительных ресурсов и времени для обучения модели

Применение:

- Сжатие данных (компрессия)
- Удаление шума из изображений
- Выявление аномалий в данных

Итог: Autoencoders демонстрируют мощный подход к самообучению, обеспечивая эффективную работу с неразмеченными данными и решая широкий спектр задач

Большие языковые модели (Large Language Models)



Ключевые особенности:

1. **Тривиальное обучение во время инференса:**
   - Модель уже обучена и может быть легко адаптирована для новых задач без значительных изменений
2. **Крайне высокая выразительность модели:**
   - Может обрабатывать сложные текстовые задачи, включая классификацию, генерацию текста, перевод и др
3. **Огромные затраты на первичное обучение:**
   - Требуется большое количество вычислительных ресурсов (GPU, TPU) и данных для начального обучения
4. **Дорогой инференс:**
   - Использование модели для предсказаний требует значительных вычислительных мощностей

Сравнение подходов:

1. **Regular classification (Обычная классификация):**
   - Входной текст обрабатывается моделью для определения класса
   - Используется стандартная архитектура для конкретной задачи
2. **Proposed method (Предложенный метод с использованием LLM, например, GPT-2):**
   - Модель обучена на универсальных задачах
   - На этапе инференса отвечает на вопросы о принадлежности текста к определённым категориям
   - Адаптация для задачи выполняется без значительного переобучения

Преимущества LLM:

Гибкость:

- Подходит для широкого спектра задач, без необходимости создания новой модели
  Мощность:
- Способна работать с огромными объёмами данных и сложными языковыми структурами

Недостатки LLM:

- **Затраты на обучение:**
  - Огромные вычислительные мощности для первичного обучения
- **Инференс:**
  - Высокая стоимость вычислений при реальном применение

Применение:

- Классификация текста
- Генерация текстов (статьи, истории, программный код)
- Перевод и обработка естественного языка

Итог: Большие языковые модели, такие как GPT, представляют собой мощный инструмент, который, несмотря на высокую стоимость, позволяет решать широкий спектр задач, связанных с обработкой текста

Качественный скачок LLM (Large Language Models)

Основная идея:

Переход к большим языковым моделям (LLM) с большим количеством параметров привел к значительному улучшению их возможностей. Эти модели способны решать задачи, которые не встречались им во время обучения, благодаря обучению на контексте

Ключевые аспекты:

1. **Обучение на контексте:**
   - Модель может решать новые задачи на основе текстовой инструкции, содержащей примеры
   - Пример: если в инструкции указано, как классифицировать текст, модель применяет этот контекст для выполнения задачи, без изменения своих весов
2. **Гибкость:**
   - LLM способны решать разнообразные задачи, такие как:
     - Ответы на вопросы
     - Переводы
     - Понимание текста
     - Решение математических задач
     - Генерация кода и резюме текста
3. **Отсутствие необходимости переобучения:**
   - Дополнительное обучение для новых задач не требуется — модель адаптируется на основе текстового ввода

Рост параметров и возможностей:

- **8 billion parameters:** Базовые задачи, такие как понимание языка и простая арифметика
- **62 billion parameters:** Добавляются задачи более высокого уровня, например, резюмирование текста и перевод
- **540 billion parameters:** Углубленное понимание языка, логические цепочки рассуждений, решение физических задач и объяснение шуток

Ключевые преимущества:

- **Широкий спектр задач:*охватывает множество доменов
- **Интуитивность:** решение новых задач возможно без дополнительного обучения
- **Масштабируемость:** повышение числа параметров ведет к улучшению производительности

Итог:

Большие языковые модели открыли новые горизонты в области ИИ, позволяя решать сложные задачи, которые раньше считались невозможными, без значительных изменений или дообучения. Их развитие продолжает расширять возможности приложений искусственного интеллекта

Zero-shot, One-shot и Few-shot Learning

Основные подходы:

Zero-shot Learning:
- Модель выполняет задачу без предоставления примеров
- Использует лишь текстовый инструктаж (prompt) для понимания задачи
- Подходит для универсальных задач, требующих обобщённых знаний

One-shot Learning:
- Модель выполняет задачу на основе одного примера.
- Один пример в контексте позволяет модели лучше понять требования задачи.

Few-shot Learning:
- Модель обучается или адаптируется на небольшом количестве примеров (обычно несколько).
- Позволяет значительно повысить точность по сравнению с zero-shot.

График:

- **Оси:**

    - **X (Number of Examples in Context):** количество предоставленных примеров
    - **Y (Accuracy):** точность выполнения задачи
- **Модели (по количеству параметров):**
    - **1.3B Params:** модель с малым количеством параметров, низкая точность
    - **138B Params:** модель средней сложности, точность увеличивается с количеством примеров
    - **175B Params:** крупная языковая модель, обеспечивает высокую точность, даже в условиях zero-shot
- **Сравнение:**
    - Без текста-подсказки (**No Prompt**) результаты хуже
    - С текстовой инструкцией (**Natural Language Prompt**) точность значительно выше, особенно для больших моделей

Выводы:

Эффективность масштабирования:
- Увеличение числа параметров модели (например, до 175B) улучшает точность и позволяет лучше справляться с задачами в zero-shot, one-shot и few-shot режимах
Важность контекста:
- Контекстные примеры (даже единичные) помогают модели лучше понять задачу, повышая точность
Zero-shot для универсальности:
- Большие модели (например, GPT) могут решать задачи без обучения на них, что делает их универсальными для широкого спектра применений

Итог: подходы zero-shot, one-shot и few-shot learning демонстрируют силу современных больших языковых моделей, позволяя адаптироваться к новым задачам с минимальным количеством примеров или даже без них