Машинное обучение: супервизированное обучение

Введение в супервизированное обучение

Машинное обучение: супервизированное обучение

- Один из самых популярных подходов в машинном обучении
- Основан на обучении с учителем: данные имеют метки
- Используется в задачах классификации и регрессии

Супервизированное обучение

Супервизированное обучение (Supervised Learning) – это метод машинного обучения, в котором модель обучается на размеченных данных, где каждому входу соответствует правильный выход.

Основные принципы супервизированного обучения

1. **Обучение на размеченных данных** – каждый объект обучающей выборки имеет известную метку (выходной результат).

2. **Обобщение** – модель должна не просто запомнить примеры, но и уметь делать предсказания на новых данных.

3. **Оптимизация** – процесс настройки модели для минимизации ошибки предсказаний.

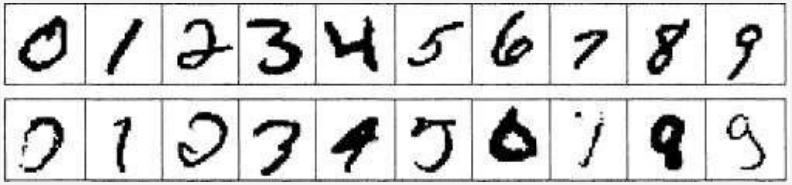
Виды задач в супервизированном обучении

Супервизированное обучение решает два типа задач:

1. **Классификация** – предсказание категориального значения (класса).
2. **Регрессия** – предсказание числового значения.

Примеры классификации:

- Определение, является ли email спамом или нет.
- Распознавание рукописных цифр.
- Диагностика болезней по медицинским данным.



Примеры регрессии:

- Прогнозирование цены недвижимости.
- Оценка спроса на товар.
- Предсказание температуры воздуха.

Разница между классификацией и регрессией

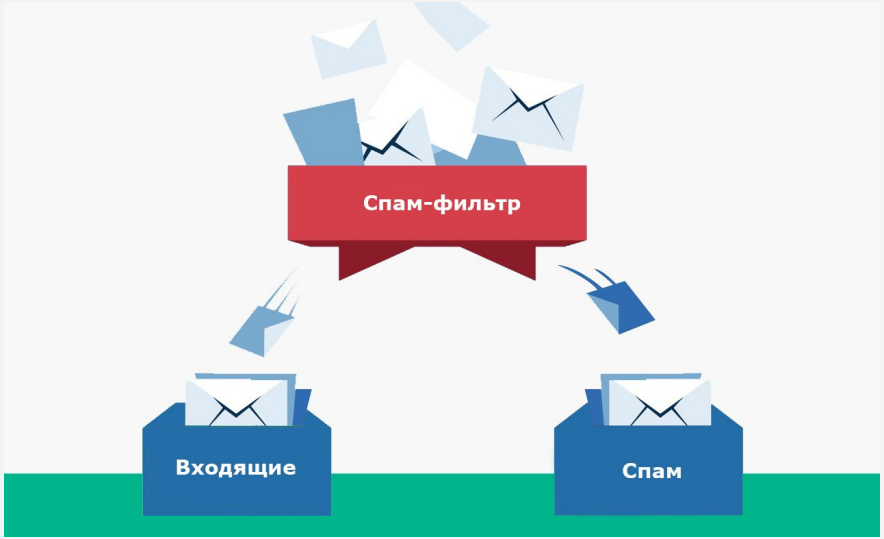
Характеристика	Классификация	Регрессия
Тип данных	Дискретные (категории)	Непрерывные (числа)
Пример	Кредитный клиент: надежный / ненадежный	Сумма кредита в долларах
Алгоритмы	Логистическая регрессия, SVM, деревья решений	Линейная регрессия, Random Forest, нейросети
Выходные значения	Метки классов (например, 0 или 1)	Числовые значения (например, 3.5, 42.7)

Пример классификации

Пример: фильтрация спама в электронной почте

📩 **Входные данные**: текст письма, тема, отправитель.

📌 **Выход**: два класса – "спам" или "не спам".



Как работает?

1. Алгоритм обучается на размеченных письмах (где уже известно, спам это или нет).
2. Выявляются ключевые слова, частота их встречаемости, структура письма.
3. Новые письма классифицируются по этим признакам.

Пример регрессии

Пример: предсказание цен на недвижимость

🏡 **Входные данные**: метраж квартиры, район, этаж, количество комнат.

💰 **Выход**: цена квартиры (числовое значение).

Как работает?

1. Алгоритм анализирует исторические данные (цены квартир за последние годы).

2. Выявляет зависимость между характеристиками квартиры и ценой.

3. Применяет найденную закономерность к новым объектам для предсказания стоимости.

Ключевые алгоритмы супервизированного обучения

1. **Линейная регрессия** – предсказание числовых значений.
2. **Логистическая регрессия** – бинарная классификация (да/нет).
3. **Метод опорных векторов (SVM)** – поиск оптимальной границы между классами.

Ключевые алгоритмы супервизированного обучения

4. **Деревья решений** – построение иерархии решений.

5. **Случайный лес (Random Forest)** – ансамбль деревьев решений.

6. **Наивный Байес (Naïve Bayes)** – быстрый алгоритм для текстовой классификации.

7. **Нейронные сети** – сложные нелинейные модели.

Линейная регрессия

Линейная регрессия (Linear Regression)

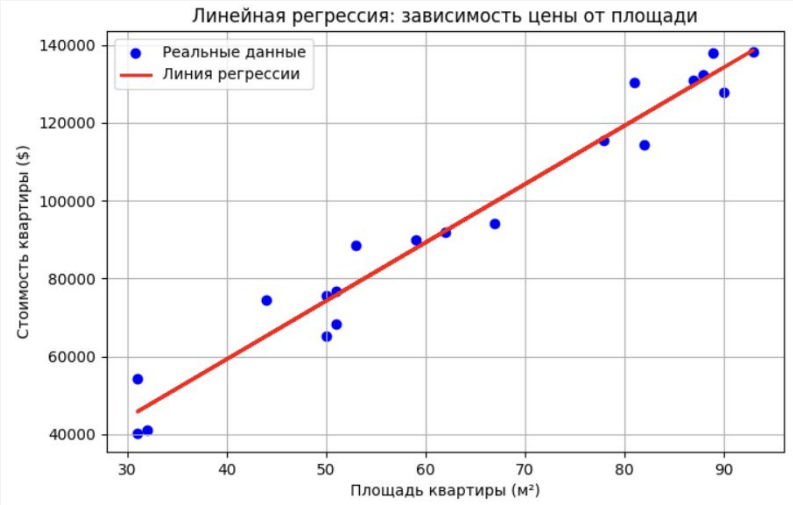
✅ Используется для прогнозирования числовых значений.

✅ Модель строит **прямую линию**, описывающую зависимость переменных.

✅ Основная формула: $$Y = wX + b$$

Визуализация линейной регрессии

- График линейной регрессии показывает **зависимость между переменными**.
- Прямая линия – это **модель**, которая предсказывает значения на основе входных данных.
- Данные, которые расположены **ближе к линии**, предсказаны с **малой ошибкой**.



Логистическая регрессия

✅ Используется для **бинарной классификации** (да/нет, 0/1).

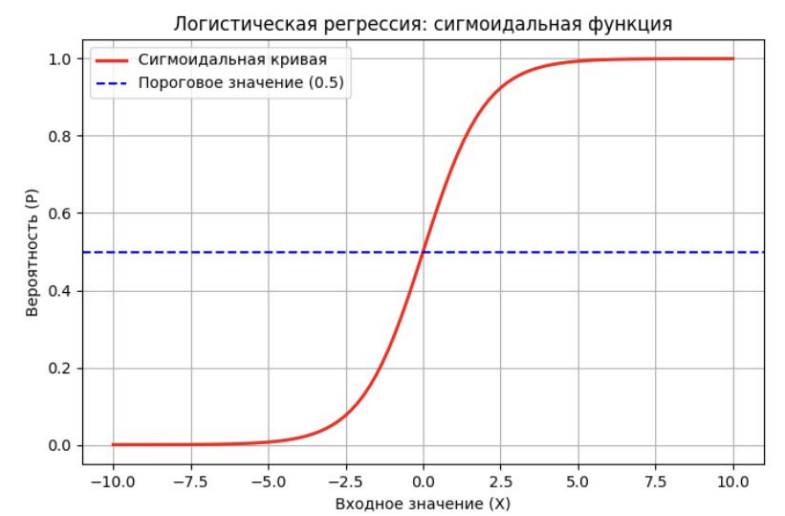
✅ Выходное значение – вероятность принадлежности к классу.

✅ Основная функция: **сигмоидальная функция (логистическая функция)**

$$P(Y = 1) = \frac{1}{1 + e^{-(wX+b)}}$$

Визуализация логистической регрессии

- Логистическая регрессия **не строит прямую**, а использует **S-образную (сигмоидальную) кривую**.
- Значения **выхода** всегда в диапазоне **(0, 1)**, что позволяет интерпретировать их как вероятность.
- Граница классификации устанавливается по порогу **0.5**:
  - **P ≥ 0.5 → Класс 1 (положительный класс)**
  - **P < 0.5 → Класс 0 (отрицательный класс)**



Метод опорных векторов (SVM)

- **Метод опорных векторов (Support Vector Machine, SVM)**

✅ Используется **для классификации и регрессии**.

✅ Находит **оптимальную границу (гиперплоскость)**, разделяющую классы.

✅ Работает **в многомерном пространстве**, что делает его эффективным для сложных данных.

✅ Использует **ядра (kernel trick)** для работы с нелинейными зависимостями.

Как работает?

1. **Разделяет данные** с максимальным зазором (margin).
2. **Выбирает ключевые точки (опорные векторы)**, которые определяют границу.
3. **Использует ядра (полиномиальные, гауссовские)** для обработки нелинейных данных.

Визуализация метода опорных векторов (SVM)

- Метод SVM разделяет данные **максимально возможным зазором (margin)**.

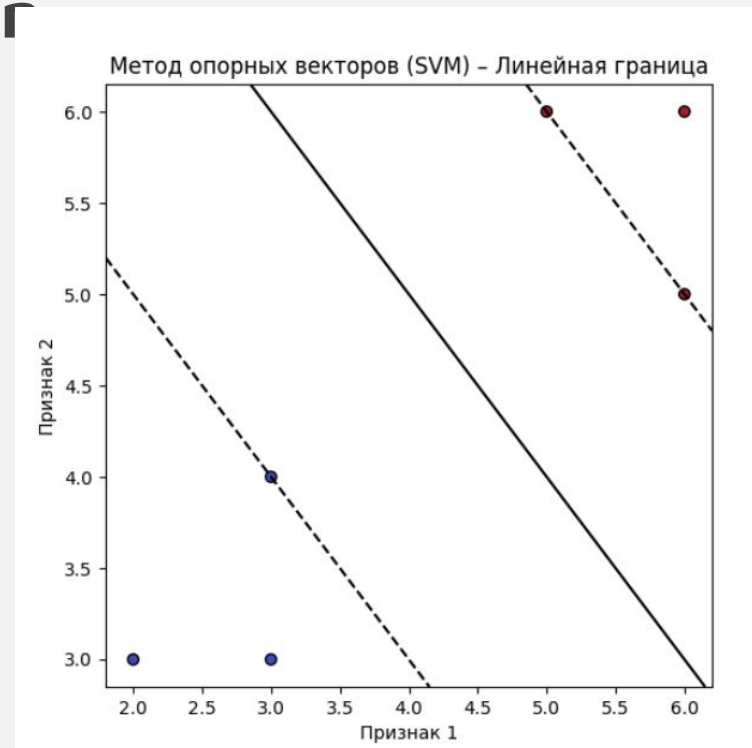
- **Опорные векторы** – это ключевые точки, которые определяют границу разделения.

- Граница классификации (гиперплоскость) проходит **между классами** так, чтобы расстояние до ближайших точек каждого класса было максимальным.

Пример кода для метода опорных векторов (SVM) в Python

```
model = SVC(kernel='linear')

model.fit(X, Y)
```

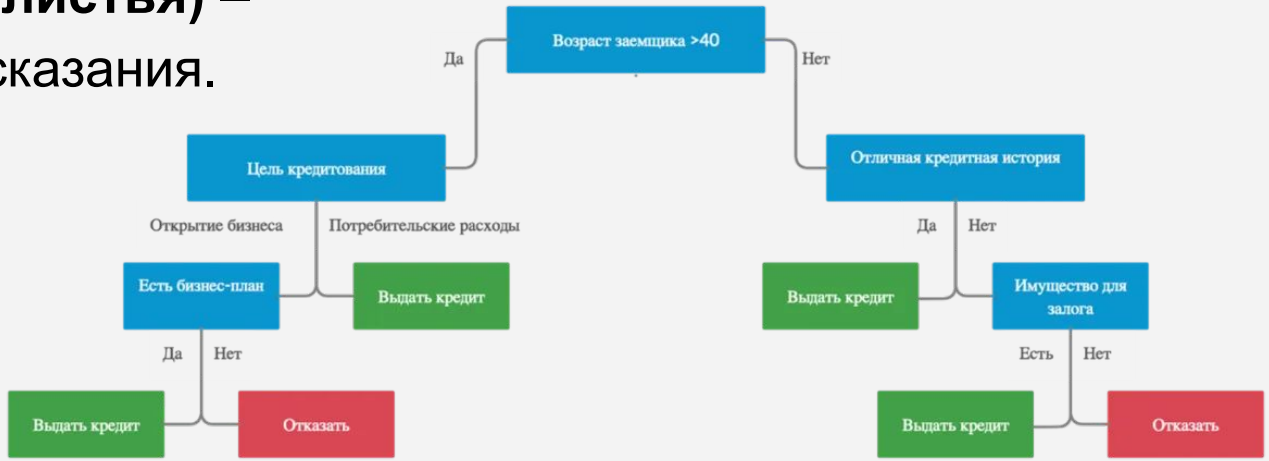


Деревья решений

- **Деревья решений (Decision Trees)** – это метод **классификации и регрессии**, который использует **иерархическую структуру решений**.
- В каждом узле выполняется **логический вопрос**, разделяющий данные.
- Ветви дерева ведут к **различным классам или предсказаниям**.

Как работает?

1. **Верхний узел (root node)** – это начальная точка.
2. **Каждое разветвление (split)** – это проверка условия.
3. **Конечные узлы (листья)** – это результат предсказания.

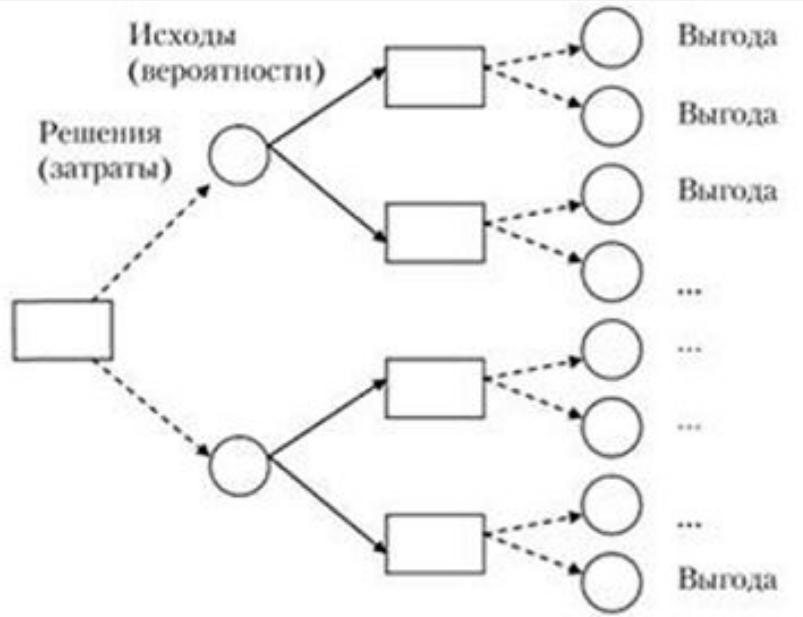


Визуализация дерева решений

- Дерево решений **разделяет данные на основе последовательных условий**.
- Каждое разветвление – это **новый логический вопрос**.
- **Листья** дерева представляют **конечный результат** (класс или числовое значение).

Как интерпретировать дерево?

1. **Начало** – корневой узел с первым условием.
2. **Разветвления** – проверка нового условия.
3. **Конечные узлы** – итоговое



Пример визуализации

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import matplotlib.pyplot as plt

model = DecisionTreeClassifier()
model.fit(X, y)
tree.plot_tree(model, filled=True)
plt.show()
```

Ансамблевые методы – случайный лес (Random Forest)

🔹 **Random Forest (Случайный лес)** – это ансамблевый метод, состоящий из **нескольких деревьев решений**.

🔹 Работает по принципу **голосования**:

- Каждый **отдельный алгоритм (дерево решений)** делает своё предсказание.
- Финальное решение принимается **путём усреднения (в регрессии) или голосования большинства (в классификации)**.

Преимущества

✅ Устойчивость к переобучению.

✅ Высокая точность за счёт использования множества моделей.

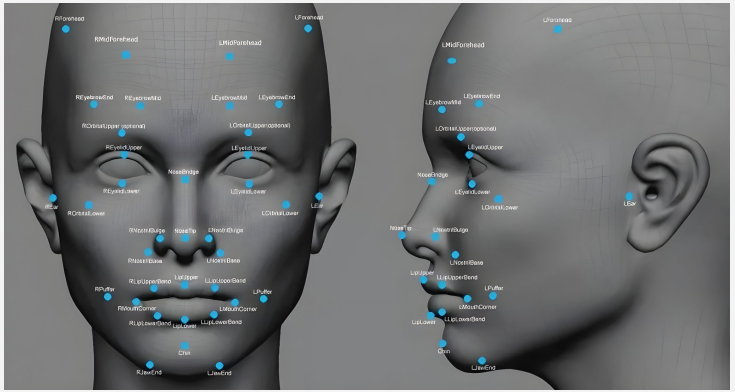
✅ Хорошо работает с большими наборами данных.

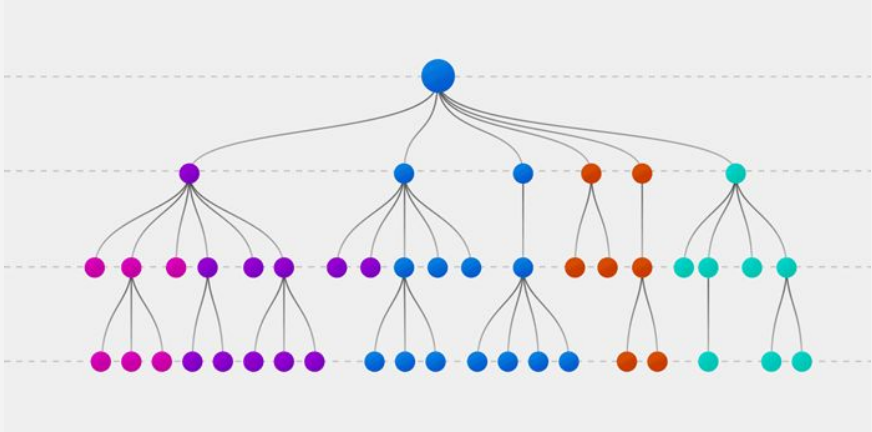
Как это работает?

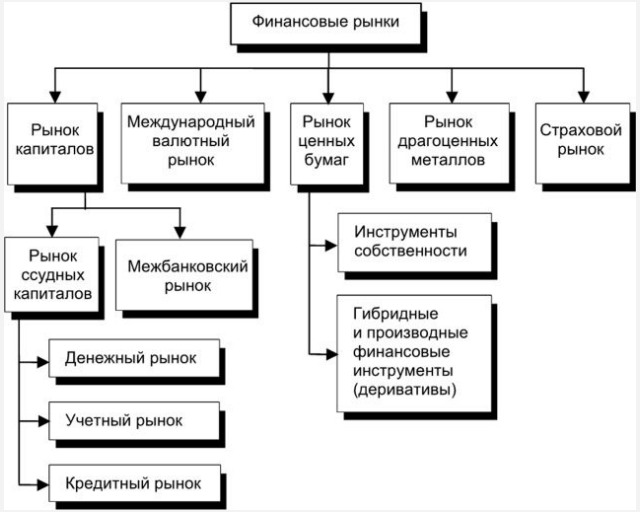
1. **Генерация нескольких деревьев решений** на разных подмножествах данных.
2. **Объединение их предсказаний** – финальный результат более надёжен.
3. **Модель устойчива к шуму**, так как единичные ошибки деревьев компенсируются большинством.

Применение:

- Распознавание лиц.
- Медицинская диагностика.
- Классификация финансовых транзакций.







Пример кода для случайного леса (Random Forest) в Python

```
from sklearn.ensemble import RandomForestClassifier


model = RandomForestClassifier(n_estimators=50,
random_state=42)

model.fit(X_train, Y_train)
```

Наивный Байес

Основан на **теореме Байеса**:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

- Используется в **текстовой классификации** (например, определение тональности отзывов).
- Предполагает, что **все признаки независимы** друг от друга (поэтому "наивный").

Нейронные сети

- **Нейронные сети (Artificial Neural Networks, ANN)** – это **алгоритмы, вдохновлённые работой мозга**.
- Они состоят из **нейронов**, соединённых в слои, и способны **извлекать сложные закономерности** из данных.
- Современные нейросети используют **глубокое обучение (Deep Learning)** для обработки изображений, текста, звука.

Как устроена нейронная сеть?

1. **Входной слой** – получает данные (например, изображение или текст).
2. **Скрытые слои** – обрабатывают информацию, выявляя **сложные зависимости**.
3. **Выходной слой** – даёт **окончательный результат** (например, «кот» или «собака»).

Как работает искусственный нейрон?

- **Искусственный нейрон** – это математическая модель, вдохновлённая работой биологического нейрона.
- Он **получает входные сигналы**, обрабатывает их и передаёт выходной сигнал.
- Использует **веса (weights), сумму входов и активационную функцию**.

Как устроен нейрон?

1. **Входные данные $x_1, x_2, ..., x_n$** – это характеристики объекта.
2. **Веса $w_1, w_2, ..., w_n$** – определяют важность каждого входа.
3. **Сумма взвешенных входов**:

$$z = w_1 x_1 + w_2 x_2 + ... + w_n x_n + b$$

4. **Активационная функция** – решает, передавать ли сигнал дальше:

$$y = f(z)$$

Метрики оценки моделей

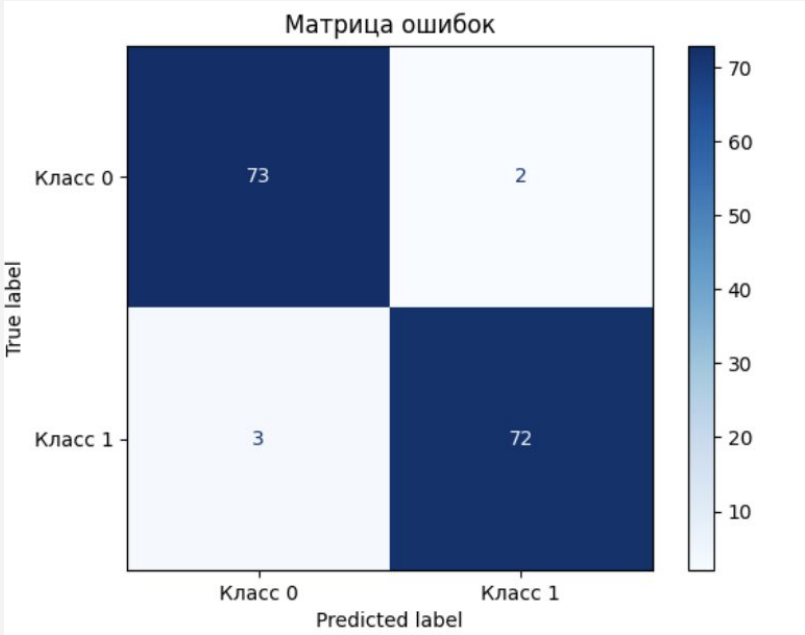
- После обучения модели важно **оценить её качество**.
- Для этого используются **разные метрики**, в зависимости от типа задачи.
- Важно понимать, **какие ошибки допустила модель** и как **улучшить её предсказания**.

Основные метрики оценки

1. **Accuracy (Точность)**
2. **Precision (Точность, положительное предсказательное значение)**
3. **Recall (Полнота, чувствительность)**
4. **F1-score**

Матрица ошибок (Confusion Matrix)

- **Матрица ошибок (Confusion Matrix)** – это таблица, которая показывает, **насколько хорошо модель предсказывает классы**.
- Позволяет понять, **где модель делает ошибки**.
- Используется в задачах **классификации**.



Как выглядит матрица ошибок?

Факт / Предсказание	**Положительный (1)**	**Отрицательный (0)**
Положительный (1) (Истинно положительные, TP)	TP (Правильные предсказания 1)	FN (Ошибка 1)
Отрицательный (0) (Истинно отрицательные, TN)	FP (Ошибка 0)	TN (Правильные предсказания 0)

Как интерпретировать?

TP (True Positive) – модель **правильно предсказала положительный класс**.

TN (True Negative) – модель **правильно предсказала отрицательный класс**.

FP (False Positive, Ошибка I рода) – модель **ошибочно предсказала положительный класс**.

FN (False Negative, Ошибка II рода) – модель **ошибочно предсказала отрицательный класс**.

Accuracy (Точность)

Количество **правильных предсказаний** среди всех предсказаний.

Применяется в **сбалансированных выборках**, но **может вводить в заблуждение**, если классы несбалансированы.

Формула:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision (Точность, положительное предсказательное значение)

Насколько модель **уверена** в правильности положительных предсказаний.

Важно при **поиске редких событий** (например, при обнаружении мошенничества).

Формула:

$$Precision = \frac{TP}{TP + FP}$$

Recall (Полнота, чувствительность)

Показывает, **насколько хорошо модель находит все положительные примеры**.

Важно в медицине, например, при **поиске заболеваний**.

Формула:

$$Recall = \frac{TP}{TP + FN}$$

F1-score

Баланс между **Precision и Recall**.

Используется, когда **важно учитывать обе метрики**.

Формула:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Ограничения супервизированного обучения

- Несмотря на эффективность, **супервизированное обучение** имеет **ряд ограничений**.

- Основные проблемы связаны с **качеством данных, зависимостью от разметки и сложностью обучения моделей**.

Основные ограничения

1️⃣ Зависимость от размеченных данных

- Требует **большого количества размеченных примеров**. Разметка **дорогая** и требует **времени** (например, в медицине). Без хороших данных модель **не сможет обучаться корректно**.

2️⃣ Проблема переобучения (Overfitting)

- Если модель **слишком сложная**, она **запоминает обучающие данные**, но **плохо работает на новых данных**. Необходима **регуляризация**, ограничение параметров модели.

Основные ограничения

3⃣ **Проблема несбалансированных классов**

- Если один класс встречается **намного чаще** другого, модель **может игнорировать редкие случаи**.Требуется **балансировка данных** или использование **взвешенных метрик**.

4⃣ **Низкая обобщающая способность**

- Если **в тестовых данных есть новые примеры**, которые модель **не видела**, она может работать некорректно. Нужны **дополнительные техники улучшения генерализации**.

Когда использовать супервизированное обучение?

- **Супервизированное обучение** идеально подходит для задач, где **известны правильные ответы** (размеченные данные).
- Используется в **классификации и регрессии**, если у нас **есть достаточное количество примеров**.

Когда супервизированное обучение эффективно?

✅ **Когда есть размеченные данные** → например, в медицине (диагностика заболеваний).

✅ **Когда нужно делать точные предсказания** → например, прогнозирование цен на рынке.

✅ **Когда важна интерпретируемость** → например, финансовые модели для оценки рисков.

Когда оно не подходит?

❌ **Если данных мало или они не размечены** → лучше использовать **обучение без учителя**.

❌ **Если система слишком сложная** → иногда требуется **глубокое обучение (Deep Learning)**.

❌ **Если данные постоянно меняются** → требуется **онлайн-обучение** или **обучение с подкреплением**.

Основные выводы по супервизированному обучению

- **Супервизированное обучение** – это один из **основных методов машинного обучения**, который работает на **размеченных данных**.
- Используется в **классификации и регрессии** для решения множества реальных задач.
- Включает в себя **разные алгоритмы**: **линейную регрессию, деревья решений, случайные леса, градиентный бустинг, нейросети**.
- Имеет **ограничения**, связанные с **зависимостью от данных**, возможностью **переобучения** и **проблемами генерализации**.