**Основные проблемы при обучении классических моделей на больших данных**

1. **Ограничения по памяти (RAM)** – Большие наборы данных не помещаются в оперативную память.
2. **Медленная скорость обучения** – Традиционные алгоритмы (например, логистическая регрессия, SVM) не оптимизированы для обработки миллиардов записей.
3. **Требование к вычислительным ресурсам** – Не все организации могут использовать мощные GPU и облачные решения.

**🔹 Методы оптимизации обучения**

**1️⃣ Использование библиотеки Dask 📊**

* Позволяет работать с **таблицами больше памяти** (out-of-core processing).
* Делит данные на **"маленькие куски" (partitions)**, обрабатывает их по очереди.
* Поддерживает **pandas-like синтаксис**.

📌 **Пример использования Dask:**

python

KopierenBearbeiten

import dask.dataframe as dd

df = dd.read\_csv("large\_dataset.csv") # Загружаем большой CSV-файл

df.describe().compute() # Выполняем агрегатные операции

✅ **Преимущество:** Работает как pandas, но эффективнее с большими данными.

**2️⃣ Online Learning (Stochastic Gradient Descent - SGD) ⚡**

* Использует **метод стохастического градиентного спуска (SGD)**.
* Модель обновляется **по одному наблюдению** за раз.
* Экономит память, так как **не загружает весь датасет в RAM**.

📌 **Пример использования SGD в scikit-learn:**

python

KopierenBearbeiten

from sklearn.linear\_model import SGDClassifier

sgd = SGDClassifier(loss="log\_loss") # Логистическая регрессия с SGD

for X\_batch, y\_batch in data\_loader: # Читаем данные пакетами (batches)

sgd.partial\_fit(X\_batch, y\_batch, classes=[0, 1])

✅ **Преимущество:** Хорошо подходит для потоковой обработки данных.

**3️⃣ Использование библиотеки VAEX 🚀**

* **Читает и обрабатывает данные в памяти GPU**.
* Позволяет **работать с миллиардами строк** быстрее, чем pandas.
* Использует **lazy evaluation** (отложенные вычисления).

📌 **Пример работы с VAEX:**

python

KopierenBearbeiten

import vaex

df = vaex.open("large\_dataset.csv")

df.mean("feature1") # Вычисление среднего без загрузки всех данных в RAM

✅ **Преимущество:** В 100 раз быстрее pandas для больших данных.

**4️⃣ Градиентный бустинг на больших данных (LightGBM, CatBoost, XGBoost) 🌲**

* **LightGBM** – использует histogram-based подход для ускорения обучения.
* **XGBoost** – поддерживает работу с дисковыми массивами.
* **CatBoost** – эффективен для категориальных данных.

📌 **Пример работы с LightGBM на больших данных:**

python

KopierenBearbeiten

import lightgbm as lgb

train\_data = lgb.Dataset("large\_train.csv", format="csv") # Читаем из файла

params = {"objective": "binary", "boosting": "gbdt"}

model = lgb.train(params, train\_data, num\_boost\_round=100)

✅ **Преимущество:** Эффективность при обработке **сотен гигабайт данных**.

**🔹 Выводы**

🔹 **Dask & Vaex** → **Анализ данных без перегрузки RAM**  
🔹 **SGD & Online Learning** → **Обучение моделей без загрузки всего набора данных**  
🔹 **LightGBM/XGBoost** → **Эффективная обработка больших данных**

✅ **Комбинирование этих техник помогает масштабировать классическое машинное обучение на уровне Big Data.**