# Отчёт по сессионному проекту по дисциплине

«Fundamentals of Data Mining and Machine Learning»


### Авторы:

Тұрғын Салтанат

Уршукова Томирис


## 1. Цель проекта

Целью данного проекта является практическое освоение базовых алгоритмов машинного обучения, а именно:

- линейной регрессии,
- логистической регрессии,
- методов классификации,

путём **реализации алгоритмов с нуля** и их сравнения с готовыми моделями библиотеки `scikit-learn`.

Дополнительной целью является анализ влияния гиперпараметров (learning rate, epochs, batch size) на процесс обучения моделей.

## 2. Описание датасета

В проекте использован открытый датасет **IMDb Video Games Dataset**, полученный с платформы Kaggle.

**Источник:** Kaggle

**Ссылка:** https://www.kaggle.com/datasets/lorentzyeung/imdb-video-games-dataset

**Размер датасета:** 14 682 наблюдения

Датасет содержит информацию о видеоиграх, включая оценки пользователей и показатели популярности.

### Основные используемые признаки:

- **User Rating** — числовая оценка пользователей
- **Popularity** — показатель популярности игры (целевая переменная)

Датасет является реальным, структурированным и удовлетворяет всем требованиям проекта.

## Влияние скорости обучения на сходимость

График “Влияние скорости обучения на сходимость при градиентном спуске” показывает, что:

lr = 0,01 продемонстрировал наилучшую сходимость к минимальному MSE.

Более высокие значения lr (например, 0,05, 0,1, 0,3) приводили к расхождению (значительному росту MSE) или нестабильному поведению, что подтверждалось такими ошибками, как

`RuntimeWarning`: переполнение встречается в квадратных и

`RuntimeWarning`: недопустимое значение, обнаруженное при вычитании.

# 3. Линейная регрессия

## 3.1 Модель

Используется одномерная линейная регрессия:

$\hat{y} = wx + b$

## 3.2 Функция потерь

В качестве функции потерь используется среднеквадратичная ошибка (MSE):

$$L = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (wx_i + b)\right)^2$$

### 3.3 Градиенты

$$\frac{\partial L}{\partial w} = -\frac{2}{n}\sum_{i=1}^{n} x_i(y_i - \hat{y_i})$$

$$\frac{\partial L}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n} (y_i - \hat{y_i})$$

Обучение производится с помощью **batch gradient descent**, реализованного вручную с использованием `numpy`.

## 4. Логистическая регрессия

### 4.1 Формулировка задачи

Задача бинарной классификации:

- 0 — низкая популярность (ниже медианы)
- 1 — высокая популярность (выше медианы)

### 4.2 Сигмоида

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

### 4.3 Функция потерь (Log-loss)

$$L = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i})]$$

### 4.4 Градиенты

$$\frac{\partial L}{\partial w} = \frac{1}{n} X^T (\hat{y} - y)$$

$$\frac{\partial L}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y_i} - y_i)$$

Обучение выполняется с использованием **mini-batch gradient descent** и L2-регуляризации.

## 5. Классификация и сравнение моделей

В проекте сравниваются две модели:

- логистическая регрессия (реализация с нуля),
- Decision Tree (библиотека `scikit-learn`).

### Используемые метрики:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix
- ROC AUC

### 5.1 Сравнение моделей

| Модель | Accuracy | Precision | Recall | F1-score | ROC AUC |
|---|---|---|---|---|---|
| Логистическая регрессия (наша) | 0.78 | 0.76 | 0.80 | 0.78 | 0.85 |
| Decision Tree (sklearn) | 0.82 | 0.81 | 0.83 | 0.82 | 0.88 |

# 6. Эксперименты

Проведены эксперименты с:

- различными значениями **learning rate**,
- количеством **epochs**,
- размером **batch size**.

Эксперименты показали, что:

- слишком большой learning rate приводит к нестабильной сходимости,
- увеличение количества эпох улучшает качество до определённого предела,
- mini-batch gradient descent обеспечивает более стабильное обучение.

## 6.1 Испытанные гиперпараметры

- **Learning rate:** 0.01, 0.05, 0.1, 0.3
- **Batch size:** 32, 64, 128
- **Epochs:** 100, 500, 1000
- **L2-регуляризация (λ):** 0.01

# 7. Выводы

В ходе проекта были реализованы и исследованы базовые алгоритмы машинного обучения.

Логистическая регрессия показала стабильные результаты на бинарной классификации, а Decision Tree продемонстрировало способность моделировать нелинейные зависимости.

Проект подтвердил важность корректного выбора гиперпараметров и понимания внутренних механизмов алгоритмов машинного обучения.