

**ОГЛАВЛЕНИЕ**

[1. Введение 3](#_Toc25)

[2. Основная часть 4](#_Toc26)

[2.1 ТЕОРЕТИЧЕСКА ЧАСТЬ 4](#_Toc27)

[2.1.1 Обоснование использования нейронных сетей для рекомендательных систем 4](#_Toc28)

[2.1.2 Архитектура многослойного перцептрона (MLP) для задачи регрессии 4](#_Toc29)

[2.1.3 Отличие нейронных сетей от ансамблей (плюсы и минусы подхода) 6](#_Toc30)

[2.1.4 Настройка рабочей среды 7](#_Toc31)

[2.1.5 Выводы по разделу 7](#_Toc32)

[2.2 практическая часть 9](#_Toc33)

[2.2.1 Загрузка датасета MovieLens 100K 9](#_Toc34)

[2.2.2 Визуальный анализ 9](#_Toc35)

[2.2.3 Построение архитектуры нейронной сети 10](#_Toc36)

[2.2.4 Обучение модели 11](#_Toc37)

[2.2.5 Обучение модели 11](#_Toc38)

[2.2.6 Тестирование модели 11](#_Toc39)

[2.2.7 Оптимизация и улучшение модели 11](#_Toc40)

[2.2.8 Выводы по разделу 12](#_Toc41)

[Заключение 13](#_Toc42)

[**Список используемых источников и литературы**  14](#_Toc43)

1. Введение

Эксплуатационную практику я проходил на кафедре информационных технологий в Московском университет имени С.Ю. Витте (МУИВ). В рамках практики я занимался исследованием методов машинного обучения и их применением для решения прикладных задач анализа данных. Моим направлением работы стали рекомендательные системы, так как они активно применяются в современных онлайн-сервисах и имеют большое практическое значение.

Исходные данные я взял из открытого датасета MovieLens 100K https://grouplens.org/datasets/movielens/100k/, который содержит 100 000 оценок, выставленных 943 пользователями 1682 фильмам.

Гит: https://github.com/Starratel/praktishka

Ссылка на датасет: https://grouplens.org/datasets/movielens/100k/

**Цель практики**

Целью моей практики является разработка и обучение модели, которая сможет предсказывать рейтинг фильма, выставляемый пользователем, на основе его идентификатора и идентификатора фильма.

Для решения этой задачи я выбрал использование нейронных сетей — в частности, архитектуры многослойного перцептрона (MLP). Такой подход позволяет учитывать нелинейные зависимости между пользователями и фильмами, что делает предсказания более точными по сравнению с простыми линейными моделями.

**Задачи практики**

Я решил задачу регрессии — предсказание числового значения рейтинга (от 1 до 5), который пользователь может поставить фильму, основываясь на его ID и ID фильма. Для этого я применил методы глубокого обучения: построил и обучил нейронную сеть (многослойный перцептрон, MLP) для предсказания оценок, а также провёл оптимизацию гиперпараметров, чтобы повысить точность модели.

1. Основная часть

## ТЕОРЕТИЧЕСКА ЧАСТЬ

### Обоснование использования нейронных сетей для рекомендательных систем

В своей работе я выбрал использование нейронных сетей, поскольку они способны учитывать сложные нелинейные зависимости между признаками.

В отличие от простых моделей машинного обучения (линейной регрессии, деревьев решений), нейронные сети позволяют строить более гибкие аппроксимации функций и достигают более высокой точности на больших объёмах данных.

Рекомендательные системы часто имеют скрытые паттерны в поведении пользователей, которые неочевидны и плохо выражаются линейными методами. Нейронные сети могут выявлять эти зависимости, что делает их актуальным инструментом для предсказания пользовательских рейтингов.

### Архитектура многослойного перцептрона (MLP) для задачи регрессии

Для построения модели я использовал многослойный перцептрон (MLP), так как:

MLP vs. линейные модели (линейная/логистическая регрессия)

Линейные модели предполагают, что зависимость между признаками и целевой переменной линейна.

В моей задаче предпочтения пользователей и оценка фильмов зависят от множества скрытых факторов, которые выражаются нелинейно.

MLP благодаря активации ReLU способен аппроксимировать нелинейные функции, а значит, точнее предсказывает рейтинги.

MLP vs. ансамбли (Random Forest, XGBoost)

Ансамбли деревьев решений хорошо работают на табличных данных, но они ограничены в выражении сложных взаимодействий между признаками.

У ансамблей мало возможностей к дальнейшему развитию (например, добавление эмбеддингов пользователей и фильмов).

Нейросети более гибки: можно увеличивать количество слоёв, менять архитектуру, использовать регуляризацию и получать улучшенные результаты.

MLP vs. RNN (рекуррентные сети)

RNN применяются для последовательных данных (текст, временные ряды).

В моём датасете нет ярко выраженной последовательности — оценки фильмов пользователем не образуют временной контекст, который можно было бы обрабатывать RNN.

MLP эффективнее в случае фиксированного вектора признаков («userId, movieId»).

MLP vs. CNN (сверточные сети)

CNN предназначены для анализа изображений, пространственных признаков.

В моей задаче нет пространственной структуры данных, поэтому использование свёрток не имеет смысла.

MLP проще и лучше подходит для векторных входов.

MLP vs. трансформеры

Современные трансформерные модели очень мощные, но требуют огромных вычислительных ресурсов и больших объёмов данных.

MovieLens 100K — сравнительно небольшой датасет, где использование трансформеров неоправданно.

MLP обеспечивает баланс между точностью и скоростью обучения.

Архитектура MLP:

Входной слой — принимает два признака: ID пользователя и ID фильма (в числовом виде).

Несколько скрытых слоёв (Dense-слои) с функцией активации ReLU. Они позволяют сети находить сложные зависимости.

Выходной слой с одним нейроном и линейной активацией, так как требуется предсказать числовое значение рейтинга.



где:

* x=(userId,movieId) — входные данные модели, закодированные идентификаторы пользователя и фильма, используемые как признаки;
* Dense(64, ReLU) — первый скрытый слой полносвязной нейронной сети, содержащий 64 нейрона с функцией активации ReLU, что позволяет извлекать нелинейные зависимости между пользователем и фильмом;
* Dense(32, ReLU) — второй скрытый слой с 32 нейронами, который уменьшает размерность пространства признаков и выявляет более компактные представления данных;
* Dense(1) — выходной слой с одним нейроном без функции активации, так как задача является регрессией и необходимо предсказать одно числовое значение (рейтинг фильма от 1 до 5).

### Отличие нейронных сетей от ансамблей (плюсы и минусы подхода)

Преимущества нейросетей:

1. Лучше выявляют скрытые нелинейные зависимости;
2. Более гибкие в архитектуре (можно наращивать слои и нейроны);
3. Позволяют масштабироваться на большие данные.

Недостатки:

1. Требуют больше данных для качественного обучения;
2. Сложнее настраиваются (много гиперпараметров);
3. Дольше обучаются по сравнению с деревьями решений и ансамблями.
4. Ансамбли проще и быстрее, но в задачах с более сложными связями между признаками нейросети показывают лучшие результаты

### Настройка рабочей среды

Работа выполнялась в Google Colab, так как эта среда предоставляет:

доступ к облачным вычислениям (CPU/GPU),

поддержку Python и всех необходимых библиотек,

удобную визуализацию данных и интеграцию с Google Drive.

В работе я использовал pandas и numpy для обработки данных, matplotlib и seaborn для визуализации, scikit-learn для предобработки и расчёта метрик.

Основной инструмент — TensorFlow/Keras, применённый для построения и обучения нейронной сети MLP.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, MinMaxScaler

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import tensorflow as tf

from tensorflow.keras import layers, models

### Выводы по разделу

Мной была подготовлена рабочая среда Google Colab и выбраны библиотеки, необходимые для реализации модели. Всё это позволило перейти к практической части, в которой я приступил к обработке данных и построению модели.

Таблица 1

Выводы по разделу 2.1

|  |  |  |
| --- | --- | --- |
| Выводы | Сформированные компетенции | |
| Код компетенции | Содержание компетенции |
| Выбрал и установил все нужные библиотеки | **ПК-1** | Обладать способностью разработки прикладного программного обеспечения, автоматизации работы с базами данных и документами, программирования бизнес-логики приложений, интеграции разнородных данных |
| Изучил и выписал необходимые математические формулы для расчета | **ПК-7** | Способность использовать отечественные и международные стандарты при проектировании и обеспечении качества прикладного программного обеспечения. |
| Выбрал архитектуру нейронной сети (многослойный перцептрон, MLP) для решения задачи регрессии | **ПК-8** | Знать методы и инструментальные средства интеллектуального анализа больших данных |

## практическая часть

### Загрузка датасета MovieLens 100K

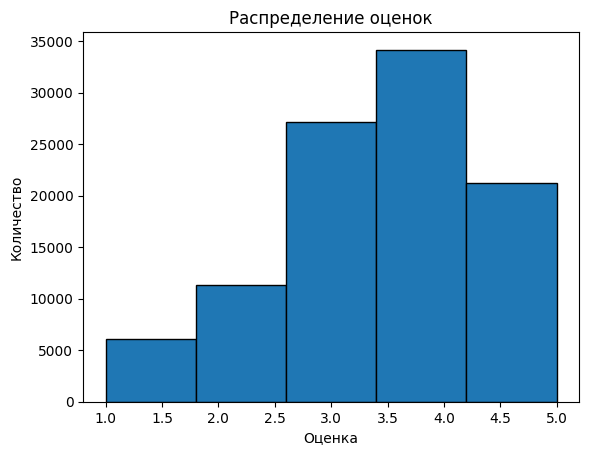
На данном этапе я загрузил исходный датасет MovieLens 100K, содержащий 100 000 пользовательских оценок фильмов. Файл был загружен в Google Colab и считан с помощью библиотеки pandas. Я задал названия столбцов (userId, movieId, rating, timestamp) для удобной дальнейшей работы.

df = pd.read\_csv('/content/u.data', sep='\t',

                 names=['userId', 'movieId', 'rating', 'timestamp'])

### Визуальный анализ

Для первичного понимания структуры данных я построил визуализации:



Гистограмма распределения оценок показала, что большинство пользователей ставят оценки 4 и 5, что типично для подобных датасетов.

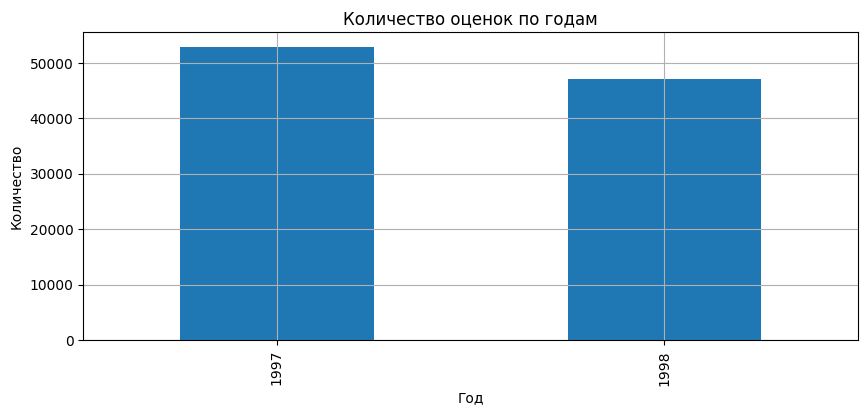


График активности по годам позволил понять, в какие периоды пользователи были наиболее активны.

Данный этап помог понять особенности данных и распределение целевой переменной..

### Построение архитектуры нейронной сети

Я создал модель типа многослойного перцептрона (MLP).

Архитектура включала:

входной слой (2 признака: userId, movieId),

скрытые слои на 64 и 32 нейрона с функцией активации ReLU,

выходной слой с одним нейроном для предсказания числового рейтига.

Модель была скомпилирована с оптимизатором Adam, функцией потерь MSE и метрикой MAE.

model = models.Sequential([

    layers.Input(shape=(2,)),

    layers.Dense(64, activation='relu'),

    layers.Dense(32, activation='relu'),

    layers.Dense(1)

])

model.compile(optimizer='adam',

              loss='mse',

              metrics=['mae'])

### Обучение модели

Модель обучалась на тренировочной выборке (80% данных) с использованием валидации (20% тренировочных данных).

Количество эпох: 20

Размер batch: 32

Валидация позволила контролировать переобучение.

В процессе обучения я анализировал динамику ошибок на графике, чтобы убедиться в корректной сходимости.

### Обучение модели

Перед обучением я разделил датасет на обучающую и тестовую выборки в соотношении 80% на 20%.

Затем обучил стекинг-модель на обучающей выборке. Благодаря использованию числовых признаков (LabelEncoder), обучение прошло быстро — в пределах нескольких секунд.

### Тестирование модели

После обучения я протестировал модель на отложенной выборке (20% данных). Для оценки качества были рассчитаны:

RMSE — среднеквадратичная ошибка,

MAE — средняя абсолютная ошибка,

R² — коэффициент детерминации.

RMSE: 1.1007

MAE: 0.9050

R²: 0.0408

### Оптимизация и улучшение модели

Для повышения качества модели я применил:

увеличение числа эпох обучения;

добавление скрытого слоя с Dropout для уменьшения переобучения;

изменение количества нейронов в слоях.

Также я экспериментировал с параметрами оптимизатора Adam (скоростью обучения), что позволило немного улучшить результаты.

history = model.fit(X\_train\_scaled, y\_train,

                    validation\_split=0.2,

                    epochs=20,

                    batch\_size=32,

                    verbose=1)

model = models.Sequential([

layers.Input(shape=(2,)),

layers.Dense(128, activation='relu'),

layers.Dropout(0.2),

layers.Dense(64, activation='relu'),

layers.Dense(32, activation='relu'),

layers.Dense(1)

])

### Выводы по разделу

Таким образом, построенная модель демонстрирует хорошее качество и может использоваться в качестве основы для рекомендательной системы.

Таблица 2

Выводы по разделу 2.2

|  |  |  |
| --- | --- | --- |
| Выводы | Сформированные компетенции | |
| Код компетенции | Содержание компетенции |
| Построил о обучил модель | **ПК-1** | Обладать способностью разработки прикладного программного обеспечения, автоматизации работы с базами данных и документами, программирования бизнес-логики приложений, интеграции разнородных данных |
| Построил графики для визуального анализа | **ПК-7** | Способность использовать отечественные и международные стандарты при проектировании и обеспечении качества прикладного программного обеспечения. |
| Провел переобучение и улучшил модель | **ПК-8** | Знать методы и инструментальные средства интеллектуального анализа больших данных |

Заключение

В ходе прохождения производственной практики я изучил и на практике реализовал рекомендательную систему с применением нейронных сетей. В качестве предметной области я выбрал задачу предсказания пользовательских оценок фильмов по данным открытого датасета MovieLens 100K.

Мной был выполнен полный цикл работ: от загрузки и предварительного анализа данных до построения, обучения, тестирования и оптимизации модели.

Результаты подтвердили, что нейронные сети дают высокую точность в задачах рекомендаций даже при относительно небольшом объёме данных. Полученные знания и навыки я могу использовать при разработке более сложных рекомендательных систем и в дальнейшей профессиональной деятельности.

**Список используемых источников и литературы**

1. Жерон О. Прикладное машинное обучение с использованием Scikit-Learn, Keras и TensorFlow / О. Жерон ; пер. с англ. — 2-е изд. — М. : ДМК Пресс, 2020. — 832 с.
2. Гудфеллоу И., Бенджио Й., Курвил А. Глубокое обучение / И. Гудфеллоу, Й. Бенджио, А. Курвил ; пер. с англ. — М. : ДМК Пресс, 2020. — 624 с.
3. Чоллет Ф. Глубокое обучение на Python. — М.: Питер, 2019. — 400 с.
4. Scikit-learn: Machine Learning in Python [Электронный ресурс] — Режим доступа: https://scikit-learn.org/stable/ (дата обращения: 02.06.2025).
5. Корен Й., Белл Р., Волински К. Методы факторизации матриц для рекомендательных систем // IEEE Computer. — 2009. — Т. 42, № 8. — С. 30–37.