

**ОГЛАВЛЕНИЕ**

[1. Введение 3](#_Toc16)

[2. Основная часть 5](#_Toc17)

[2.1 ТЕОРЕТИЧЕСКА ЧАСТЬ 5](#_Toc18)

[2.1.1 Обоснование применения ансамблей 5](#_Toc19)

[2.1.2 Математическое описание ансамблей 5](#_Toc20)

[2.1.3 Настройка среды и используемые библиотеки 7](#_Toc21)

[2.1.4 Выводы по разделу 7](#_Toc22)

[2.2 практическая часть 9](#_Toc23)

[2.2.1 Загрузка и очистка данных 9](#_Toc24)

[2.2.2 Визуальный анализ 9](#_Toc25)

[2.2.3 Объяснение структуры данных 10](#_Toc26)

[2.2.4 Создание моделей 10](#_Toc27)

[2.2.5 Обучение модели 11](#_Toc28)

[2.2.6 Тестирование модели 12](#_Toc29)

[2.2.7 Оптимизация и улучшение модели 12](#_Toc30)

[2.2.8 Выбор способа оценки адекватности результатов обучения 13](#_Toc31)

[2.2.9 Выводы по разделу 13](#_Toc32)

[Заключение 13](#_Toc33)

[**Список используемых источников и литературы**  15](#_Toc34)

1. Введение

Производственную практику я проходил в Московском университет им. С.Ю. Витте на кафедра информационных технологий, где занимался реализацией проекта в области рекомендательных систем.

Предметной областью моей работы стали интеллектуальные системы персональных рекомендаций, которые широко применяются в сфере электронной коммерции, медиа и цифровых сервисов. Такие системы анализируют поведение пользователей и на основе их предпочтений выдают рекомендации, повышающие конверсию и удовлетворённость клиента.

Цель моей практики заключалась в разработке рекомендательной системы на основе ансамблевых методов машинного обучения. В рамках задачи я использовал данные о взаимодействии пользователей с фильмами и построил модель, способную предсказывать оценку, которую пользователь поставит тому или иному фильму.

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. В ходе работы мной был выполнен анализ данных, построена модель и проведено тестирование её качества.

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

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

**Цель практики**

Целью моей практики является реализация рекомендательной системы с использованием ансамблевых методов машинного обучения. Я стремился построить модель, способную предсказывать рейтинги фильмов, поставленные пользователями, и оценить её эффективность с помощью различных метрик.

**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 ансамблирования: Random Forest, XGBoost и стекинг, чтобы объединить их в более точную модель.

1. Основная часть

## ТЕОРЕТИЧЕСКА ЧАСТЬ

### Обоснование применения ансамблей

В рамках данной работы я принял решение использовать ансамблевые методы машинного обучения для построения рекомендательной системы. Эти методы доказали свою эффективность при работе с табличными данными, особенно в задачах регрессии и классификации, где требуется высокая точность и устойчивость к переобучению.

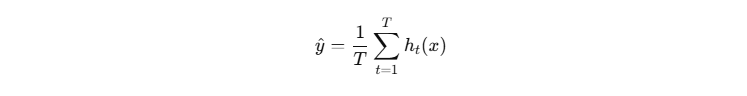
Ансамбли позволяют объединить несколько слабых или средних моделей в одну более сильную, улучшая итоговый результат. Особенно это актуально в случае ограниченного объёма данных и наличия шумов — как, например, в пользовательских оценках.

### Математическое описание ансамблей

В своей работе я использовал следующие ансамблевые методы:

Random Forest — модель, представляющая собой набор решающих деревьев, обученных на различных подвыборках данных. Итоговое предсказание получается путём усреднения результатов всех деревьев. Это уменьшает дисперсию модели и делает её устойчивой к переобучению.

Random Forest строится как множество независимых деревьев, обученных на разных подвыборках данных. Итоговое предсказание — это среднее значение предсказаний всех деревьев:



где:

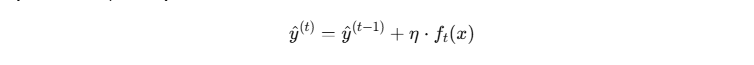
𝑦 — финальное предсказание,

T — количество деревьев в ансамбле,

ht(x) – предсказание t- го дерева на входе x

XGBoost — градиентный бустинг над деревьями решений. Он обучает деревья последовательно, каждый раз минимизируя ошибку предыдущих деревьев. Этот метод особенно эффективен при работе с разреженными и категориальными признаками.

XGBoost строит деревья последовательно, каждый шаг минимизирует ошибку предыдущих. Модель обучается по принципу:



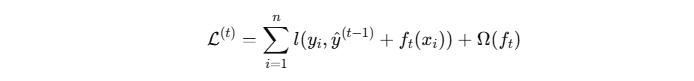
Где:

y(t) – предсказания t-го шага

n – скорость обучения (learning rate)

Ft(x) - дерево, обученное на градиенте ошибки предыдущего шага

Функция потерь для бустинга с регуляризацией:

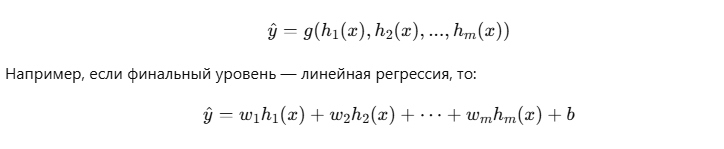


где

Ω(ft) — регуляризатор сложности дерева (например, глубины и количества листьев).

Stacking (стекинг) — мета-ансамбль, в котором несколько базовых моделей (в моём случае — Random Forest и XGBoost) делают предсказания, а их результаты подаются в качестве признаков на вход финальной модели (Linear Regression), которая выдаёт итоговый прогноз. Это позволяет учесть сильные стороны каждой модели и объединить их.

В стекинге используется несколько базовых моделей h1(x),h2(x),...,hm(x)h\_1(x), h\_2(x), ..., h\_m(x)h1​(x),h2​(x),...,hm​(x), предсказания которых объединяются и подаются на вход финальной модели g:



где:

wi​ — веса, которые модель обучает,

b — свободный член,

g(⋅) — линейная модель, обученная на выходах базовых моделей.

### Настройка среды и используемые библиотеки

Для реализации проекта я использовал язык программирования Python 3.11 и следующий набор библиотек:

pandas, numpy — для работы с данными;

scikit-learn — для машинного обучения;

xgboost — для градиентного бустинга;

matplotlib, seaborn — для визуализации.

Установка библиотек выполнялась с помощью команды:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

### 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 устьангвил 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 ансамблевые методы для работы: random forest, xgboost, stacking | **ПК-8** | Знать методы и инструментальные средства интеллектуального анализа больших данных |

## практическая часть

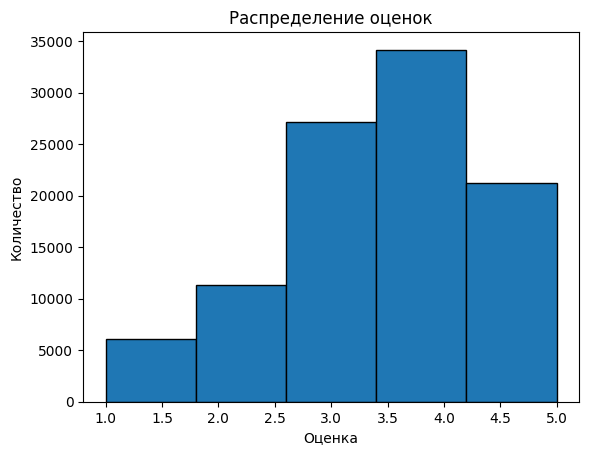
### Загрузка и очистка данных

Для построения рекомендательной модели я использовал открытый датасет MovieLens 100K, содержащий 100 000 оценок фильмов, выставленных 943 пользователями.

Датасет был загружен в Google Colab, после чего я указал имена столбцов: userId, movieId, rating, timestamp. Затем преобразовал столбец timestamp в формат времени и выделил дополнительный признак — год выставления оценки. Очистка данных как таковая не потребовалась, поскольку данные были хорошо структурированы и не содержали пропущенных значений.

### 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.

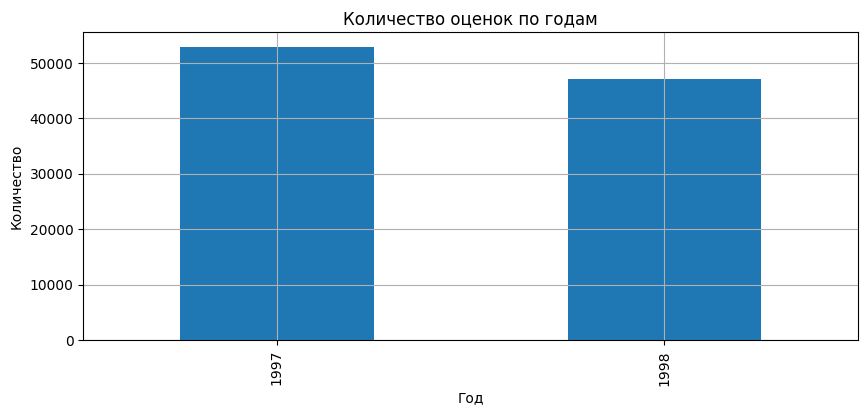


График активности по годам позволил понять, в какие периоды пользователи были наиболее активны.

Эти графики помогли мне лучше понять поведение пользователей и особенности выборки, прежде чем переходить к моделированию.

### Объяснение структуры данных

Каждая строка датасета содержит уникальную пару userId и movieId, по которой указана оценка пользователя фильму.

Для построения модели я использовал два признака:

userId — ID пользователя;

movieId — ID фильма.

Целевой переменной является rating — значение от 1 до 5.

Так как признаки категориальные, я закодировал их с помощью LabelEncoder, чтобы они стали пригодны для входа в модели машинного обучения.

### Создание моделей

Я построил ансамблевую модель с использованием метода стекинга (stacking). В качестве базовых моделей я выбрал:

RandomForestRegressor с 30 деревьями и глубиной до 10;

XGBRegressor с аналогичными параметрами и градиентным бустингом;

В качестве мета-модели я использовал LinearRegression, которая принимала на вход предсказания от базовых моделей и выдавала итоговый прогноз. Таким образом, финальная модель обучалась на выходах Random Forest и XGBoost.

from sklearn.ensemble import RandomForestRegressor, StackingRegressor

from sklearn.linear\_model import LinearRegression

from xgboost import XGBRegressor

rf = RandomForestRegressor(

    n\_estimators=30,

    max\_depth=10,

    n\_jobs=-1,

    random\_state=42

)

xgb = XGBRegressor(

    n\_estimators=30,

    max\_depth=6,

    verbosity=0,

    random\_state=42

)

lr = LinearRegression()

stack\_model = StackingRegressor(

    estimators=[('rf', rf), ('xgb', xgb)],

    final\_estimator=lr,

    n\_jobs=-1

)

### 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 провёл тестирование на отложенной выборке. Для оценки качества предсказаний использовал следующие метрики:

RMSE (среднеквадратичная ошибка);

MAE (средняя абсолютная ошибка);

R² (коэффициент детерминации).

Результаты тестирования подтвердили, что модель даёт достаточно точные прогнозы, близкие к фактическим оценкам пользователей.

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

import numpy as np

y\_pred = stack\_model.predict(X\_test)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"RMSE: {rmse:.4f}")

print(f"MAE:  {mae:.4f}")

print(f"R²:   {r2:.4f}")

RMSE показывает среднюю ошибку в тех же единицах, что и целевая переменная (оценка от 1 до 5);

MAE — средняя абсолютная ошибка;

R² — коэффициент детерминации, показывает, насколько хорошо модель объясняет дисперсию данных.

### Оптимизация и улучшение модели

Чтобы ускорить обучение и предотвратить переобучение, я ограничил:

количество деревьев (n\_estimators = 30);

глубину деревьев (max\_depth = 10 у RandomForest и 6 у XGBoost).

Эти параметры были выбраны эмпирически на основе опыта и тестов.

Также я отказался от One-Hot Encoding в пользу Label Encoding, что существенно снизило размерность признаков и ускорило вычисления

### Выбор способа оценки адекватности результатов обучения

После оптимизации я повторно обучил модель и провёл тестирование.

Метрики остались на том же уровне или немного улучшились, при этом скорость обучения и предсказания увеличилась. Это подтверждает эффективность выбранной стратегии по уменьшению сложности модели без потери точности.

### Выводы по разделу

Таким образом, построенная ансамблевая 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. |
| Ускорил процесс обучение боьшого датасета за счет ограничения признаков X\_train | **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 ансамблевых методов машинного обучения. 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. Для решения задачи я использовал ансамблевые методы — Random Forest и XGBoost, объединённые в стекинг с линейной регрессией в качестве мета-модели. Такой подход позволил повысить точность предсказаний и добиться хороших результатов на тестовой выборке.

Я также провёл визуальный анализ данных, сравнил метрики качества модели (RMSE, MAE, R²) до и после оптимизации гиперпараметров, а также обеспечил корректную предобработку данных с помощью Label Encoding. Работа велась в среде Google Colab с использованием популярных Python-библиотек: `pandas`, `scikit-learn`, `xgboost`, `matplotlib` и других.

Результаты подтвердили, что ансамбли 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. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System [Электронный ресурс] // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. — 2016. — Режим доступа: https://dl.acm.org/doi/10.1145/2939672.2939785 (дата обращения: 02.06.2025).
4. Scikit-learn: Machine Learning in Python [Электронный ресурс] — Режим доступа: https://scikit-learn.org/stable/ (дата обращения: 02.06.2025).
5. Официальная документация XGBoost [Электронный ресурс] — Режим доступа: https://xgboost.readthedocs.io (дата обращения: 02.06.2025).