

Информационно-технологический проект

# Разработка интеллектуальной системы

## «Подбор фильма по интересам»

---

Автор: Краснобрыж Платон Валерьевич

Руководитель: Мещеряков Никита

# Обоснование актуальности

Зрители часто тратят на поиск фильма больше времени, чем на просмотр, сталкиваясь с **«парадоксом выбора»**.

Популярные стриминги навязывают контент в своих интересах, скрывая механизмы рекомендаций.

**Актуальность проекта** — в создании прозрачного инструмента, который подбирает фильмы математически точно, основываясь на объективной схожести сюжетов.



# Цель и Задачи

---

## ❶ Цель

Создание программного продукта на языке **Python**, который автоматизирует подбор фильмов, используя алгоритм k-ближайших соседей (KNN), и предоставляет пользователю топ-5 рекомендаций с наивысшим рейтингом.

## ❷ Задачи

- Найти и выполнить предобработку датасета (очистка, One-Hot Encoding).
- Реализовать и сравнить три математические модели поиска (Cosine, Euclidean, Manhattan).
- Разработать алгоритм гибридной фильтрации: поиск по схожести + сортировка по рейтингу.
- Протестировать систему на реальных запросах и оценить качество выдачи.

# Этапы работы над проектом

---



## 1. Подготовительный

Выбор темы, анализ проблемы, поиск датасета (IMDb) и изучение теории.



## 2. Основной

Написание кода в Google Colab, обучение ML моделей, интерактивный поиск.



## 3. Финальный

Тестирование на жанрах, сравнение метрик, оформление документации и презентации.

# Ресурсы проекта

---

## Материальные:

ПК, Интернет.

## Трудовые:

Разработчик, консультации руководителя.

## Информационные:

Документация Scikit-learn, Pandas, датасет IMDb.

## Финансовые:

0 руб. (бесплатное ПО).

## Программные:

Google Colab, язык Python.

# Обзор проекта (Часть 1. Подготовка Данных)

## Подготовка данных

- Использован датасет **IMDb** (~5500 фильмов).
- Проверяем, есть ли пропуски.
- Жанры преобразованы в бинарные векторы методом One-Hot Encoding (создано >150 признаков).
- Год выпуска нормализован (StandardScaler) для баланса весов с жанрами.
- Рейтинг исключен из обучения, чтобы искать фильмы по смыслу, а не по оценке.

```
...   ⚒ Статистика пропусков до обработки:  
main_genre      0  
side_genre       0  
Year             0  
Rating           0  
dtype: int64  
-----  
✨ Пропусков в жанрах нет.  
-----  
✅ Финальная таблица признаков собрана.  
Размер X_final: (5562, 158)
```

...

✓ Файл открыт! Размер таблицы: (5562, 10)

	Movie_Title	Year	Director	Actors	Rating	Runtime(Mins)	Censor	Total_Gross	main_genre	side_genre
0	Kantara	2022	Rishab Shetty	Rishab Shetty, Sapthami Gowda, Kishore Kumar G...	9.3	148	UA	Gross Unkown	Action	Adventure, Drama
1	The Dark Knight	2008	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart, M...	9.0	152	UA	\$534.86M	Action	Crime, Drama
2	The Lord of the Rings: The Return of the King	2003	Peter Jackson	Elijah Wood, Viggo Mortensen, Ian McKellen, Or...	9.0	201	U	\$377.85M	Action	Adventure, Drama

# Обзор проекта (Часть 2. Модели)

## Архитектура решения

Использован алгоритм **k-Nearest Neighbors (KNN)**.

Настройка: Поиск 20 ближайших соседей (n\_neighbors=20) для формирования широкого пула кандидатов.

Обучено 3 модели с разными метриками расстояния:

### 1. Cosine

Лучше всего определяет жанровую схожесть.

### 2. Euclidean

Классическая геометрия (сильнее учитывает год)

### 3. Manhattan

Устойчива к выбросам.

```
# Создаем словарь для хранения моделей
models = {}
|
print("Начинаем обучение моделей...")

# МОДЕЛЬ 1: KNN с Косинусным расстоянием (Cosine)

print("1. Обучаем модель с метрикой Cosine...")
model_cosine = NearestNeighbors(n_neighbors=20, metric='cosine', n_jobs=-1)
model_cosine.fit(X_final) # Обучаем на нашей таблице (Жанры + Год)
models['Cosine'] = model_cosine

# МОДЕЛЬ 2: KNN с Евклидовым расстоянием (Euclidean)

print("2. Обучаем модель с метрикой Euclidean...")
model_euclidean = NearestNeighbors(n_neighbors=20, metric='euclidean', n_jobs=-1)
model_euclidean.fit(X_final)
models['Euclidean'] = model_euclidean

# МОДЕЛЬ 3: KNN с Манхэттенским расстоянием (Manhattan)

# Расстояние "городских кварталов" (сумма разниц по модулю).
print("3. Обучаем модель с метрикой Manhattan...")
model_manhattan = NearestNeighbors(n_neighbors=20, metric='manhattan', n_jobs=-1)
model_manhattan.fit(X_final)
models['Manhattan'] = model_manhattan
```

# Обзор проекта (Часть 3. Результат)

## Демонстрация работы

1. Пользователь вводит название фильма.
2. Система находит **20 «соседей»** по векторному пространству.
3. Алгоритм сортирует их по рейтингу.
4. Выдает **Топ-5 лучших** рекомендаций.

```
... ⚡ Введите название фильма на английском:  
Название фильма: The Matrix  
  
⚡ Введите название фильма на английском:  
Название фильма: The matrix  
🎬 Вы выбрали: The Matrix (1999)  
Жанры: Action, Sci-Fi  
★ Рейтинг: 8.7  
=====
```

# Сравнение моделей

## ● МОДЕЛЬ: Cosine

1. Terminator 2: Judgment Day (1991)  
Жанры: Action, Sci-Fi  
★ 8.6 | dist: 0.0483
2. The Terminator (1984)  
Жанры: Action, Sci-Fi  
★ 8.1 | dist: 0.1342
3. The Matrix Reloaded (2003)  
Жанры: Action, Sci-Fi  
★ 7.2 | dist: 0.0151
4. The Wolverine (2013)  
Жанры: Action, Sci-Fi  
★ 6.7 | dist: 0.1663
5. The Matrix Revolutions (2003)  
Жанры: Action, Sci-Fi  
★ 6.7 | dist: 0.0151

## ● МОДЕЛЬ: Euclidean

1. Terminator 2: Judgment Day (1991)  
Жанры: Action, Sci-Fi  
★ 8.6 | dist: 0.4956
2. The Terminator (1984)  
Жанры: Action, Sci-Fi  
★ 8.1 | dist: 0.9292
3. The Boondock Saints (1999)  
Жанры: Action, Thriller  
★ 7.7 | dist: 1.4142
4. The Matrix Reloaded (2003)  
Жанры: Action, Sci-Fi  
★ 7.2 | dist: 0.2478
5. Arlington Road (1999)  
Жанры: Action, Crime, Drama  
★ 7.2 | dist: 1.4142

## ● МОДЕЛЬ: Manhattan

1. Terminator 2: Judgment Day (1991)  
Жанры: Action, Sci-Fi  
★ 8.6 | dist: 0.4956
2. The Terminator (1984)  
Жанры: Action, Sci-Fi  
★ 8.1 | dist: 0.9292
3. The Boondock Saints (1999)  
Жанры: Action, Thriller  
★ 7.7 | dist: 2.0000
4. The Matrix Reloaded (2003)  
Жанры: Action, Sci-Fi  
★ 7.2 | dist: 0.2478
5. Three Kings (1999)  
Жанры: Action, Adventure, Comedy  
★ 7.1 | dist: 2.0000

# Выводы

---

## ❖ Достижения

- **Цель достигнута:** Разработан функциональный скрипт на Python.
- **Задачи решены:** Данные обработаны, модели обучены, поиск работает.

## ☒ Результаты

- **Продукт:** Интеллектуальный помощник решает проблему выбора за секунды.
- **Эффективность:** Сравнение показало, что метрика Cosine дает наиболее разнообразные рекомендации, а Euclidean и Manhattan сильнее зависят от года выпуска.

# Перспективы развития проекта

---



## Telegram-бот

Интеграция скрипта с  
мессенджером для удобного  
доступа с телефона.



## Расширение базы

Добавление новых фильмов  
2024-2025 годов и российских  
картин.



## Учет актеров

Добавление режиссеров и  
актерского состава в признаки  
модели для точности.

# Источники информации

---

-  Документация Scikit-learn: [scikit-learn.org](http://scikit-learn.org)
-  Документация Pandas: [pandas.pydata.org](http://pandas.pydata.org)
-  Kaggle Dataset (IMDb Movies)
-  Материалы курса по Machine learning и Deep learning

# Смотрите хорошее кино!

Проект имеет открытый исходный код. Вы можете изучить  
реализацию и протестировать алгоритмы.



<https://github.com/platonkrasnobryz6323-blip/film-selection>