ALEKSANDRA KWAST

Programista Python Developer – praca zaliczeniowa

Spis treści

1. Cel projektu

Celem niniejszego projektu było stworzenie interaktywnego systemu rekomendacji filmów, który umożliwia użytkownikowi wyszukiwanie i otrzymywanie propozycji filmowych na podstawie różnych kryteriów, takich jak historia wyszukiwań, podobieństwo do podanego tytułu, minimalna ocena czy preferowany gatunek. Projekt został zrealizowany w języku Python z wykorzystaniem bibliotek takich jak pandas, scikit-learn, rapidfuzz oraz numpy.

2. Opis realizacji

System opiera się na danych z pliku imdbpl.csv, zawierającego informacje o filmach, takie jak tytuł, ocena IMDb, liczba ocen, rok produkcji, czas trwania oraz przynależność do wybranych gatunków filmowych. Dane te są przetwarzane i standaryzowane, a następnie wykorzystywane do obliczania podobieństw między filmami za pomocą algorytmu k-NN (k-Nearest Neighbors).

Użytkownik może korzystać z systemu w trybie interaktywnym, wpisując tytuły filmów lub komendy sterujące, takie jak:

- recommend – rekomendacje na podstawie historii,
- surprise me – losowa propozycja filmu,
- minrating – ustawienie minimalnej oceny,
- filter genre – filtrowanie po gatunku,
- clear filters – usunięcie aktywnych filtrów.

Projekt zawiera również funkcję zapisu historii wyszukiwań do pliku CSV, co umożliwia analizę preferencji użytkownika w przyszłości.

2.1. Archiwum danych

Archiwum danych używane w projekcie zawiera informacje o filmach i jest zapisane w pliku imdbPL.csv. Plik ten zawiera blisko 10k rekordów, z których każdy reprezentuje jeden film. Dane te obejmują szeroki zakres lat produkcji filmów — od 1913 do 2014 roku.

2.1.1. Struktura danych

Dane w pliku imdbPL.csv są zorganizowane w następujące kolumny:

fn, tid, title, wordsInTitle, url, imdbRating, ratingCount, duration, year, type, nrOfWins, nrOfNominations, nrOfPhotos, nrOfNewsArticles, nrOfUserReviews, nrOfGenre, Action, Adult, Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, FilmNoir, GameShow, History, Horror, Music, Musical, Mystery, News, RealityTV, Romance, SciFi, Short, Sport, TalkShow, Thriller, War, Western

2.1.2. Typy informacji

Każdy rekord w archiwum zawiera m.in.:

- tytuł filmu,
- ocenę IMDb,
- liczbę ocen użytkowników,
- rok produkcji,
- czas trwania filmu (w sekundach),
- przynależność do jednego lub wielu gatunków filmowych (reprezentowanych jako kolumny binarne).

2.1.3. Idea użycia bazy danych

Zebrane dane są wykorzystywane w systemie rekomendacji do obliczania podobieństw między filmami oraz do filtrowania i sortowania wyników na podstawie preferencji użytkownika. Informacje o ocenach, liczbie ocen, roku produkcji i gatunkach filmowych pozwalają na dokładne dopasowanie rekomendacji do indywidualnych potrzeb użytkownika.

3. Dostępne funkcje

System rekomendacji filmów został zaprojektowany z myślą o użytkowniku – jego wygodzie, ciekawości i potrzebie odkrywania wartościowych tytułów. Interfejs tekstowy, choć prosty w formie, oferuje bogaty zestaw funkcji, które pozwalają na dynamiczne sterowanie rekomendacjami i filtrowanie wyników zgodnie z indywidualnymi preferencjami. Poniżej przedstawiono szczegółowy opis dostępnych komend:

- **Wyszukiwanie filmu po tytule**
  Użytkownik może rozpocząć interakcję, wpisując nazwę dowolnego filmu. System automatycznie analizuje wprowadzone dane i porównuje je z tytułami w bazie, wykorzystując algorytmy dopasowania tekstowego. W przypadku niejednoznaczności, użytkownik otrzymuje sugestię najbliższego dopasowania i może potwierdzić lub odrzucić propozycję. To rozwiązanie pozwala na szybkie odnalezienie filmu nawet przy literówkach czy niepełnych nazwach.
- **recommend – rekomendacje na podstawie historii**
  Po wpisaniu tej komendy system analizuje dotychczasową historię wyszukiwań użytkownika i proponuje filmy, które są najbardziej zbliżone do wcześniej przeglądanych tytułów. Wykorzystywany jest algorytm k-NN (k-nearest neighbors), który na podstawie cech takich jak ocena IMDb, rok produkcji, czas trwania i gatunki, oblicza podobieństwo między filmami. To funkcja idealna dla tych, którzy chcą otrzymywać trafne, spersonalizowane propozycje.
- **surprise me – losowa propozycja filmu**
  Dla użytkowników szukających inspiracji lub chcących wyjść poza schemat, dostępna jest komenda surprise me. System losowo wybiera film z bazy danych, który spełnia określone kryteria jakościowe (np. minimalna ocena) oraz ewentualne filtry gatunkowe. Co ważne, unika powtórzeń, nie proponując filmów już wcześniej oglądanych. To doskonały sposób na odkrycie czegoś nowego i nieoczywistego.
- **minrating [ocena] – ustawienie minimalnej oceny**
  Użytkownik może określić dolny próg jakości filmów, wpisując np. minrating 7.5. Od tego momentu system będzie uwzględniał w rekomendacjach wyłącznie filmy, które osiągnęły wskazaną ocenę lub wyższą. To przydatne narzędzie dla osób, które chcą ograniczyć propozycje do tytułów wysoko ocenianych przez społeczność IMDb.
- **filter genre [gatunek] – filtrowanie po gatunku**
  Pozwala zawęzić rekomendacje do konkretnego gatunku filmowego, np. filter genre Drama. Po aktywacji tego filtra system będzie uwzględniał wyłącznie filmy przypisane do wskazanej kategorii. To funkcja szczególnie przydatna, gdy użytkownik ma ochotę na konkretny klimat – komedię, thriller, science fiction czy dramat.

- **clear filters – usunięcie aktywnych filtrów**
  Komenda ta przywraca domyślne ustawienia systemu, usuwając wszystkie wcześniej ustawione filtry, takie jak minimalna ocena czy gatunek. Dzięki temu użytkownik może ponownie przeglądać pełną gamę filmów bez ograniczeń.
- **genres – lista dostępnych gatunków**
  Wyświetla pełną listę gatunków filmowych, które można wykorzystać w filtrze filter genre. Ułatwia to użytkownikowi wybór poprawnej nazwy gatunku i zapobiega błędom składniowym.
- **exit – zakończenie sesji**
  Po zakończeniu pracy z systemem, użytkownik może wpisać exit, co powoduje zapis historii wyszukiwań i rekomendacji do pliku historia_uzytkownika.csv. To pozwala na zachowanie kontekstu i analizę preferencji w przyszłości.

4. Pomysły na dalsze usprawnienie programu

Obecna wersja systemu rekomendacji filmów stanowi solidną podstawę funkcjonalną, jednak istnieje wiele możliwości dalszego rozwoju, które mogłyby znacząco zwiększyć jego użyteczność, elastyczność i atrakcyjność dla użytkownika. Poniżej przedstawiono propozycje usprawnień, które warto rozważyć w kolejnych etapach rozwoju projektu:

- **Wprowadzenie profilu użytkownika**
  Aktualnie, historia wyszukiwania zapisywana jest lokalnie i tymczasowo. Rozszerzenie systemu o możliwość tworzenia profili użytkowników pozwoliłoby na trwałe przechowywanie preferencji, historii oraz personalizację rekomendacji między sesjami. Umożliwiłoby to także obsługę wielu użytkowników w jednej instancji programu.
- **Rozbudowa filtrów wyszukiwania**
  Aktualnie użytkownik może ustawić minimalną ocenę i jeden filtr gatunkowy. W przyszłości warto umożliwić filtrowanie po wielu gatunkach jednocześnie, roku produkcji, czasie trwania filmu czy liczbie ocen. Pozwoliłoby to na bardziej precyzyjne dopasowanie wyników do oczekiwań użytkownika.
- **Integracja z zewnętrznymi bazami danych (np. TMDb, OMDb)**
  Zastosowana baza danych zawiera podstawowe informacje o filmach. Integracja z zewnętrznymi API umożliwiłaby wzbogacenie danych o opisy fabularne, obsadę, plakaty, zwiastuny czy recenzje, co zwiększyłoby wartość informacyjną rekomendacji.
- **System oceniania i opinii użytkownika**
  Dodanie możliwości oceniania rekomendowanych filmów przez użytkownika pozwoliłoby systemowi lepiej rozpoznawać preferencje i dostosowywać przyszłe propozycje. Opinie mogłyby być również zapisywane w historii i wykorzystywane do analizy trendów.

Naturalny język zapytań
Zamiast sztywnych komend, użytkownik mógłby wpisywać zapytania w języku naturalnym, np. „poleć mi dramat z lat 90. z oceną powyżej 8”. Wprowadzenie takiej funkcjonalności zwiększyłoby intuicyjność i dostępność systemu.

- **Interfejs graficzny (GUI)**
  Choć interfejs tekstowy jest funkcjonalny, dodanie graficznego interfejsu użytkownika (np. w formie aplikacji desktopowej lub webowej) mogłoby znacząco poprawić komfort obsługi, szczególnie dla mniej zaawansowanych użytkowników.
- **Tryb rekomendacji kontekstowej**
  Możliwość wyboru rekomendacji zależnych od nastroju, pory dnia czy okazji (np. „coś lekkiego na wieczór”, „film na weekend z rodziną”) uczyniłaby system bardziej elastycznym i przyjaznym w codziennym użytkowaniu.
- **Wersja mobilna lub offline**
  Stworzenie wersji mobilnej lub działającej offline zwiększyłoby dostępność systemu w różnych warunkach – np. w podróży, bez dostępu do internetu.

5. Struktura programu

5.1. Import i przygotowanie środowiska

Pierwsze linijki kodu zestaw bibliotek, które nadają programowi funkcjonalność. pandas i numpy odpowiadają za manipulację danymi, datetime i csv za rejestrację historii, random za element losowości, a scikit-learn dostarcza narzędzi do uczenia maszynowego. rapidfuzz pełni rolę strażnika językowego, umożliwiając dopasowanie tytułów filmów mimo literówek czy niepełnych nazw.

5.2. Wczytywanie i przetwarzanie danych

Moduł ten rozpoczyna się od załadowania pliku imdbPL.csv, który stanowi serce systemu – archiwum filmowe. Dane są oczyszczane z braków, a następnie przekształcane w zestaw cech numerycznych i binarnych. Gatunki filmowe są reprezentowane jako kolumny zero-jedynkowe, co pozwala na ich łatwe uwzględnianie w analizie. Dane są standaryzowane, a na ich podstawie budowany jest model k-NN, który umożliwia wyszukiwanie podobnych filmów w przestrzeni cech.

5.3. Zmienne globalne i pamięć użytkownika

Tutaj definiowane są zmienne sterujące: min_rating i genre_filter, które odpowiadają za aktywne filtry, oraz user_history i user_history_log, które rejestrują interakcje użytkownika. To swoista pamięć programu – dzięki niej program „wie”, co już było oglądane i może unikać powtórzeń.

5.4. Funkcje pomocnicze

Ten moduł zawiera funkcje wspierające logikę programu:

- get_genres(row) tłumaczy binarne dane gatunkowe na czytelny opis.
- matches_filters(film) sprawdza, czy dany film spełnia aktywne kryteria.
- find_best_match(title_input) to inteligentne dopasowanie tytułu – nie tylko szuka, ale też pyta użytkownika o potwierdzenie.
- save_history_to_file() dba o trwałość danych, zapisując historię do pliku CSV.

5.5. Mechanizmy rekomendacyjne

To serce systemu – zestaw funkcji, które odpowiadają za generowanie rekomendacji:

- recommend_similar(title_input) znajduje filmy podobne do wskazanego tytułu.
- recommend_from_history() analizuje historię użytkownika i proponuje filmy zgodne z jego gustem.
- surprise_me() to element losowości – wybiera film, który może zaskoczyć, ale nadal spełnia określone kryteria.

Każda z tych funkcji korzysta z modelu k-NN i filtrów, tworząc rekomendacje, które są zarówno trafne, jak i zróżnicowane.

5.6. Interfejs użytkownika

Moduł interactive_loop() to brama do świata programu. Obsługuje wszystkie komendy wpisywane przez użytkownika, od wyszukiwania tytułów po zarządzanie filtrami.

5.7. Sekcja testów i uruchamianie programu

Końcowa część programu pełni funkcję inicjalizacyjną i przygotowuje użytkownika do pracy z systemem rekomendacji. Zamiast statycznych testów, jak miało to miejsce we wcześniejszej wersji, obecna implementacja wprowadza bardziej angażujący i spersonalizowany sposób rozpoczęcia sesji. Po uruchomieniu programu użytkownik proszony jest o podanie trzech swoich ulubionych filmów. System wykorzystuje te dane jako punkt wyjścia do wygenerowania pierwszych rekomendacji – są one oparte na analizie podobieństwa do wskazanych tytułów. Dzięki temu już na starcie użytkownik otrzymuje propozycje dopasowane do własnego gustu. Następnie uruchamiany jest tryb interaktywny (interactive_loop()), który umożliwia dalszą eksplorację bazy filmów, ustawianie filtrów, wyszukiwanie tytułów oraz korzystanie z funkcji takich jak recommend, surprise me czy minrating. Cała logika programu zostaje w tym momencie oddana w ręce użytkownika, który może w pełni sterować systemem za pomocą prostych komend tekstowych.Taka struktura uruchomienia programu nie tylko zwiększa zaangażowanie użytkownika, ale również pozwala na natychmiastowe dostosowanie rekomendacji do jego indywidualnych preferencji.

6. Kod źródłowy

```
import pandas as pd
import datetime
import csv
import random
from sklearn.neighbors import NearestNeighbors
from sklearn.preprocessing import StandardScaler
from rapidfuzz import process

# Wczytanie danych z pliku imdbpl.csv
df = pd.read_csv("imdbpl.csv")

feature_cols = [
    'imdbRating', 'ratingCount', 'duration', 'year',
] + [
    'Action','Adult','Adventure','Animation','Biography','Come-
dy','Crime','Documentary','Drama',
    'Family','Fantasy','FilmNoir','GameShow','History','Horror','Mu-
sic','Musical','Mystery',
    'News','RealityTV','Ro-
mance','SciFi','Short','Sport','TalkShow','Thriller','War','Western'
]

df_clean = df.dropna(subset=['title'] + feature_cols).reset_in-
dex(drop=True)
X = df_clean[feature_cols]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

knn = NearestNeighbors(n_neighbors=6, metric='euclidean')
knn.fit(X_scaled)

# Filtry
min_rating = 5.0
genre_filter = None

# Historia użytkownika
user_history_log = []
user_history = []

def get_genres(row):
    return ', '.join([g for g in feature_cols[4:] if row[g] == 1])

def find_best_match(title_input):
    titles = df_clean['title'].tolist()
    match_data = process.extractOne(title_input, titles)
    if match_data:
```

```
        match, score, _ = match_data
        if score < 85:
            print(f"⚠ Znaleziono podobny tytuł: {match} (trafność:
{score}%)")
            confirm = input("Czy chodziło Ci o ten film? (t/n):
").strip().lower()
            if confirm != 't':
                return None
        return match
    return None

def save_history_to_file():
    if not user_history_log:
        return
    with open("historia_uzytkownika.csv", "a", newline='', encod-
ing='utf-8') as file:
        writer = csv.writer(file)
        for record in user_history_log:
            writer.writerow(record)

def surprise_me():
    candidates = df_clean[df_clean['imdbRating'] >= min_rating]
    if genre_filter:
        candidates = candidates[candidates[genre_filter] == 1]
    candidates = candidates[~candidates['title'].isin(user_history)]  #
unikaj powtórzeń

    if candidates.empty:
        print("❗ Brak filmów do zaproponowania w trybie 'surprise
me'.")
        return

    film = candidates.sample(1).iloc[0]
    print("\n🎁 Niespodzianka! Spróbuj tego filmu:")
    print(f"👉 {film['title']} (rok: {int(film['year'])}, ocena:
{film['imdbRating']}, gatunki: {get_genres(film)})")

    user_history.append(film['title'])
    user_history_log.append([film['title'],
datetime.datetime.now().isoformat(), 'surprise'])

def matches_filters(film):
    global min_rating, genre_filter
    if film['imdbRating'] < min_rating:
        return False
    if genre_filter:
        return film[genre_filter] == 1
    return True
```

```
def recommend_similar(title_input):
    title_match = find_best_match(title_input)
    if not title_match:
        print(f"❌ Nie znaleziono filmu podobnego do: {title_input}")
        return
    idx = df_clean[df_clean['title'] == title_match].index[0]
    distances, indices = knn.kneighbors([X_scaled[idx]])
    print(f"\n🎬 Rekomendacje dla: {title_match} (ocena:
{df_clean.iloc[idx]['imdbRating']})")
    found = False
    for i in indices[0][1:]:
        film = df_clean.iloc[i]
        if matches_filters(film):
            print(f"👉 {film['title']} (rok: {int(film['year'])},
ocena: {film['imdbRating']}, gatunki: {get_genres(film)})")
            found = True
    if not found:
        print("❗ Brak rekomendacji spełniających aktywne filtry.")

def recommend_from_history():
    if not user_history:
        print("👎 Brak historii wyszukiwań.")
        return
    print(f"\n📋 Twoja historia: {', '.join(user_history)}")
    user_vectors = []
    for title in user_history:
        idx = df_clean[df_clean['title'] == title].index[0]
        user_vectors.append(X_scaled[idx])
    import numpy as np
    avg_vector = np.mean(user_vectors, axis=0).reshape(1, -1)
    distances, indices = knn.kneighbors(avg_vector)
    shown = set(user_history)
    found = False
    print("\n🎯 Rekomendacje na podstawie historii:")
    for i in indices[0]:
        film = df_clean.iloc[i]
        if film['title'] not in shown and matches_filters(film):
            print(f"👉 {film['title']} (rok: {int(film['year'])},
ocena: {film['imdbRating']}, gatunki: {get_genres(film)})")
            shown.add(film['title'])
            found = True
    if not found:
        print("❗ Brak rekomendacji spełniających aktywne filtry.")

def list_genres():
    print("\n🎭 Dostępne gatunki:")
    for g in feature_cols[4:]:
        print(f"• {g}")
```

```
def get_favorite_movies():
    print("🎥 Podaj 3 swoje ulubione filmy (po jednym na raz):")
    count = 0
    while count < 3:
        inp = input(f"⭐ Ulubiony film {count+1}: ").strip()
        match = find_best_match(inp)
        if match:
            user_history.append(match)
            user_history_log.append([match,
datetime.datetime.now().isoformat(), 'favorite'])
            print(f"☑ Dodano: {match}")
            count += 1
        else:
            print("❌ Nie znaleziono filmu. Spróbuj jeszcze raz.")

def interactive_loop():
    global min_rating, genre_filter
    print("\n🟢 Witaj w systemie rekomendacji filmów!")
    print("Wpisz nazwę filmu, lub komendę:")
    print(" ◇ recommend – rekomenduj na podstawie historii")
    print(" ◇ surprise me – zaproponuj film losowo")
    print(" ◇ minrating [ocena] – ustaw minimalną ocenę (np. minrating
7.5)")
    print(" ◇ filter genre [GATUNEK] – filtruj tylko po gatunku (np.
Drama)")
    print(" ◇ clear filters – usuń wszystkie filtry")
    print(" ◇ genres – pokaż dostępne gatunki")
    print(" ◇ exit – zakończ")

    while True:
        inp = input("\n📝 Komenda lub tytuł: ").strip()
        if inp.lower() == "exit":
            save_history_to_file()
            print("📁 Historia zapisana do: historia_uzytkownika.csv")
            break
        elif inp.lower() == "recommend":
            recommend_from_history()
        elif inp.lower() in ["surprise me", "suprise me", "surprize
me"]:
            surprise_me()
        elif "me" in inp.lower() and " " in inp.lower():
            print("❗ Czy chodziło Ci o komendę 'surprise me'? Sprawdź
pisownię.")
        elif inp.lower().startswith("minrating"):
            try:
                min_rating = float(inp.split()[1])
                print(f"📊 Ustawiono minimalną ocenę: {min_rating}")
            except:
                print("❗ Użycie: minrating [liczba]")
```

```
        elif inp.lower().startswith("filter genre"):
            try:
                genre_input = inp.split(" ", 2)[2].strip()
                if genre_input in feature_cols:
                    genre_filter = genre_input
                    print(f"🎭 Filtr gatunku ustawiony na: {genre_fil-
ter}")
                else:
                    print("❗ Nieprawidłowy gatunek.")
            except:
                print("❗ Użycie: filter genre [nazwa gatunku]")
        elif inp.lower() == "clear filters":
            min_rating = 0.0
            genre_filter = None
            print("🧹 Filtry wyczyszczone.")
        elif inp.lower() == "genres":
            list_genres()
        else:
            match = find_best_match(inp)
            if match:
                user_history.append(match)
                user_history_log.append([match,
datetime.datetime.now().isoformat(), 'search'])
                recommend_similar(match)
            else:
                print("❌ Nie znaleziono filmu.")

# 🔧 Start: użytkownik podaje 3 ulubione filmy
get_favorite_movies()

# 🎯 Wstępne rekomendacje na ich podstawie
recommend_from_history()

# 🔁 Uruchom interaktywny tryb
interactive_loop()
```