

# ZDANIE SPRAWY

System rekomendacji w oparciu o uczenie maszynowe

Przez  
Krzysztof Sereda



Wydział Elektryczny PW  
Informatyka Stosowana

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>2</b>
1.1	Content-Based Filtering . . . . .	2
1.2	Collaborative Filtering . . . . .	3
1.2.1	User-Based Collaborative Filtering . . . . .	3
1.2.2	Item-Based Collaborative Filtering . . . . .	3
1.3	Matrix Factorization . . . . .	3
1.4	Neural Matrix Factorization . . . . .	3
1.4.1	Zastosowania NeuMF . . . . .	3
<b>2</b>	<b>Plan działania</b>	<b>4</b>
2.1	Wykorzystanie rzeczywistych danych . . . . .	4
2.2	Przygotowanie danych . . . . .	4
2.2.1	links.csv . . . . .	4
2.2.2	movies_metadata.csv . . . . .	4
2.2.3	ratings_prepare.ipynb . . . . .	5
2.2.4	Wczytanie danych . . . . .	6
2.2.5	Czyszczenie danych . . . . .	6
2.2.6	Dostosowanie danych do ważnych rekomendacji . . . . .	7
2.3	Trening modeli . . . . .	8
<b>3</b>	<b>Realizacja</b>	<b>8</b>
3.1	Uproszczony wstęp matematyczny - Matrix Factorization . . . . .	8
3.2	Uproszczony wstęp matematyczny - Neural Matrix Factorization . . . . .	11
3.3	Wizualizacja wektorów osadzeń . . . . .	14
<b>4</b>	<b>Trenowanie modeli</b>	<b>17</b>
<b>5</b>	<b>Testy</b>	<b>19</b>
5.1	Porównanie wyników . . . . .	20
<b>6</b>	<b>Podsumowanie</b>	<b>21</b>
6.1	Wykorzystanie modeli w różnych dziedzinach . . . . .	22

# 1 Wstęp

Systemy rekomendacji stały się nieodłącznym elementem nowoczesnych aplikacji i usług online, pomagając użytkownikom w odkrywaniu treści, które mogą ich zainteresować. Od platform streamingowych, takich jak Netflix i Spotify, po sklepy internetowe, takie jak Amazon, systemy rekomendacji wpływają na to, jakie filmy, muzyka czy produkty są sugerowane użytkownikom.

Główne cele systemów rekomendacji to poprawa doświadczeń użytkowników oraz zwiększenie zaangażowania i lojalności klientów. Dzięki analizie ogromnych zbiorów danych, systemy te są w stanie dostarczać spersonalizowane rekomendacje, które są dopasowane do indywidualnych preferencji i zachowań użytkowników. Istnieje wiele metod i algorytmów stosowanych w systemach rekomendacji, od prostych metod filtrowania opartego na treści, po bardziej zaawansowane techniki, takie jak filtrowanie kolaboratywne, sieci neuronowe i modele oparte na głębokim uczeniu.

Filtrowanie kolaboratywne, zarówno w wersji opartej na użytkownikach, jak i na przedmiotach, jest jedną z najpopularniejszych technik stosowanych w systemach rekomendacji. Metoda ta wykorzystuje dane o preferencjach użytkowników w celu identyfikacji wzorców i zależności, które mogą być następnie używane do przewidywania przyszłych preferencji. Modele te są często wspomagane przez techniki takie jak faktoryzacja macierzy, która pozwala na wydobycie ukrytych cech zarówno użytkowników, jak i przedmiotów, oraz na bardziej precyzyjne rekomendacje.

W ostatnich latach, wraz z rozwojem technologii głębokiego uczenia, pojawiły się nowe możliwości w zakresie systemów rekomendacji. Modele oparte na sieciach neuronowych, takie jak model NeuMF (Neural Matrix Factorization), łączą w sobie zalety tradycyjnych metod faktoryzacji macierzy z mocą sieci neuronowych, umożliwiając jeszcze dokładniejsze i bardziej spersonalizowane rekomendacje.

Celem niniejszej pracy jest analiza i porównanie różnych podejść do rekomendacji filmów, ze szczególnym uwzględnieniem metod faktoryzacji macierzy i modeli głębokiego uczenia. W pracy zostaną przedstawione zarówno teoretyczne podstawy tych metod, jak i praktyczne zastosowania, wraz z implementacją i ewaluacją wyników. Szczególny nacisk zostanie położony na ocenę skuteczności modeli rekomendacji oraz na identyfikację czynników wpływających na jakość dostarczanych rekomendacji.

W ramach przeprowadzonych badań zostaną wykorzystane rzeczywiste dane z serwisów filmowych, co pozwoli na realistyczną ocenę efektywności różnych podejść. Wyniki tych badań mogą przyczynić się do lepszego zrozumienia mechanizmów działających w systemach rekomendacji oraz do opracowania bardziej zaawansowanych i efektywnych algorytmów rekomendacyjnych, które mogą znaleźć zastosowanie w praktyce.



Rysunek 1: Przedstawienie przykładowych wykorzystanych narzędzi

## 1.1 Content-Based Filtering

Filtracja oparta na zawartości (Content-Based Filtering) polega na rekomendowaniu użytkownikowi elementów, które są podobne do tych, które wcześniej preferował. W tym podejściu kluczową rolę odgrywają cechy (atrybuty) elementów. Na przykład, jeśli użytkownik lubi filmy science fiction z dużą ilością akcji, system będzie analizował cechy takich filmów, takie jak gatunek, obsada, reżyseria czy słowa kluczowe, i na tej podstawie wyszukiwał oraz rekomendował filmy o podobnych atrybutach. Podejście to opiera się na założeniu, że preferencje użytkownika można modelować na podstawie cech elementów, które użytkownik wcześniej ocenił pozytywnie. W praktyce często stosuje się różne techniki uczenia maszynowego oraz metody przetwarzania tekstu, aby wyodrębnić i analizować istotne atrybuty elementów, co pozwala na bardziej precyzyjne rekomendacje.

## 1.2 Collaborative Filtering

Filtracja współpracy (Collaborative Filtering) opiera się na analizie interakcji między użytkownikami i elementami, takimi jak oceny filmów przez różnych użytkowników. W przeciwieństwie do filtracji opartej na zawartości, to podejście nie wymaga znajomości cech rekomendowanych elementów. Zamiast tego, system rekomendacji analizuje wzorce w zachowaniach użytkowników, takie jak oceny czy kliknięcia, aby sugerować elementy, które mogą spodobać się podobnym użytkownikom. Collaborative Filtering można podzielić na dwie główne kategorie: filtrowanie kolaboratywne oparte na użytkownikach (User-Based Collaborative Filtering) oraz filtrowanie kolaboratywne oparte na przedmiotach (Item-Based Collaborative Filtering).

### 1.2.1 User-Based Collaborative Filtering

W podejściu User-Based Collaborative Filtering system rekomendacji identyfikuje użytkowników, którzy mają podobne wzorce zachowań do użytkownika docelowego. Na przykład, jeśli dwóch użytkowników oceniło wiele tych samych filmów w podobny sposób, system uzna ich za podobnych. Następnie, na podstawie preferencji podobnych użytkowników, system rekomenduje elementy, które użytkownik docelowy może uznać za interesujące. W praktyce często stosuje się metryki takie jak współczynnik korelacji Pearsona czy podobieństwo kosinusowe do mierzenia podobieństwa między użytkownikami.

### 1.2.2 Item-Based Collaborative Filtering

W podejściu Item-Based Collaborative Filtering system rekomendacji skupia się na identyfikowaniu podobieństw między elementami na podstawie wzorców ocen użytkowników. Na przykład, jeśli dwa filmy są często oceniane w podobny sposób przez różnych użytkowników, system uzna je za podobne. Następnie, na podstawie preferencji użytkownika docelowego, system rekomenduje filmy, które są podobne do tych, które użytkownik już ocenił pozytywnie. Podejście to ma tę zaletę, że jest często bardziej skalowalne i stabilne w miarę dodawania nowych użytkowników i elementów, ponieważ podobieństwa między elementami zmieniają się wolniej niż podobieństwa między użytkownikami.

## 1.3 Matrix Factorization

Matrix Factorization (Faktoryzacja Macierzy) jest jedną z najpopularniejszych i najskuteczniejszych technik stosowanych w systemach rekomendacyjnych, szczególnie w kontekście filtracji kolaboratywnej. Technika ta polega na dekompozycji macierzy ocen użytkowników na dwie macierze o niższym wymiarze, które reprezentują ukryte cechy użytkowników oraz elementów. W klasycznym podejściu, każda komórka macierzy ocen zawiera ocenę przyznaną przez użytkownika danemu elementowi, a celem faktoryzacji macierzy jest przewidzenie brakujących ocen na podstawie znanych danych.

## 1.4 Neural Matrix Factorization

Neural Matrix Factorization (NeuMF) stanowi zaawansowaną ewolucję tradycyjnej faktoryzacji macierzy, integrującą jej zalety z potęgą sieci neuronowych. NeuMF jest w stanie uchwycić nieliniowe zależności między użytkownikami a elementami, co prowadzi do bardziej precyzyjnych i spersonalizowanych rekomendacji.

### 1.4.1 Zastosowania NeuMF

NeuMF znalazł szerokie zastosowanie w różnych dziedzinach, w tym w platformach streamingowych, sklepach internetowych oraz serwisach społecznościowych. Dzięki swojej zdolności do modelowania skomplikowanych wzorców interakcji, NeuMF jest szczególnie skuteczny w kontekstach, gdzie tradycyjne metody mogą zawodzić.

## 2 Plan działania

Celem niniejszej pracy jest analiza i porównanie skuteczności różnych metod rekomendacji filmów, ze szczególnym uwzględnieniem faktoryzacji macierzy (MF) oraz Neural Matrix Factorization (NeuMF). Eksperyment opiera się na wykorzystaniu rzeczywistych danych z The Movies Dataset, który zawiera informacje o filmach, ocenach użytkowników oraz metadanych filmów. W celu przeprowadzenia analizy i treningu modeli przygotowano osobne pliki Jupyter Notebook (.ipynb).

### 2.1 Wykorzystanie rzeczywistych danych

#### The Movies Dataset

Metadata on over 45,000 movies. 26 million ratings from over 270,000 users.



Rysunek 2: The Movies Dataset

Do przeprowadzenia eksperymentu wykorzystano dane z The Movies Dataset, które są powszechnie stosowane w badaniach nad systemami rekomendacji. Dane te obejmują:

- **Ratings:** Zawiera oceny filmów przyznane przez użytkowników.
- **Movies Metadata:** Zawiera szczegółowe informacje o filmach, takie jak tytuł, gatunek, obsada itp.
- **Links:** Zawiera powiązania między identyfikatorami filmów w różnych bazach danych (np. IMDb, TMDb).

### 2.2 Przygotowanie danych

Proces przygotowania danych został podzielony na kilka etapów, z których każdy został zrealizowany w osobnym pliku Jupyter Notebook:

- `links_prepare.ipynb`
- `movies_metadata_prepare.ipynb`
- `ratings_prepare.ipynb`

#### 2.2.1 links.csv

W tym kroku przygotowałem dane wiążące identyfikator `movieId` z `imdbId` pozbywając się nie potrzebnej w moim przypadku kolumny `tmdbId`

Pierwszy obraz przedstawia surowe dane, które zawierały dodatkową kolumnę `tmdbId`. W drugim obrazie można zobaczyć przekształcone dane, gdzie usunięto kolumnę `tmdbId`, pozostawiając tylko kolumny `movieId` i `imdbId`. Tak przekształcone dane zostały zapisane do nowego pliku `cleaned_links.csv`.

#### 2.2.2 movies\_metadata.csv

Działania związane z `movies_metadata.csv` polegały na wstępnym zapoznaniu się z danymi, które będą używane w dalszych analizach i przekształceniach. W szczególności zweryfikowane zostały braki w danych oraz dokonano wstępnej analizy zawartości. Celem było upewnienie się, że dane są kompletne i poprawne, co jest kluczowe dla dokładności późniejszych analiz.

	movied	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0
...	...	...	...
45838	176269	6209470	439050.0
45839	176271	2028550	111109.0
45840	176273	303758	67758.0
45841	176275	8536	227506.0
45842	176279	6980792	461257.0

45843 rows × 3 columns

(a) Surowe dane

	movied	imdbId
0	1	114709
1	2	113497
2	3	113228
3	4	114885
4	5	113041
...	...	...
45838	176269	6209470
45839	176271	2028550
45840	176273	303758
45841	176275	8536
45842	176279	6980792

45843 rows × 2 columns

(b) Dane przekształcone

Rysunek 3: Przekształcenie danych links.csv

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	release_date		
count	45466		4494	45466	45466	7782	45466	45449	45455	45466	44512	...	45379	4.546
unique	5		1698	1226	4069	7673	45436	45417	92	43373	44307	...		17336
top	False	{'id': 415931, 'name': 'The Bowery Boys', 'pos...	0	{['id': 18, 'name': 'Drama']}	http://www.georgecarlin.com	141971	tt1180333	en	Hamlet	No overview found.	...		2008-01-01	
freq	45454		29	36573	5000	12	3	3	32269	8	133	...		136
mean	NaN		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	1.126
std	NaN		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	6.433
min	NaN		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	0.000
25%	NaN		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	0.000
50%	NaN		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	0.000
75%	NaN		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	0.000
max	NaN		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	2.787
11 rows × 14 columns														

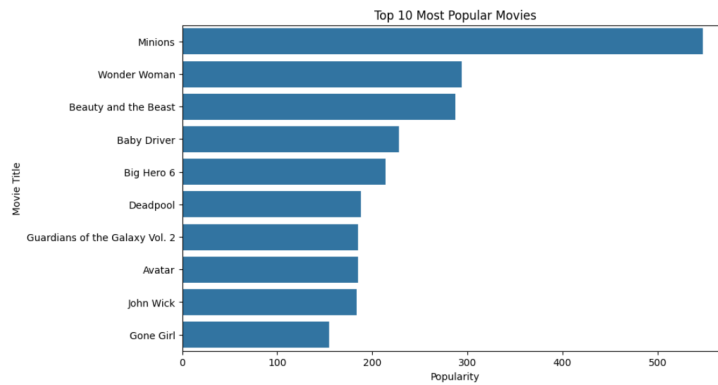
11 rows × 24 columns

Rysunek 4: Surowe dane z movies\_metadata.csv

W wyniku wstępnej analizy i przekształceń, usunięto zbędne kolumny oraz poprawiono formatowanie danych. Upewniono się również, że wszystkie istotne informacje, takie jak identyfikatory filmów i tytuły, są kompletne i gotowe do użycia w dalszych etapach projektu. Tak przygotowane dane zapisano do pliku `cleaned_movies_metadata.csv`, który posłużył jako baza do mapowania identyfikatorów filmów na tytuły w systemie rekomendacji.

### 2.2.3 ratings\_prepare.ipynb

Celem pliku `ratings_prepare.ipynb` było przygotowanie danych o ocenach filmów, aby mogły one być użyte do trenowania modeli rekomendacyjnych. Poniżej przedstawiam szczegółowy opis kroków wykonanych w notebooku `ratings_prepare.ipynb`.



Rysunek 5: Przykładowa wizualizacja

	id	imdb_id	title
45456	84419	0038621	House of Horrors
45457	390959	0265736	Shadow of the Blair Witch
45458	289923	0252966	The Burkittsville 7
45459	222848	0112613	Caged Heat 3000
45460	30840	0102797	Robin Hood
45461	439050	6209470	Subdue
45462	111109	2028550	Century of Birthing
45463	67758	0303758	Betrayal
45464	227506	0008536	Satan Triumphant
45465	461257	6980792	Queerama

Rysunek 6: Przetworzone dane z movies\_metadata.csv

## 2.2.4 Wczytanie danych

Plik ten zawierał następujące kolumny:

- **userId** – identyfikator użytkownika,
- **movieId** – identyfikator filmu,
- **rating** – ocena nadana przez użytkownika,
- **timestamp** – znacznik czasu.

## 2.2.5 Czyszczenie danych

Następnie przeprowadziłem wstępną analizę i czyszczenie danych, w następujących krokach:

### 1. Usunięcie kolumny timestamp

Pierwszym krokiem było usunięcie kolumny **timestamp**. Kolumna ta zawiera informacje o czasie, w którym użytkownik dokonał oceny. Ponieważ dla naszych celów rekomendacyjnych nie jest istotne,

kiedy dokładnie użytkownik ocenił film, usunięcie tej kolumny upraszcza dane i zmniejsza ich rozmiar, co jest korzystne dla wydajności przetwarzania.

## 2. Filtrowanie użytkowników z co najmniej 20 ocenami

Następnie przefiltrowałem użytkowników, aby zachować tylko tych, którzy ocenili co najmniej 20 filmów. Użytkownicy z mniejszą liczbą ocen mogą nie dostarczyć wystarczających danych do nauczania modeli rekomendacyjnych, co mogłoby negatywnie wpłynąć na ich jakość.

## 3. Zachowanie ocen powyżej 3

Kolejnym krokiem było zachowanie tylko tych ocen, które są większe niż 3. Ocenę powyżej 3 są uznawane za pozytywne, co oznacza, że użytkownik lubił dany film.

## 4. Filtrowanie filmów ocenionych co najmniej 20 razy

Dla filmów zastosowano podobne filtrowanie jak dla użytkowników. Zachowano tylko te filmy, które były ocenione co najmniej 20 razy. Filmy z mniejszą liczbą ocen mogą nie dostarczyć wystarczających danych do nauczania modeli, co mogłoby prowadzić do mniej precyzyjnych rekomendacji.

## 5. Konwersja ocen (binaryzacja)

Ostatnim krokiem była konwersja wszystkich ocen powyżej 3 na wartość 1, czyli binaryzacja ocen. Dzięki temu każda pozytywna ocena jest traktowana jako 1 - reszta przyjmuje wartości 0, co upraszcza modelowanie i skupia się na tym, czy użytkownik lubił dany film, czy nie.

Następnie dokonałem mapowania identyfikatorów użytkowników i filmów na nowe identyfikatory, aby uprościć modelowanie:

1. **Mapowanie identyfikatorów użytkowników:** Wszystkim unikalnym identyfikatorom użytkowników przypisano nowe, kolejne identyfikatory zaczynając od 0. Takie podejście ułatwia późniejszą pracę z modelami rekomendacyjnymi, które często wymagają, aby identyfikatory były liczbami całkowitymi zaczynającymi się od 0.
2. **Mapowanie identyfikatorów filmów:** Analogicznie jak w przypadku użytkowników, wszystkie unikalne identyfikatory filmów zostały przemapowane na nowe, kolejne identyfikatory zaczynające się od 0. To również uprościło implementację modeli rekomendacyjnych.

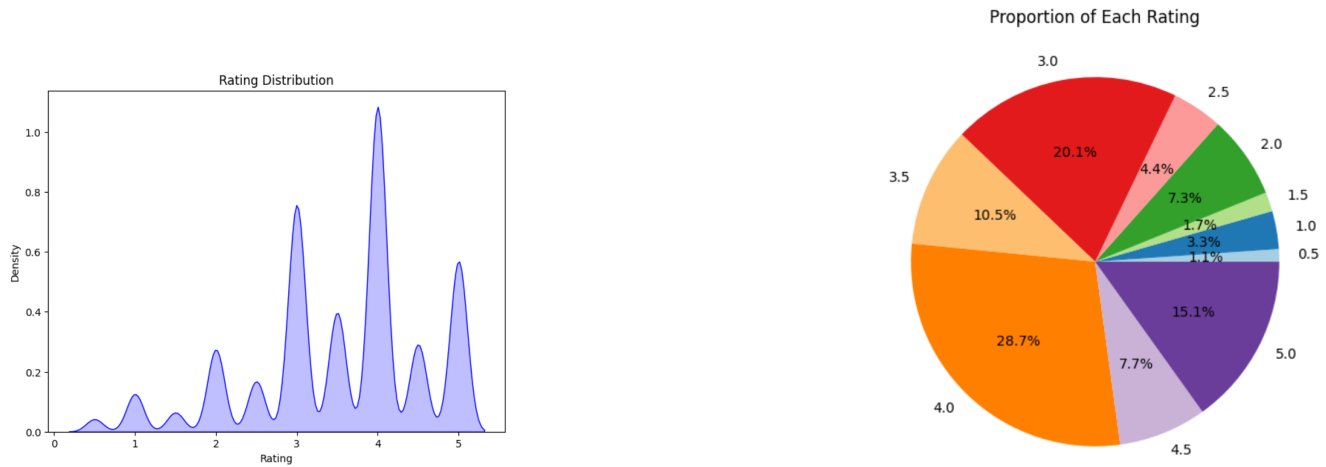
### 2.2.6 Dostosowanie danych do ważnych rekomendacji

W celu zapewnienia, że model będzie generował jedynie istotne rekomendacje, wykorzystano pliki `movies_metadata.csv` i `links.csv`. Na ich podstawie dokonano dodatkowego przefiltrowania danych:

- **Filtrowanie filmów na podstawie `movies_metadata.csv` i `links.csv`:** Zachowano jedynie te filmy, które występowały w obu plikach. Dzięki temu upewniono się, że tylko filmy z odpowiednimi metadanymi oraz poprawnymi powiązaniem są brane pod uwagę w systemie rekomendacji.

Tak przygotowane dane zapisano do pliku `full_ratings.csv`, który posłużył jako baza do trenowania modeli rekomendacyjnych.





Rysunek 7: Przykładowe wizualizacje ratings\_small.csv

## 2.3 Trening modeli

Po przygotowaniu danych przystąpiono do treningu modeli rekomendacyjnych. Proces ten został również zrealizowany w osobnych plikach Jupyter Notebook:

- **MF\_model.ipynb**: Plik ten zawiera kod do trenowania modelu faktoryzacji macierzy (Matrix Factorization). Model został zdefiniowany przy użyciu bibliotek TensorFlow i Keras, a następnie wytrenowany na przygotowanych danych. Po zakończeniu treningu, model zapisano w formacie HDF5.
- **NeuMF\_model.ipynb**: Plik ten zawiera kod do trenowania modelu Neural Matrix Factorization (NeuMF). Model łączy w sobie tradycyjną faktoryzację macierzy z głębokimi sieciami neuronowymi, co pozwala na uchwycenie bardziej skomplikowanych wzorców w danych. Model został również zdefiniowany przy użyciu TensorFlow i Keras, a następnie wytrenowany i zapisany w formacie HDF5.

## 3 Realizacja

Część związaną z realizacją projektu zacząłem od obserwacji zachowań na, samodzielnie wygenerowanych, małych zbiorach danych. Pozwoliło to lepiej zrozumieć matematyczne podstawy wykorzystywanych podejść jak i wizualizację wyników.

### 3.1 Uproszczony wstęp matematyczny - Matrix Factorization

Matrix Factorization to technika używana do przewidywania ocen w systemach rekomendacji. Każdy użytkownik i każdy film są reprezentowani przez wektory w przestrzeni o określonej liczbie wymiarów (embedding vectors). Wartości tych wektorów są uczone w trakcie treningu modelu.

#### Dane

Założmy, że mamy 3 użytkowników ( $U_0, U_1, U_2$ ) oraz 3 filmy ( $M_0, M_1, M_2$ ). Poniżej przykładowa tabela ocen:

User	Movie	Rating
$U_0$	$M_0$	5
$U_0$	$M_1$	3
$U_1$	$M_1$	4
$U_1$	$M_2$	2
$U_2$	$M_0$	1
$U_2$	$M_2$	4

	$M_0$	$M_1$	$M_2$
$U_0$	5	3	0
$U_1$	0	4	2
$U_2$	1	0	4

### Inicjalizacja Wektorów Osadzeń i Biasów

Założmy, że wektory osadzeń mają wymiar 2 ( $embedding\_size = 2$ ). Na początku, wektory osadzeń i biasy są inicjalizowane losowo.

### Początkowe Wektory Osadzeń (Embeddings):

$$U_0 : \begin{bmatrix} u_{00} \\ u_{01} \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix}$$

$$U_1 : \begin{bmatrix} u_{10} \\ u_{11} \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix}$$

$$U_2 : \begin{bmatrix} u_{20} \\ u_{21} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.8 \end{bmatrix}$$

$$M_0 : \begin{bmatrix} m_{00} \\ m_{01} \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}$$

$$M_1 : \begin{bmatrix} m_{10} \\ m_{11} \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.1 \end{bmatrix}$$

$$M_2 : \begin{bmatrix} m_{20} \\ m_{21} \end{bmatrix} = \begin{bmatrix} 0.4 \\ -0.6 \end{bmatrix}$$

### Początkowe Biasy:

$$b_{u_0} = 0.05$$

$$b_{u_1} = -0.02$$

$$b_{u_2} = 0.01$$

$$b_{m_0} = 0.02$$

$$b_{m_1} = -0.03$$

$$b_{m_2} = 0.04$$

### Obliczanie Prognozowanej Oceny

Dla każdej oceny obliczamy prognozowaną ocenę jako iloczyn skalarny wektorów osadzeń użytkownika i filmu dodając biasy.

**Prognozowana ocena dla  $U_0$  i  $M_0$ :** 1. Wektory osadzeń:

$$U_0 : \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix}$$

$$M_0 : \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}$$

2. Iloczyn skalarny:

$$dot\_product = 0.1 \cdot 0.2 + (-0.2) \cdot 0.3 = 0.02 - 0.06 = -0.04$$

3. Biasy:

$$\begin{aligned}\text{user\_bias} &= 0.05 \\ \text{item\_bias} &= 0.02\end{aligned}$$

4. Prognozowana ocena:

$$\text{rating\_prediction} = -0.04 + 0.05 + 0.02 = 0.03$$

### Aktualizacja Wag

Podczas treningu modelu, wagi są aktualizowane w celu minimalizacji błędu. Obliczamy gradienty i używamy ich do aktualizacji wag.

**Aktualizacja dla  $U_0$  i  $M_0$ :** Załóżmy, że używamy prostego gradientu, aby zaktualizować wagi.

**Funkcja Straty (Loss Function)** Używamy średniego błędu kwadratowego (MSE):

$$\text{loss} = (\text{rating\_prediction} - \text{rating})^2$$

Dla  $U_0$  i  $M_0$ :

$$\begin{aligned}\text{rating} &= 5 \\ \text{rating\_prediction} &= 0.03 \\ \text{loss} &= (0.03 - 5)^2 = (-4.97)^2 = 24.7009\end{aligned}$$

**Gradienty** Obliczamy gradienty względem wag osadzeń i biasów. Dla uproszczenia, obliczymy przybliżone gradienty:

1. Gradienty dla wektorów osadzeń:

$$\begin{aligned}\frac{\partial \text{loss}}{\partial u_{00}} &= \frac{\partial \text{loss}}{\partial \text{rating\_prediction}} \cdot \frac{\partial \text{rating\_prediction}}{\partial u_{00}} \\ \frac{\partial \text{loss}}{\partial u_{00}} &= 2 \cdot (\text{rating\_prediction} - \text{rating}) \cdot m_{00} \\ \frac{\partial \text{loss}}{\partial u_{00}} &= 2 \cdot (0.03 - 5) \cdot 0.2 = 2 \cdot (-4.97) \cdot 0.2 = -1.988\end{aligned}$$

2. Aktualizacja wag: Załóżmy współczynnik uczenia (learning rate)  $\eta = 0.01$ :

$$\begin{aligned}u_{00} &:= u_{00} - \eta \cdot \frac{\partial \text{loss}}{\partial u_{00}} \\ u_{00} &:= 0.1 - 0.01 \cdot (-1.988) = 0.1 + 0.01988 = 0.11988\end{aligned}$$

Podobnie aktualizujemy pozostałe wagi.

### Macierz po jednej iteracji

Początkowe wartości osadzeń i biasów zmieniają się po jednej iteracji. Oto zaktualizowane wartości:

### Wektory Osadzeń po Aktualizacji:

$$U_0 : \begin{bmatrix} 0.11988 \\ -0.18012 \end{bmatrix}$$

$$U_1 : \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix}$$

$$U_2 : \begin{bmatrix} -0.3 \\ 0.8 \end{bmatrix}$$

$$M_0 : \begin{bmatrix} 0.21988 \\ 0.31988 \end{bmatrix}$$

$$M_1 : \begin{bmatrix} -0.5 \\ 0.1 \end{bmatrix}$$

$$M_2 : \begin{bmatrix} 0.4 \\ -0.6 \end{bmatrix}$$

### Biasy po Aktualizacji:

$$b_{u_0} = 0.06988$$

$$b_{u_1} = -0.02$$

$$b_{u_2} = 0.01$$

$$b_{m_0} = 0.03988$$

$$b_{m_1} = -0.03$$

$$b_{m_2} = 0.04$$

## 3.2 Uproszczony wstęp matematyczny - Neural Matrix Factorization

Neural Matrix Factorization (NeuMF) to zaawansowana technika używana do przewidywania ocen w systemach rekomendacji, która łączy tradycyjną faktoryzację macierzy z mocą sieci neuronowych. NeuMF jest w stanie uchwycić nieliniowe zależności między użytkownikami a filmami, co prowadzi do bardziej precyzyjnych rekomendacji.

### Dane

Mamy 3 użytkowników ( $U_0, U_1, U_2$ ) i 3 filmy ( $M_0, M_1, M_2$ ). Poniżej przedstawiamy tabelę ocen:

User	Movie	Rating
$U_0$	$M_0$	5
$U_0$	$M_1$	3
$U_1$	$M_1$	4
$U_1$	$M_2$	2
$U_2$	$M_0$	1
$U_2$	$M_2$	4

### Inicjalizacja Wektorów Osadzeń i Biasów

Założmy, że wektory osadzeń mają wymiar 2 ( $embedding\_size = 2$ ). Na początku, wektory osadzeń i biasy są inicjalizowane losowo.

### Początkowe Wektory Osadzeń (Embeddings):

$$U_0 : \begin{bmatrix} u_{00} \\ u_{01} \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix}$$

$$U_1 : \begin{bmatrix} u_{10} \\ u_{11} \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix}$$

$$U_2 : \begin{bmatrix} u_{20} \\ u_{21} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.8 \end{bmatrix}$$

$$M_0 : \begin{bmatrix} m_{00} \\ m_{01} \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}$$

$$M_1 : \begin{bmatrix} m_{10} \\ m_{11} \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.1 \end{bmatrix}$$

$$M_2 : \begin{bmatrix} m_{20} \\ m_{21} \end{bmatrix} = \begin{bmatrix} 0.4 \\ -0.6 \end{bmatrix}$$

### Początkowe Biasy:

$$b_{u_0} = 0.05$$

$$b_{u_1} = -0.02$$

$$b_{u_2} = 0.01$$

$$b_{m_0} = 0.02$$

$$b_{m_1} = -0.03$$

$$b_{m_2} = 0.04$$

### Obliczanie Prognozowanej Oceny

Dla każdej oceny obliczamy prognozowaną ocenę przy użyciu iloczynu skalarnego wektorów osadzeń użytkownika i filmu oraz przez wielowarstwowy perceptron (MLP), a następnie łączymy wyniki.

**Generalized Matrix Factorization (GMF)** GMF jest rozszerzeniem tradycyjnej faktoryzacji macierzy, gdzie iloczyn skalarny wektorów osadzeń użytkownika i filmu jest zastąpiony przez iloczyn elementowy:

Iloczyn elementowy, w przeciwieństwie do tradycyjnego iloczynu skalarnego, polega na mnożeniu odpowiednich elementów dwóch wektorów przez siebie, co pozwala na uchwycenie bardziej złożonych i nieliniowych relacji między użytkownikami a elementami.

**Iloczyn Skalarny** W tradycyjnej faktoryzacji macierzy, iloczyn skalarny wektorów osadzeń użytkownika ( $p_u$ ) i filmu ( $q_i$ ) jest obliczany jako suma iloczynów odpowiednich elementów tych wektorów:

$$\text{dot\_product} = \sum_k p_{u,k} \cdot q_{i,k}$$

**Iloczyn Elementowy** W GMF, iloczyn skalarny zastępowany jest przez iloczyn elementowy, który tworzy nowy wektor poprzez mnożenie odpowiednich elementów dwóch wektorów:

$$\text{element\_wise\_product} = p_u \odot q_i = \begin{bmatrix} p_{u,1} \cdot q_{i,1} \\ p_{u,2} \cdot q_{i,2} \\ \vdots \\ p_{u,k} \cdot q_{i,k} \end{bmatrix}$$

**Przykład** Dla lepszego zrozumienia, rozważmy przykład z użytkownikiem  $U_0$  i filmem  $M_0$ . Załóżmy, że ich wektory osadzeń są następujące:

$$p_{U_0} = \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix}, \quad q_{M_0} = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}$$

**Iloczyn Skalarny:**

$$\text{dot\_product} = 0.1 \cdot 0.2 + (-0.2) \cdot 0.3 = 0.02 - 0.06 = -0.04$$

**Iloczyn Elementowy:**

$$\text{element\_wise\_product} = \begin{bmatrix} 0.1 \cdot 0.2 \\ -0.2 \cdot 0.3 \end{bmatrix} = \begin{bmatrix} 0.02 \\ -0.06 \end{bmatrix}$$

Iloczyn tworzy nowy wektor, który może być następnie przetwarzany przez kolejne warstwy sieci neuronowej, umożliwiając bardziej złożone operacje i modelowanie nieliniowych zależności.

**Multi-Layer Perceptron (MLP)** MLP przekształca wektory osadzeń użytkownika i filmu za pomocą kilku warstw ukrytych. Na przykład, dla uproszczonego modelu z jedną warstwą ukrytą:

1. Konkatenacja wektorów:

$$[p_{U_0}, q_{M_0}] = [0.1, -0.2, 0.2, 0.3]$$

2. Warstwa ukryta:

$$h = \sigma(W \cdot [p_{U_0}, q_{M_0}] + b)$$

Założmy:

$$W = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.8 \end{bmatrix}, \quad b = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$$

$$h = \sigma \left( \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.8 \end{bmatrix} \cdot \begin{bmatrix} 0.1 \\ -0.2 \\ 0.2 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} \right)$$

$$h = \sigma \left( \begin{bmatrix} 0.1 \cdot 0.1 + 0.2 \cdot (-0.2) + 0.3 \cdot 0.2 + 0.4 \cdot 0.3 \\ 0.5 \cdot 0.1 + 0.6 \cdot (-0.2) + 0.7 \cdot 0.2 + 0.8 \cdot 0.3 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} \right)$$

$$h = \sigma \left( \begin{bmatrix} 0.01 - 0.04 + 0.06 + 0.12 \\ 0.05 - 0.12 + 0.14 + 0.24 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} \right)$$

$$h = \sigma \left( \begin{bmatrix} 0.15 \\ 0.31 \end{bmatrix} \right)$$

3. Aktywacja:

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

$$h = \begin{bmatrix} \sigma(0.15) \\ \sigma(0.31) \end{bmatrix} \approx \begin{bmatrix} 0.5374 \\ 0.5770 \end{bmatrix}$$

**Łączenie wyników GMF i MLP** Połączenie wyników GMF i MLP za pomocą warstwy wyjściowej:

$$\text{NeuMF}(U_0, M_0) = w^T \cdot [\text{GMF}(U_0, M_0), \text{MLP}(U_0, M_0)] + b$$

Założmy:

$$w = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}, \quad b = 0.1$$

$$\text{NeuMF}(U_0, M_0) = 0.1 \cdot 0.02 + 0.2 \cdot (-0.06) + 0.3 \cdot 0.5374 + 0.4 \cdot 0.5770 + 0.1$$

$$\text{NeuMF}(U_0, M_0) = 0.002 - 0.012 + 0.1612 + 0.2308 + 0.1$$

$$\text{NeuMF}(U_0, M_0) = 0.482$$

### 3.3 Wizualizacja wektorów osadzeń

W tym eksperymencie zastosowałem model Matrix Factorization, aby wytrenować wektory osadzeń użytkowników na podstawie ocen filmów. Użyłem danych ocen sześciu użytkowników dla trzech filmów. Celem było zobrazowanie, jak użytkownicy z podobnymi preferencjami znajdują się bliżej siebie w przestrzeni osadzeń po treningu modelu.

#### Dane

Wykorzystane dane:

```
ratings_data = {
    'userId': [0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5],
    'movieId': [0, 1, 0, 1, 0, 2, 1, 2, 0, 2, 1, 0],
    'rating': [5, 5, 5, 5, 1, 1, 5, 1, 1, 1, 5, 1]
}
```

#### Hipoteza

Użytkownicy o id = (2,4) oraz id = (0,1) ocenili te same filmy tymi samymi ocenami zatem powinni oni znaleźć się dość blisko na wykresie przedstawiającym ich umiejscowienie w przestrzeni.

#### Kod w Pythonie

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Embedding, Dot, Add, Flatten, Input
import matplotlib.pyplot as plt

# Dane ocen
ratings_data = {
    'userId': [0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5],
    'movieId': [0, 1, 0, 1, 0, 2, 1, 2, 0, 2, 1, 0],
    'rating': [5, 5, 5, 5, 1, 1, 5, 1, 1, 1, 5, 1]
}

ratings = pd.DataFrame(ratings_data)

def make_tf_dataset(data, batch_size=2):
    dataset = tf.data.Dataset.from_tensor_slices(({
        'userId': data['userId'].values,
```

```

    'movieId': data['movieId'].values
}, data['rating'].values))

dataset = dataset.shuffle(buffer_size=len(data)).batch(batch_size)
return dataset

train_dataset = make_tf_dataset(ratings)

class MatrixFactorization(Model):
    def __init__(self, num_users, num_items, embedding_size):
        super(MatrixFactorization, self).__init__()
        self.user_embedding = Embedding(num_users, embedding_size, embeddings_initializer='he_normal', name="user_e")
        self.item_embedding = Embedding(num_items, embedding_size, embeddings_initializer='he_normal', name="item_e")
        self.user_bias = Embedding(num_users, 1, name="user_bias")
        self.item_bias = Embedding(num_items, 1, name="item_bias")

    def call(self, inputs):
        user_vector = self.user_embedding(inputs['userId'])
        item_vector = self.item_embedding(inputs['movieId'])
        user_bias = self.user_bias(inputs['userId'])
        item_bias = self.item_bias(inputs['movieId'])
        dot_user_item = tf.reduce_sum(user_vector * item_vector, axis=1, keepdims=True)
        return dot_user_item + user_bias + item_bias

embedding_size = 2
mf_model = MatrixFactorization(num_users=6, num_items=3, embedding_size=embedding_size)
optimizer = tf.keras.optimizers.Adam(learning_rate=0.1)
mf_model.compile(optimizer=optimizer, loss='mean_squared_error')

sample_data = {
    'userId': tf.constant([0, 1, 2, 3, 4, 5]),
    'movieId': tf.constant([0, 1, 2, 1, 2, 0])
}
mf_model(sample_data)

history_user_embeddings = []
history_item_embeddings = []
history_user_biases = []
history_item_biases = []

def store_weights():
    history_user_embeddings.append(mf_model.user_embedding.get_weights()[0].copy())
    history_item_embeddings.append(mf_model.item_embedding.get_weights()[0].copy())
    history_user_biases.append(mf_model.user_bias.get_weights()[0].copy())
    history_item_biases.append(mf_model.item_bias.get_weights()[0].copy())

store_weights()

num_epochs = 50
for epoch in range(num_epochs):
    mf_model.fit(train_dataset, epochs=1, verbose=0)
    store_weights()

evaluation = mf_model.evaluate(train_dataset)
print(f'\nEwaluacja Modelu - Strata: {evaluation}')
```

```

def plot_embeddings(embeddings, title):
    plt.figure(figsize=(8, 6))
    for i in range(embeddings.shape[0]):
        plt.scatter(embeddings[i, 0], embeddings[i, 1], label=f'User {i}')
    plt.text(embeddings[i, 0], embeddings[i, 1], str(i))
    plt.title(title)
    plt.xlabel('Wymiar 1')
    plt.ylabel('Wymiar 2')

```

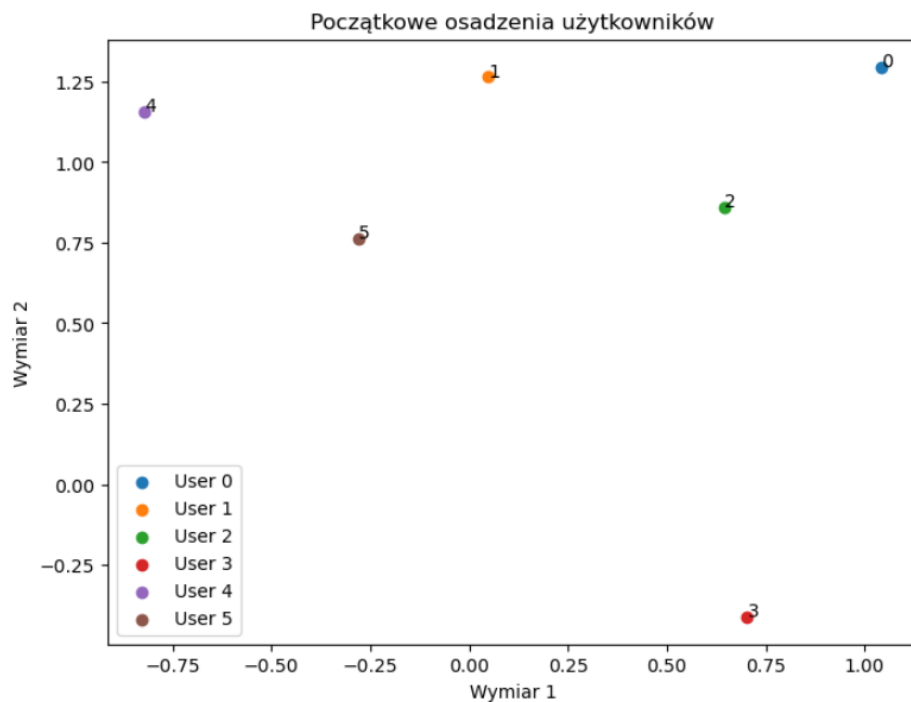


```
plt.legend()
plt.show()

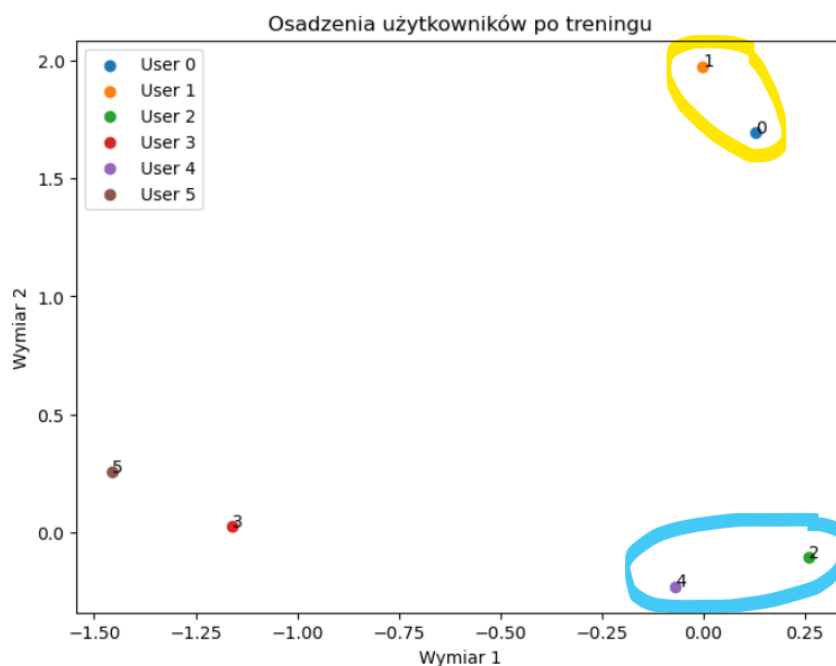
plot_embeddings(history_user_embeddings[0], 'Początkowe osadzenia użytkowników')
plot_embeddings(history_user_embeddings[-1], 'Osadzenia użytkowników po treningu')
```

## Wyniki

Na poniższych wykresach przedstawiono osadzenia użytkowników przed i po treningu modelu Matrix Factorization.



Rysunek 8: Początkowe osadzenia użytkowników

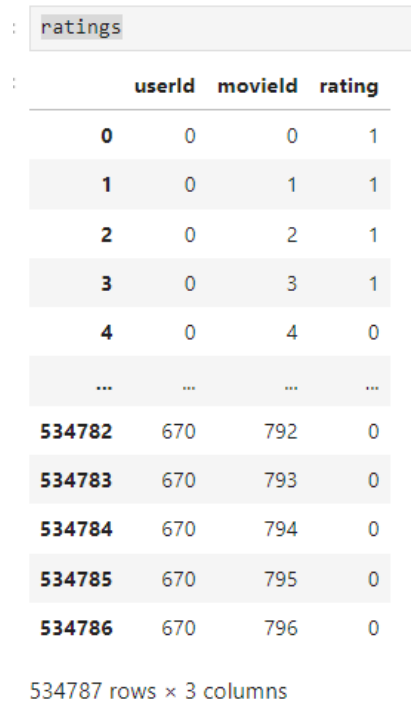


Rysunek 9: Osadzenia użytkowników po treningu

Na wykresie przed treningiem, osadzenia użytkowników są losowe i nie mają wyraźnego wzorca. Po treningu modelu widzimy, że użytkownicy 0 i 1 oraz 2 i 4, którzy lubią te same filmy, są blisko siebie w przestrzeni wektorowej.

## 4 Trenowanie modeli

W celu przeprowadzenia eksperymentu z użyciem systemów rekomendacji, wykorzystałem dwa notebooki: jeden do nauki modelu Matrix Factorization (MF) oraz drugi do nauki modelu Neural Matrix Factorization (NeuMF). Dane użyte w tych notebookach pochodzą z przygotowanych wcześniej plików, które finalnie składają się na przygotowany do nauki `full_ratings.csv`. W pliku tym widnieją wszystkie możliwe kombinacje ocen filmów przez użytkowników. Jeżeli pewna osoba oceniła przykładowo 5 filmów ocenami będącymi wyższymi niż 3, wówczas będzie ona miała 5 wystąpień w której kolumna 'ratings' przyjmie 1, reszta filmów nie była pozytywnie oceniona zatem przyjmie wartości 0.



	userid	movieid	rating
0	0	0	1
1	0	1	1
2	0	2	1
3	0	3	1
4	0	4	0
...	...	...	...
534782	670	792	0
534783	670	793	0
534784	670	794	0
534785	670	795	0
534786	670	796	0

534787 rows × 3 columns

Rysunek 10: Przygotowane opinie użytkowników do nauki modeli

- Liczba unikalnych użytkowników: 671
- Liczba unikalnych filmów: 797
- Liczba ocen: 534787

Do nauki modeli użyłem następujących parametrów:

- Liczba epok ( $n\_epochs$ )
- Rozmiar wsadu ( $batch\_size$ )
- Wymiar osadzeń ( $embedding\_size$ ) dla MF oraz wymiarów ukrytych ( $latent\_dim\_mf, latent\_dim\_mlp$ ) i warstw gęstych ( $dense\_layers\_1, dense\_layers\_2$ ) dla NeuMF
- Współczynnik uczenia ( $learning\_rate$ )
- Podział danych na walidacyjne ( $val\_split$ )

## Parameters

```
embedding_size = 4
learning_rate = 0.001
num_epochs = 10
batch_size=16
val_split=0.2
```

(a) Parametry modelu MF

## Parameters

```
num_epochs = 20
learning_rate = 0.001
batch_size = 128
val_split = 0.1
latent_dim_mf = 16
latent_dim_mlp = 4
dense_layers = [8,8]
reg_layers = [0.01, 0.01]
activation_dense = "relu"
reg_mf: int = 0
reg_mlp: int = 0.01
```

(b) Parametry modelu NeuMF

Rysunek 11

Wyniki eksperymentu z dostosowywania parametrów dla modelu MF:

n	embedding size	learning rate	n_epochs	batch_size	val_split	loss	accuracy	precision	recall	auc
1	4	0.01	5	128	0.2	0.0583	0.9335	0.6108	0.1891	0.8160
2	2	0.01	5	128	0.2	0.0598	0.9319	0.6194	0.1710	0.7989
3	8	0.01	5	128	0.2	0.0593	0.9340	0.6206	0.2377	0.8218
4	16	0.01	5	128	0.2	0.0613	0.9339	0.6119	0.2471	0.8139
5	8	0.1	5	128	0.2	0.1200	0.8935	0.2220	0.1857	0.6568
6	8	0.01	10	128	0.2	0.0596	0.9334	0.6134	0.2416	0.8218
7	8	0.01	10	32	0.2	0.0762	0.9204	0.4103	0.1885	0.7400
8	8	0.01	5	64	0.05	0.0675	0.9276	0.5213	0.1923	0.7713
9	8	0.001	5	128	0.2	0.0461	0.9396	0.7601	0.2575	0.9166
10	4	0.001	10	16	0.2	0.0510	0.9362	0.7158	0.2166	0.8653

Rysunek 12: Przygotowane opinie użytkowników do nauki modeli

Wyniki eksperymentu z dostosowywania parametrów dla modelu NeuMF:

n	latent_dim_mf	latent_dim_mlp	dense_layers_1	dense_layers_2	learning_rate	n_epochs	batch_size	val_split	loss	accuracy	precision	recall	auc
1	4	16	16	4	0.01	5	128	0.2	0.0485	0.9380	0.7052	0.2468	0.8822
2	4	16	16	4	0.01	10	128	0.2	0.0490	0.9373	0.7022	0.2552	0.8807
3	4	16	16	4	0.001	10	128	0.2	0.0474	0.9391	0.6921	0.2862	0.8915
4	4	16	16	4	0.001	10	128	0.2	0.0485	0.9383	0.6914	0.2546	0.8786
5	4	4	8	8	0.001	10	128	0.2	0.0486	0.9372	0.6828	0.2618	0.8845
6	16	8	8	8	0.001	10	32	0.1	0.0360	0.9559	0.8168	0.5182	0.9223
7	16	8	8	8	0.001	20	128	0.1	0.0324	0.9624	0.8523	0.5815	0.9140

Rysunek 13: Przygotowane opinie użytkowników do nauki modeli

Analizując powyższe wyniki, zauważyłem kilka kluczowych zależności:

- **Embedding size:** Wzrost rozmiaru osadzeń (embedding size) miał zmienny wpływ na wyniki modelu. Większe rozmiary osadzeń (np. 16) nie zawsze przekładały się na lepsze wyniki, a czasem prowadziły do pogorszenia wartości loss i auc.
- **Learning rate:** Współczynnik uczenia miał znaczący wpływ na wyniki. Niższy współczynnik uczenia (0.001) przyniósł lepsze wyniki w przypadku wszystkich miar, w szczególności dla loss i auc.

- **Batch size:** Eksperymentowanie z różnymi rozmiarami wsadów (batch size) wskazało, że standardowy rozmiar 128 działał najlepiej, choć większe wsady (64) również dawały przyzwoite wyniki.
- **Validation split:** Wartość `val_split` nie miała znaczącego wpływu na wyniki w porównaniu do innych parametrów, jednak niższa wartość (0.05) dawała nieco lepsze wyniki w niektórych przypadkach.

Podsumowując, przeprowadzone eksperymenty wykazały, że modele NeuMF osiągają lepsze wyniki w porównaniu do modeli MF w kontekście rekomendacji filmów, co sugeruje ich większą przydatność w rzeczywistych zastosowaniach.

## 5 Testy

W celu przeprowadzenia testów, wyimportowałem najlepsze modele wytrenowane podczas procesu dostosowywania parametrów. Wykorzystałem dwa modele: Matrix Factorization (MF) oraz Neural Matrix Factorization (NeuMF), które osiągnęły najlepsze wyniki na zbiorze walidacyjnym.

Następnie napisałem prostą aplikację w frameworku Flask, która umożliwia dokonywanie predykcji rekomendacji filmów dla użytkowników. Aplikacja przyjmuje żądania HTTP POST zawierające identyfikator użytkownika (*userId*) oraz typ modelu (*modelType*) i zwraca JSON z najlepszymi rekomendacjami filmowymi.

Aplikacja korzysta z wcześniej przygotowanych danych, takich jak *ratings\_small.csv*, *cleaned\_movies\_metadata.csv*, *user\_id\_mapping.csv* oraz *imdb\_id\_mapping.csv*. Dzięki temu możliwe jest mapowanie identyfikatorów użytkowników i filmów oraz zwracanie w odpowiedzi zarówno identyfikatorów *imdb\_id*, jak i tytułów filmów.

Poniżej przedstawiam ogólną architekturę aplikacji:

- Import modeli: Modele MF i NeuMF są ładowane z plików zapisanych w formacie HDF5.
- Mapowanie danych: Aplikacja korzysta z plików *user\_id\_mapping.csv* oraz *imdb\_id\_mapping.csv* w celu mapowania identyfikatorów użytkowników i filmów na odpowiednie wartości używane w modelach.
- Predykcja: Na podstawie przesłanych danych aplikacja generuje predykcje rekomendacji filmów dla użytkownika, który nie oglądał jeszcze pewnych filmów.
- Zwracanie wyników: Wynikiem działania aplikacji jest lista rekomendacji zawierająca *imdb\_id*, tytuł filmu oraz procentowe oszacowanie trafności predykcji.

Przykładowe żądanie HTTP POST:

```
POST /recommend HTTP/1.1
Host: localhost:5000
Content-Type: application/json

{
  "userId": 129,
  "modelType": "neumf"
}
```

Przykładowa odpowiedź JSON:

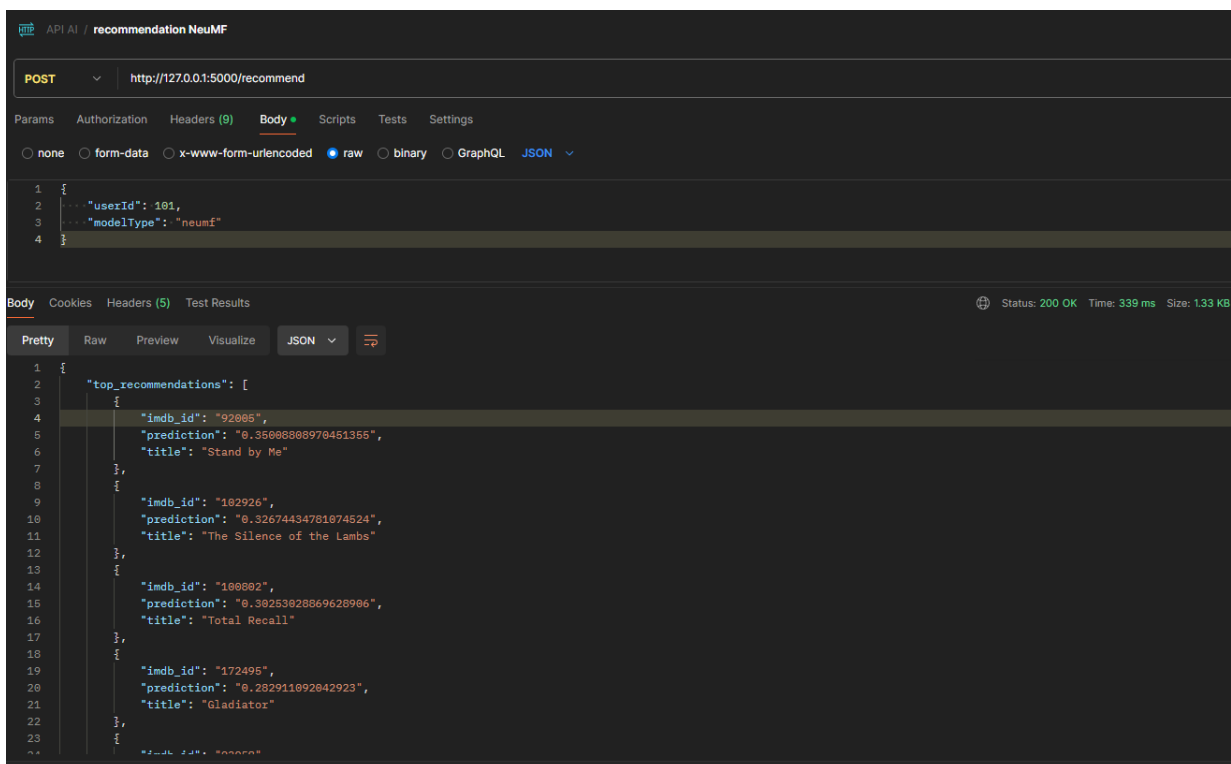
```
{
  "top_recommendations": [
    {
      "imdb_id": "120794",
      "title": "The Prince of Egypt",
      "prediction": "68.55%"
    },
    {
```

```

    "imdb_id": "162222",
    "title": "Cast Away",
    "prediction": "59.55%"
  },
  {
    "imdb_id": "169547",
    "title": "American Beauty",
    "prediction": "45.55%"
  }
]
}

```

Możliwość wyboru modelu pozwala nam zobaczyć oraz porównać zwracane predykcje, przykładowo dla tego samego użytkownika.



Rysunek 14: Podgląd użycia aplikacji udostępniającej API do wykorzystania modeli

Dzięki tej aplikacji możliwe jest łatwe i szybkie uzyskiwanie rekomendacji filmowych na podstawie ocen użytkowników oraz modeli MF i NeuMF, co pozwala na praktyczne wykorzystanie wytrenowanych modeli w systemach rekomendacji filmów.

## 5.1 Porównanie wyników

W celu porównania wyników działania modeli MF oraz NeuMF, przeprowadziłem testy dla tego samego użytkownika i porównałem zwrócone rekomendacje filmowe. Poniżej przedstawiam wyniki dla użytkownika o identyfikatorze 101:

```
{
  "top_recommendations": [
    {
      "imdb_id": "107290",
      "prediction": "0.44888967275619507",
      "title": "Jurassic Park"
    },
    {
      "imdb_id": "114746",
      "prediction": "0.4343724250793457",
      "title": "Twelve Monkeys"
    },
    {
      "imdb_id": "102926",
      "prediction": "0.4127289652824402",
      "title": "The Silence of the Lambs"
    },
    {
      "imdb_id": "109830",
      "prediction": "0.41246360540390015",
      "title": "Forrest Gump"
    },
    {
      "imdb_id": "119654",
      "prediction": "0.3829708397388458",
      "title": "Men in Black"
    },
    {
      "imdb_id": "112573",
      "prediction": "0.38149788975715637",
      "title": "Braveheart"
    },
    {
      "imdb_id": "110912",
      "prediction": "0.37743884325027466",
      "title": "Pulp Fiction"
    },
    {
      "imdb_id": "114369",
      "prediction": "0.3586675524711609",
      "title": "Se7en"
    },
    {
      "imdb_id": "119116",
      "prediction": "0.33611422777175903",
      "title": "The Fifth Element"
    },
    {
      "imdb_id": "114814",
      "prediction": "0.322351336479187",
      "title": "The Usual Suspects"
    }
  ],
  "userId": "100"
}

{
  "top_recommendations": [
    {
      "imdb_id": "137523",
      "prediction": "0.6427794694900513",
      "title": "Fight Club"
    },
    {
      "imdb_id": "110912",
      "prediction": "0.6170629262924194",
      "title": "Pulp Fiction"
    },
    {
      "imdb_id": "169547",
      "prediction": "0.4850800931453705",
      "title": "American Beauty"
    },
    {
      "imdb_id": "102926",
      "prediction": "0.46006089448928833",
      "title": "The Silence of the Lambs"
    },
    {
      "imdb_id": "68646",
      "prediction": "0.44368165731430054",
      "title": "The Godfather"
    },
    {
      "imdb_id": "109830",
      "prediction": "0.4355073869228363",
      "title": "Forrest Gump"
    },
    {
      "imdb_id": "114369",
      "prediction": "0.3468163013458252",
      "title": "Se7en"
    },
    {
      "imdb_id": "114814",
      "prediction": "0.32843178510665894",
      "title": "The Usual Suspects"
    },
    {
      "imdb_id": "172495",
      "prediction": "0.3278125524520874",
      "title": "Gladiator"
    },
    {
      "imdb_id": "119217",
      "prediction": "0.318348228931427",
      "title": "Good Will Hunting"
    }
  ],
  "userId": "100"
}
```

Rysunek 15: Porównanie wyników dla modeli kolejno MF oraz NeuMF

Jak można zauważyć na zamieszczonym obrazie porównawczym, oba modele zwracają podobne rekomendacje filmowe, ale istnieją pewne różnice w rankingach i przewidywanych wartościach trafności.

## 6 Podsumowanie

W ramach tego projektu zrealizowałem zaawansowany system rekomendacji filmów, obejmujący przygotowanie danych, implementację modeli oraz ich ewaluację. Udało mi się przygotować dane z The Movies Dataset, zawierające informacje o ocenach filmów, metadanych filmów oraz powiązaniach między identyfikatorami filmów. Przeprowadziłem operacje takie jak usunięcie niepotrzebnych kolumn z `links.csv`, przekształcenie danych w `movies_metadata.csv` oraz przygotowanie danych z `ratings.csv`, aby zawierały tylko istotne oceny filmów.

Zaimplementowałem dwa modele rekomendacji: Matrix Factorization (MF) i Neural Matrix Factorization (NeuMF). Model MF reprezentuje użytkowników i filmy jako wektory w przestrzeni o określonej liczbie wymiarów, a model NeuMF łączy tradycyjne podejście MF z głębokimi sieciami neuronowymi. Przeprowadziłem

eksperymenty z różnymi konfiguracjami hiperparametrów, co pozwoliło na dostrojenie modeli i poprawę ich wyników.

Modele oceniałem, wykorzystując miary takie jak loss, accuracy, precision, recall i AUC. Analiza wyników pozwoliła na zidentyfikowanie najlepszych konfiguracji hiperparametrów. Następnie stworzyłem aplikację we frameworku Flask, umożliwiającą predykcję ocen filmów na podstawie ocen użytkowników. Aplikacja korzysta z przygotowanych danych, efektywnie mapując identyfikatory filmów i użytkowników oraz zwracając szczegółowe informacje o rekomendowanych filmach.

Testy wykazały, że oba modele, MF i NeuMF, dostarczają trafne rekomendacje, jednak NeuMF osiągnął nieco lepsze wyniki, szczególnie pod względem precyzji. Przykłady zwróconych rekomendacji potwierdziły skuteczność obu podejść.

Podsumowując, wykonanie tego projektu było dla mnie ciekawym doświadczeniem, gdyż temat ten jest dość trudny, a w internecie nie ma jeszcze dużej ilości wiedzy na temat NeuMF. Projekt ten dostarczył mi cennych doświadczeń w zakresie implementacji i ewaluacji systemów rekomendacji, stanowiąc fundament do dalszych prac i ulepszeń.

## 6.1 Wykorzystanie modeli w różnych dziedzinach

Modele rekomendacyjne, takie jak MF i NeuMF, mają szerokie zastosowanie nie tylko w kontekście rekomendacji filmowych. Mogą być one z powodzeniem używane w różnych dziedzinach, takich jak:

### E-commerce

W sektorze e-commerce, modele rekomendacyjne mogą być wykorzystywane do personalizacji ofert produktowych dla klientów. Dzięki analizie wcześniejszych zakupów oraz ocen produktów przez klientów, modele mogą sugerować produkty, które mają największe prawdopodobieństwo zainteresowania klienta, co zwiększa prawdopodobieństwo zakupu.

### Social Media

W mediach społecznościowych, takie modele mogą być używane do rekomendacji treści, znajomych oraz grup. Analiza interakcji użytkowników z różnymi postami, stronami i innymi użytkownikami pozwala na dostarczanie spersonalizowanych rekomendacji, które zwiększają zaangażowanie użytkowników na platformie.

### Streaming muzyki

W usługach streamingowych muzyki, modele rekomendacyjne mogą analizować historię odtwarzania użytkownika oraz jego oceny, aby proponować nowe utwory i artystów, które mogą przypaść mu do gustu. Dzięki temu użytkownik może odkrywać nową muzykę dostosowaną do jego preferencji.

### Medycyna

W medycynie, modele rekomendacyjne mogą odgrywać kluczową rolę w personalizacji opieki zdrowotnej. Przykładowe zastosowania obejmują:

**Rekomendacja terapii** Analiza danych pacjentów z historią leczenia i odpowiedzi na różne terapie może pomóc w rekomendowaniu najbardziej skutecznych terapii dla nowych pacjentów z podobnymi profilami zdrowotnymi.

**Personalizacja leków** Modele rekomendacyjne mogą sugerować leki na podstawie wcześniejszych reakcji pacjentów na leki i ich indywidualnych cech zdrowotnych, co może prowadzić do bardziej efektywnego i spersonalizowanego leczenia.

**Rekomendacja badań diagnostycznych** Na podstawie danych o symptomach pacjentów i wcześniejszych wynikach badań, modele mogą rekomendować najbardziej odpowiednie testy diagnostyczne, zwiększając szanse na szybsze i dokładniejsze postawienie diagnozy.