Metody Odkrywania Wiedzy

Projekt 2018 Z

Predykcja ocen książek – badania

Igor Markiewicz
Aleksander Droszcz

Prowadzący – dr inż. Paweł Cichosz

Spis treści

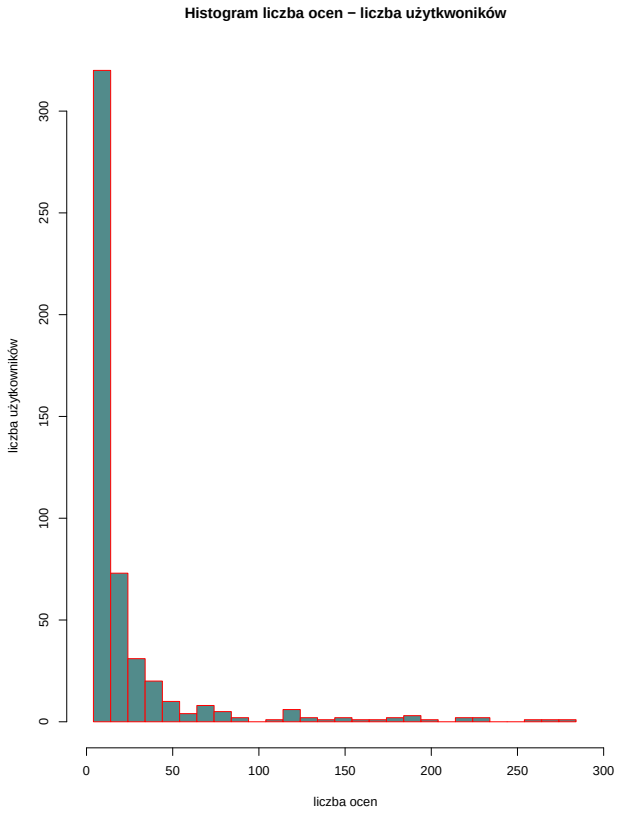
1 Założenia

Po wstępnych testach zostały poczynione następujące założenia:

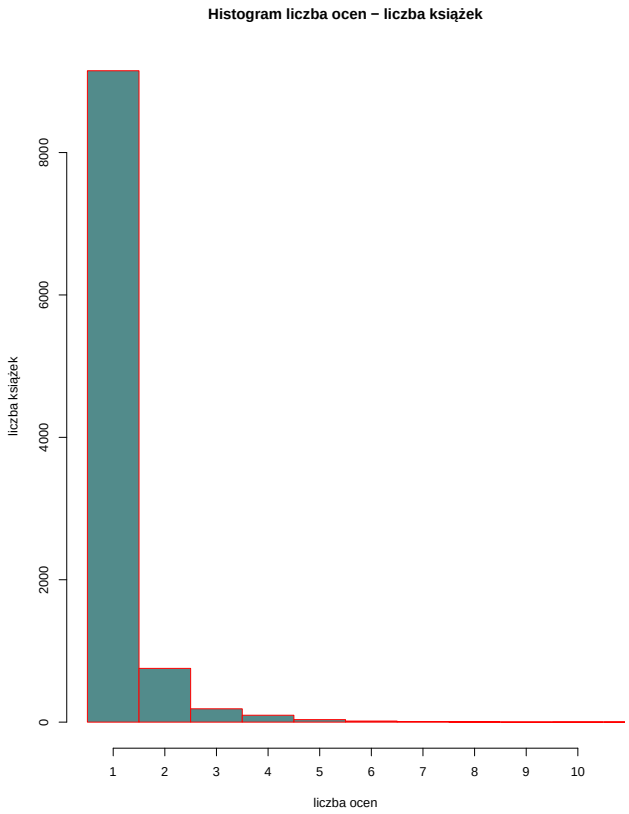
- zrezygnowano z użycia metody Item Based Collaborative Filtering – dla mniejszych danych zwracała wartości nieokreślone, dla większych jej działanie trwało bardzo długi oraz było bardzo zasobożerne pamięciowo
- na wstępie zostało ustawione ziarno standardowego generatora liczb pseudolosowych na wartość 1648
- w każdym teście zastosowano 5. krotną walidację krzyżową
- do zbioru ucząco – testowego zostało wybranych losowo 500. użytkowników wraz z ich wszystkimi ocenami, ale tylko takich których liczba ocen jest równa co najmniej 5
- procedura testowa polega tym że jeśli istnieje n ocen, a użytkownik ma ich $m \leq n$, to wybieramy losowo $k \leq m$ ocen przeznaczonych dla algorytmu do odtworzenia $n-k$ ocen. Po odtworzeniu przeprowadzamy testy na nieużytych $m-k$ ocenach przeznaczonych do validacji. W przypadku testów zdecydowano się na testowanie pojedynczej oceny (parametr *given = -1*)
- procedura optymalizacji parametrów polegała na przyjęciu wartości predefiniowanych a następnie iteracyjnym testowaniu jednego parametru, ustawieniu jego najlepszej wartości i przejściu do kolejnego testowanego parametru. Uniknięto w ten sposób zbyt duzej liczby obliczeń.

2 Badania

2.1 Wstępna analiza danych

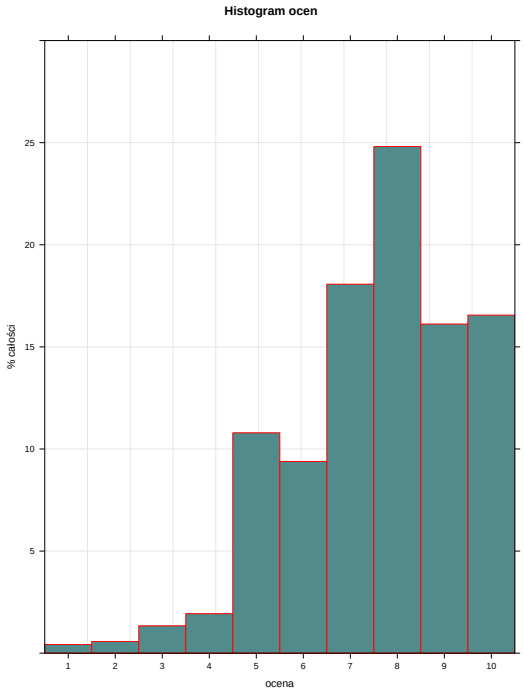


Rys. 1: Histogram liczba ocen – liczba użytkowników



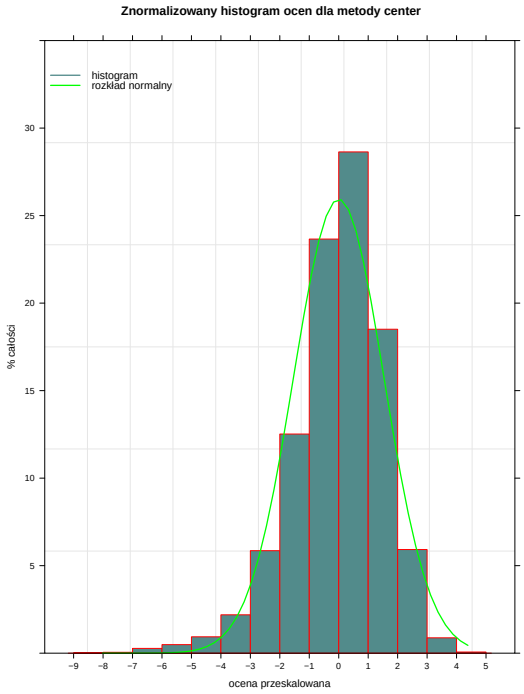
Rys. 2: Histogram liczba ocen – liczba książek

Możemy zauważyć że zarówno zależność ilości użytkowników od liczby wystawionych ocen jak również zleżność liczby książek od liczby ocen jest szybko malejąca. Sprawdzono że dla wylosowanych 500. użytkowników istniej 10260 unikatowych książek oraz 12092 ocen co stanowi $\frac{12092}{500 \cdot 10260} \cdot 100 \approx 0.24\%$ całej macierz. Możemy więc wnioskować że macierz ocen jest siline rzadka, co zapewne będzie miało przełożenie na otrzymane wyniki.

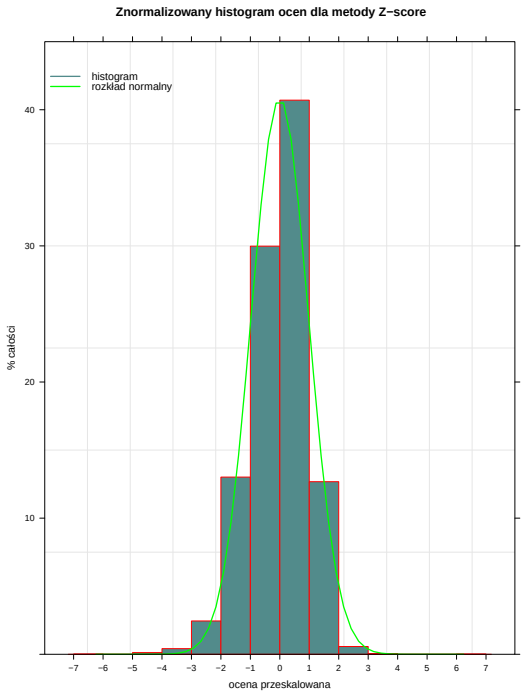


Rys. 3: Histogram rozkładu ocen

Możemy zauważyć że rozkład ocen jest niesymetryczny i użytkownicy mają tendencję do wystawiania wyższych ocen.



Rys. 4: Znormalizowany histogram ocen – metoda *center*

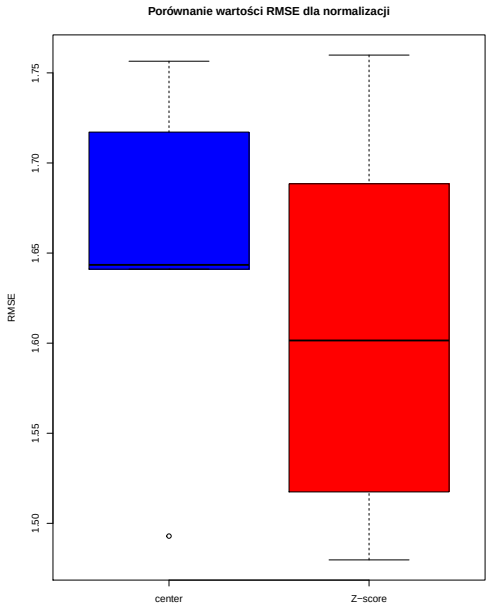


Rys. 5: Znormalizowany histogram ocen – metoda *Z–score*

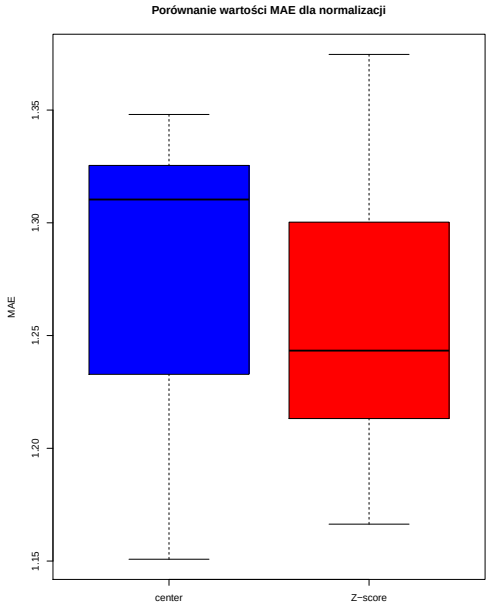
W przypadku metody *Z–score* otrzymujemy bardziej symetryczny histogram.

2.2 Optymalizacja algorytmu User Based Collaborative Filtering

2.2.1 Testy rodzaju normalizacji danych dla UBCF (center, Z–score)



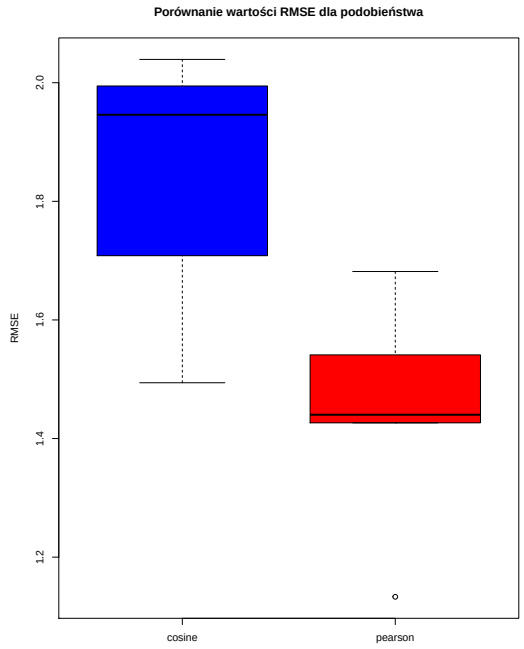
Rys. 6: UBCF - test rodzajów normalizacji - błąd RMSE



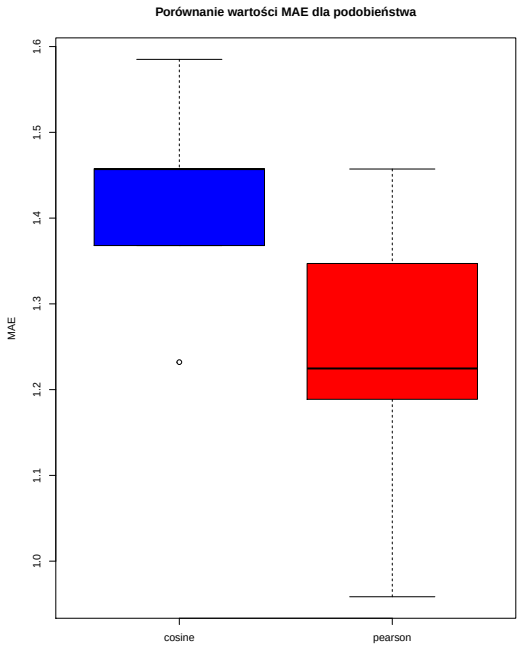
Rys. 7: UBCF - test rodzajów normalizacji - błąd MAE

Jak widać na zamieszczonych powyżej wykresach, lepszą metodą w przypadku algorytmu UBCF okazuje się być metoda 'Z-score', która osiąga mniejsze błędy RMSE oraz MAE.

2.2.2 Testy metryk podobieństwa dla UBCF(cosine, Pearson)



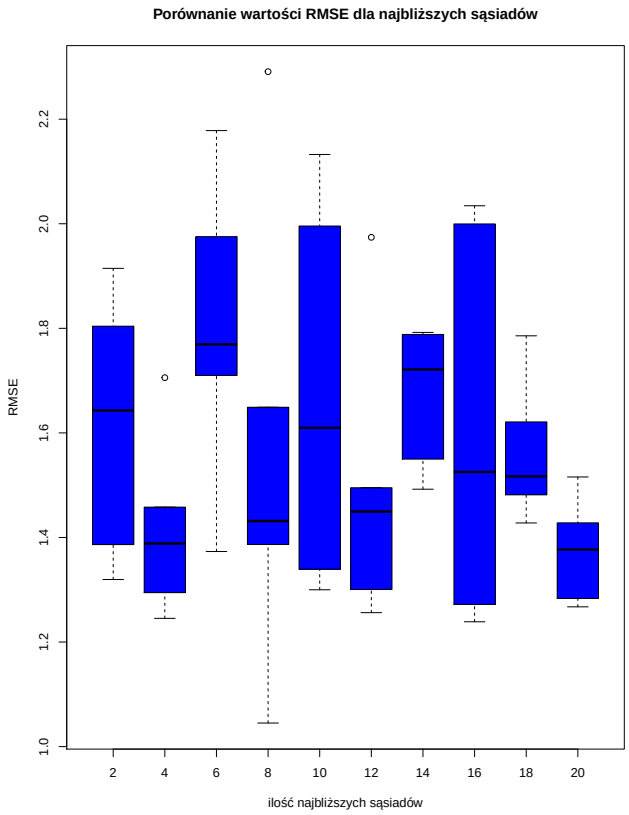
Rys. 8: UBCF - test metryk podobieństwa - błąd RMSE



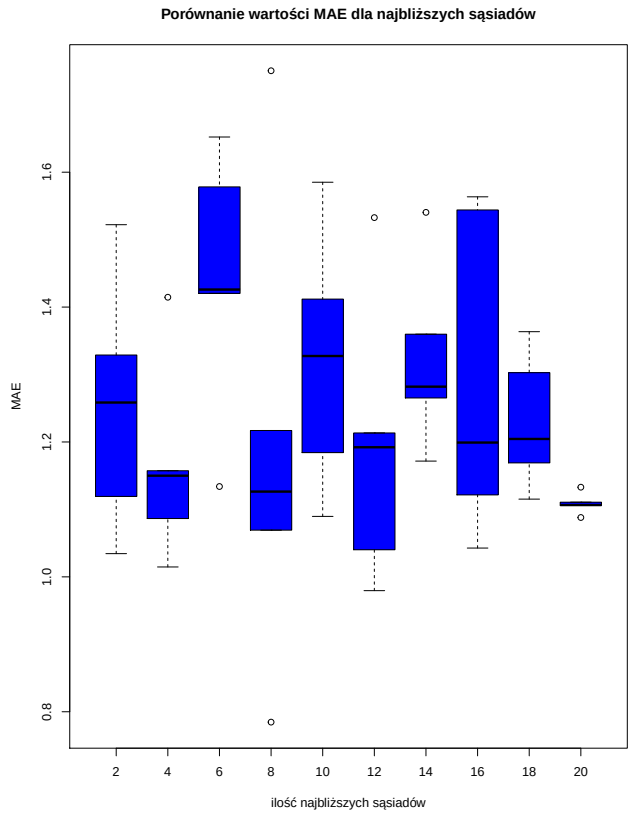
Rys. 9: UBCF - test metryk podobieństwa - błąd MAE

Możemy zauważyć iż lepsze wyniki zostały osiągnięte dla współczynnika Pearsona jako metryki podobieństwa.

2.2.3 Testy dla różnej ilości najbliższych sąsiadów



Rys. 10: UBCF - test liczby sąsiadów - błąd RMSE

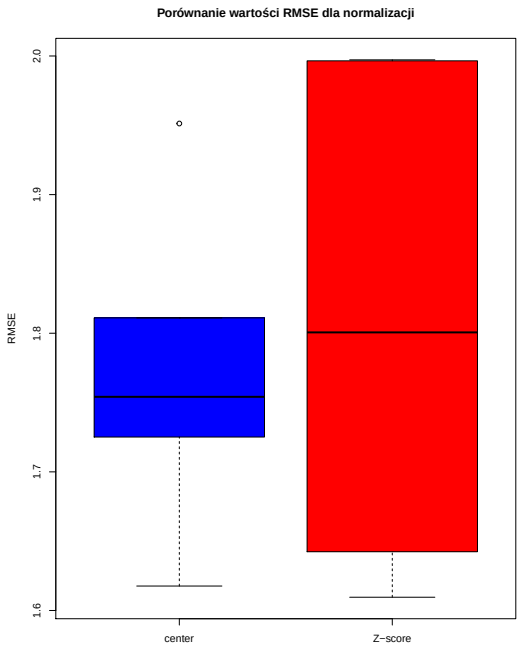


Rys. 11: UBCF - test liczby sąsiadów - błąd MAE

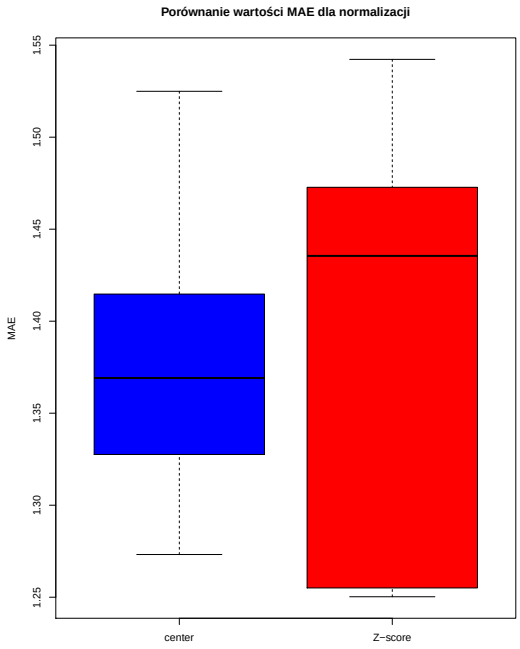
Jako kompromis między ilością najbliższych sąsiadów oraz miarami błedów, zostało przyjęte 8. najbliższych sąsiadów.

2.3 Optymalizacja algorytmu Funk SVD

2.3.1 Testy rodzaju normalizacji danych dla SVDF(center, Z–score)



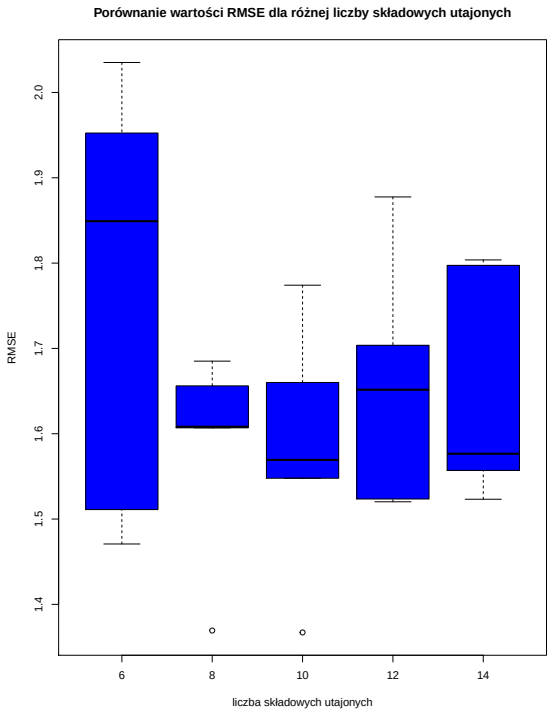
Rys. 12: SVDF - test rodzajów normalizacji - błąd RMSE



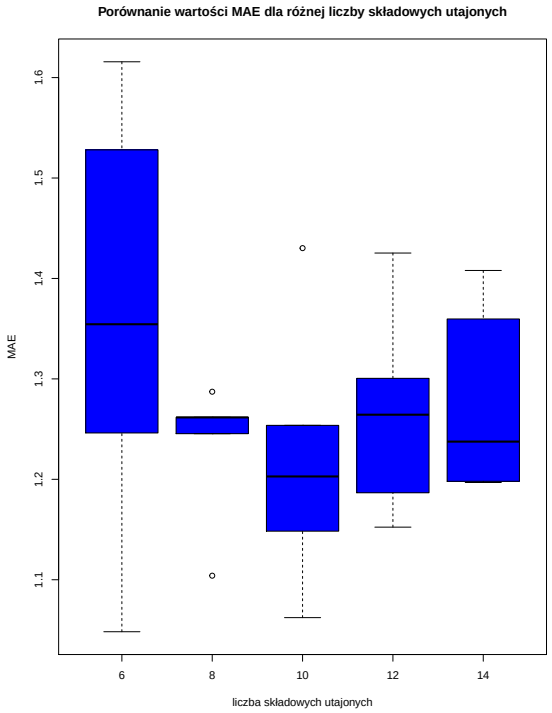
Rys. 13: SVDF - test rodzajów normalizacji - błąd MAE

W przypadku normalizacji danych dla SVDF korzystniej wypadła normalizacja typu center.

2.3.2 Testy dla różnej liczby składowych utajonych SVDF



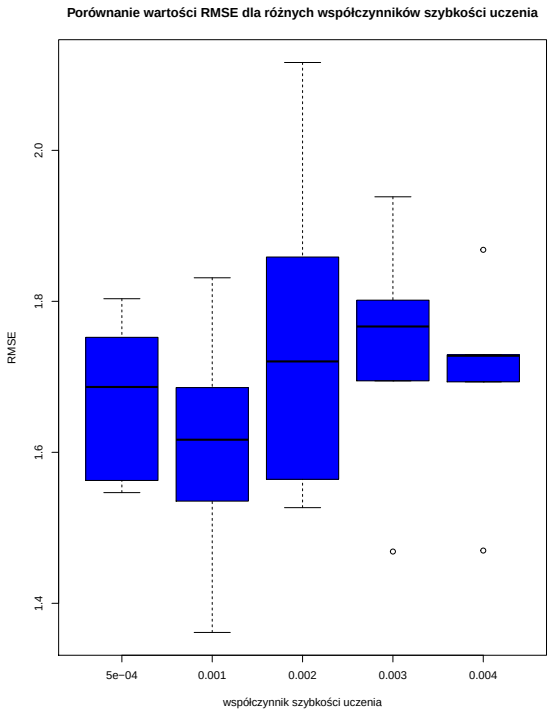
Rys. 14: SVDF - test liczby składowych utajonych - błąd RMSE



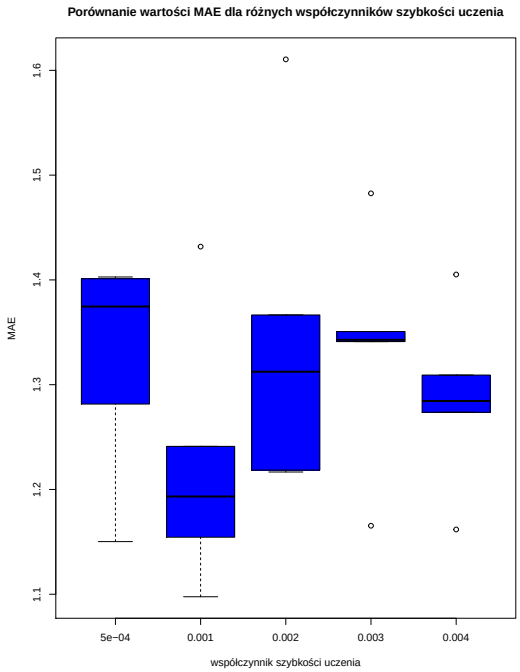
Rys. 15: SVDF - test liczby składowych utajonych - błąd MAE

Jako suboptymalną liczbę składowych utajonych modelu ustalono liczbę 10.

2.3.3 Testy dla różnych współczynników szybkości uczenia SVDF



Rys. 16: SVDF - test wartości współczynnika szybkości uczenia - błąd RMSE

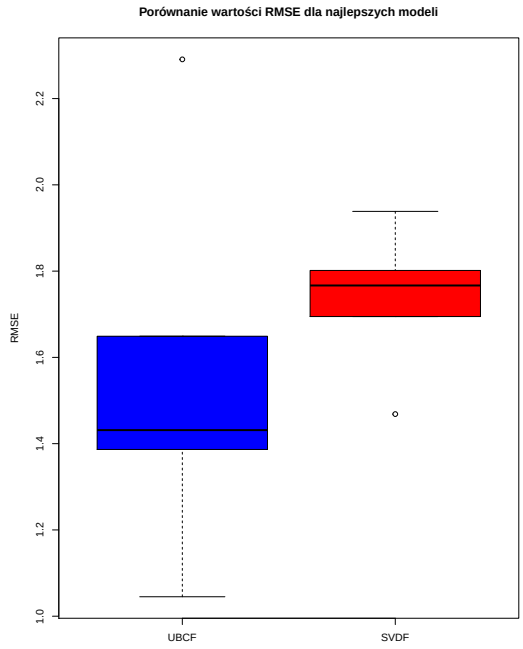


Rys. 17: SVDF - test wartości współczynnika szybkości uczenia - błąd MAE

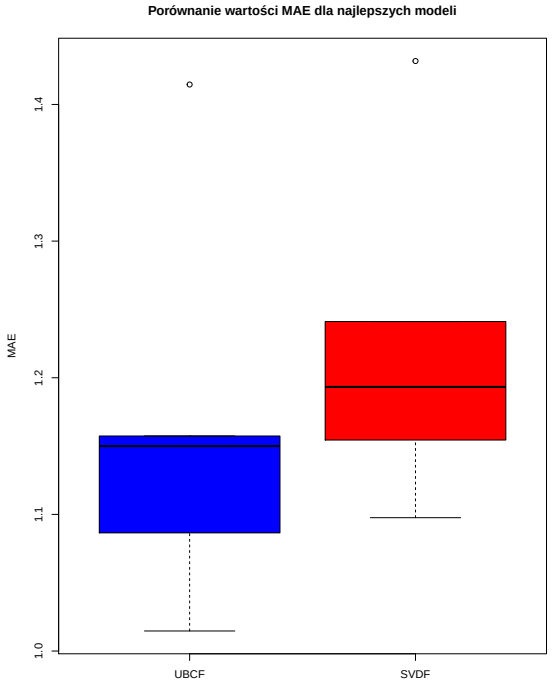
Jako suboptymalną wartość współczynnika szybkości uczenia przyjęto wartość 0.001

2.4 Porównanie najlepszych modeli

Dla najlepszych parametrów zostały porównane poniżej modele UBCF oraz SVDF



Rys. 18: Porównanie algorytmów UBCF oraz SVDF - błąd RMSE



Rys. 19: Porównanie algorytmów UBCF oraz SVDF - błąd MAE

Z wykresów pudełkowych wynika że lepsze rezultaty osiąga algorytm UBCF. W celu ilościowego porównania obu algorytmów przeprowadzono również test statystyczny t – Studenta dla dwóch populacji:

$$\begin{cases} H_0 : \mu_{UBCF} - \mu_{SVDF} = 0 \\ H_1 : \mu_{UBCF} - \mu_{SVDF} \neq 0 \end{cases}$$

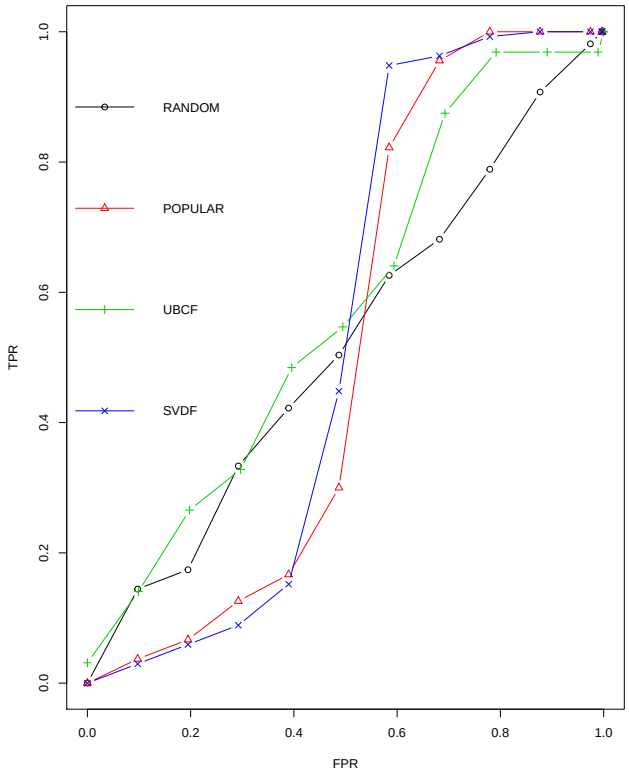
gdzie μ_{UBCF}, μ_{SVDF} oznaczają średnie arytmetyczne RMSE/MAE (z pięciu prób dla najlepszych modeli). W efekcie otrzymano

- dla RMSE: p–value = 0.5464
- dla MAE: p–value = 0.4937

W obu przypadkach z racji na duże wartości p–value nie mamy podstaw do odrzucenia hipotezy zerowej o równości średnich w obu populacjach. Jednym z powodów niewykrycia przez test statystyczny istotej różnicy między średnimi może być mała liczność próby.

2.5 Binaryzacja danych

Po wyznaczeniu optymalnych wartości parametrów dla algorytmów UBCF oraz SVDF zostały one porównane z algorytmami o bardziej trywialnej zasadzie działania (RANDOM - losowe przydzielanie ocen, POPULAR - predykcje bazujące na popularności przedmiotów, nie na subiektywnych odczuciach użytkownika). W tym celu została użyta funkcja 'evaluate' z pakietu 'Recommenderlab' z parametrem 'type' ustawionym na wartość 'topNList'. To umożliwiło binaryzację danych i wygenerowanie krzywych ROC dla każdego z czterech algorytmów. Binaryzacja oznaczała, że każda ocena większa niż 0 (po normalizacji) jest uznawana jako pozytywna. Została wykonana ewaluacja typu 'split' (losowy podział na zbiór uczący 90 % oraz testowy 10 %) a następnie testy typu 'all-but–2' (dwie losowo wybrane, znane oceny przeznaczone do testów, reszta znanych ocen na obliczenie podobieństwa z wyuczonymi wektorami). Parametr 'topNList' oznacza, iż dla każdego użytkownika generowana była lista N polecanych książek. Jeżeli książka o pozytywnej ocenie pojawiała się według danego algorytmu na liście, to był to przykład prawdziwy pozytywny. Jeżeli książka o pozytywnej ocenie nie pojawiała się na liście, to był to przykład fałszywy negatywny. Analogicznie sytuacja wyglądała w przypadku książek o ocenach negatywnych dla danego użytkownika. Wyniki te były dla wszystkich użytkowników uśredniane. Parametrem, na podstawie którego generowane były krzywe ROC, była długść listy N (od 1 do wszystkich książek znajdujących się w macierzy ocen). Poniżej przedstawione zostały otrzymane krzywe ROC.



Rys. 20: Krzywe ROC dla wszystkich omawianych algorytmów

Widać po uzyskanych krzywych ROC, iż algorytmy UBCF oraz SVDF uzyskały lepsze wyniki niż losowe dobieranie rekomendacji. Algorytm SVDF osiągnał bardzo podobne rezultaty jak algorytm POPULAR, a trochę lepsze od nich osiągnął algorytm UBCF. Widać, iż różnice między trywialnymi metodami a algorytmami UBCF oraz SVDF nie są znaczące. W dużej mierze zapewne wynika to z jakości rozpatrywanego zbioru danych. Większość książek miały tylko jedną ocenę, co uniemożliwia skomplikowaną predykcję. Należy również zaznaczyć, iż niektóre funkcje w pakiecie Recommenderlab odbiegają od klasycznych form badania algorytmów. Przede wszystkim generowanie krzywych ROC wyłącznie na podstawie długości rekomendowanej listy N książek, a nie na podstawie zmieniania wartości oceny, która rozdziela oceny negatywne i pozytywne (binaryzacja).

3 Uwagi i wnioski

- stwierdzono że pięciokrotna walidacja krzyżowa może nie być wystarczająca (w części testów możemy zaobserwować zaburzone wyniki takie jak duże rozstępy kwartylowe, nakładanie się skrajnych kwartyli na minimum lub maksimum wyznaczone na podstawie IQR, czy mała wiarygodność testu statystycznego)

- z racji na bardzo rzadką macierz ocen przeprowadzenie testów było silnie utrudnione – być

może pomogłoby wczytanie większej liczby uzytkowników, jednak byłoby to okupione dużo bardziej czasochłonnymi obliczeniami

- zauważono również że istniej możliwość występowania problemu z ziarnem generatora – wydaje się że mogło być ono nadpisywane przez niektóre funkcje, co mogło sprawić że testy dla różnych parametrów nie były wykonywane na takim samym podziale zbiorów podczas walidacji krzyżowej
- problemy z pakietem 'Recommenderlab' – błędy w dokumentacji, brak opisu wszystkich funkcjonalności, dziwne zachowania niektórych metod, brak niektórych funkcjonalności pomagajacych przy testowaniu

4 Bibliografia

[1] https://grouplens.org/datasets/book-crossing/

[2] https://cran.r-project.org/web/packages/recommenderlab/vignettes/recommenderlab.pdf

[3] https://cran.r-project.org/web/packages/recommenderlab/recommenderlab.pdf

[4] https://en.wikipedia.org/wiki/Collaborative_filtering

[5] https://cran.r-project.org/web/packages/rrecsys/rrecsys.pdf

[6] https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_systems)#Funk_SVD

[7] http://nicolas-hug.com/blog/matrix_facto_1

[8] http://nicolas-hug.com/blog/matrix_facto_2

[9] http://nicolas-hug.com/blog/matrix_facto_3

[10] https://www.slideshare.net/DKALab/collaborativefilteringfactorization