Escola de Engenharia da Universidade do Minho
Departamento de Informática
Mestrado em Matemática e Computação
Mestrado Integrado em Engenharia Informática

Sistema de Recomendação SBSFlix

Joel Morais, A70841
Luísa Caldas, PG35974
Tiago Fraga, A74092
Valéria Romanciuc, PG33724

3 de janeiro de 2019

Conteúdo

Capítulo 1

Introdução

Tal como vimos no primeiro trabalho, um sistema de recomendação é uma ferramenta que procura prever a preferência de um utilizador num determinado tema, fornecendo-lhe recomendações relacionadas com o mesmo. O objetivo deste trabalho é desenvolver um sistema de recomendação, sendo que escolhemos criar um para recomendações de filmes, uma vez que foi um dos temas abordados no primeiro trabalho.

Vamos começar por descrever o *dataset* escolhido e os seus atributos, passando de seguida para os *workflows* criados no KNIME para o desenvolvimento do modelo. Além disso, veremos também quais os paradigmas adotados no modelo e quais as vantagens e desvantagens da adoção de cada um.

Capítulo 2

Descrição do *dataset*

O *dataset* escolhido para este trabalho contém dados obtidos através do *site* "MovieLens"e é composto por 100 mil *ratings* atribuídos a cerca de 9 mil filmes diferentes, por parte de 600 utilizadores [1]. A cada utilizador está associado um conjunto de filmes vistos por si, bem como os respetivos *ratings* atribuídos.

Este *dataset* é composto pelos seguintes atributos:

- **MovieId:** número único atribuído a cada filme;
- **Title:** título do filme;
- **Genres:** géneros atribuídos a cada filme;
- **UserId:** número único atribuído a cada utilizador;
- **Rating:** avaliação (de 0 a 5) dado a um filme por utilizador.

De forma a facilitar a utilização dos géneros de um filme posteriormente no *dataset*, foi utilizado o nodo "Java Snippet" para separar os diferentes géneros. Assim, foram criados 17 novos atributos binários, em que cada um representa um género diferente.

Capítulo 3

Abordagens utilizadas

Ao longo deste trabalho, utilizamos diferentes abordagens no sistema de recomendação criado, nomeadamente: *top-n*, abordagem por conteúdo e abordagem colaborativa.

3.1 Top-N

Este método de recomendação baseia-se em recomendar filmes a um utilizador tendo por base os *ratings* ou as visualizações de outros utilizadores.

No nosso sistema de recomendação foram criados dois tipos diferentes de recomendações Top-N. Numa primeira fase, decidimos fazer as recomendações tendo por base a média de *ratings*, isto é, retornamos os 10 filmes com a maior média de *ratings* do *dataset*, excluindo aqueles que o utilizador para a qual é feita a recomendação já tenha visto. Apesar disto, também é relevante ter em conta o número de visualizações por filme para as recomendações a serem feitas. Isto ajuda o utilizador a ter uma ideia mais precisa sobre o "valor" da média dos *ratings*, pois se, por exemplo, apenas uma pessoa tiver visualizado um filme e dado 5, esse filme terá uma média de *ratings* de 5, mas este não é um valor muito representativo.

Numa segunda fase, decidimos fazer uma recomendação tendo por base o número de visualizações, isto é, retornámos os 10 filmes (excluindo os já vistos pelo utilizador) mais vistos pelos utilizadores.

3.2 *Content-based Filtering*

A maneira mais simples de recomendar filmes tendo por base o seu conteúdo é considerar uma base de dados e criar regras que retornem consequentes, isto é, se um utilizador viu dois filmes, então cada um destes filmes vai implicar ver um outro. Esta foi outra abordagem utilizada no trabalho, em que foi criada uma lista de consequentes, aplicando regras de associação. Através dessas regras,

obtivemos o conjunto de consequentes para os filmes vistos por um dado utilizador, sendo este devolvido ao utilizador (excluindo os repetidos).

Uma desvantagem desta estratégia é o facto de, muitas vezes, a lista de consequentes englobar os filmes já vistos pelo utilizador em questão, resultando numa lista vazia para a recomendação.

Outra estratégia baseada em conteúdo adotada foi a criação de *clusters* com base nos géneros de filmes. Nesta abordagem, os filmes são agrupados por *clusters*, de forma a determinarmos qual o *cluster* predominante (aquele que contém o maior número de filmes vistos pelo utilizador). De seguida, são considerados todos os filmes desse *cluster* que ainda não foram vistos, e devolvidos os 10 filmes com maior média de *ratings*.

3.3 *Collaborative Filtering*

Para usarmos uma estratégia de filtragem colaborativa, criámos *clusters* com base nos diferentes utilizadores. De seguida, determinámos em que *cluster* se encontrava o utilizador em questão e, através da distância euclidiana, encontrámos o utilizador mais próximo deste. Isto permitiu obter a lista de filmes visualizados pelo segundo utilizador, do qual excluímos os filmes vistos pelo primeiro, obtendo, assim, um conjunto de recomendações.

Esta abordagem assenta na ideia de que utilizadores com gostos semelhantes terão "boas" recomendações a fazer uns aos outros. No entanto, em caso de utilizadores com um gosto muito diversificado, podemos obter recomendações que não são relevantes para os restantes utilizadores do *cluster*.

Capítulo 4

Descrição dos *workflows*

No nosso *workflow* (ver figura 4.1) temos diversos *wrapped metanodes* criados, de forma a ajudar na organização do trabalho. Apresentamos, de seguida, a descrição de cada um deles.

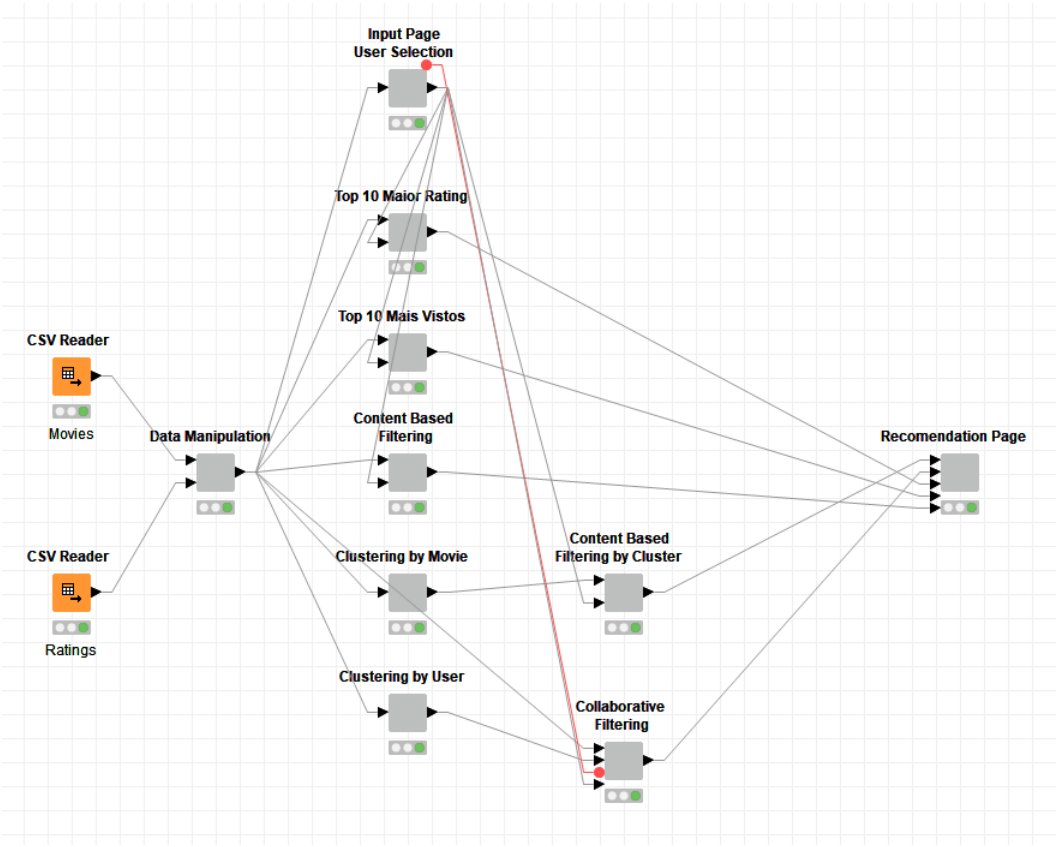

Figura 4.1: *Workflow* Geral

4.1 *Data Manipulation*

Neste *workflow* é realizado um primeiro tratamento dos dados. Primeiramente, uma vez que existem dois ficheiros diferentes com informação (um para os filmes e outro para os *ratings*), é feito um *joiner* que junta os dois ficheiros através da coluna *movieId*. Em seguida, é feita a separação dos géneros (tal como referido no capítulo 2).

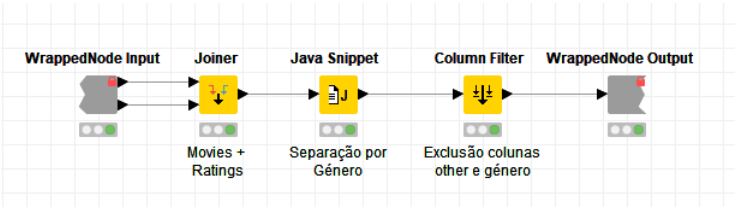

Figura 4.2: *Wrapped Metanode: Data Manipulation*

4.2 *Input Page: User Selection*

Neste *workflow* é feita a filtração dos filmes vistos pelo utilizador e também é criado um *metanode* ("Input Page") para a interface do sistema de recomendação.

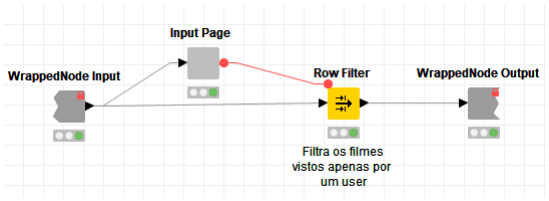

Figura 4.3: *Wrapped Metanode: Input Page: User Selection*

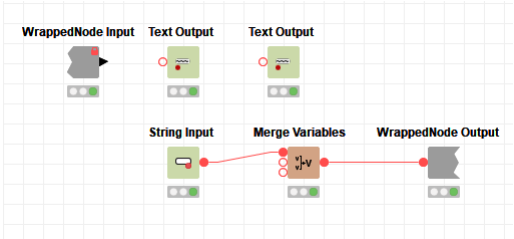

Figura 4.4: *Wrapped Metanode: Input Page*

4.3 Top 10 Maior *Rating*

Este *workflow* foi criado com o objetivo de obtermos os 10 filmes com melhor *rating* que um dado utilizador ainda não tenha visto. Assim, começamos por receber o conjunto de filmes com a média de *ratings* respetiva, bem como o conjunto de filmes visto pelo utilizador. Depois de juntarmos ambos através do *Joiner* e excluirmos do *dataset* os filmes vistos pelo *user* (através do *Row Filter*), ordenamos os filmes por ordem decrescente de média de *rating* e selecionamos apenas as 10 primeiras linhas.

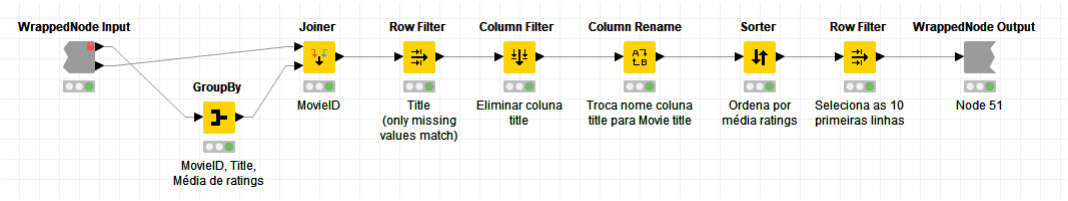

Figura 4.5: *Wrapped Metanode:* Top 10 Maior *Rating*

4.4 Top 10 Mais Vistos

À semelhança do *workflow* anterior, este tem o objetivo de devolver os 10 filmes mais vistos do *dataset*, mas que um dado utilizador ainda não tenha visto.

O processo é análogo ao anterior: começamos por obter o conjunto de filmes com o número de visualizações respetivo, bem como o conjunto de filmes visto pelo utilizador. Depois de juntarmos os dois conjuntos, excluímos os filmes visualizados pelo utilizador e obtemos os 10 primeiros filmes ordenando o conjunto por ordem decrescente de visualizações.

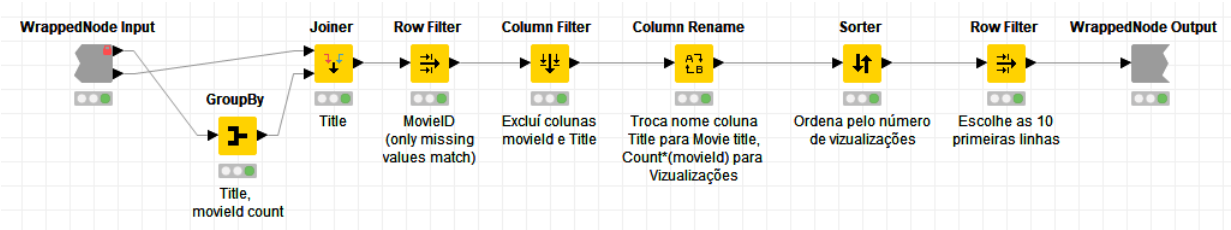

Figura 4.6: *Wrapped Metanode:* Top 10 Mais Vistos

4.5 *Content Based Filtering*

Com este *wrapped metanode* pretendemos obter um conjunto de filmes "semelhantes" aos que um dado utilizador já viu. Para isso, começamos por obter uma lista de filmes vistos por cada utilizador com o nodo *GroupBy* e criamos regras de associação para cada filme com o nodo *Association Rule Learner*. Assim, obtemos pares de antecedentes e consequentes para poder ser feita a recomendação. Por exemplo, a um utilizador que tenha visto o filme "The Silence of The Lambs" (antecedente), será recomendado o "The Sixth Sense" (consequente), segundo estas regras de associação.

De seguida, obtemos o conjunto de filmes dado pelas regras dos consequentes para os filmes já vistos pelo utilizador, através do *Joiner*. Depois de eliminarmos os repetidos desta lista, eliminamos, também, aqueles que o utilizador já viu da lista de consequentes. No caso do utilizador selecionado para este *workflow*, a lista retornada era vazia. Esta é uma desvantagem desta abordagem, uma vez que pode retornar um conjunto vazio para a recomendação.

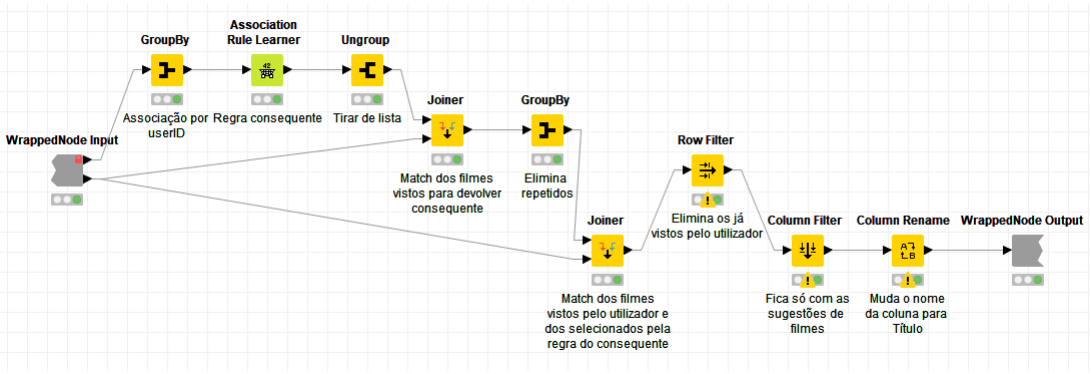

Figura 4.7: *Wrapped Metanode: Content Based Filtering*

4.6 *Clustering by Movie*

Neste *metanode* são criados *clusters* por géneros de filme, usando, para isso, o PCA para redimensionar o *dataset*. Assim, em primeiro lugar, é feita uma divisão dos dados em duas dimensões através do PCA. Em seguida, é feita uma agregação por géneros (pelos 17 atributos criados), *movieId*, título e média de *ratings*.

Para determinar o número de *clusters* necessários, foi feito um *workflow* chamado *Loop Clusters*, que usa o *k-Means* para calcular o erro, através da métrica *Mean Squared Error* (MSE). Este *loop* corre 10 vezes, vendo o erro para até 10 *clusters* (ver figura 4.9).

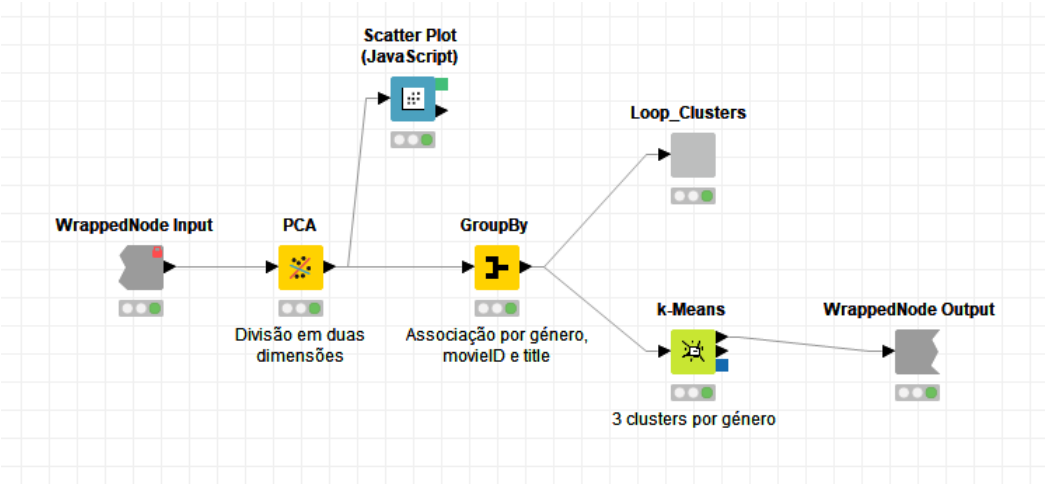

Figura 4.8: *Wrapped Metanode: Cluster by Movie*

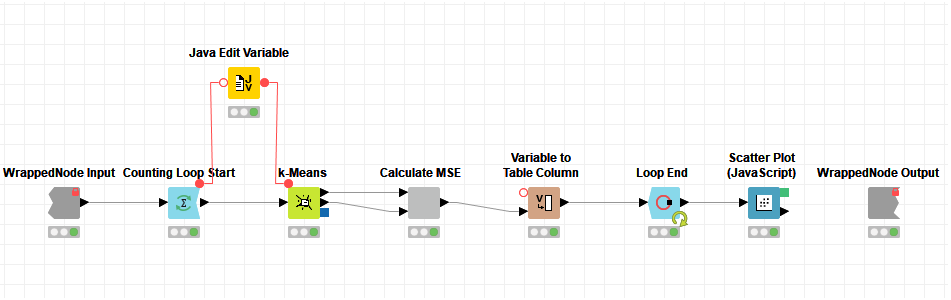

Figura 4.9: *Wrapped Metanode: Loop Cluster*

O cálculo do MSE é feito noutro *workflow* (ver figura 4.10), aplicando a fórmula

$$\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2,$$

onde n representa o número de observações, y_i representa o valor da previsão e $\hat{y}_i$ representa o valor observado.

Para determinar o valor ótimo de *clusters* a usar, foi utilizado o *Elbow method*. Como podemos ver na figura 4.11, segundo este método, o melhor número de *clusters* é 3, uma vez que é o ponto a partir do qual a diferença dos erros é bastante semelhante.

Posto isto, é feito um novo *K-Means* (ver figura 4.8) com 3 *clusters*, ficando os nossos filmes agrupados em 3 *clusters* diferentes.

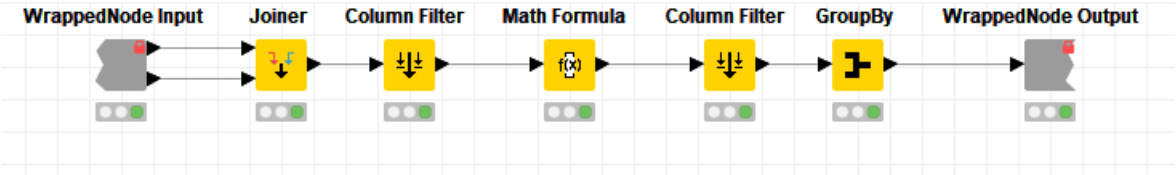

Figura 4.10: *Wrapped Metanode: Calculate MSE*

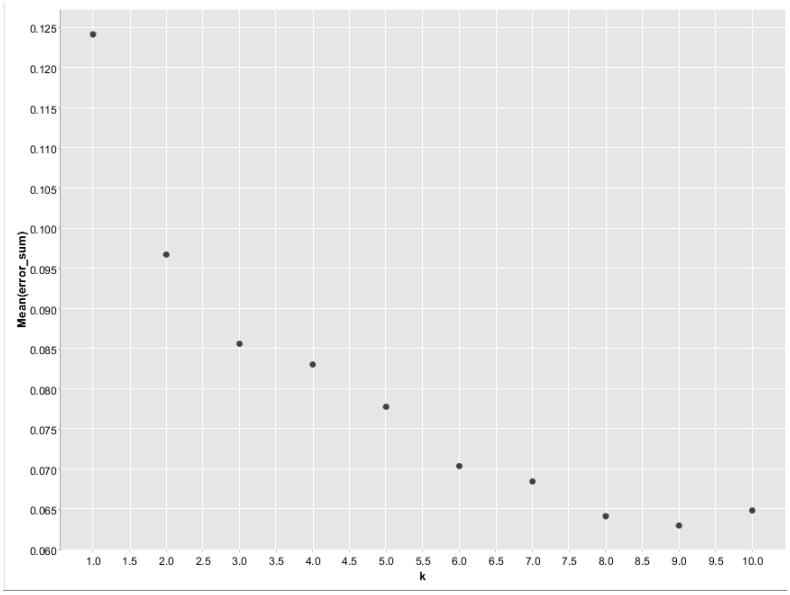

Figura 4.11: *Scatter plot* com o número de *clusters* por erro.

4.7 *Content Based Filtering by Cluster*

Neste *metanode*, temos como objetivo obter um conjunto de filmes recomendados com base nos *clusters* por géneros, de acordo com os género mais visto por um dado utilizador.

Para isso, começamos por obter o conjunto de filmes vistos pelo utilizador, aos quais associamos um dado *cluster* através do *Joiner*. De seguida, obtemos o *cluster* predominante, isto é, aquele onde se inserem mais filmes visualizados. Finalmente, obtemos os 10 filmes com melhor pontuação desse *cluster*, mas que ainda não tenham sido vistos pelo utilizador.

4.8 *Clustering by User*

Neste workflow são seguidos os mesmos passos que no anterior, com a exceção de que é feita uma agregação por *userId* e, além disso, é feita a média de todos os géneros individualmente e dos

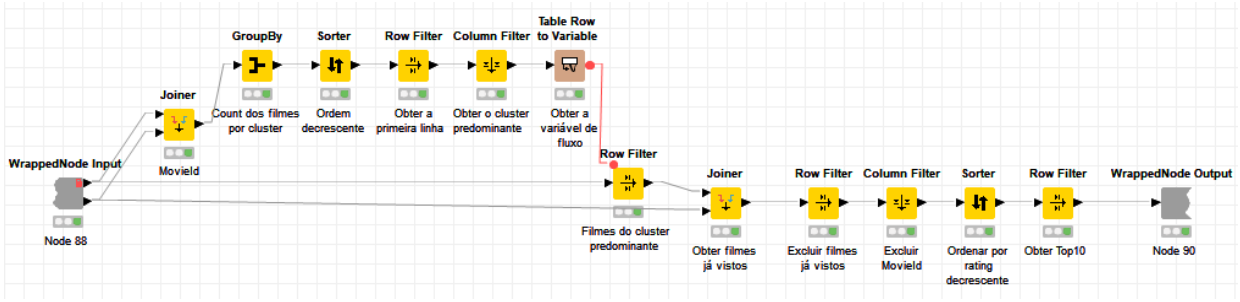

Figura 4.12: *Wrapped Metanode: Content Based Filtering by Cluster.*

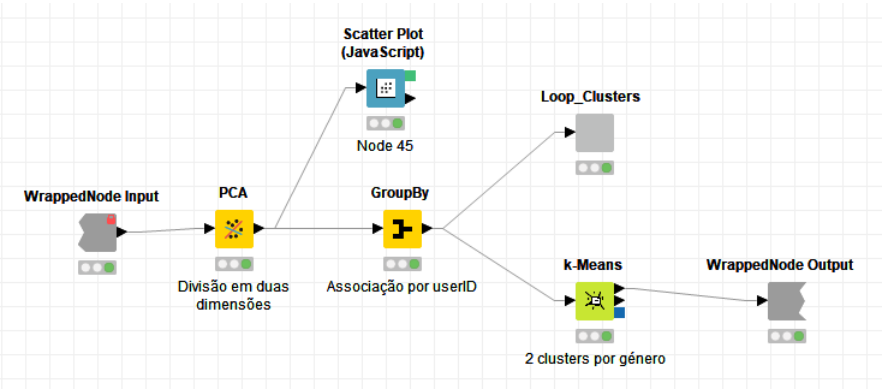

Figura 4.13: *Wrapped Metanode: Clustering by User.*

ratings. Para determinar o número ótimo de *clusters*, foi usado o mesmo método que na secção anterior, sendo o resultado 2 *clusters*.

4.9 *Collaborative Filtering*

Com este *workflow*, pretendemos devolver ao utilizador um conjunto de recomendações com base nos filmes preferidos do utilizador mais "próximo" deste dentro do *cluster* onde se insere.

Assim, começamos por separar o utilizador em questão dos restantes, através da aplicação de dois *Row Filters*. Pretendemos encontrar o utilizador mais "próximo" do primeiro, usando, para isso, a distância euclidiana. Estes dados servem de input ao nodo *Similarity Search*, que nos devolve o *userID* do utilizador mais próximo.

De seguida, obtemos os filmes vistos por esse utilizador, dos quais excluímos o conjunto de filmes visto pelo utilizador inicial, para o qual estamos a criar a recomendação. Finalmente, ordenamos por ordem decrescente de *rating* esta última lista e selecionamos os 10 primeiros para a recomendação.

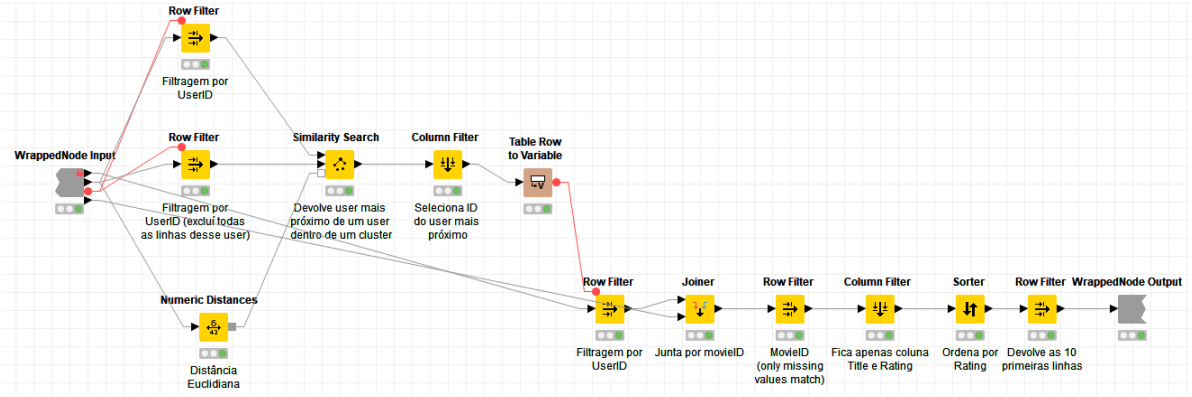

Figura 4.14: *Wrapped Metanode: Collaborative Filtering.*

4.10 *Recommendation Page*

Para a interface da página de recomendações, criámos uma tabela de recomendações por cada abordagem mencionada anteriormente (ver figura 4.15).

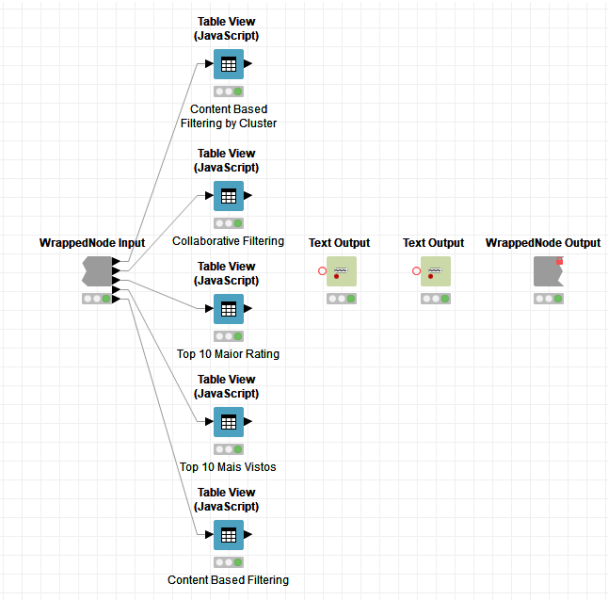

Figura 4.15: *Wrapped Metanode: Recommendation Page.*

Capítulo 5

Sistema de recomendação desenvolvido

Neste capítulo, apresentamos a interface do sistema de recomendação desenvolvido, a que chamámos SBSFlix.

Este é composto por duas páginas: a primeira, de apresentação, onde o utilizador deve introduzir o seu *userID*. Já a segunda página apresenta as recomendações para este utilizador, de acordo com as abordagens descritas no capítulo anterior.

Figura 5.1: Página Principal

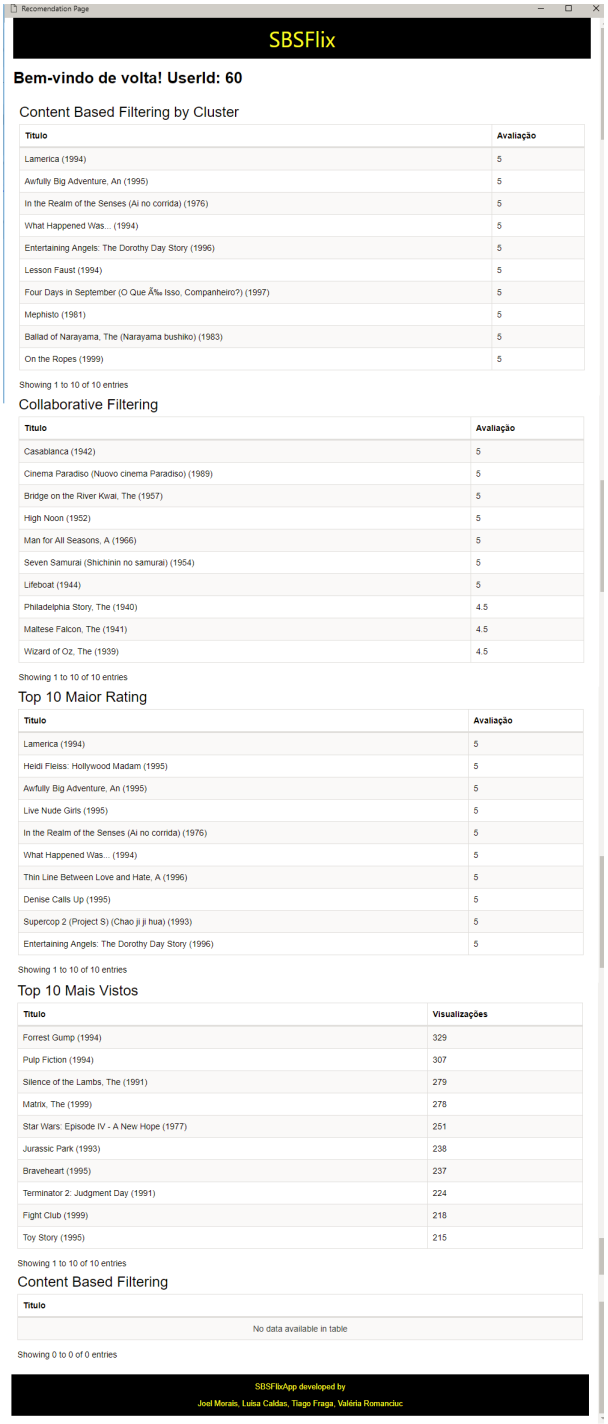

Figura 5.2: Página com as Recomendações

Capítulo 6

Conclusão

O nosso sistema de recomendação apresenta várias estratégias para alguns dos paradigmas. Apresentamos duas estratégias de Top-N (Top 10 Maior Rating e Top 10 Mais Vistos), duas de filtragem baseada em conteúdo (*Content Based Filtering* e *Content Based Filtering by Cluster*) e uma de filtragem colaborativa (*Collaborative Filtering*). A conjunção destas abordagens no nosso sistema permite-nos classificá-lo como híbrido.

Uma conclusão que pudemos tirar foi que, ao apresentar apenas a opção de filtragem baseada em conteúdo utilizando a regra do consequente no nosso *dataset*, corremos o risco de não ter recomendações disponíveis para o utilizador, uma vez que ele já pode ter visto todos os filmes a serem recomendados. Uma vez que esta foi uma das nossas primeiras ideias, a partir daqui decidimos seguir outras direções mais favoráveis. Assim, podemos concluir que, no geral, as outras recomendações são muito mais fiáveis que esta.

De forma a melhorar o nosso sistema, poderíamos tentar contornar o problema do *cold start*, isto é, conseguirmos criar recomendações para um utilizador novo. Para isso, teríamos de guardar um conjunto de filmes vistos por esse utilizador, sendo este posteriormente cruzado com a nossa base de dados para obter as recomendações, tal como foi descrito no capítulo 4.

Bibliografia

[1] MovieLens. https://grouplens.org/datasets/movielens/?fbclid=IwAR3yOgPw83_jYokReO_nNpCDjwdMtT9HY1J3bJYaHHHWCzGz1HpmtcogtQs. Consultado a 15-12-2018.