RECOMENDAÇÃO DE ANIMES UTILIZANDO MACHINE LEARNING, UMA ABORDAGEM BASEADA EM AVALIAÇÕES DOS USUÁRIOS.

Douglas Waltricke Freitas[1]

Gustavo dos Santos Lucca[2]

Resumo: O presente artigo decorre do desenvolvimento de uma aplicação web para gerar recomendações de animes assertivas baseadas em avaliações, utilizando *Machine learning*, uma subárea da inteligência artificial, especializada em usufruir de um conjunto de dados para fazer classificações e predições. Utilizando os conceitos de *Machine learning* com algoritmos de similaridade de usuário e vizinhança, é possível gerar recomendações e projetar a nota que o usuário irá dar para as obras sugeridas após assisti-las, com margem de erro de até 1 ponto.

Palavras-chave: *Machine learning*, sistema de recomendação, inteligência artificial

1 INTRODUÇÃO

O mercado de streaming de vídeos movimenta bilhões de dólares por ano no mundo todo e um dos seus principais recursos para evitar a evasão de seus clientes é gerar boas recomendações de obras a serem assistidas. Tão mportante é esta tarefa, que a empresa líder neste mercado, *Netflix* ®, lançou em 2009 um desafio de 1 milhão de dólares, para desenvolvedores que conseguissem melhorar seu algoritmo de recomendação em 10% [1].

Nos últimos anos o mercado de streaming de vídeos ocidental vem se expandindo na categoria de animes. Inclusive uma das maiores plataformas do mundo de *streaming* de vídeo, é focada neste tipo de conteúdo, a *Crunchyroll*. Plataforma esta que possui mais de 20 milhões de usuários, tornando-a líder no mercado de *streaming* de animes no globo [2].

Outras plataformas de exibição de animes como *Hidive*, *Funimation, Anime planet,* entre outros, vem ganhando espaço no mercado, e a própria *Netflix* que no passado era focada em filmes, seriados e documentários, já vem se mostrando interessada nesta categoria.

De acordo com Ted Sarandos, *Chief Content Officer* (Chefe oficial de conteúdo) na *Netflix*, a mesma pretende expandir seu catálogo em 2018 com um

[1] Graduando em Engenharia da Computação. E-mail: douglas.waltricke.f@gmail.com
[2] Coordenador da Engenharia da Computação. E-mail: gustavo.lucca@satc.edu.br

investimento de 8 bilhões de dólares, destinados a 90 obras originais, sendo delas, 30 animes [3].

Atualmente as plataformas de *streaming* focadas em animes não contam com sistemas de recomendações de animes ou possuem sistemas de recomendação que utilizam abordagens focadas no anime e não no usuário, assim como a *Crunchyroll* que tende a recomendar obras parecidas com as assistidas em termos de características. Porém essa abordagem tem como objetivo apenas recomendar algo que o usuário possa gostar e não tem por finalidade garantir isso e nem classificar o quanto o mesmo irá gostar da obra.

Visando atender a uma necessidade de mercado, ou seja, um sistema de recomendação de animes com predição de nota, que não apenas diga se o usuário irá gostar ou não da obra, mas faça uma predição aproximada da nota que o usuário irá dar para a obra após assistir, este artigo decorre da fundamentação e implementação de tal sistema e aferição dos resultados. Este sistema é dividido em uma aplicação *backend* e uma aplicação *frontend.*

2 SISTEMA DE RECOMENDAÇÃO

Sistemas de recomendação são basicamente softwares, ferramentas ou técnicas para recuperar, filtrar dados e prover significantes recomendações para usuários ativos [4]. As recomendações são basicamente de dois tipos: personalizadas e as não personalizadas. As personalizadas são geradas para um indivíduo único, com base em dados do mesmo, já as não personalizadas são as geradas para um grupo inteiro. Quanto a forma de efetuar uma recomendação, ainda pode ser dividida em 3 principais categorias: as recomendações por filtros colaborativos, baseadas no item ou conteúdo e as abordagens híbridas [4].

Recomendação por filtro colaborativo é a mais bem-sucedida e difundida forma de serviço de recomendação personalizada, seu ponto chave é achar usuários ou itens similares através da pontuação dos usuários para os itens [5]. Todavia esta depende diretamente do conjunto de dados disponível para alimentar o algoritmo e de uma quantidade suficiente de dados do usuário, o que pode torná-la inviável para alguns cenários. Porém com um factual conjunto de dados e volume apropriado, as

recomendações tendem a ser assertivas, caso utilizem os métodos adequados de classificação, similaridade e recomendação.

Sua principal desvantagem é conhecida como *cold start* (em tradução livre, início devagar)*,* é a situação onde o usuário ainda não efetuou avaliação de itens no sistema, logo não há dados do usuário para alimentar o algoritmo.

Já as recomendações baseadas no item ou conteúdo, também conhecidas como recomendações por filtros cognitivos, efetuam recomendações de itens baseados em comparação entre conteúdo dos itens [4], logo não sofre o problema do *cold start.* Um exemplo de funcionamento dessa abordagem, é o caso em que o usuário assistiu um anime do gênero de drama e ação, com lançamento de 2017, o sistema tende a gerar recomendação de itens que possuem as mesmas características citadas: drama e ação, com lançamento em 2017. Porém essa abordagem possui o problema da redundância, tende a recomendar sempre itens parecidos. Nesse caso, seriam recomendados somente animes de características iguais ao assistido ou muito parecidas.

No exemplo citado é possível observar três características, duas de gênero e uma de ano de lançamento. Logo todas as recomendações com a abordagem baseada no item ou conteúdo, seriam de itens com essas características, tornando redundante as recomendações. Entretanto quando se trata de animes, o usuário pode desejar recomendações de animes diferentes em termos de características, mas que se enquadrem no seu gosto.

Também como concluído em [6] são necessárias em torno de 19 características de um anime para gerar uma boa recomendação, o que pode demandar muito tempo para registro do anime no catálogo. E muitas vezes essas características são impossíveis de se obter sem ter assistido e analisado a obra, tornando inviável essa abordagem de recomendação. Em contrapartida a recomendação baseada no item ou conteúdo, não sofre com o *cold start,* pois ela consegue gerar recomendação com pouco ou quase nenhuma informação do usuário sobre o quanto ele gostou da obra, pois a recomendação baseada no item ou conteúdo utiliza das características do item para a recomendação.

Já a recomendação hibrida, trata do agregado de diversas técnicas, que incluem filtros colaborativos e recomendações baseadas no conteúdo ou item, base

de conhecimento e outras técnicas [4]. Porém estas tendem a ser muito mais complexas de serem implementadas, pois fazem a junção de diversas abordagens. Dessa forma em determinados casos ela pode até mesmo ser inviável.

2.4 MÉTODOS DE CLASSIFICAÇÃO

Para poder fazer uma recomendação e predição, antes é necessário classificar os dados, por exemplo, caso deseje-se recomendar um anime para um usuário, baseado na similaridade de usuários, primeiro é preciso classificar quais são os usuários similares a ele em seu gosto [4].

A classificação faz parte da mineração de dados, por sua vez a mineração de dados é um termo usado para descrever a descoberta de conhecimento em bancos de dados, conhecida também como *KDD* (*Knowlodge-discovery databases*), essa mineração de dados também é um processo que usa técnicas de estatística, matemática, inteligência artificial e aprendizado de máquina para extrair e identificar informações úteis e conhecimentos relevantes de uma variedade de grandes conjuntos de dados [7]. Entre os métodos de classificação, destaca-se o *Pearson correlation Similarity*, que apresenta uma equação para obtenção de similaridade entre usuários. Esta considera conjuntos de itens de classificação de dois usuários e remove o valor médio de todas as classificações [8].

A classificação do usuário i e j é Iij (Iij = Ii∩Ij), e a equação de cálculo da semelhança de *Pearson* é dada por.

$$Sim(i,j) = \frac{\sum_{x \in Iij} (rix - ri')(rjx - ri')}{\sqrt{\sum_{x \in Iij} (rix - ri')^2} \sqrt{\sum_{x \in Iij} (rjx - rj')^2}} \quad (1)$$

Fonte: [8]

Sendo ri e rj valores de pontuação do usuário i e j para os itens respectivos representados por x. Já r é a média da pontuação do conjunto analisado.

De forma complementar ao método de *Pearson,* pode ser utilizado o *K-nearest neighbor* (*KNN,* em tradução livre, K vizinho mais próximo), este pode ser interpretado também como: quantidade K de vizinhos mais próximos. Trata-se de um

método para obter a classificação de objetos que se encontrem o mais próximo possível do objeto em comparação [7].

Após escolher a modelagem matemática que define os critérios para considerar um usuário similar a outro (através do método de *Pearson correlation similarity*), faz-se necessário também, obter uma certa quantidade de usuários similares, pois nem sempre o usuário mais parecido, terá pontuado todos os itens do conjunto de dados. Logo não seria possível recomendar algo que o usuário mais similar não tenha pontuado, entrando em um ciclo de erro para o sistema de recomendação.

Nesse caso onde o usuário mais similar encontrado não pontuou em todos os itens do conjunto de dados, é necessário o tratamento através do algoritmo *KNN* que tem por objetivo definir a vizinha em ordem de similaridade, ou seja, o vizinho mais próximo é o mais similar e assim sucessivamente. Pois em tal situação, pode-se usar os dados de outros usuários similares (vizinhos) caso o mais similar (vizinho mais próximo) não tenha pontuado em todos os itens do conjunto de dados.

Ainda sobre o algoritmo *KNN* existe o conceito de *Threshold user neighborhood* que é o limiar da vizinhança, geralmente denotado por um valor absoluto que define até quanto de similaridade um usuário tem com outro para ser considerado da mesma vizinhança.

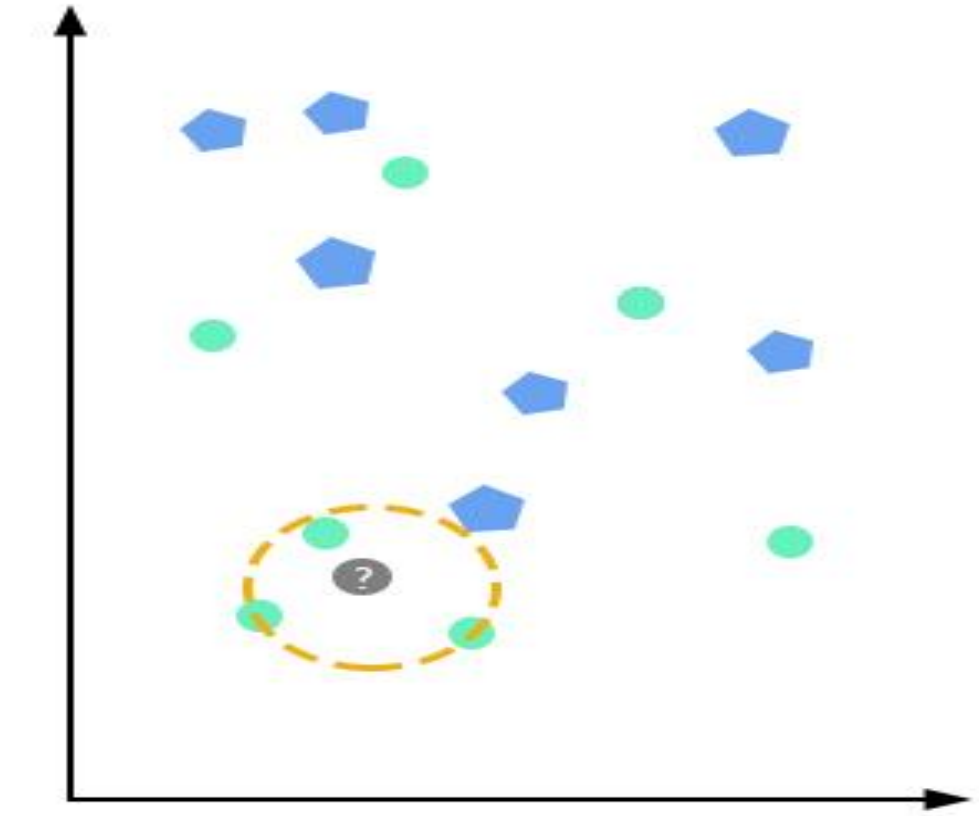
O conceito *Threshold user neighborhood* é de suma importância e varia perante a aplicação. Um exemplo para esse conceito seria dizer que usuários são parecidos quando a pontuação deles para o mesmo anime varia em 1 ponto, em uma escala de 0 a 10, isso pode ser considerado uma margem satisfatória de similaridade, logo o valor absoluto 1 poderia ser usado como *Threshold user neighborhood* em um sistema de recomendação de animes que utilizasse o *KNN*.

Todavia se o algoritmo de *KNN* estivesse analisando a similaridade de genoma, para fins de predição de potenciais doenças, 1 ponto, dependendo da escala, é um valor muito alto, gerando um erro de classificação, por colocar em grupos iguais, objetos muitos distintos, neste exemplo, os genomas.

Na figura 3, define-se o círculo pontilhado como vizinhança do item representado pela circunferência com interrogação, os itens que estão dentro ou

tocam a área de vizinhança, são tratados como vizinhos. O tamanho desse círculo é definido pelo anteriormente citado *Threshold user neighborhood.*

Figura 3 - Gráfico de exemplo do KNN



Fonte: O autor.

Ainda na figura 3, é possível verificar que o *KNN* faz uma classificação com uma abordagem diferente de outros métodos de classificação, que comumente traçam retas ou planos, para fazer a separação dos itens. Caso o *Threshold user neighborhood* fosse definido com um valor maior, o círculo pontilhado que destaca o ponto de interrogação do gráfico, seria maior, abrangendo mais itens como do mesmo grupo.

2.5 *DATASET*

Machine learning pode ser aplicada para uma larga gama de situações problema, desde detecção de fraudes fiscais, até recomendações, porém isso só pode ser feito devido à grande quantidade de dados fornecidas pelos *Datasets* [9].

Hays e Efros discutem em [11] o problema de preenchimento de buracos em uma fotografia, dando como exemplo uma foto que precisa ser tratada para remoção de uma parte em específico sem deixar rastros que algo foi removido da imagem, eles então concluíram que o algoritmo utilizado para resolução deste problema tinha um desempenho baixo quando usado uma coleção de 10 mil fotos para treinamento do algoritmo, porém utilizando uma coleção de dados de 2 milhões de fotos, foi possível atingir um limiar excelente de preenchimento desse mesmo buraco [10] [11]. Provando então a importância de um conjunto de dados com um volume satisfatório, para treinamento do algoritmo.

2.6 AVALIAÇÃO DAS RECOMENDAÇÕES

Dentre as metodologias mais utilizadas para avaliação de recomendação e predição, as que mais se destacam são *MAE* (*Mean Absolut Error*, em tradução livre, erro absoluto médio), *RMSE* (*Root Mean Squared Error,* em tradução livre, Raiz média do erro quadrático) e, sendo empregadas até mesmo em áreas como meteorologia, qualidade do ar [12]. A equação do *MAE* quantifica a média absoluta da magnitude de erros de um conjunto de predições, sem considerar sua direção. Ou seja, a diferença entre a predição e a real observação, onde todas as individuais diferenças têm peso igual [12]. *MAE* é dado pela equação (1).

$$MAE = \frac{1}{n}\sum_{j=1}^{n} |yj - \hat{}yj| \qquad (1)$$

Semelhante ao *MAE*, o RMSE também determina a média de erro, porém quadrática e na raiz, o que em geral implica em um valor maior para o erro, o que pode ser desejado pelo tipo de aplicação, a fim de penalizar erros, com um valor maior de relevância [12]. O *RMSE* é dado pela equação (2).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n} (yj - \hat{}yj)^2} \qquad (2)$$

3 METODOLOGIA

De modo a respeitar as boas práticas de programação, e possibilitar a criação de um sistema escalável e de fácil manutenção, o mesmo foi divido em duas aplicações, uma aplicação *backend* e uma aplicação *frontend.*

3.1 APLICAÇÃO *BACKEND*

A aplicação *backend* foi desenvolvida utilizando a linguagem JAVA, o banco de dados escolhido foi o *POSTGRES SQL* e como software de gerenciamento de banco de dados foi utilizado o *pgAdmin* 4. Também foi utilizado o framework *ORM Hibernate* para persistência dos dados e mapeamento objeto-relacional.

A aplicação *backend* está exposta como *API* para ser consumida e comunicar-se com a aplicação *frontend*, a comunicação segue a arquitetura *RESTful* e transportando dados no formato *JSON*.

Esta *stack* de tecnologias foi escolhida também, pela preocupação com o desempenho da aplicação, logo foi necessário selecionar tecnologias que são preparadas para trabalhar com volume alto de dados e que tenham maturidade no mercado, tal como comunidade, para facilitar o desenvolvimento com *plugins* e *libraries* da comunidade. Tecnologias como JAVA, *Hibernate* e *POSTGRES SQL* são utilizadas em grandes empresas, com volume alto de dados, logo se mostraram uma escolha interessante. No lado do *frontend* temos o framework *angularJS* também usado em grandes aplicações, inclusive em sistemas *google*.

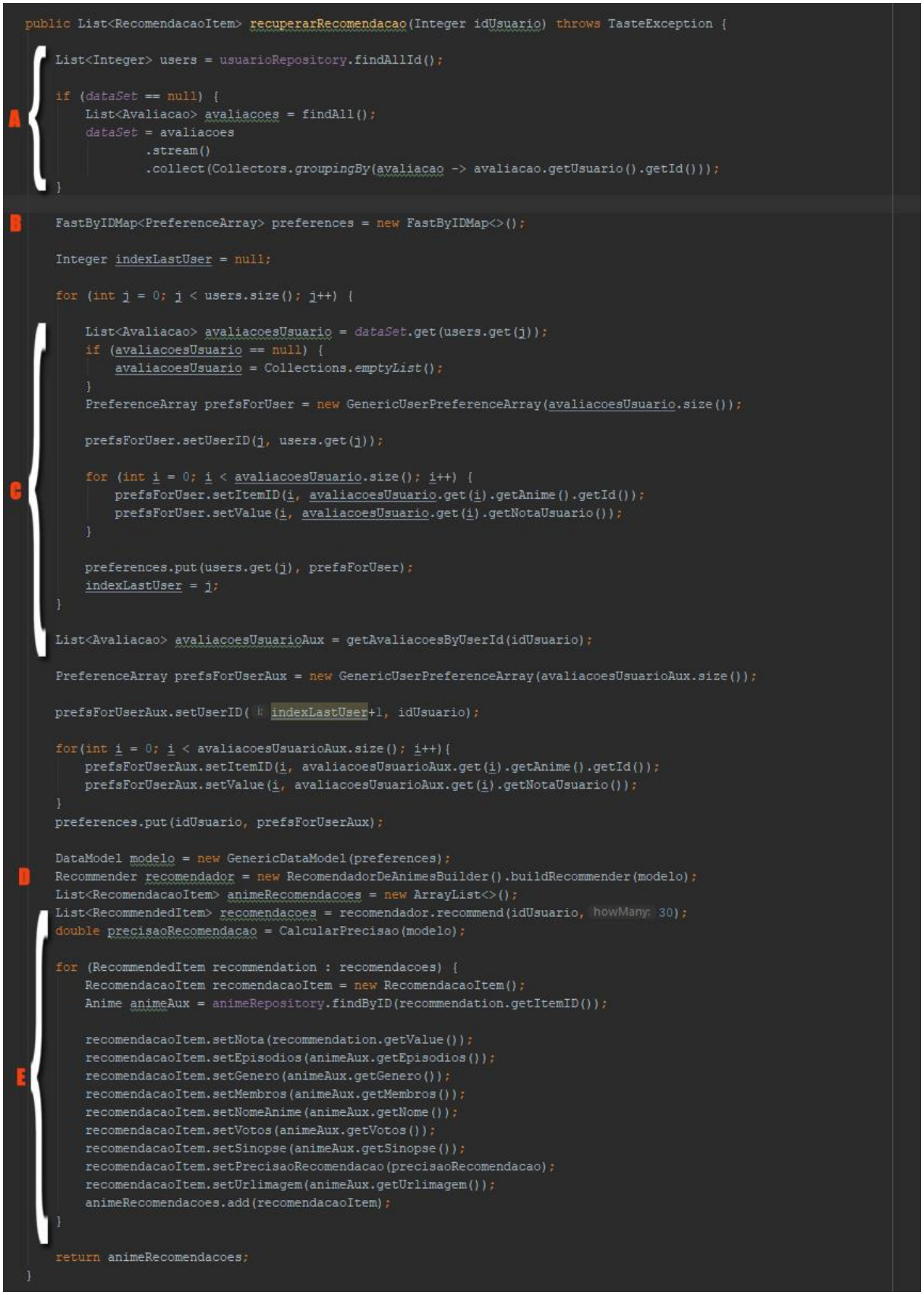
Figura 4 – A estrutura da aplicação *backend*

Fonte: O autor

3.1.2 GERANDO AS RECOMENDAÇÕES COM A NOTA ESTIMADA

Os conceitos de *user similarity* e *k-nearest neighbor* foram aplicados utilizando a biblioteca científica *Apache Mahout,* o qual possui implementação dos conceitos, através dos algoritmos citados na referência bibliográfica, sendo eles *Pearson correlation similarity e k-nearest neighborhood.*

Figura 8 – Método responsável pela lógica das recomendações.



O método responsável por gerar as recomendações, recebe o identificador do usuário (Id) que deseja a recomendação, em seguida, recupera o *dataset* previamente cadastrado no banco, mas que já está em nível de memória neste escopo, com mais de 128.000 avaliações (mais de 2.000 usuários) *[8A].* Após isso é criada uma estrutura de *array* otimizada para situações com volume alto de dados, como está, a estrutura é a *FastByIDMap [8B]*, disponibilizada também pela biblioteca científica *Apache Mahout.*

Em seguida são recuperadas as avaliações dos usuários para os animes que eles já avaliaram, e gerado um *array* de preferências desses usuários. Na sequência é criado o conjunto de treinamento com o *array* chamado *preferences,* este conjunto de treinamento será consumido pelos algoritmos de *Machine learning [8C]*.

Utilizando o padrão de projeto *builder* é gerado um objeto de recomendações, que de fato será responsável por gerar as recomendações, utilizando os algoritmos de *Machine learning [8D]*.

As recomendações são geradas apenas com a nota calculada e estimada pelos algoritmos de *Machine learning* e o nome do anime, logo é necessário buscar pelo Id do anime as demais informações para preencher devidamente o *array* de recomendações com mais detalhes do anime recomendado, como sinopse e imagem, de modo a entregar para o usuário final as informações pertinentes do título recomendado [8E].

No que tange aos algoritmos de *Machine learning*, é utilizado o algoritmo de *PearsonCorrelationSimilarity* (implementação do *Apache Mahout*) para definir os usuários similares, este algoritmo utiliza-se da correlação de *Pearson* para definir a similaridade entre os usuários, em seguida a vizinhança é definida, usando o conceito de *K-nearest neighbor* através do algoritmo *ThresholdUserNeighborhood* (implementação do *Apache Mahout*), definindo a vizinhança dos usuários. Nessa parte também é definido o *threshold.* Por fim é gerada a classe de recomendação, com similaridade, vizinhança e modelo definidos anteriormente.

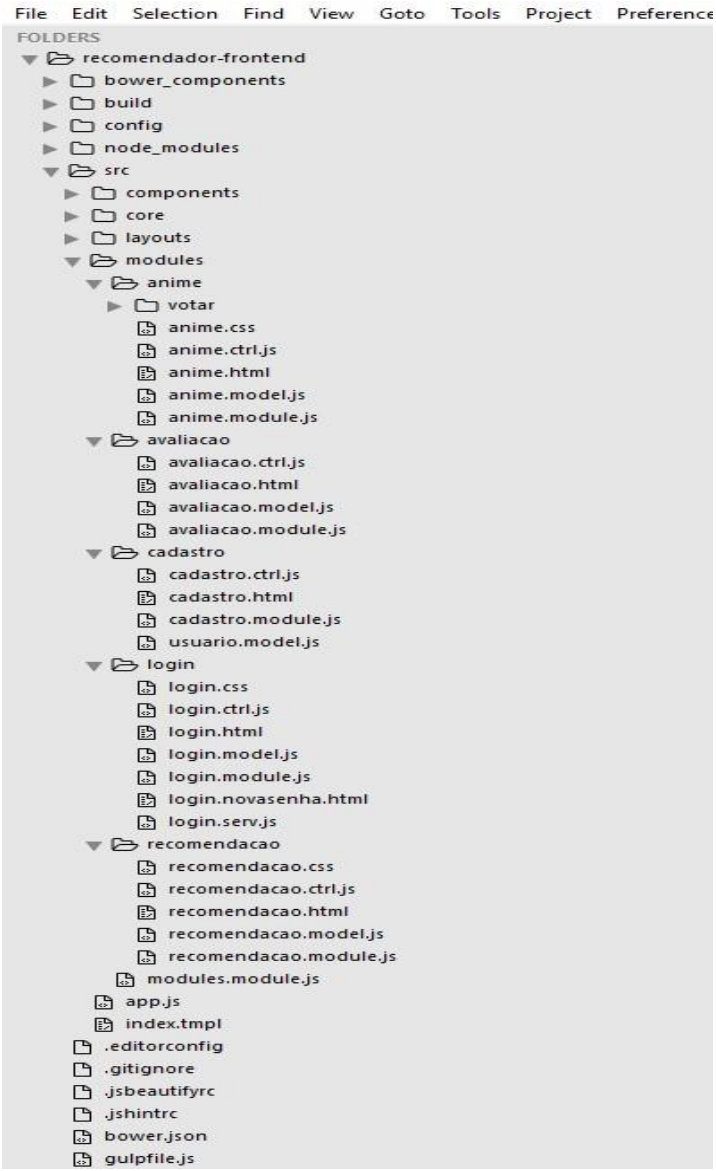
Na aferição das recomendações, optou-se pela utilização do *MAE* como método de avaliação das predições. Por não se julgar interessante ou necessária penalidades mais relevantes para erros, que é uma particularidade do *RMSE.*

3.2 APLICAÇÃO *FRONTEND*

A aplicação *frontend* foi desenvolvida utilizando a linguagem *ECMAScript* 6, porém como a maioria das funcionalidades do ECMAScript 6 ainda não são implementadas pelos navegadores, foi necessário a utilização do transpilador *Babel* que tem por finalidade transformar o código criado de *ECMAScript* 6 para *ECMAScript* 5, que é compatível com os navegadores em suas implementações atuais.

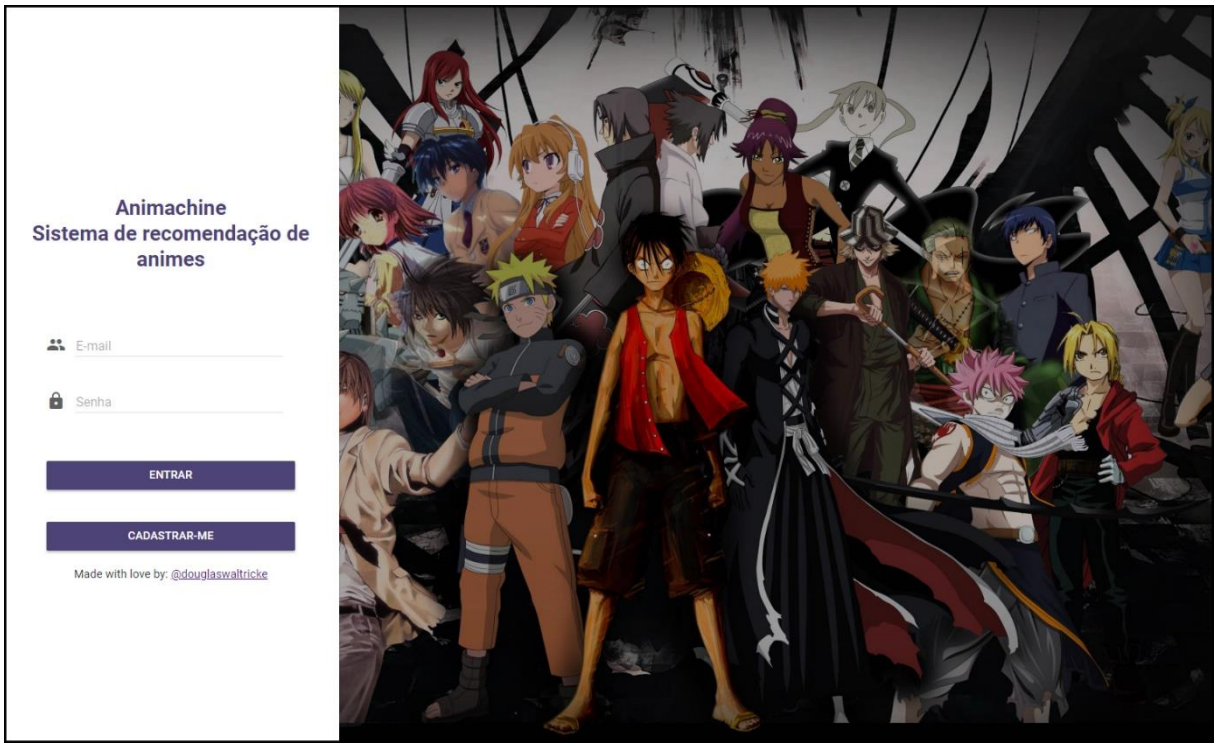
Como *frameworks* foram utilizados o AngularJS 1.x e AngularJS Material 1.1.x. Como gerenciador de dependências foi escolhida o *node package manager* (npm). Na parte de CSS da aplicação *frontend* foi utilizado o pré-processador *Postcss.* Com o surgimento de diversas “tarefas” para serem executadas de modo a rodar a aplicação *frontend,* como pré-processamento da folha de css, transpilação do *ECMAScript* 6, acionamento do servidor web entre outras, optou-se pela utilização de uma ferramenta de automatização de tarefas, a escolhida foi o *GULP.*

Figura 10 – Estrutura da aplicação *frontend*



Na figura 11 é possível visualizar a tela de *login* da interface *web* do *frontend*. O sistema de autenticação é baseado em token, a cada requisição feita ao *backend,* o token é transportado e é verificado a autenticidade do usuário para tráfego das informações.

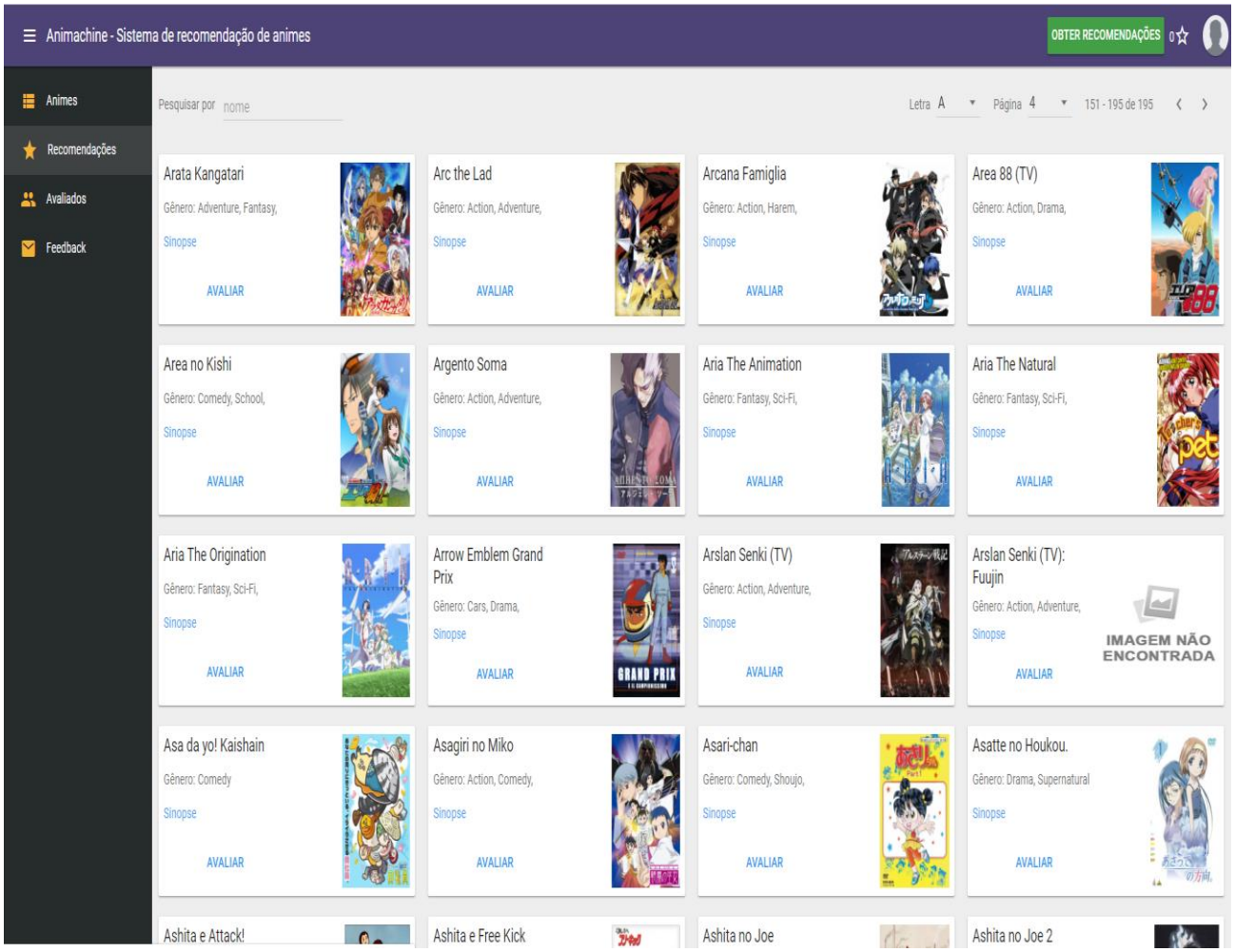
Figura 11 – Tela de *login*.



Fonte: O autor

Na figura 12 é exibido a listagem dos animes, por ser um volume grande de animes, mais de 3.800 títulos diferentes, foi aplicado conceitos de usabilidade, seguindo as heurísticas de Nielsen, conceitos como: consistência e padronização no layout, estética e design minimalista, *feedbacks* em casos de erros ou sucesso em operações, tudo isso para uma boa navegação do usuário final e desempenho do próprio sistema. Logo não são carregados todos os títulos de uma única vez, a listagem é feita através de um sistema de paginação, com filtros por letra, nome do anime e pagina.

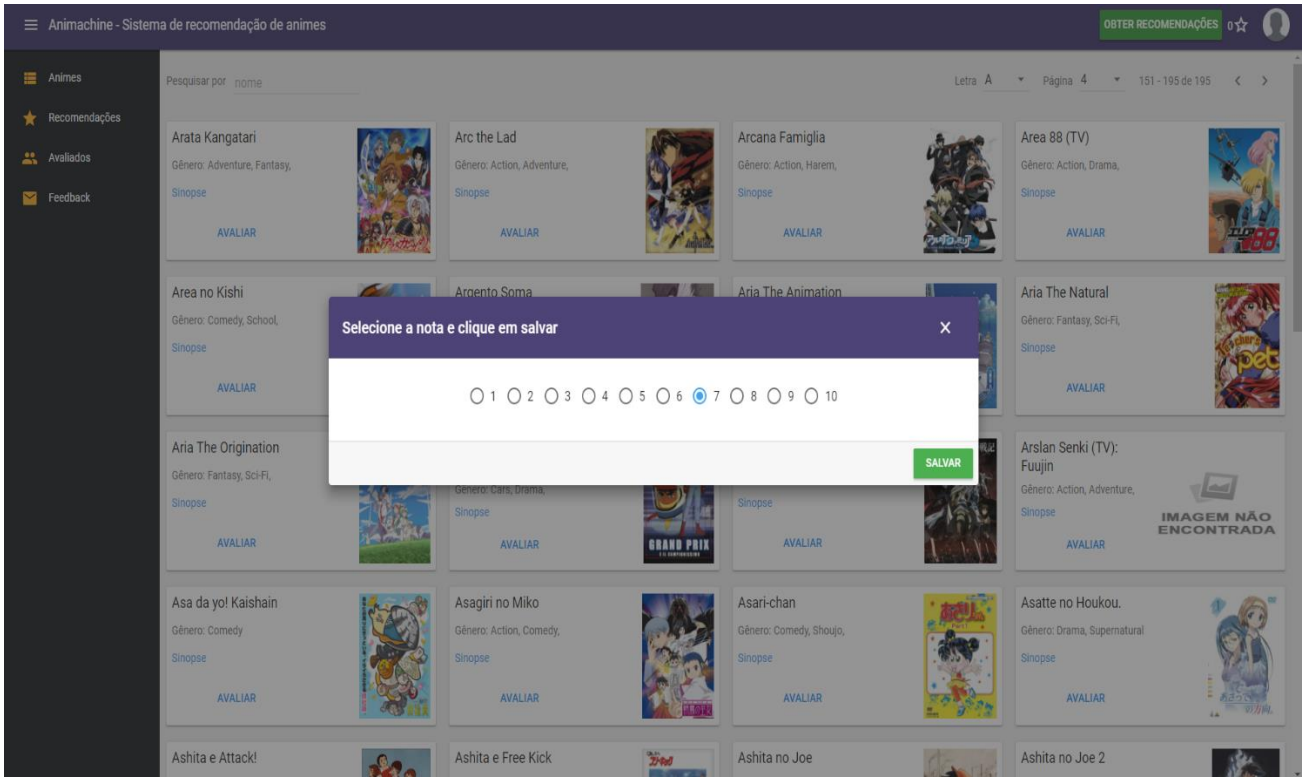
Figura 12 – Listagem dos animes.



Fonte: o autor

Na figura 13 é possível visualizar como o usuário irá avaliar um anime que ele já assistiu.

Figura 13 – Avaliação de um anime.



Fonte: O autor

4 RESULTADOS

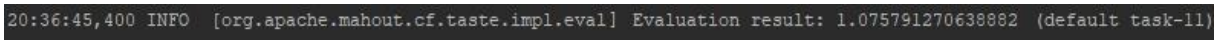
Utilizando um *dataset* de teste, com mais de 128.000 avaliações de pouco mais de 2.000 usuários diferentes, já foi possível obter uma margem de erro de aproximadamente 1,075 pontos, utilizando o método de aferição *MAE (*implementação do *Apache Mahout),* este método é apresentado na figura 14. Já na figura 15 é possível visualizar uma avaliação de precisão, da própria recomendação gerada.

Figura 14 – Método de aferição da recomendação.

```
public void CalcularPrecisao(DataModel modelo) throws TasteException {
    RecommenderEvaluator precisao = new MeanAbsoluteErrorRecommenderEvaluator();
    RecommenderBuilder recomendadorBuilder = new RecomendadorDeAnimesBuilder();
    double precisaoValor = precisao.evaluate(recomendadorBuilder, dataModelBuilder: null, modelo, trainingPercentage: 0.5, evaluationPercentage: 0.5);
    System.out.println(precisaoValor);
}
```

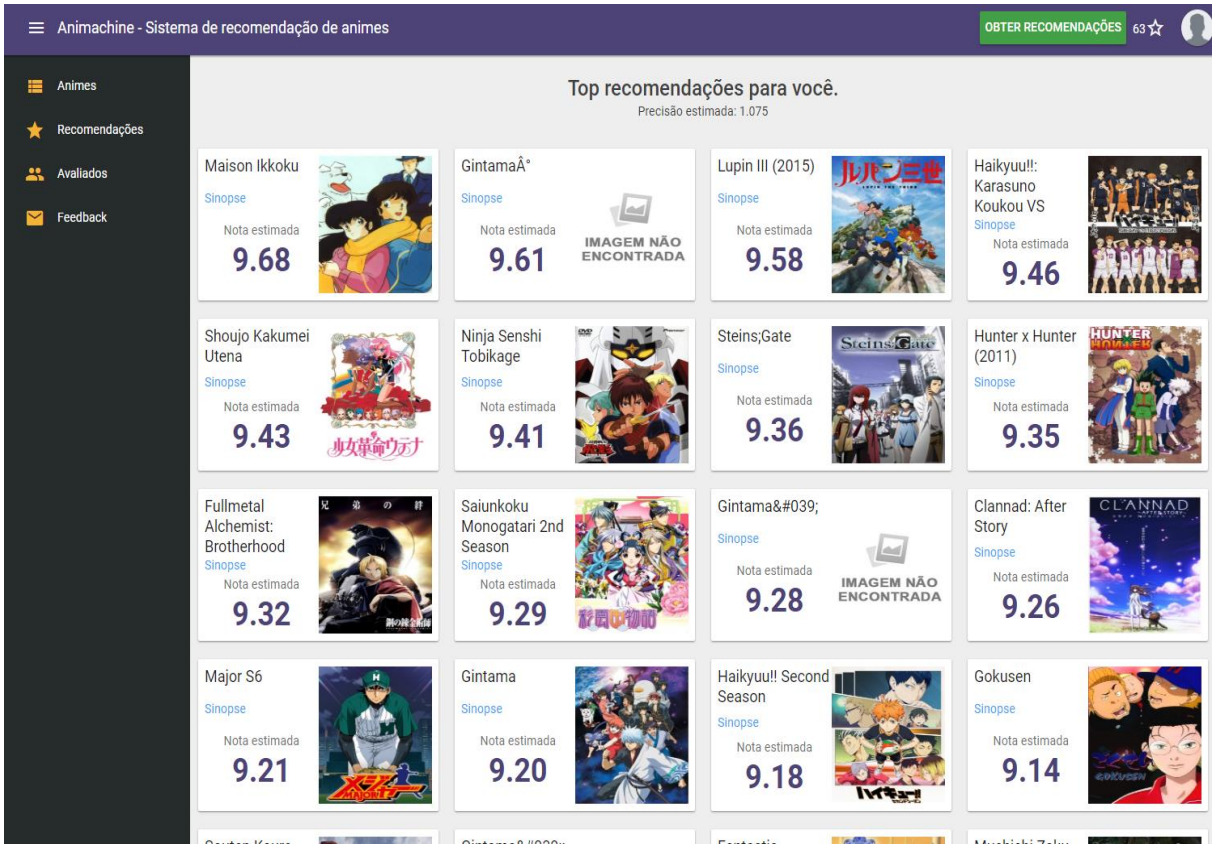
Fonte: O autor

Figura 15 – Precisão da recomendação.



Fonte: O autor

Figura 16 – Tela do sistema web com as recomendações e a nota estimada.

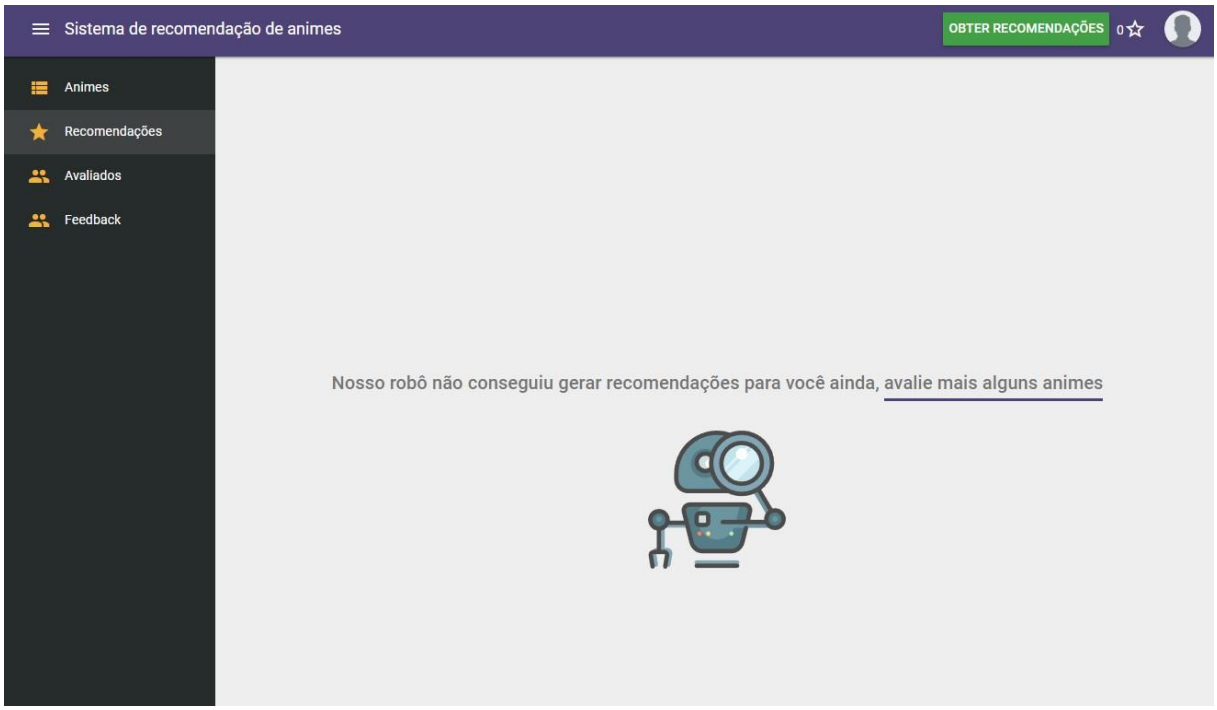


Fonte: O autor

Em cenários cujo *dataset* de treinamento foi limitado a 20.000 avaliações (pouco mais de 300 usuários), foi possível aferir uma queda drástica na precisão, chegando a 3 pontos na margem de erro.

Caso o sistema fosse incapaz de gerar as recomendações por falta de avaliações suficientes do usuário, um *feedback* seria emitido no lugar das recomendações, pedindo para o usuário efetuar mais algumas avaliações, como apresentado na figura 17.

Figura 17 – *Feedback*, na situação o qual não é possível gerar a recomendação.



Fonte: O autor

4 CONSIDERAÇÕES FINAIS

Foi possível obter o sistema web de recomendações de animes inicialmente proposto. Tal sistema foi desenvolvido em duas partes principais, a aplicação *backend* e *frontend*. No que tange aplicação *backend* a maior preocupação era o desempenho e escalabilidade, e nos testes foi possível aferir que ambos foram satisfatórios. Na aplicação *frontend* além de desempenho, existia a preocupação da interface ser intuitiva para o usuário interagir e receber sua recomendação de forma simples e prática, e com próprio *feedback* dos usuários, foi possível concluir que este objetivo foi concluído.

Também foi atingida uma precisão satisfatória nas recomendações, como apresentado nos resultados, com margem de erro de até 1,075.

REFERÊNCIAS

[1] Netflix, “Netflix Prize,” *NetflixPrize*, 2009. [Online]. Available: https://www.netflixprize.com/.

[2] Cruchyroll, “Largest Anime Streaming Service,” 2017. [Online]. Available: http://www.crunchyroll.com/anime-press-release/2017/02/09-1/largest-anime-streaming-service-crunchyroll-surpasses-one-million-paid-subscribers.

[3] N. Y. Times, “Netflix Says It Will Spend Up to $8 Billion on Content Next Year,” 2017. [Online]. Available: https://www.nytimes.com/2017/10/16/business/media/netflix-earnings.html.

[4] K. Shah, A. Salunke, S. Dongare, and K. Antala, “Recommender Systems : An overview of different approaches to recommendations,” 2017.

[5] X. Zang *et al.*, “A new weighted similarity method based on neighborhood user contributions for collaborative filtering,” *Proc. - 2016 IEEE 1st Int. Conf. Data Sci. Cyberspace, DSC 2016*, pp. 376–381, 2017.

[6] H. Cho, M. L. Schmalz, S. A. Keating, and J. H. Lee, “Information Needs for Anime Recommendation: Analyzing Anime Users’ Online Forum Queries,” *Proc. ACM/IEEE Jt. Conf. Digit. Libr.*, pp. 3–4, 2017.

[7] A. Danades, D. Pratama, D. Anggraini, and D. Anggriani, “Comparison of Accuracy Level K-Nearest Neighbor Algorithm and Support Vector Machine Algorithm in Classification Water Quality Status,” pp. 137–141, 2016.

[8] B. Guo, S. Xu, D. Liu, L. Niu, F. Tan, and Y. Zhang, “Collaborative Filtering Recommendation Model with User Similarity Filling,” pp. 1151–1154, 2017.

[9] J. W. Richards, M. Fetherolf, F. O. By, and B. Cronin, “Henrik Brink.”

[10] N. Peter and S. Russell, *Inteligência Artificial*. 2013.

[11] J. Hays and A. A. Efros, “Scene completion using millions of photographs,” *Commun. ACM*, vol. 51, no. 10, p. 87, 2008.

[12] T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)? -Arguments against avoiding RMSE in the literature,” *Geosci. Model Dev.*, vol. 7, no. 3, pp. 1247–1250, 2014.

[13] Smola, A. (2018). Introduction to Machine Learning. 1st ed. Cambridge: cambridge university press,p.30.
[14] Apache. Mahout [ONLINE] avaliable: https://mahout.apache.org/docs/0.13.0/api/docs/mahout-mr/

ABSTRACT

This article is based on the development of a web application to generate assertive recommendations of anime on a specific level using Machine learning, a subarea of artificial intelligence, specialized in the use of a set of data to make classifications and predictions. Using the concepts of Machine learning with algorithms of user similiraty and neighborhood, it was possible to obtain a high precision, in agreeing the note that the user will give to the anime suggested after watching them.

Key-words: *Machine Learning*; Recommendation system; Artificial intelligence.