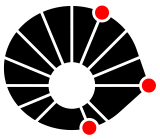
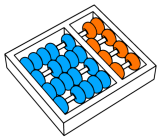




Revisão de técnicas de Ciência de Dados para Sistemas de Recomendação

Raissa C. Correia

Relatório Técnico - IC-PFG-19-01
Projeto Final de Graduação
2019 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

The contents of this report are the sole responsibility of the authors.
O conteúdo deste relatório é de única responsabilidade dos autores.

Revisão de técnicas de Ciência de Dados para Sistemas de Recomendação

Raissa C. Correia*

Resumo

Esse projeto tem como propósito comparar algumas das principais abordagens utilizadas atualmente em sistemas de recomendação, e suas respectivas virtudes e pontos fracos. Para isso foram feito alguns códigos em Python utilizando o *Numpy*,o *SciPy* e o *Surprise*, bibliotecas muito consagradas para sistemas de recomendação no segmento de ciencia de dados muitas presentes no *SciKit-Learn* e no Pandas para testar as principais abordages: *Content-Based*, *Colaborative Based* e Fatoração Matricial. Foi possível concluir que as técnicas de fatoração matricial são mais robustas por oferecer um resultado mais acurado em avaliações mais extremas como 5 e 1 estrela. No entando foi possível notar que variações de algoritmos como K-Nearest Neighbors foi capaz de obter bons resultados com uma implementação mais simples, especialmente se os dados são pré-analisados, com uma clusterização, com uma redução de dimensionalidade entre outras técnicas. O mais importante é concluir que algoritmo algum é um solução definitiva para o problema, tudo depende do dataset, do contexto da plataforma que a utiliza, de como os dados foram preparados e se as interações implícitas são levadas em conta e quais levar em conta.

1 Introdução

Com o advento das redes sociais, do e-commerce, e de plataformas de conteúdo como *Netflix*, *Spotify* e *Youtube*, surgiu a necessidade de algoritmos capazes de recomendar o conteúdo ao usuário para facilitar seu consumo e manter o usuário interessado na plataforma. Nesse mesmo período ocorreram grandes avanços no aprendizado de máquina, e na ciência de dados aplicada, graças ao aumento de oferta de poder computacional e a disponibilidade de dados abundante.

Tais sistemas tem um valor mercadológico significativo uma vez que recomendações são fundamentais para guiar os diferentes perfis de usuário no imenso catálogo de possibilidades com relação ao que assistir ou comprar. Entretanto, tal recomendação não podendo ser simplista, pois isso pode levar o usuário a se cansar, mas não pode ser uma recomendação complexa demais a ponto de não ser viável computacionalmente.

Como os contextos de e-commerce e de plataformas de conteúdo são muito distintos, e usuários recém-cadastrados são muito distintos de usuários com histórico de consumo, há

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

diversas abordagens que são adequadas, uma vez que o contexto da aplicação e do usuário são fundamentais.

Dado este cenário, este trabalho tem como objetivo avaliar os principais problemas encontrados nos sistemas de recomendação destacando como as abordagens evoluiram com o tempo. Para permitir esta avaliação, conduziremos experimenntos com as técnicas mais relevantes avaliando seus respectivos desempenhos de acordo com os principais critérios definidos na literatura sobre o dataset *MovieLens*. Este dataset é o mais amplamente utilizado para validar sistemas de recomendação devido a sua qualidade e simplicidade, o que nos permitirá conduzir uma discussão atualizada sobre tais abordagens, seus prós e contras e as implicações disso no comportamento das recomendações nas plataformas atuais.

1.1 Objetivos

Esse projeto tem como objetivo implemetar, analisar e comparar as mais populares abordagens para sistemas de recomendação, e concluir como isso molda o funcionamento do e-commerce e das plataformas de conteúdo atualmente.

1.2 Organização do Texto

As próximas seções descrevem o problema e detalham a metodologia e resultados obtidos e o conteúdo deste trabalho está distribuído da seguinte forma:

- No capítulo 2 trataremos dos fundamentos teóricos necessários para tais sistemas, entender seus principais problemas, e as abordagens mais famosas.
- No capítulo 3 trataremos do problema da alta dimensionalidade dos dados, e como as técnicas envolvendo fatoração matricial properaram justamente por atacar tal problema
- No capítulo 4 descrevemos os experimentos realizados e os resultados obtidos
- O capítulo 5 descreve as conclusões obtidas com o estudo.

2 Fundamentos em Sistemas de Recomendação

2.1 Principais Problemas dos Sistemas de Recomendação

Um sistema de recomendação representa um conjunto de técnicas utilizados para fornecer sugestões de itens a serem recomendados para um usuário. Segundo Çano et. al. [1], os principais problemas que os sistemas de recomendação encontram são:

- **Dados Esparsos:** Os dados são sempre esparsos uma vez que a maioria dos usuários consome apenas uma mínima fração do conteúdo total numa plataforma. Isso é especialmente mais difícil em e-commerce pois existem centenas de categorias de produtos e a maioria dos usuários não tem histórico de compras em diversas categorias. Neste sentido, surge o problema de se recomendar mais produtos do mesmo tipo que o

usuário acabou de comprar. Já em plataformas de consumo de conteúdo como na netflix é um pouco mais simples traçar perfis de usuário e coeficientes de proximidade entre esses usuários e as diversas categorias. É claro, isso traz um custo computacional bem maior, mas veremos abordagens para combater este problema.

- **Escalabilidade:** A escalabilidade é um dos problemas mais graves pois boa parte dos algoritmos que veremos tem uma complexidade proporcional ao produto do número de usuários, por número de itens, por número de features. Entre as diversas formas de resolver a questão muitas delas envolvem reduzir o universo de itens, e analisar as features mais isoladamente possível.

- **Itens Sinônimos:** Há itens que são similares demais entre si, e que recomendar um sinônimo após o usuário ter consumido um item, leva a rejeição pelo tédio no caso de um conteúdo, ou rejeição por já ter algo quase igual no caso de um e-commerce. Então é preciso conseguir fazer uma analise de dados de tal forma que distingui o similar do sinônimo.

- **Diversidade:** Esse problema é o oposto do itens sinônimos, as pessoas gostam de itens similares, mas até certo ponto, pois há o problema de criar tédio e repetitividade, portanto deve-se recomendar itens mais "distantes", mas quais? E com que frequência ou taxa?

- **O problema da ovelha cinza:** Quando se tenta criar clusters de usuários, existem sempre usuários que não se encaixam muito bem em nenhum cluster, e podem se encaixar um pouco em diversos grupos, e tem um histórico de escolhas incomum ou esparso entre as diversas categorias, sendo quase impossível traçar um perfil previsível num modelo simples. Na realidade modelos simples com poucas features criam muitas ovelhas cinzas, e modelos muitos complexos fazem com que usuários com perfis de consumo mais previsíveis caiam em overfitting.

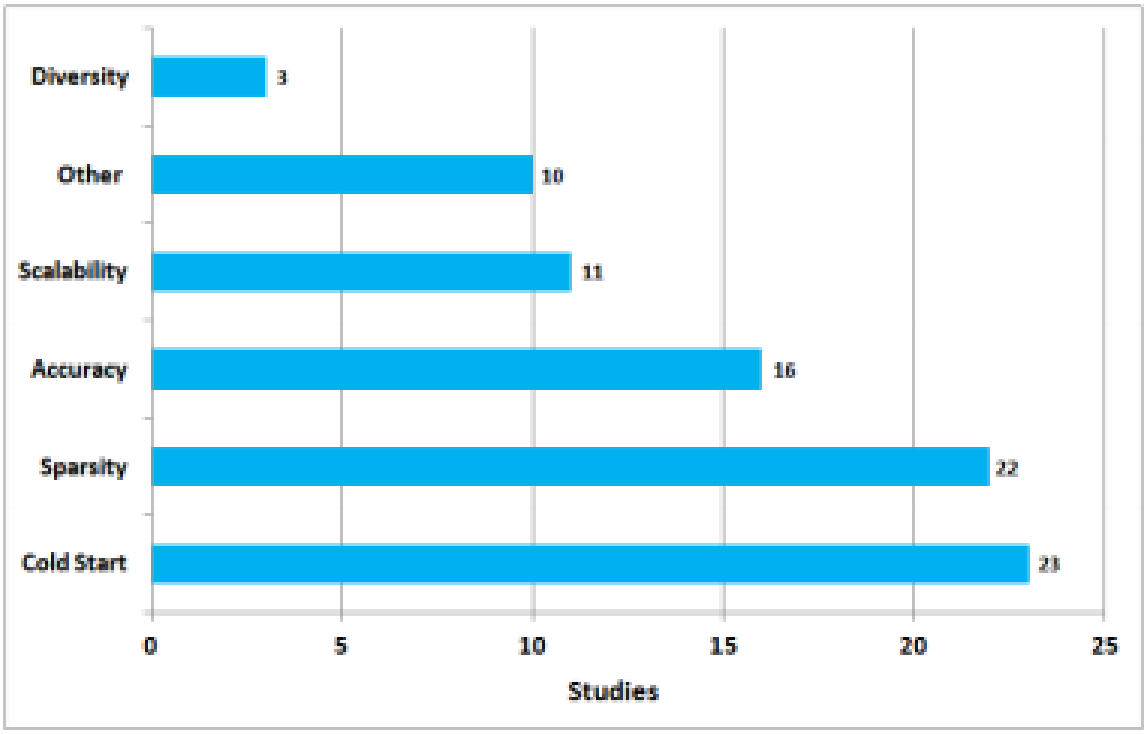
- **Cold Start:** Quando o usuário ainda não tem nenhum histórico na plataforma, realizar as recomendações se torna uma tarefa praticamente impossível. Existe a ajuda do contexto, e do perfil do usuário como país, gênero, idade, porém esse problema é um dos que mais se busca combater pois se trata justamente de dar uma impressão de que o sistema é bom e eficaz para o usuário que acabou de chegar. Sendo este com uma importância comercial imensa.

Podemos observar na figura 1 abaixo, a relação de quanto estudos abordam cada um dos problemas mais comuns.

2.2 Principais Abordagens de Filtragem

- **Collaborative Filtering:** Segundo os surveys realizados no assunto nos últimos 10 anos: [2] [3] [4]. Existem duas grandes abordagens de filtragem colaborativa, a model-based e a memory-based. A memory-based é a mais simples, pois trata-se de criar uma "fórmula de proximidade" entre itens ou entre usuários, e para realizar a

Figura 1: Problemas mais abordados nos estudos



predição usa-se pesos baseados nas avaliações. Os métodos memory-based são de fácil implementação e baixo custo computacional pois não se preocupa com a descrição do item, mas possuem sérias limitações com o "cold start"e dados esparsos. Em outra seção veremos uma lista de formas de correlação, como a de Pearson, a de semelhança de cossenos dentre outros. A model-based se trata de um conjunto amplo de diferentes métodos de aprendizado de máquina e mineração de dados para se chegar numa predição. É usado com frequência uma combinação dos métodos como redução de dimensionalidade, para evitar a "maldição da alta dimensionalidade", além de evitar os dados esparsos. Com isso essa abordagem, tem um trade-off mais equilibrado entre performance e escalabilidade, ao custo de construir o modelo.

- **Content-Based Filtering:** É baseado na ideia de que temos itens muito bem descritos, e usuários vagamente descritos, um cenário muito comum no e-commerce, boa parte da informação que se sabe do usuário é na realidade de contexto, mas a medida que o usuário cria um histórico de preferência a similaridade disso com os itens é usada na recomendação. Atualmente este método de filtragem é usado em conjunto com a colaborativa, criando sistemas híbridos de recomendação.

2.3 Análise de Correlação entre Usuários e Itens

Normalmente as correlações retornam um valor real entre -1 e +1, mas algumas retornam um valor entre 0 e 1, o que pode ser util se o algoritmo ou o contexto não se aplica para correlações negativas. As mais famosas são:

2.3.1 Similaridade de Cossenos

Se trata do "inner product" entre 2 vetores não nulos, de forma que se o ângulo entre eles for 0^o temos +1 de correlação, se forem perpendiculares a correlação é 0, e se forem em direções opostas, em 180^o a correlação é -1, independente da magnitude. Uma vantagem dessa técnica é sua simplicidade, especialmente com vetores esparsos e não ter problemas com altas dimensões, pois o princípio se mantém.

$$simil = \cos\theta = \frac{A.B}{(||A||.||B||)} \tag{1}$$

A similaridade de cossenos pode ser suavizada com a seguinte equação,mas no contexto desse projeto não iremos aplicá-la:

$$softcosine(a,b) = \frac{\sum_{i,j}^{n} s_{ij} a_i b_j}{\sqrt{\sum_{i,j}^{n} s_{ij} a_i a_j}\sqrt{\sum_{i,j}^{n} s_{ij} b_i b_j}} \tag{2}$$

Nesta é possível levar em conta a similaridade entre as features, o que pode ser muito importante em alguns sistemas.

2.3.2 Distância Euclidiana

Util para avaliar a soma da distância decompondo entre cada uma das dimensões, porém justamente por isso ela sobre muito com a maldição da dimensionalidade, pois um item muito próximo em uma das dimensões pode-se beneficiar disso. Onde N é o numero de dimensões do espaço e x e y são as coordenadas que definem cada um dos vetores característicos dos itens, temos a distância definida da seguinte forma:

$$euclideandist = \sqrt{\sum_{i}^{N}(y_i - x_i)^2} \tag{3}$$

2.3.3 Correlação de Pearson

Diz o quanto os itens são correlacionados através da distância da média de sua própria distribuição.

$$Pearson(u,v) = \frac{1 - (u - u_m)(v - v_m)}{||u||.||v||} \tag{4}$$

2.3.4 Mean squared difference

Uma forma muito comum de calcular divergência entre avaliações de usuários para assim agrupá-los em uma categoria ou não.

$$MSD(u,v) = \frac{1}{|I_{uv}|} \sum_{i \in |I_{uv}|} (r_{ui} - r_{vi})^2 \tag{5}$$

Iuv é o número de itens avaliado por ambos os usuários e a similaridade é dado por:

$$\frac{1}{1 + MSD(u,v)} \tag{6}$$

3 Fatoração Matricial

Uma das formas mais eficientes de se descobrir as características latentes é através das técnicas de fatoração matricial, a qual ganhou muita notoriedade após o lançamento do Netflix Prize que durou de 2006 a 2009. Neste contexto as soluções mais bem sucedidas envolveram versões de fatoração matricial. Uma das grandes vantagens de usar tais técnicas é sua fácil escalabilidade, pois a adição de novos itens não envolve recalcular tudo novamente. Adicionar novas features, usuários ou itens é bastante simples, inclusive em termos de custo computacional, e se torna mais fácil recomendar com base em preferências implícitas do usuário, por conta da sua capacidade de serem representadas, o que é especialmente importante nos casos onde as avaliações explícitas sejam muito esparsas.

O que todas as técnicas têm em comum é tornar a relação usuário-item fruto de uma combinação de vetores de características, através do padrão de avaliação dos itens.

3.1 Redução de Dimensionalidade

Quando mencionamos o paradigma model-based na filtragem colaborativa, algo que é fundamental no processamento de dados para tornar a recomendação computacionalmente possível em larga escala é a redução de dimensionalidade, além de evitar a "maldição da dimensionalidade".

Os maiores problemas que enfrentamos com espaço de features de dimensões muito altas, é principalmente a explosão de combinações possíveis e por isso demandar conjuntos de treinamento muito grandes. Além de matrizes com muitas dimensões serem muito esparsas, isso deve ser evitado ao máximo. Outro problema que surge é o cálculo de distância, numa busca pelos vizinhos mais próximos, ou uma clusterização. A tendência é que se escolha alguns que são muito próximos em algumas dimensões, em prol de outros que são mais ou menos próximos na maioria das dimensões, se aplicarmos a fórmula de euclides em 2 pontos num espaço de 10 dimensões, por exemplo, o ponto [3,0,0,0,0,0,0,0,0,0] é mais próximo da origem que o ponto [1,1,1,1,1,1,1,1,1,1], mas a depender do contexto uma proximidade maior em todas as dimensões pode ser mais ou menos valiosa que ser mais distante em uma unica dimensão, mas o que importa é que a informação nesse caso é perdida.

Os métodos de redução de dimensionalidade são divididos em extração e seleção de features. A seleção é feita usando-se de métodos de classificação e regressão para se reduzir um espaço de mais dimensões em menos, e assim pegando um subconjunto representativo, como é o caso do PCA que utilizamos para verificar se o espaço de vetores características de usuários pode ser reduzido para menos que 18 dimensões, sendo cada dimensão um gênero do dataset Movielens.

A extração de features consiste em utilizar tensores para uma transformação multilinear de um espaço de features de maior dimensão num de menor dimensão. Uma das abordagens mais comuns em sistemas de recomendação é a fatoração matricial, pois como estamos lidando com dados esparsos, a matriz de correlação entre usuários e itens, pode ser transformada num produto matricial de 2 matrizes bem menores, uma de tamanho usuários x features e outra de tamanho itens x features que serão menos esparsas.

Para fins de tornar esse processo ainda mais computacionalmente prático, existe uma versão da fatoração matricial amplamente utilizada nos sistemas de recomendação que é o SVD, single value decomposition, que falaremos sobre na próxima seção.

3.2 Descrição do Modelo de Fatoração Matricial

A ideia é mapear tanto os usuários como os itens em fatores latentes de dimensionalidade f (número de features), de tal forma que as iterações usuário-item são modeladas como inner products neste espaço. Cada item i é associado a um vetor qi, e cada usuário a um vetor pu, ambos de tamanho f. Dessa forma o rating estimado do usuário u pelo item i se dá pelo dot product:

$$\hat{r}_{ui} = q_i^T . p_u \tag{7}$$

Essa equação é utilizada no NMF(Non-Negative Matrix Factorization) da biblioteca Surprise.

O maior desafio é computar o mapeamento de cada item e usuário para cada vetor. Esse modelo é muito próximo do SVD(Single Value Decomposition), uma técnica amplamente conhecida para identificar fatores semânticos latentes num conjunto de informação. No entanto, como os dados são muito esparsos na matriz usuário-item, aplicar o SVD pode levar ao overfitting, além do SVD original ser indefinido para matrizes incompletas.

Para aprender os vetores Pu e Qi o sistema busca minimizar o regularized squared error do conjunto de avaliações conhecidos. Onde k é o conjunto de treinamento:

$$min_{p,q} = \sum_{u,i \in k} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{8}$$

Esta minimização é justamente a implementação do SVD na biblioteca surprise como pode-se observar na referência [10].

3.3 Algoritmos de Aprendizagem

Os 2 grandes algoritmos de aprendizagem são: *Stochastic Gradient Descent* e *Alternating Least Squares*. Simon Funk popularizou a solução do *Stochastic Gradient Descent*

Tanto no caso do SVD quanto do NMF ambas as minimizações são calculadas por sucessivas épocas de iteração usando o SGD(stochastic gradient descent), pode ser usado também o ALS(alternating least squares), porém este é mais usado em datasets, em conjuntos maiores que a relação entre os vetores em questão.

Para cada caso de treinamento o sistema prevê a matriz Rui e computa a predição associada o erro. Então modifica os parâmetros com uma magnitude proporcional na direção oposta ao gradiente.

$$e_{ui} = r_{ui} - q_i^T p_u \tag{9}$$

$$q_i \leftarrow q_i + \gamma(e_{ui}.p_u - \lambda.q_i) \tag{10}$$

$$p_u \leftarrow p_u + \gamma(e_{ui}.q_i - \lambda.p_u) \tag{11}$$

Apesar dessa técnica sem em geral mais rápida e eficaz que o ALS. O ALS se sobressai em 2 situações além da citada acima: em paralelismo, pois cada Qi e cada pu pode ser calculado de forma totalmente paralela, e a segunda situação é em sistemas centrados em dados implícitos

3.4 Inclusão do Bias e mudanças no tempo

Uma das vantagens de fatoração matricial em relação a filtragem colaborativa. Uma das melhores formas de entender o bias num sistemas de recomendação é, temos 3 componentes:

$b_{ui} = \mu + b_i + b_u$

O primeiro é o Bias do item: se os filmes têm uma avaliação média de 3.7 e o filme em questão por ser muito famoso como Avangers tem uma avaliação média de 4.2 ele tem um bias +0,5. O bu é o viés do usuário na hora de avaliar, se em média as avaliações são de 3.7 mas João é um usuário mais crítico e sua avaliação é em média 3.1, seu Bu é de -0.6. O seria a avaliação prevista em condições normais, portanto a nota esperada para o filme seria.

$b_{ui} = 3.7 + 0,5 + (-0,6) = 3.6$

$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$

Portanto a recomendação se vale de 4 fatores: (média global), bi(viés do item), bu(viés do usuário) e qiTpu(interação usuário-item). E assim o sistema busca pelos quadrados mínimos dessa função. Essa é a versão com bias, dos algoritmos de fatoração matricial da biblioteca surprise que podem ser encontrados na mesma referência [10] que justamente por levar isso em conta possui resultados significamente melhores.

$$min_{p,q,b} = \sum_{u,i \in k} (r_{ui} - \mu - b_i - b_u - p_u^T q_i)^2 + \lambda(||q_i||^2 + ||p_u||^2 + b_i^2 + b_u^2) \tag{12}$$

É importante mencionar que também existe uma dinâmica temporal em todos estes 4 fatores justamente por usuários mudarem de preferências e itens entrarem mais em alta em certas épocas, e existem diversas técnicas para captar a versão temporal de tais parâmetros.

$$\hat{r}_{ui}(t) = \mu + b_i(t) + b_u(t) + q_i^T(t)p_u(t) \tag{13}$$

4 Metodologia

4.1 Ambiente de Desenvolvimento

4.1.1 Uso do Deep Learning Questionado

O Python 3 nos ultimos anos passou a ser de longe a linguagem mais amplamente usada para o desenvolvimento, não apenas do aprendizado de máquina, e da construção das redes neurais, mas também da parte de análise, filtragem, clustrização dos dados necessárias para a recomendação.

A partir do Python foram desenvolvidos umas série de frameworks, com funções para facilitar o treinamento, a construção das camadas assim como a função de ativação dos neurônios. O *Tensorflow* passou a ser o mais utilizado, chegando a 50% das pesquisa[5], o *PyTorch* tambem é muito usado, porém recentemente tem sido menos utilizado. Tais estatística de frameworks de Deep Learning mais comuns podemos observar na figura 2.

Como o intuito neste artigo é justamente avaliar os métodos do recomendador, faremos o código da forma mais tradicional e ortodoxa, usando as ferramentas mais comuns. No entando após um tempo de desenvolvimento deste projeto, vários artigos sobre a discussão de aplicação dos frameworks de deep learning nos sistemas de recomendação e se isso tem de fato levado a um progresso significativo, para que o custo de desenvolvimento e de poder computacional valesse a pena. A conclusão que as técnicas com deep learning não vão além da busca por fatores latentes levando em conta as interações implícitas dos usuários como o SVD++ faz[6].

4.1.2 Visualização dos Dados

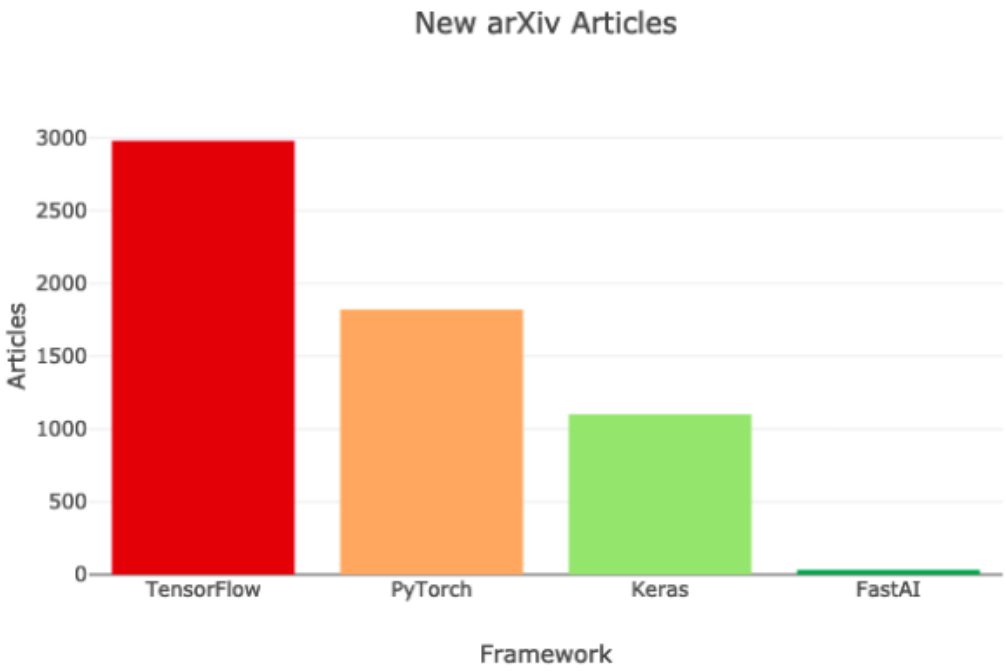
Para a bilbioteca de visualização de dados, uma breve pesquisa leva a conclusão que o matplotlib é de longe o mais usado, e cobre a grande maioria das situações com uma sintaxe extremamente prática. Alguns usam o seaborn especialmente para scatter plot já que tal tipo de plot é uma de suas especialidades. A conclusão no fim das contas, é que será usado o matplotlib sempre que possível, e caso não seja possível será usado o seaborn ou o vega para completar, tanto pelo tipo de plot quanto para a boa apresentação dos dados.

4.2 Aplicação dos Algoritmos de Recomendação

4.2.1 Dados de Entrada

Seguindo a linha de fazer o ambiente de desenvolvimento o mais ortodoxo possível, temos também um segundo fato: A grande maioria dos artigos sobre sistemas de recomendação envolvem datasets de filmes, seguido de músicas e o tema do e-commerce fica um tanto de lado nas pesquisas. Conforme podemos ver na figura 3 A principal razão para isto é limitação de dataset com qualidade o bastante para se testar as hipóteses. De longe o

Figura 2: Frameworks mais referenciados no arxiv de acordo com [5]



dataset mais utilizado nos artigo é o do MovieLens e ele será central no nossa pesquisa por isso. O uso do movielens em relação aos outros datasets pode ser observado na figura 4

Uma das grandes qualidades do MovieLens é sua estrutura. Em primeiro lugar o dataset não é esparso, as tabelas são completas, não há usuários com dados incompletos, ou itens com dados incompletos.

Segundo que ele não tem um excesso do dados do qual se necessita extrair um subconjunto, ele tem o necessário para performar o algoritmo recomendador.

O terceiro fator é seu tamanho, o dataset completo contém um número imenso de ratings, em torno de 20 milhões e isso permite realizar um conjunto de treinamento e testes muito grande, mesmo para técnicas que demandam mais dados de treinamento.

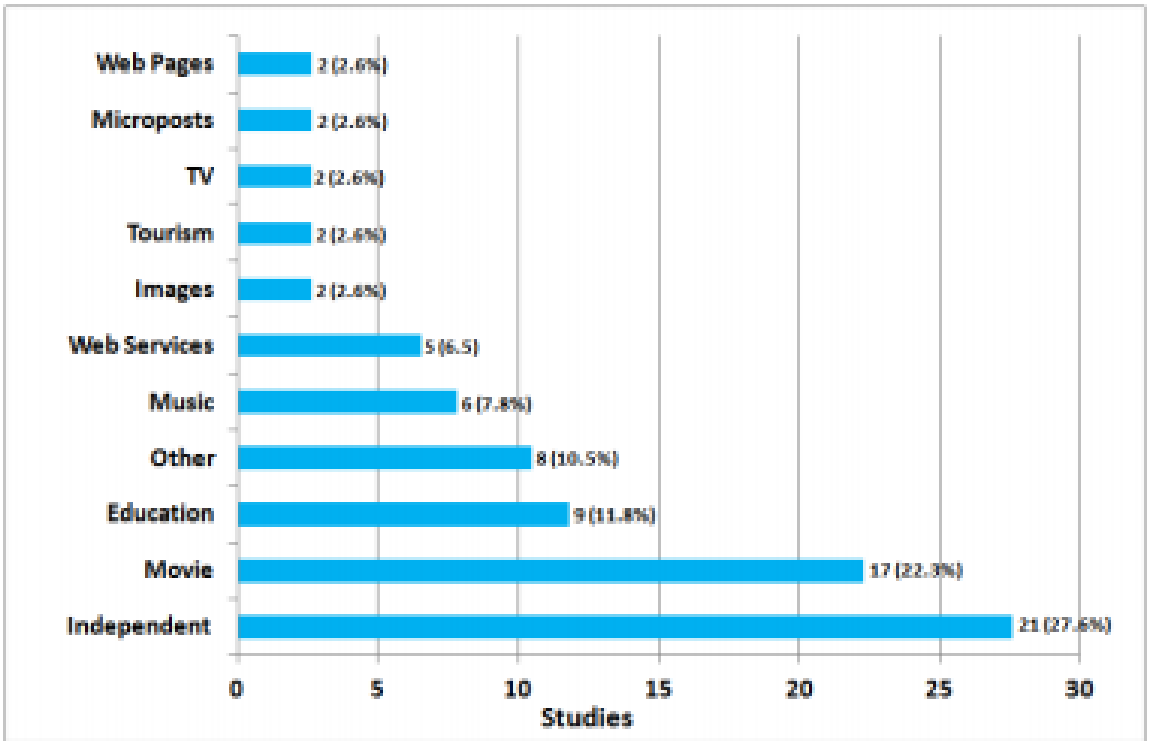
O quarto e ultimo fator para sua escolha é o arquivo em si, é um conjunto de CSV's, um formato extremamente popular tanto para o uso de bibliotecas simples como numpy, passando por pandas que é essencial para se lidar com grandes dados.

4.3 Preparação dos Dados

4.3.1 "Long Tail Crop"

A ideia do "Long Tail Crop"é conforme sugerido por algumas fontes, tirar a parte "ruidosa"demais dos dados, que são usuarios que avaliaram pouquissimos filmes, e filmes que foram muito pouco avaliados. Esses dados poluem a analise pois não são de fatos filmes

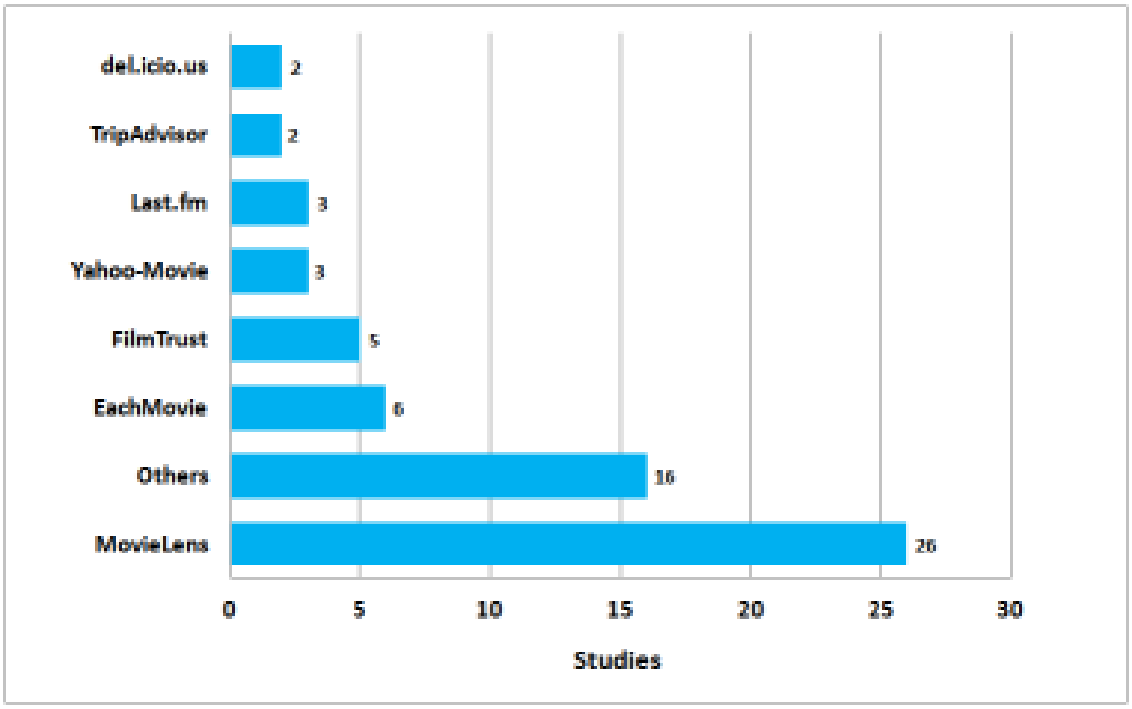
Figura 3: Áreas de aplicação mais comuns nos estudos de acordo com a Pesquisa de Sistemas de Recomendação Híbridos



ou usuários que podem ter seus perfis traçados com solidez. Normalmente em plataformas de conteúdo, e e-commerce, tais itens não são recomendados pelos algoritmos, são deixados a critérios dos usuários achá-los, ou então ficarem presentes numa seção de "novidades", e no caso dos usuários a eles são oferecidos indicações genéricas e bastante variadas entre os clusters de itens para criar uma vitrine da plataforma para este novo usuário.
Há contrapontos conforme no artigo [7] da Universidade da Pensylvania de 2009, que um dos diferenciais do Netflix é repensar o uso da Long Tail para além de ser meramente cortada, mas sim a capacidade de ser usada para recomendar nichos e identificar o gosto dos usuários por estes nichos. Uma análise bem mais complexa, que não abordaremos nos experimentos, mas que é essencial para atacar o problema das recomendações genérica e monótonas. No dataset do movielens, há uma seleção prévia de filmes e usuários para compor o dataset em sua versão menor, o 100K, podemos ver essa distribuição de avaliações feitas por usuários na figura 5 e feitos nos filmes na figura 6

Portanto após esses resultados foi decidido que ficaremos apenas com os 200 usuários que mais avaliaram filmes, e com os 2000 filmes mais avaliados, e assim os datasets foram recalculados com a retirada das respectivas linhas. Dessa forma temos um conjunto de dados com bem menos ruído para trabalharmos com o PCA e a Clusterização.

Figura 4: Datasets mais utilizados

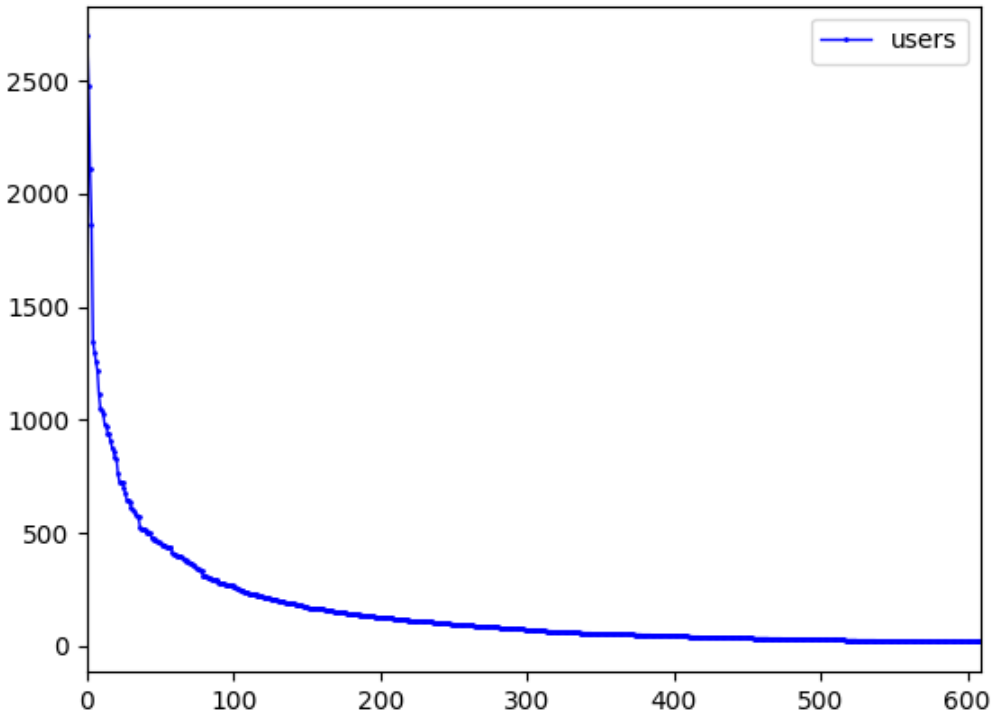


4.3.2 Reduçao de Dimensionalidade

A redução de dimensionalidade é um processo fundamental para tornar a clusterização viável, uma vez que ela precisa do calculo de distâncias entre os pontos, esse calculo é profundamente afetado pela alta dimensionalidade dos dados.
Foi escolhido utilizar o genero dos filmes como seu vetor de caracterização, que possuem 18 dimensões, o número de gêneros, afinal um filme pode ser de multiplos gêneros. Esse vetor de caracterização é composto por 0 e 1's indicando se o filme tem ligação com este gênero ou não, a partir do arquivo original do dataset "movies.csv".
Poderia ter sido escolhido para caracterizar os filmes as tags, mas que não foram escolhidas por sua complexidade de 1128 dimensões, e pelas palavras que compõem o título dos filmes - já que os filmes não possuem descrições além das tags - mas isso excederia nossa capacidade computacional para realizar as demonstrações.
Os usuários para serem condizentes com os filmes são definidos pelos gêneros dos filmes que eles avaliaram, esses valores de correlação usuário-gênero são definidos pela equação:

$$GenderUserCorrelation = \frac{1}{NumRatings} * \sum_{rating \in ratings} (Rating * MovieGenderVector) \tag{14}$$

Figura 5: Long Tail Plot dos Usuários do Dataset ML-100K

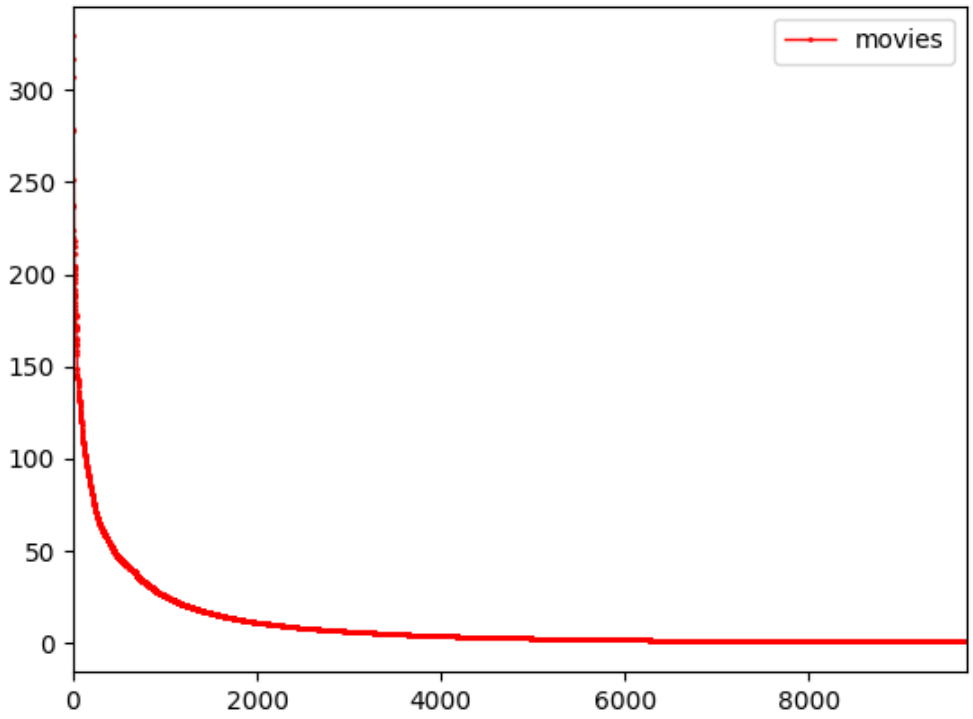


Onde NumRatings é o numero de avaliações que o usuário fez, ratings é o universo de avaliações que o usuário fez, e o MovieGenderVector é o vetor tamanho 18 que 0's e 1's que define se o filme tem alguma relação com aquele gênero.

Dos diversos algoritmos de redução de dimensionalidade escolhemos o PCA [8] devido a sua velocidade de execução, sua ampla utilização, estar muito bem documentado e implementado pela biblioteca Scikit-Learn, além de ser um algoritmo clássico para se avaliar a contribuição das features para a caracterização dos itens.

O resultado foi um novo espaço de feature com 17 dimensões, essas novas features são uma combinação linear das antigas 18 definidas a partir da matriz de tamanho 18x17 obtida a partir de: "pca.components". As duas matrizes seja dos filmes, seja dos usuários pode ser vista na saída pcaresults.txt. A contribuição de cada feature para os usuários pode ser vista na figura 7 e para os filmes na figura 8

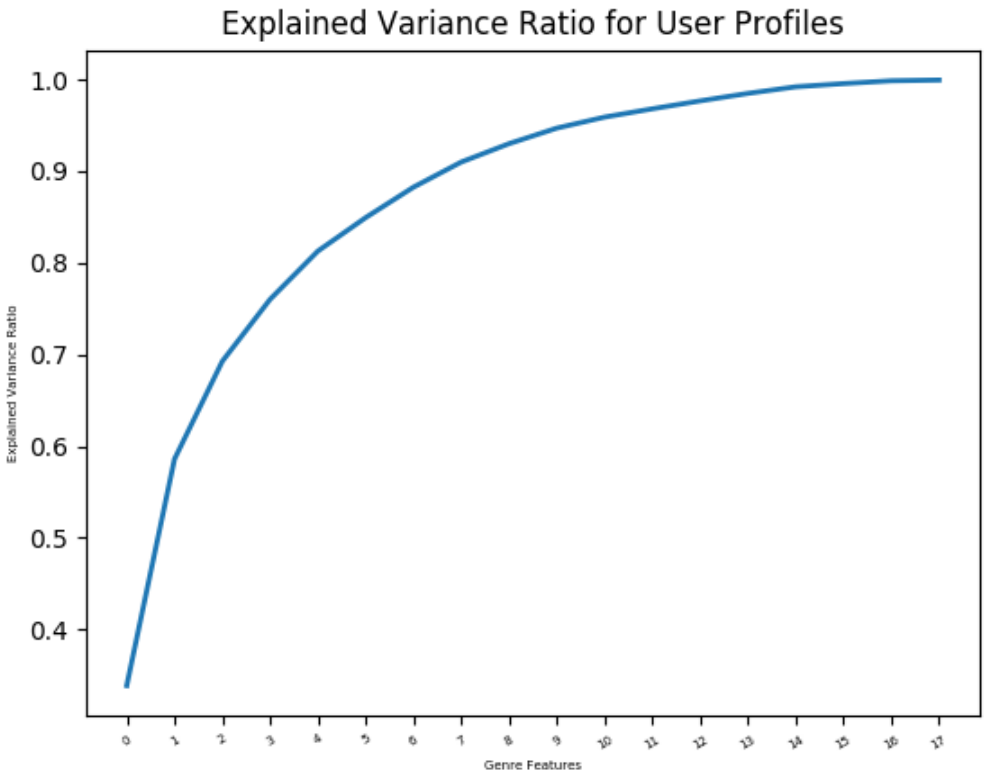
Figura 6: Long Tail Plot dos Filmes do Dataset ML-100K



4.3.3 Clusterização dos Filmes e Usuários

Após a redução de dimensionalidade, temos os dados prontos para serem clusterizados, o algoritmos escolhido foi o HDBSCAN. [9] Este algoritmo é uma versão aperfeiçoada do DBSCAN por permitir aplicar a dataset com densidades variáveis e uma seleção mais robusta dos parâmetros. Este algoritmo precisa apenas do numero mínimo de itens em uns cluster para performar, mas diferente de alguns outros algoritmos concorrentes ele pode deixar alguns itens do dataset fora de qualquer cluster, é permitido adicionar o numero minimo de clusters também, esse número pode ser muito bem determinado através da interface com o usuário que a plataforma de conteúdo terá. Diferentemente do K-means, ele não exige uma forma geométrica especifica, o que o torna adequado nessa situação com 17 dimensões, onde seria impossível saber qual é o formato das nuvens de pontos que os filmes ou os usuários estão formando. Porém ao executarmos o algoritmo com a escolha de parâmetros mais liberal possível para a formação dos clusters, como permitir que eles sejam os menores possíveis tivemos, no dataset testado que é o menor uma imensa formação de outliners, especialmente entre usuários. Nos filmes os outlines foram 26,45% e entre

Figura 7: Relevancia das Features nos Usuários - PCA



os usuários tivemos 73,34%, portanto a ideia de clusterizar esse dataset foi abandonada. Porém o princípio por tras da clusterização não deve ser ignorado em datasets maiores, mais diversos, e com a necessidade de dividirmos os grupos. Estes resultados podem ser vistos no arquivo HBDSCANresults.txt.

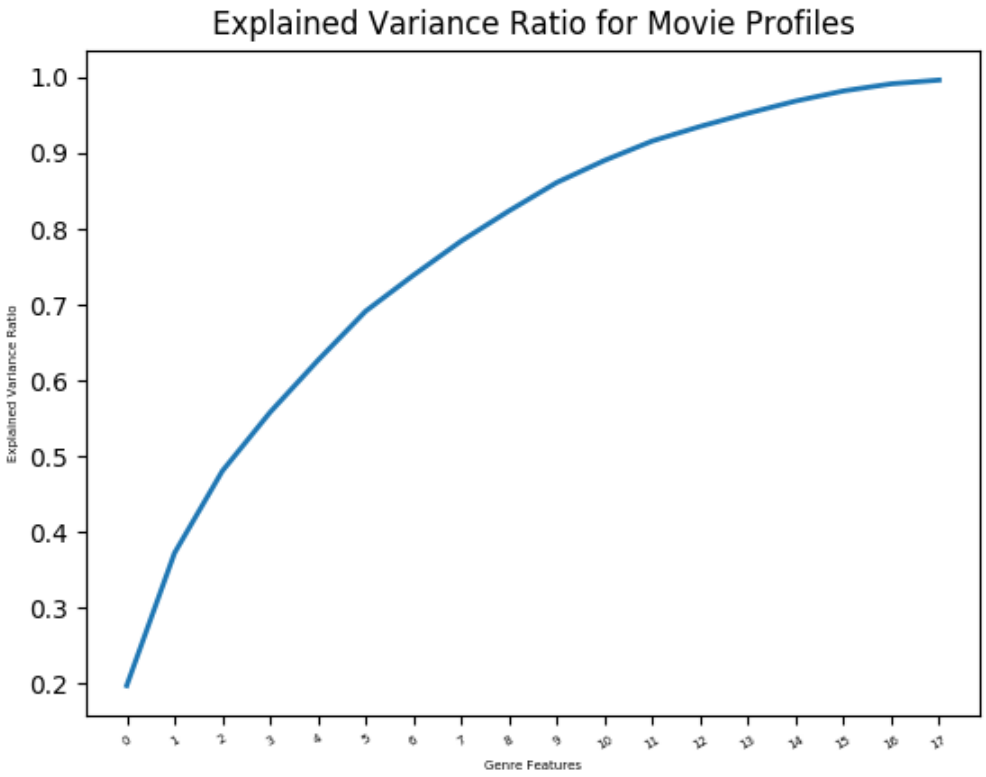
4.3.4 Métodos de Avaliação

Conforme na seção 3.7.1 da nossa referência [1] as formas mais comuns de métodos de avaliação são nos artigos: Comparação com um método similar, e em segundo lugar uma pesquisa entre os usuários, e em terceiros uma combinação de ambos. Como podemos ver na figura 9.

Justamente por ser um sistema com um comportamento parcialmente aleatório, em que os resultados individuais não podem ser objetivamente comparados, uma avaliação de diversos fatores, principalmente em como atacam cada um dos grandes problemas abordados no início e seu custo computacional são o melhor parâmetro para avaliar tais sistemas.

As características mais avaliadas e comparadas no contexto dos sistemas de recomendação

Figura 8: Relevancia das Features nos Filmes - PCA



são: Acurácia, Satisfação do Usuário, Diversidade, Complexidade Computacional e apresentação de novidades ao usuário, nessa ordem em termos de número de estudos, como podemos ver na figura 10

Cada uma dessas características tem diversas formas de ser comparada, e aqui vão as principais, como podemos ver na figura 11:

Para fins deste artigo usaremos não as mais famosas nesse levantamento, mas as mais comentadas históricamente, além de ser o garantidamente possível no tempo e na amostragem que possuímos: Para a acurácia: RMSE, MAE e F1. Para a diversidade: Ranking Distances e Long Tail Plot Para a complexidade: O tempo de execução

- O **Long Tail Plot** é um gráfico de número de avaliações no eixo Y, e movieID no eixo X. Isso é ordenado do mais para o menos avaliado, a maioria dos filmes são pouco conhecidos, pouco assistidos e pouco avaliados, enquanto alguns poucos concentram muitas avaliações e fama. Se fizer um Long Tail Plot dos recomendados e tiver uma "cauda longa"há uma recomendação adequadas de itens não apenas famosos.

- Uma das mais utilizadas que não poderemos testar nesse artigo, pois necessita de uma

Figura 9: Metodologias de Avaliação mais utilizadas

Methodology	Studies
Comparison with similar method	58
User survey	14
Comparison and user survey	3
No evaluation	1

Figura 10: Características mais avaliadas nos estudos de acordo com a Pesquisa de Sistemas de Recomendação Híbridos

Recommendation characteristic	Studies
Accuracy	62
User satisfaction	10
Diversity	7
Computational complexity	6
Novelty-Serendipity	4

plataforma funcional, é o MARK, **medium average recall for k**, que é a taxa com a qual um item é visualizado dado que foi incluso na lista de recomendados.

- O método de **raking distance**, é comparar a similaridade dos top-n itens recomendados, e com isso avaliar a diversidade das recomendações.

- O tempo de execução será medido com as funções de obter o data e hora do sistema operacional do próprio python.

- **F1:** Para definir o F1 é preciso deixar claro que só se aplica ao "hit set"ou seja os que estão no top-N recomendados e no conjunto de teste, e que dependem do recall e da precisão. Com isso determinamos:

$$\textbf{Recall} = \frac{TamanhodeHitSet}{TamanhodoTestSet} \tag{15}$$

$$\textbf{Precisão} = \frac{TamanhodeHitSet}{N} \tag{16}$$

Conforme N cresce o recall aumenta e a precisão cai, e vice-versa, por isso F1 é uma boa métrica por combinar os 2 na seguinte equação:

$$\textbf{F1} = \frac{2 * recall * precisão}{recall + precisão} \tag{17}$$

Figura 11: Métodos de Avaliação nos estudos de acordo com a Pesquisa de Sistemas de Recomendação Híbridos

Characteristic	Metrics	Studies
Accuracy	Precision	31
	MAE	27
	Recall	23
	F1	14
	RMSE	6
	Other	15
User satisfaction	Qualitative Subjective Assessment	10
Diversity	Coverage	4
	Ranking distances	3
Complexity	Execution time	6
Novelty-Serendipity	Surprisal	2
	Coverage in Long-Tail	1
	Expected Popularity Complement	1

Isso para cada usuário e assim tirar a média desses F1 para obter-se uma pontuação para o sistema em geral.

5 Experimentos e Resultados

Nesse momento utilizamos o Scikit-Surprise, uma biblioteca extremamente bem embasada nos principais artigos contendo a implementação dos principais algoritmos, com a capacidade justamente de fazer de forma pratica: A divisão do dataset em conjunto de treinamento e teste, *loops de cross-validation* e uma *Grid Search* dos diversos parâmetros em cima de um algoritmo para saber quais parâmetros passados possuem o melhor RMSE ou MAE. Os resultados das comparações entre os algoritmos com os parâmetros padrão pode ser visto na tabela 1. Os resultados da Grid Search com o SVD pode ser visto na tabela 2 e com o KNN na tabela 3.

Os algoritmos presente no Surprise que utilizamos para comparação foram os baseados em proximidade, as equações nas quais as avaliações previstas são baseadas estão na referência[16]

- **KNN-Basic**: É um algoritmo basico de filtragem colaborativa, que pode ser baseada em similaridade de usuários, em média 6 vezes mais rápida, ou baseada em similaridade entre itens. Apesar da similaridade de itens ser mais confiável e resultar em predições melhores, pois ao rodar o HDBSCAN pudemos observar que os itens são bem mais clusterizáveis em torno de proximidade ou distribuição de densidade que os usuários. È muito questionável se a complexidade multiplicar por 6 vale a pena, em geral não. A

medida de similaridade pode ser baseada em "MSD" *Mean Square Difference*,"cosine"a similaridade de cossenos que já discutimos, e "pearson"também já abordada, no nosso caso foi escolhida a MSD justamente por como ela trata melhor os dados do nosso contexto, e ao fazer a "Grid Search"obtivemos um resultado pior com a opção cosine. Com esses resultados descobertos seguimos em frente com essas configurações para os outros algoritmos dessa categoria.

- **KNN Means**: Este leva em consideração a média de avaliação de um usuário, se for baseada em usuário ou a avaliação média de um item caso seja baseada em conteúdo. Essa é uma forma menos custosa que calcular os bias, e é capaz de obter um resultado um pouco melhor que o KNN Básico, por levar em conta tais *outliners*

- **KNN Z-Score**: Este é analogo ao Means, mas ele utiliza no lugar das médias, os desvios padrão dos usuários ou dos itens. Oferecendo um intermediário de qualidade entre o Means e o Baseline.

- **KNN Baseline**: Nesse caso leva-se em conta no KNN a baseline que é muito bem descrita na referência[15] A baseline é composta do bias do usuário, o quanto as avaliações do usuário desviam das avaliações médias, do bias do item, que é o quanto o item é acima ou abaixo da média nas avaliações, e da avaliação média dos itens. Portanto tal algoritmo é mais bem sucedido que o KNN Basico especialmente para itens e usuários mais fora da média. Para evitar o overfitting os 2 bias e a avaliação prevista tem fatores *lambda* que são determinados por iterações de *Cross Validation* pois variam conforme o dataset.

O que pode ser observado é que a diferença no tempo de execução entre os diferentes tipos de KNN é pequena, por isso a versão mais complexa que é a "Baseline"certamente vale a pena nas aplicações comerciais.
Além disso foi observado que fazer a recomendação calculada com base em similaridade de itens é mais demorada, mas possui resultados melhores, e se essa diferença é significativa depende muito do dataset e da natureza do negócio que estiver usando o algoritmo.
Também concluimos que os algoritmo baseados em proximidade são mais lentos que os baseados em fatoração matricial,e demandam muito recálculo com a adição de novos usuários e itens, sendo estes adequados após uma pré-clusterização dos usuários e dos itens, onde os respectivos bias não sofreram alteração tão grandes com o tempo, nem serão necessárias novas cross validations constantes para o recalculos dos fatores lambda.
Outra coisa importante é o calculo de distância nos algoritmos baseados no KNN, e como isso os afeta, isso depende muito dos dados e do numero de dimensões que estamos lidando, no caso do suprise de acordo com esta referência[12] tem as seguintes similaridades: cosine, msd, Pearson e Pearson Baseline, e conforme dito acima a MSD se relevou a mais adequada para o nosso caso e na "Grid Search"obtivemos os melhores resultados com ela.
Também foi necessário escolher uma medida de acurácia[11], nas resultados gerais nos arquivos ".txt"calculamos todas as 4 disponíveis: RMSE, MAE, MSE e FCP, nos gráficos foram usadas apenas RMSE, MAE e o F1 calculados a partir da precisão e do recall, no entanto as 4 medidas de acurácia caminham muito juntas não dando resultados discordantes entre

si, se um algoritmo tem um RMSE melhor que o outro, terá um MAE, um FCP e um F1 melhor que o outro também.
Os algoritmos a seguir são os baseados em fatoração matricial, e suas respectivas equações se encontram na referência [10]

- **SVD**: O SVD foi muito popularizado durante o Netflix Prize e pela solução do Simon Funk, ele possui a versão com bias e sem bias, mas todas as vezes que foi testado usei a versão com bias. O objeto é descobrir um numero pré-definido de características latentes dos itens e o interesse dos usuários por tais características. Obviamente quanto mais características melhores os resultados, mas o tempo da execução cresce consideravelmente[13]. os 3 algoritmos de fatoração matricial tem 8 parâmetros que podem ser ajustados, que são as taxas de aprendizado aplicadas pelo SGD ou pelo ALS para o Bias do Usuario, o Bias do Item, o Qi que é a relação do itens com cada um dos fatores latentes, e o Pu que é a afinidade do usuários pelos fatores latentes. Os outros 4 parâmetros são os termos de regulalização desses mesmos itens para evitar o overfitting ou o underfitting.

- **NMF**: Este algoritmo é analogo ao anterior, a grande diferença é que os fatores relacionados aos usuários e aos itens nunca são negativos, justamente por isso o nome *Non-Negative*. Nos testes realizados as previsões foram de qualidade similar, as vezes levemente melhores que no SVD, sem grandes diferenças significativas, porém esta versão é mais sensível as condições iniciais, e isso pode ser aproveitado se for usado no contexto adequado, onde uma correlação negativa não faria sentido.

- **SVD++**: Este algoritmo foi o que obteve as melhores predições, porém ele tem uma complexidade computacional muito alta, e foi o que demorou mais para executar as recomendações, em média 10 vezes mais que o KNN que por sua vez é mais lente que o SVD ou o NMF. No entanto ele tem a grande vantagem de calcular as interações implícitas dos usuários, se baseando num perfil mais detalhado. Isso acrescenta um parâmetro a mais o "Yj", e sua respectiva taxa de aprendizado e parâmetro de regularização[14] O dado implícito que esse dataset permite explorar é basicamente se o usuário avaliou o filme ou não, levando em conta que o usuário normalmente avalia algo quando foi acima da média ou abaixo da média, e acaba esquecendo de avaliar quanto o filme foi "mediano"na sua opinião, isso permite dar uma noção melhor em como recomendar os filmes medianos para um determinado grupo de usuários que preferem certas características latentes por vê-las como boas ou ruins.

Há outros algoritmos no Scikit-Surprise, porém eles não foram extensamente avaliados, foram apenas executados, por conta de sua menor relevância tanto nos artigos como no mercado. Seus resultados foram semelhantes ou piores, que os baseados em KNN, porém suas respectivas implementações podem ser encontradas no repositório.
Além das tabelas, o resultados das medidas de RMSE, MAE e F1 para cada uma das 5 iterações de *Cross Validation*, podemos observar nas figuras 12, 13, 14 respectivamente.
Por fim temos a comparação de erros por rating dos algoritmos assim como a rating prevista para cada uma das quase 21000 previsões, presente respectivamente nas figuras 15 e 16.

Resultado da Grid Search:

Tabela 1: Comparing the Best Algorithm Results

Algorithm	RMSE	MAE	MSE	FCP	Precision	Recall
KNN Basic	0.9406	0.7215	0.8848	0.6690	0.7875	0.2958
KNN Means	0.8909	0.6801	0.7937	0.6522	0.8432	0.2845
KNN Z-Score	0.8882	0.6734	0.7889	0.6511	0.8260	0.2771
KNN Baseline	0.8625	0.6602	0.7438	0.6698	0.8353	0.2721
SVD	0.8688	0.6685	0.7547	0.6592	0.8447	0.2522
NMF	0.9124	0.7021	0.8324	0.6452	0.8099	0.2738
SVD++	0.8571	0.6582	0.7347	0.6832	0.8642	0.2557
Co-Clustering	0.8961	0.6850	0.8031	0.6623	0.8400	0.2844
Slope One	0.8961	0.6850	0.8031	0.6623	0.8400	0.2844
Baseline	0.8712	0.6697	0.7591	0.6772	0.8773	0.2297
Normal	1.4217	1.1348	2.0212	0.4986	0.5933	0.2465

Tabela 2: Best Result SVD

Metric	Biased	Num Factors	Learning Rate	Regularization Term
RMSE	True	50	0.005	0.4
MAE	True	50	0.005	0.4

6 Conclusão

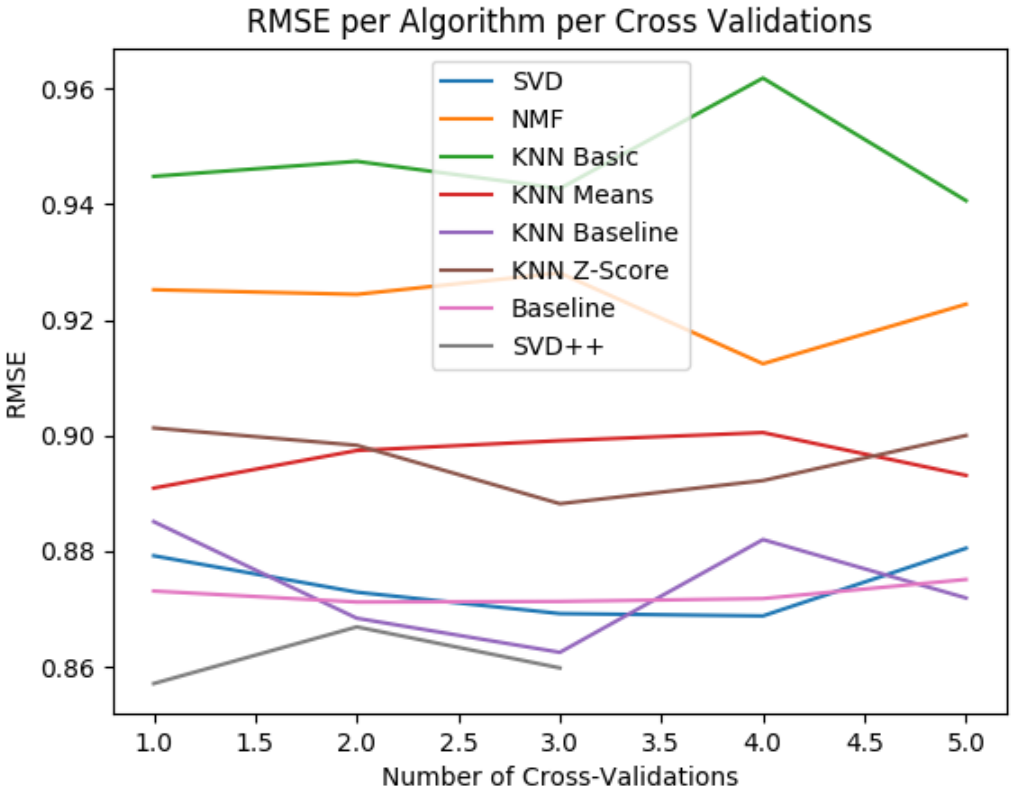
A principal conclusão é que a qualidade da recomendação vai muito além do algoritmo escolhido para fazer a previsão a partir do histórico de avaliações. Para garantir uma plataforma de consumo ou de conteúdo adequada e interessante depende dos seguintes fatores:

- Habilidade e Capacidade de clusterizar os itens por categorias, para garantir que a pagina inicial do usuário contenha dos top-N de diversas categorias, garantindo assim a diversidade de oferta de itens.
- Lidar com itens pouco avaliados, e saber como a Long Tail pode ser explorada através do oferecimento a usuários com mais histórico, assim resolver o problema do *Long tail* e da "ovelha cinza".
- Lidar com usuários que tem pouco histórico, e saber oferecer um misto do "genérico"da plataforma, com alguns itens que satisfazem o pouco que o usuário avaliou, tratando assim o *cold start*.
- Quanto ao problema dos dados esparsos, uma mistura de algoritmos adequados e as medidas já citadas são capazes de lidar com isso, na medida do possível.
- O problema dos itens sinônimos é necessário um sistema de constante exclusão do que foi consumido ou comprado e recalcular para aquele usuario, além da identificação do

Tabela 3: Best Result KNN

Metric	K	Similarity Option	Min Neighbors e	User Based
RMSE	55	MSD	1	False
MAE	55	MSD	1	False

Figura 12: RMSE dos Algoritmos Mencionados

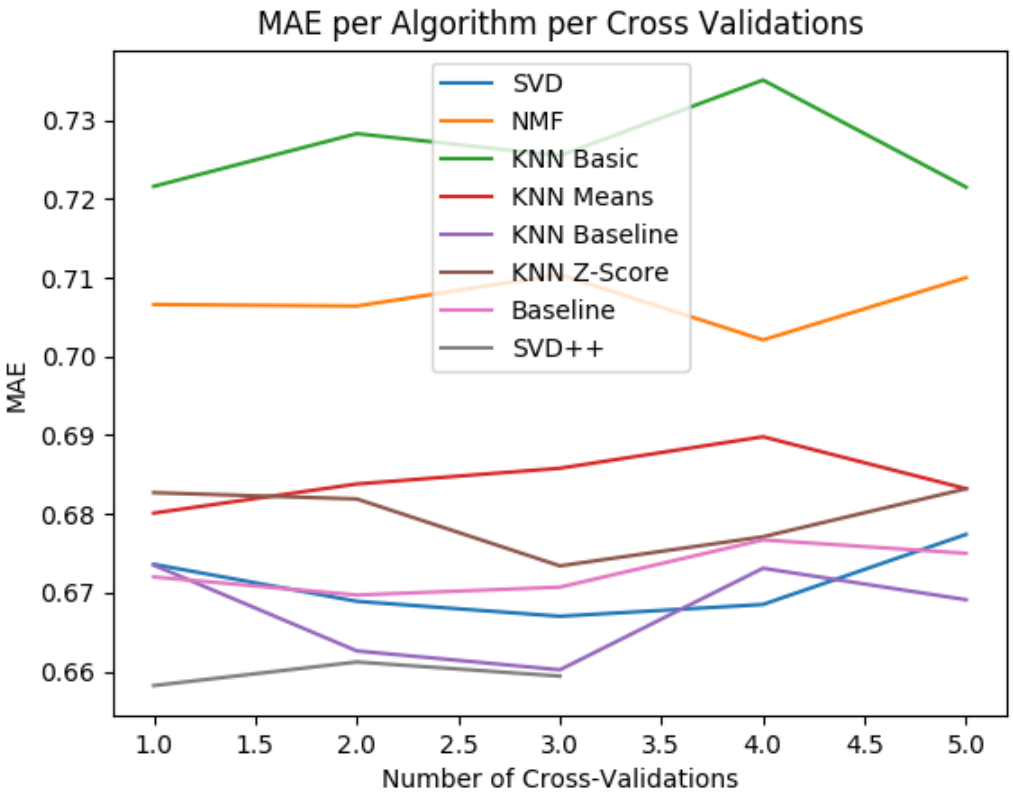


que é idêntico e do que é similar. Calcular essas similaridade é bastante custoso, por isso continua sendo um grande problema no e-commerce.

- O que é SVD++ ensinou para as plataformas de uma forma geral é que na maioria dos contextos, mais importante do que pedir para o usuário dar uma nota de 1 a 5, é apenas o *like/deslike* é suficiente, em conjunto com os dados de interação implícita. Tanto que hoje apenas plataformas como *e-commerce*, o *Google Maps* e a *Play Store* que mantém o sistema de 1 a 5 estrelas para avaliação.

- Outra conclusão é que é muito dificil conseguir algoritmos melhores sem um custo

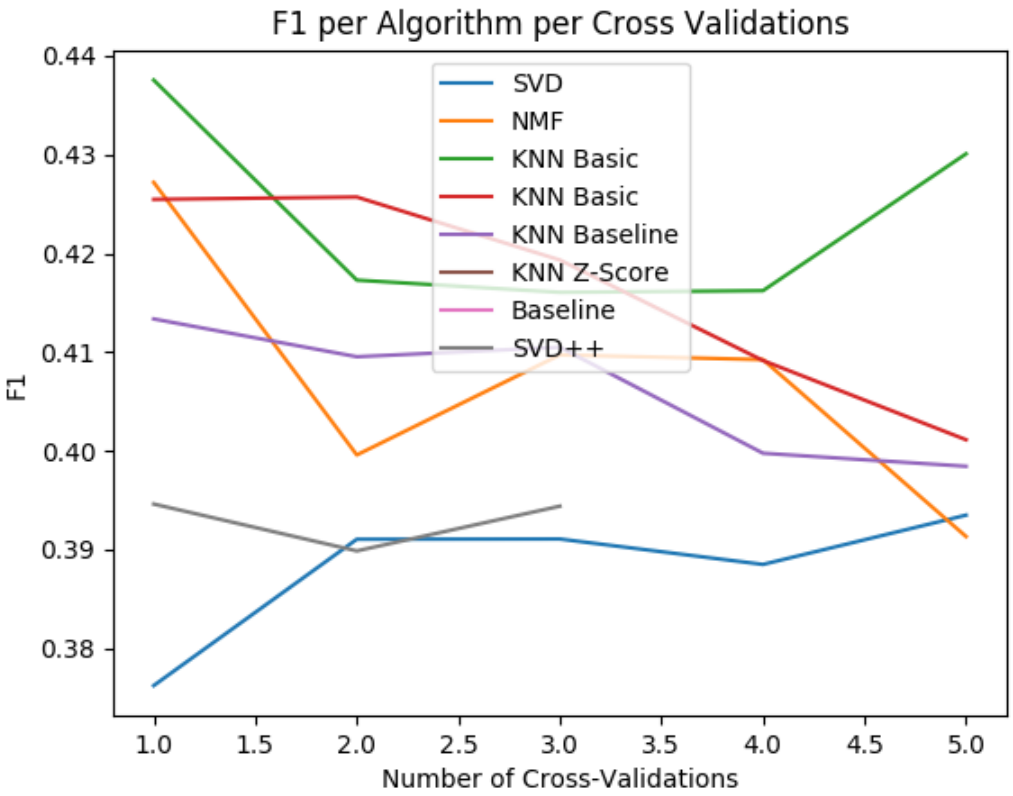
Figura 13: MAE dos Algoritmos Mencionados



computacional muito grande, e evoluir o algoritmo em si para além do que temos hoje é muito dificil. Portanto o mais importante nos ultimos anos tem sido aplicar os algoritmos e a preparação dos dados corretamente de acordo com o contexto.

Repositório Git do Projeto: https://github.com/raissaccorreia/movieRecommender

Figura 14: F1 dos Algoritmos Mencionados



Referências

[1] Erion Çano, Maurizio Morisio(2019), *Hybrid Recommender Systems: A Systematic Literature Review*, Intelligent Data Analysis.

[2] Xiaoyuan Su and Taghi M. Khoshgoftaar (2009), *Survey of Collaborative Filtering Techniques*, Department of Computer Science and Engineering, Florida Atlantic University https://www.hindawi.com/journals/aai/2009/421425/

[3] Francesco Ricci, Lior Rokach and Bracha Shapira (2011), *Introduction to Recommender Systems Handbook*, Springer Science+Business Media http://www.inf.unibz.it/ ricci/papers/intro-rec-sys-handbook.pdf

[4] John S. Breese, David Heckerman and Carl Kadie(2013), *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*, Microsoft Research https://arxiv.org/abs/1301.7363

Figura 15: Distribuição de Erro nas Recomendações

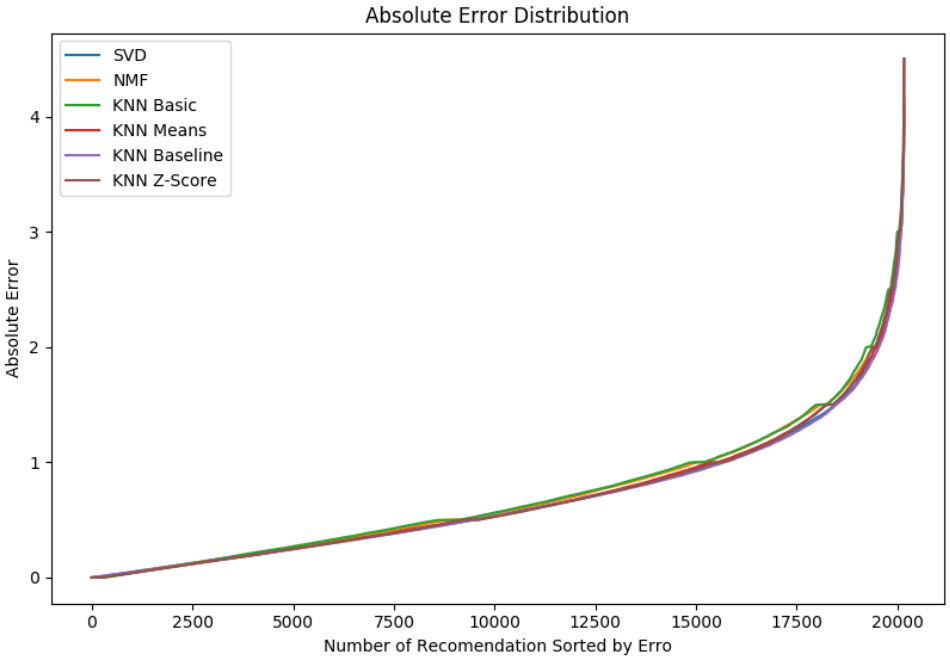
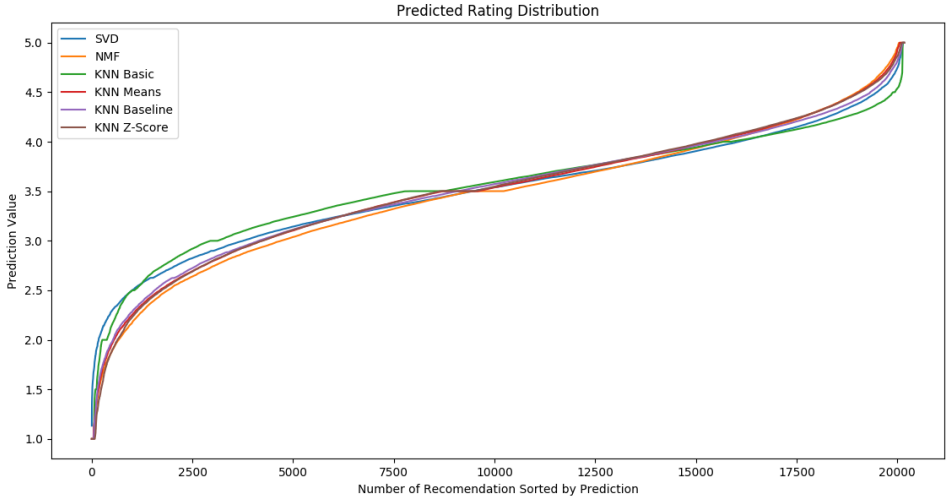


Figura 16: Avaliação Prevista nas Recomendações



[5] Jeff Hale (2019), *Which Deep Learning Framework is Growing Fastest?*, Towards Data Science https://towardsdatascience.com/which-deep-learning-framework-is-growing-fastest-3f77f14aa318

[6] Maurizio Ferrari Dacrema,Paolo Cremonesi,Dietmar Jannach (2019), *Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches*, Thirteenth ACM Conference on Recommender Systems https://arxiv.org/abs/1907.06902

[7] (2009), *Rethinking the Long Tail Theory: How to Define 'Hits' and 'Niches'*, University of Pensylvania https://knowledge.wharton.upenn.edu/article/rethinking-the-long-tail-theory-how-to-define-hits-and-niches/

[8] *Principal Component Analysis*, Scikit-Learn https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[9] *HDBSCAN Clustering Algorithm*, PyPI https://hdbscan.readthedocs.io/en/latest/ https://pypi.org/project/hdbscan/

[10] *Matrix Factorization Algorithms Surprise*, Scikit-Surprise https://surprise.readthedocs.io/en/stable/matrix$_f actorization.html$

[11] *Accuracy Metrics Suprise*, Scikit-Surprise https://surprise.readthedocs.io/en/stable/accuracy.html

[12] *Similarity Metrics Suprise*, Scikit-Surprise https://surprise.readthedocs.io/en/stable/similarities.html

[13] Yehuda Koren,Robert Bell and Chris Volinsky (2009), *Matrix Factorization Techniques for Recommender Systems*, Yahoo Research and ATT Labs—Research https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf

[14] Yehuda Koren (2008), *Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model*, ATT Labs—Research https://www.cs.rochester.edu/twiki/pub/Main/HarpSeminar/Factorization$_M eets_t he_N eighborhood-_a$ $_M ultifaceted_C ollaborative_F iltering_M odel.pdf$

[15] Yehuda Koren (2010), *Factor in the Neighbors: Scalable and Accurate Collaborative Filtering*, Yahoo Research http://courses.ischool.berkeley.edu/i290-dm/s11/SECURE/a1-koren.pdf

[16] *k-NN inspired algorithms*, Scikit-Surprise https://surprise.readthedocs.io/en/stable/knn$_i nspired.html$