

# Sistema de Recomendação

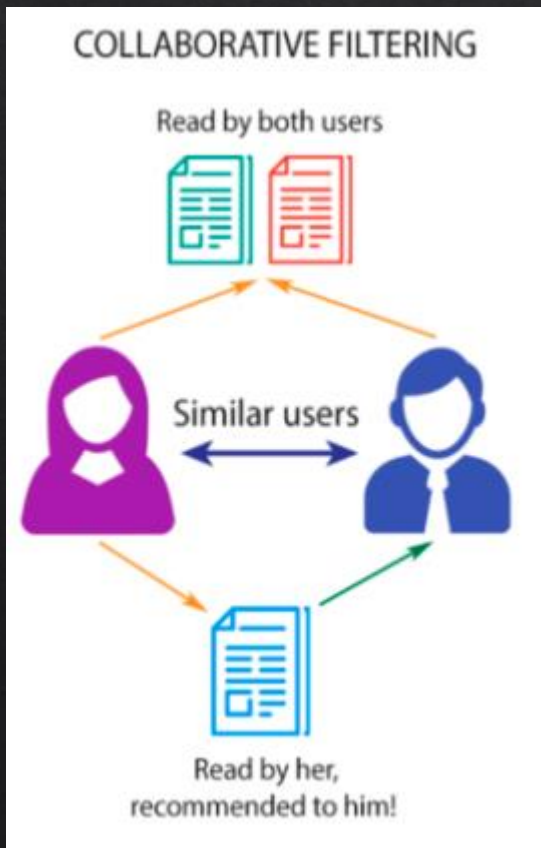
## Modelos de Predição

# Modelos de Predição

## Filtragem Colaborativa Baseada no Modelo

Baseada nas compras e avaliações do usuário

Sugestões de listas para compras futuras



Recomendações de itens com base nas escolhas de usuários similares

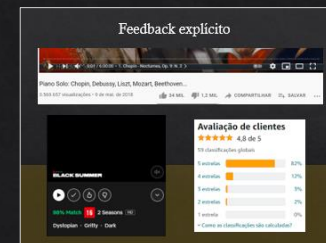
Usuários com interesses similares

Utiliza as avaliações dos usuários

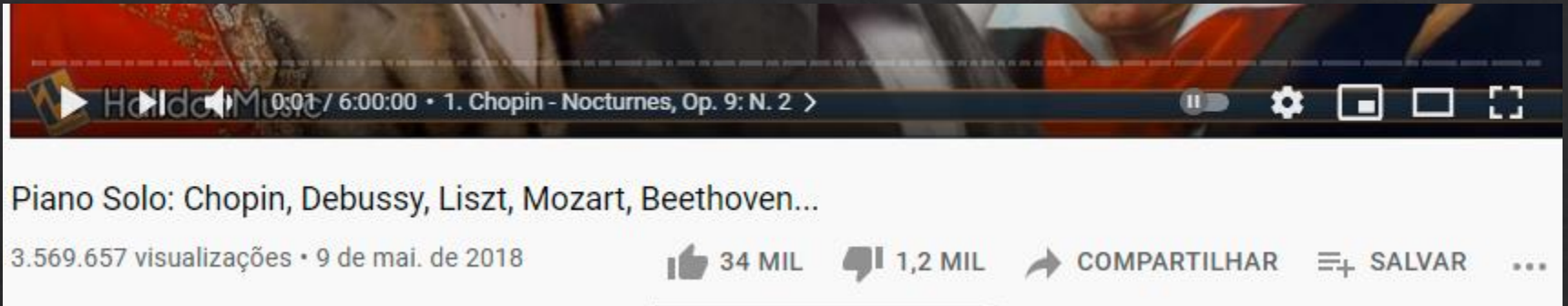
Feedback implícito



Feedback explícito



# Feedback explícito

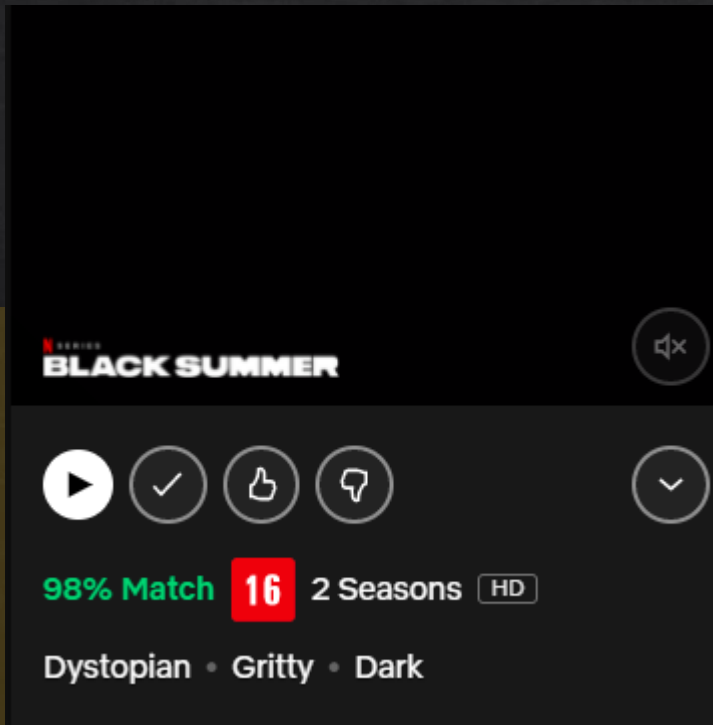


YouTube video player interface for a piano solo performance. The video title is "Piano Solo: Chopin, Debussy, Liszt, Mozart, Beethoven...". The video has 3,569,657 views and was uploaded on May 9, 2018. The player shows a progress bar at 0:01 / 6:00:00. The video is titled "1. Chopin - Nocturnes, Op. 9: N. 2". The interface includes standard YouTube controls: play/pause, volume, settings, full screen, and a share button. Below the video, there are engagement metrics: 34 MIL likes, 1,2 MIL comments, and a share button labeled "COMPARTILHAR". There is also a "SALVAR" (Save) button and a menu icon.

Piano Solo: Chopin, Debussy, Liszt, Mozart, Beethoven...

3.569.657 visualizações • 9 de mai. de 2018

34 MIL 1,2 MIL COMPARTILHAR SALVAR ...



Netflix interface for the series "Black Summer". The title "BLACK SUMMER" is displayed in white text on a black background. Below the title, there are icons for play, checkmark, like, and comment. The series is rated "98% Match" with a red "16" age rating. It is labeled "2 Seasons" and "HD". The genres "Dystopian", "Gritty", and "Dark" are listed at the bottom.

**BLACK SUMMER**

98% Match **16** 2 Seasons HD

Dystopian • Gritty • Dark



Customer rating breakdown for "Black Summer". The title "Avaliação de clientes" is displayed. The overall rating is 4.8 de 5, represented by five stars. There are 59 global ratings. The breakdown shows the percentage of ratings for each star level: 5 stars (82%), 4 stars (12%), 3 stars (3%), 2 stars (2%), and 1 star (0%). A link "Como as classificações são calculadas?" is provided at the bottom.

### Avaliação de clientes

★★★★★ 4,8 de 5

59 classificações globais

5 estrelas	82%
4 estrelas	12%
3 estrelas	3%
2 estrelas	2%
1 estrela	0%

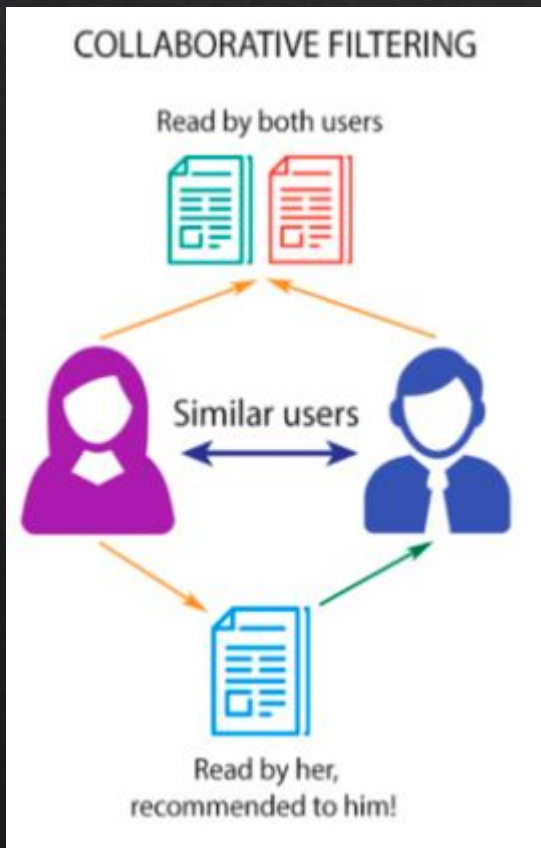
▼ Como as classificações são calculadas?

# Modelos de Predição

## Filtragem Colaborativa Baseada no Modelo

Baseada nas compras e avaliações do usuário

Sugestões de listas para compras futuras



Recomendações de itens com base nas escolhas de usuários similares

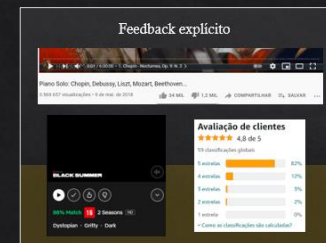
Usuários com interesses similares

Utiliza as avaliações dos usuários

Feedback implícito



Feedback explícito





# Feedback implícito

Nota inferida do comportamento do usuário

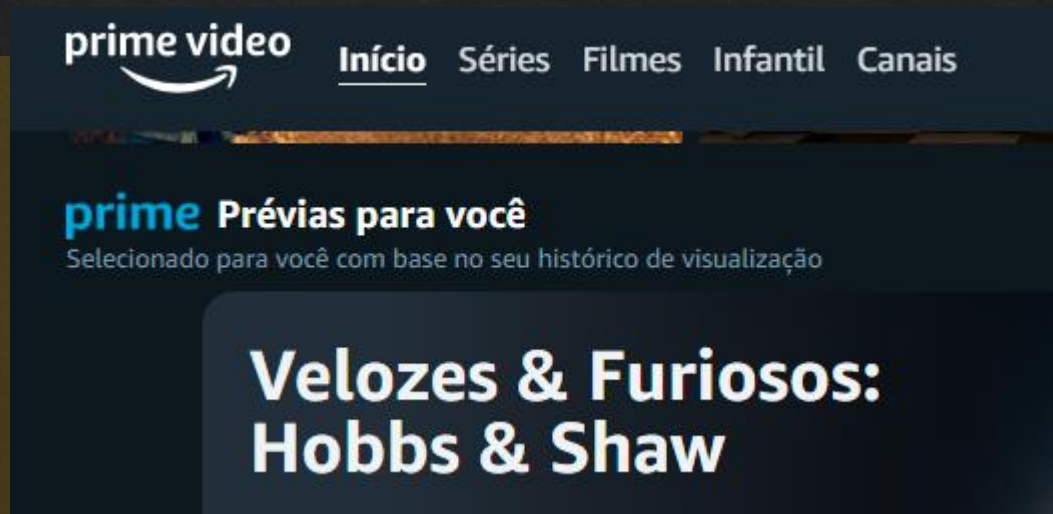
Histórico de compra

Histórico de navegação

Padrões de pesquisa

Sabe-se o que o usuário gosta

O que ele não gosta?



# Modelos de Predição

## Filtragem Colaborativa Baseada no Modelo

Bibliotecas disponíveis

Algoritmos de  
Clustering

KNN

K-means

Surprise

Algoritmos baseado em  
matriz de fatorização

SVD

Non-Ve  
MF

FMP

Surprise

fastai

# Modelos de Predição

## Filtragem Colaborativa Baseada no Modelo

### Scikit Surprise - Tools

#### Divisão do dataset em treino e teste

```
from surprise.model_selection import train_test_split
```

#### Retorna a acurácia de um algoritmo utilizando validação cruzada

```
from surprise.model_selection import cross_validate
```

#### Retorna a acurácia de um algoritmo utilizando validação cruzada em várias combinações de parâmetros – encontra os melhores parâmetros

```
from surprise.model_selection import GridSearchCV
```

#### Retorna a acurácia de um algoritmo utilizando validação cruzada em várias combinações randômicas de parâmetros – encontra os melhores parâmetros

```
from surprise.model_selection import RandomizedSearchCV
```

# Considerando: Filtragem Colaborativa

## K-Vizinhos mais próximos (kNN)

Criação de uma matriz: linhas (usuário)  
colunas (itens)

	$i_1$	$i_2$	$i_3$	$i_n$
$u_1$				
$u_2$				
$u_3$				
$u_a$				
$u_n$				



Importar o pacote



```
from sklearn.neighbors import NearestNeighbors
```

Aprendizado não supervisionado

## Hiperparâmetros

```
[ ] NearestNeighbors(n_neighbors=5, radius=1.0, algorithm='auto', leaf_size=20, metric='minkowski', p=2, metric_params=None)
```

Grid Search CV

<https://scikit-learn.org/stable/modules/neighbors.html#unsupervised-neighbors>



# Considerando: Filtragem Colaborativa

## K-Vizinhos mais próximos (kNN)

Criação de uma matriz: linhas (usuário)  
colunas (itens)

**Scikit Surprise**

	$i_1$	$i_2$	$i_3$	$i_n$
$u_1$				
$u_2$				
$u_3$				
$u_a$				
$u_n$				

Importar o pacote

```
!pip install surprise  
from surprise.prediction_algorithms import knns
```

Aprendizado não  
supervisionado

```
knns.KNNBasic(k=40, min_k=1, sim_options={}, verbose=True, **kwargs) #algoritmo básico para filtragem colaborativa  
knns.KNNWithMeans(k=40, min_k=1, sim_options={}, verbose=True, **kwargs)  
#algoritmo básico para filtragem colaborativa com avaliações médias de cada usuário  
knns.KNNWithZScore(k=40, min_k=1, sim_options={}, verbose=True, **kwargs)  
#algoritmo básico para filtragem colaborativa considerando normalização do score z de cada usuário  
knns.KNNBaseline(k=40, min_k=1, sim_options={}, bsl_options={}, verbose=True, **kwargs)  
#algoritmo básico para filtragem colaborativa considerando uma classificação de linha de base
```

# Considerando: Filtragem Colaborativa

## K-Vizinhos mais próximos (kNN)



```
knns.KNNBasic(k=40, min_k=1, sim_options={}, verbose=True, **kwargs) #algoritmo básico para filtragem colaborativa
knns.KNNWithMeans(k=40, min_k=1, sim_options={}, verbose=True, **kwargs)
#algoritmo básico para filtragem colaborativa com avaliações médias de cada usuário
knns.KNNWithZScore(k=40, min_k=1, sim_options={}, verbose=True, **kwargs)
#algoritmo básico para filtragem colaborativa considerando normalização do score z de cada usuário
knns.KNNBaseline(k=40, min_k=1, sim_options={}, bsf_options={}, verbose=True, **kwargs)
#algoritmo básico para filtragem colaborativa considerando uma classificação de linha de base
```

### Métricas de Similaridade

`sim_options={name='nome da similaridade', user_based='se as similaridades serão computadas entre usuários ou itens'}`

`name`='nome da similaridade': cosseno (similaridade de cosine);  
msd (semelhança da diferença média quadrática);  
pearson (coeficiente de correlação de pearson);  
pearson\_baseline (coeficiente de correlação de pearson  
[shrunk])

Distância euclidiana: cálculo da distância entre dois pontos

Similaridade: cálculo de quão perto estão dois pontos

# Considerando: Filtragem Colaborativa

## K-Vizinhos mais próximos (kNN)

```
from surprise import Trainset, dataset
knn = knns.KNNBasic()
knn.fit(data_train)
predict = knn.test([data_test])
```

Treinamento do modelo

Realizando recomendações a outros usuários

```
from collections import defaultdict
def top_recommend(predict, topN=3):
    top_recs = defaultdict(list)
    for uid, iid, true_r, est, _ in predict:
        top_recs[uid].append((iid, est))
    for uid, user_ratings in top_recs.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_recs[uid] = user_ratings[:topN]
    return top_recs
```

3 principais recomendações  
para cada usuário

Filtragem colaborativa item-item

# Considerando: Filtragem Colaborativa

## Clusterização - Surprise

### Co-clustering

```
class surprise.prediction_algorithms.co_clustering.CoClustering
```

Bases: `surprise.prediction_algorithms.algo_base.AlgoBase`

A collaborative filtering algorithm based on co-clustering.

This is a straightforward implementation of [George:2005].

Basically, users and items are assigned some clusters  $C_u$ ,  $C_i$ , and some co-clusters  $C_{ui}$ .

Clusters are assigned using a straightforward optimization method, much like k-means.

- Parameters:
- `n_cltr_u` (*int*) – Number of user clusters. Default is `3`.
  - `n_cltr_i` (*int*) – Number of item clusters. Default is `3`.
  - `n_epochs` (*int*) – Number of iteration of the optimization loop. Default is `20`.
  - `random_state` (*int*, RandomState instance from numpy, or `None`) – Determines the RNG that will be used for initialization. If *int*, `random_state` will be used as a seed for a new RNG. This is useful to get the same initialization over multiple calls to `fit()`. If RandomState instance, this same instance is used as RNG. If `None`, the current RNG from numpy is used. Default is `None`.
  - `verbose` (*bool*) – If True, the current epoch will be printed. Default is `False`.



# Considerando: Filtragem Colaborativa

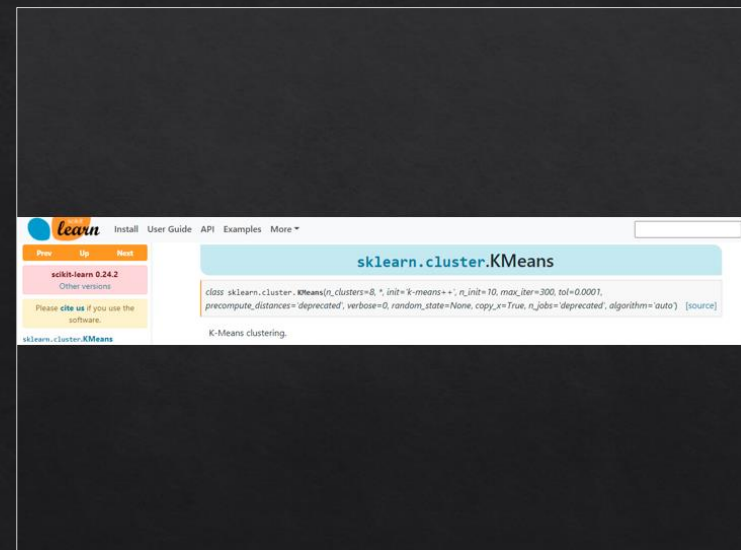
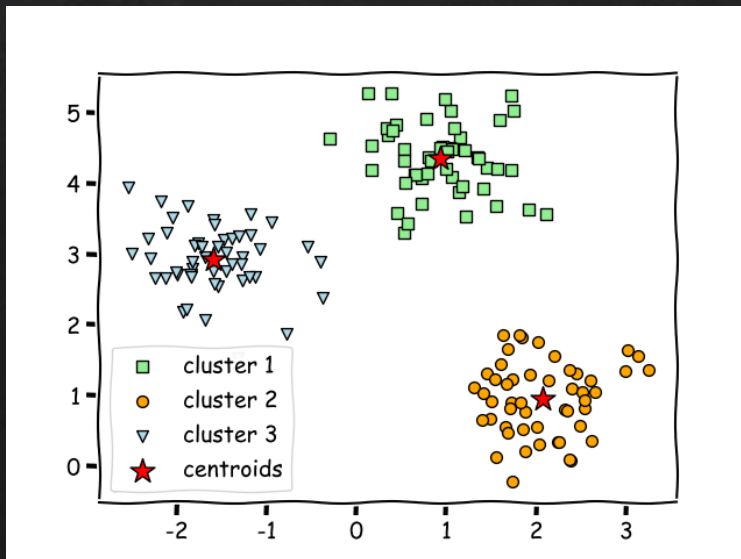
## Clusterização – K – means - Sklearn

Seleção do k (número de clusters)

Seleção randômica de centroides

Repetição das etapas anteriores (realocação iterativa)

Avaliação da distância de cada ponto ao centroides





## sklearn.cluster.KMeans

```
class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.0001,
precompute_distances='deprecated', verbose=0, random_state=None, copy_x=True, n_jobs='deprecated', algorithm='auto') \[source\]
```

K-Means clustering.

# Modelos de Predição

## Filtragem Colaborativa Baseada no Modelo

### Bibliotecas disponíveis

#### Algoritmos de Clustering

KNN

K-means

Surprise

#### Algoritmos baseado em matriz de fatorização

SVD

Non-Ve  
MF

FMP

Surprise

fastai

#### Considerando: Filtragem Colaborativa Matriz de Fatorização

##### Matrix Factorization-based algorithms

`surprise.prediction_algorithms.matrix_factorization.svd`

State: `surprise.prediction_algorithms.matrix_factorization.svd`

The famous SVD algorithm, as popularized by Simon Funk during the Netflix Prize. When baselines are not used, this is equivalent to Probabilistic Matrix Factorization (see below).

`surprise.prediction_algorithms.matrix_factorization.nmf`

State: `surprise.prediction_algorithms.matrix_factorization.nmf`

A collaborative filtering algorithm based on Non-negative Matrix Factorization.

`sklearn.decomposition.TruncatedSVD`

(this is `sklearn.decomposition.TruncatedSVD`)

Dimensionality reduction using truncated SVD (aka LSA).

# Considerando: Filtragem Colaborativa

## Matriz de Fatorização

### Matrix Factorization-based algorithms

```
class surprise.prediction_algorithms.matrix_factorization.SVD
```

Bases: `surprise.prediction_algorithms.algo_base.AlgoBase`

The famous SVD algorithm, as popularized by [Simon Funk](#) during the Netflix Prize. When baselines are not used, this is equivalent to Probabilistic Matrix Factorization [salakhutdinov2008a] (see [note](#) below).

```
class surprise.prediction_algorithms.matrix_factorization.NMF
```

Bases: `surprise.prediction_algorithms.algo_base.AlgoBase`

A collaborative filtering algorithm based on Non-negative Matrix Factorization.

### `sklearn.decomposition.TruncatedSVD`

```
class sklearn.decomposition.TruncatedSVD(n_components=2, *, algorithm='randomized', n_iter=5, random_state=None, tol=0.0)
```

[\[source\]](#)

Dimensionality reduction using truncated SVD (aka LSA).



# Considerando: Filtragem Colaborativa

## Matriz de Fatorização

