

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Tarcisio Moreira da Silva

Sistemas de Recomendação de Livros

Belo Horizonte
2022

Tarcisio Moreira da Silva

Sistemas de Recomendação de Livros

Trabalho de Conclusão de Curso apresentado
ao Curso de Especialização em Ciência de
Dados e Big Data como requisito parcial à
obtenção do título de especialista.

Belo Horizonte
2022

SUMÁRIO

1. Introdução	4
1.1. Contextualização	4
1.2. O problema proposto	4
2 Coleta de dados	5
3. Processamento/Tratamento de Dados	7
3.1 Ferramentas utilizadas	7
3.2 Importação dos dados	9
3.3 Visualizando informações dos conjuntos de dados	11
3.4 Visualizando a aparência dos Resultados	12
3.5 Removendo as colunas	14
3.6 Renomeando DataFrames	14
3.7 Associando DataFrames	16
3.7.1 Campos duplicados	17
4. Análise e Exploração dos Dados	17
4.1 Pré-Análise para exploração	17
4.2 Descrição e Exploração de informações	23
5. Criação de Modelos de Machine Learning.....	29
5.1 Filtragem colaborativa com NearestNeighbors.....	29
5.2 Filtragem colaborativa com PCA e K-Means	32
6. Interpretação dos Resultados	40
7. Apresentação dos Resultados	46
8. Links	46
REFERÊNCIAS	47

1. Introdução

1.1. Contextualização

Neste contexto, de um mundo onde existe uma diversidade muito grande de produtos (sejam eles bens de consumo, filmes, vídeos, posts, etc), é que surge a necessidade de uma ferramenta que consiga entender e disponibilizar para os usuários os produtos mais relevantes, de forma que o processo de escolha/compra seja viável.

Assim surgiram os sistemas de recomendação que, hoje, são muito presentes no nosso dia a dia.

Um sistema de recomendação é uma das principais aplicações da ciência de dados. Toda empresa consumidora de Internet requer um sistema de recomendação como Netflix, Youtube, feed de notícias etc. O que você quer mostrar de uma grande variedade de itens é um sistema de recomendação.

Um sistema de recomendação de livros é um tipo de sistema de recomendação em que temos que recomendar livros semelhantes ao leitor com base em seu interesse. O sistema de recomendação de livros é usado por sites online que fornecem e-books como google play books, open library, good Read's etc.

1.2. O problema proposto

Usaremos um método de **filtragem baseada em colaboração** para construir dois soluções de sistemas de recomendação de livros e avaliar o desempenho dos resultados. Isso ajudará o leitor a encontrar os melhores livros do seu interesse.

Para primeiro problema usaremos o algoritmo **Nearest Neighbor**, para criarmos uma máquina não supervisionada.

Iremos construir 1 máquina que com base nas escolhas de leituras de outras pessoas, o livro seja recomendado a outras pessoas com interesse semelhante.

Exemplo:

Mauricio leu e gostou do livro: "O código Limpo".

Fernanda: também leu e gostou desse livro

Agora Mauricio leu e gostou do livro "Pai rico, pai pobre".

Então temos que recomendar o livro "Pai rico, pai pobre" para Fernanda.

Para segundo problema iremos criar uma máquina de aprendizagem não-supervisionado K-means e PCA.

A forma para identificar se o usuário gosta do livro é se ele avaliou o mesmo acima da média de suas avaliações.

Iremos agrupar os perfis semelhantes de comportamento de avaliações dos livros, para listar os livros com melhores avaliações dos usuários de acordo com o perfil de cada usuário.

2 Coleta de dados

Como foi solicitado mais de um *dataset* de fontes diferentes eu utilizei para enriquecimento o conjunto de dados **Books data**. Este conjunto de dados é inspirado no Gooreads e foram adicionados os gêneros de livros usando a API do Google, os dados foram coletados no site <https://www.kaggle.com/datasets/heryhelder/books-data> e contém 112623 linhas de uma lista de livros. O conjunto de dados **Books data** contém uma tabela de livros com os campos abaixo.

Tabela 1: Estrutura Dataset Books data

Campos	Descrição	Tipo
Name	Título do Livro	object
Authors	Nome do autor do Livro	object
ISBN	Identificador Padrão Internacional de Numeração de Livro	object
PublishYear	Ano de publicação do Livro	float64
Language	Idioma do livro	object
Description	Descrição do livro	object
Genres	Gêneros de livros	object

Já o Conjunto de dados de livros **Book-Crossing Dataset** - Livros lidos por usuários e classificações fornecidas por eles na Amazon, coletados em <http://www2.informatik.uni-freiburg.de/~ciegler/BX/>.

Este conjunto de dados foi compilado por Cai-Nicolas Ziegler em 2004, é composto por três tabelas para usuários, livros e avaliações. As classificações explícitas são expressas em uma escala de 1 a 10 (valores mais altos denotando maior apreciação) e a classificação implícita é expressa por 0.

Contém 278.858 usuários (anônimos, mas, com informações demográficas) fornecendo 1.149.780 avaliações (explícitas/implícitas) sobre 271.379 livros.

O conjunto de dados Book-Crossing compreende três tabelas.

- **BX-Usuários**

Contém os usuários, observe que os IDs de usuário (`User-ID`) foram anonimizados e mapeados para números inteiros. Dados demográficos são fornecidos (`Location`, `Age`) se disponíveis. Caso contrário, esses campos contêm valores NULL.

Tabela 2: Estrutura Dataset BX-Usuários

Campos	Descrição	Tipo
User-ID	Identificador do usuário	int64
Location	Localização geográfica	object
Age	Idade biológica do usuário	float64

- **BX-Livros**

Os livros são identificados por seus respectivos ISBN. ISBNs inválidos já foram removidos do conjunto de dados. Além disso, algumas informações baseadas em conteúdo são fornecidas (`Book-Title`, `Book-Author`, `Year-Of-Publication`, `Publisher`), obtidas da Amazon Web Services. Observe que no caso de vários autores, apenas o primeiro é fornecido. URLs com links para imagens de capa também são fornecidas, aparecendo em três tipos diferentes (`Image-URL-S`, `Image-URL-M`, `Image-URL-L`), ou seja, pequeno, médio e grande. Esses URLs apontam para o site da Amazon.

Tabela 3: Estrutura Dataset BX-Livros

Campos	Descrição	Tipo
ISBN	Identificador Padrão Internacional de Numeração de Livro	object
Book-Title	Título do livro	object
Book-Author	Nome do autor do livro	object
Year-Of-Publication	Ano de publicação do livro	object
Publisher	Nome do editor	object
Image-URL-S	Imagem da capa do livro tamanho pequeno	object
Image-URL-M	Imagem da capa do livro tamanho média	object
Image-URL-L	Imagem da capa do livro tamanho grande	object

- **BX-Book-Ratings**

Contém as informações de classificação do livro. As classificações ('Book-Rating') são explícitas, expressas em uma escala de 1 a 10 (valores mais altos denotando maior valorização), ou implícitas, expressas por 0.

Tabela 4: Estrutura Dataset **BX-Book-Ratings**

Campos	Descrição	Tipo
User-ID	Identificador do usuário	int64
ISBN	Identificador Padrão Internacional de Numeração de Livro	object
Book-Rating	Avaliação do <u>livro</u>	int64

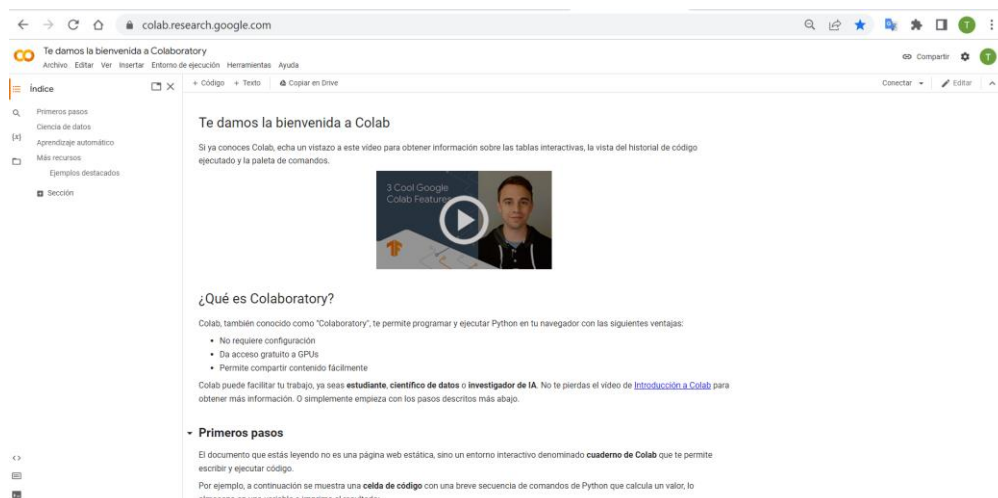
3. Processamento/Tratamento de Dados

Nessa seção é analisado os dados que são necessários para continuidade do trabalho. Os dados serão formatados e/ou utilizados filtros caso for preciso. Os dados que não forem necessários serão removidos.

3.1 Ferramentas utilizadas

Como ferramenta para desenvolvimento, foi escolhida a ferramenta em nuvem Google Colab, que permite criar e executar códigos na linguagem Python, disponível em <https://colab.research.google.com/>.

Figura 1: Screenshot Google Colab



Fonte: Autor

Foi escolhido o Google Colab pois não é preciso fazer nenhuma instalação na sua máquina local. Permite trabalhar de qualquer máquina que tenha acesso a internet.

Algumas das bibliotecas python já estão instaladas por padrão, mas caso seja necessário existe uma forma de instalar outras. Pode-se compartilhar com outros o acesso a um arquivo Colab.

3.1.1 Bibliotecas Instaladas

Para realizar o processamento e o tratamento dos dados, foi necessário importar algumas bibliotecas conforme a figura 2 abaixo

Figura 2: Screenshot importação de bibliotecas

```
# 1. Importando Bibliotecas
import pandas as pd # lib pandas
import numpy as np # lib numpy

# lib sklearn
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score, silhouette_samples
from sklearn.neighbors import NearestNeighbors

# Lib Kmeans
#Lib PCA análise de componentes principais
#Lib Avalia algoritmos de clusterização
#lib NearestNeighbors

# Lib Necessário para visualização de gráficos
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns
# Lib utilização dos graficos
#Lib conjunto de mapas de cores
#lib Seaborn atua em cima do matplotlib e ajuda a melhorar o visual dos gráficos

# biblioteca de visualização de dados do Python 3d matplotlib
from mpl_toolkits.mplot3d import Axes3D

#bibliotecas visualização de dados do Python
import cufflinks as cf # para conectar o plotly ao panda
import plotly
import plotly.offline as py
import plotly.graph_objects as go
import plotly.io as pio
from plotly.offline import plot, iplot
pio.renderers.default = "colab"

#configuração para o colab
```

Fonte: Autor

A tabela 5 mostra de forma detalhada as principais bibliotecas importadas.

Tabela 5: Bibliotecas utilizadas

Biblioteca	Descrição	Comando(s) utilizados
Pandas	Pacote de ferramentas para análise de dados e manipulação, construída sobre a base da linguagem de programação python.	import pandas as pd
Numpy	Pacote de ferramentas utilizada para realizar cálculos em Arrays multidimensionais, construída sobre a base de linguagem python.	import numpy as np

Matplotlib	Pacote de ferramentas utilizada para criação de gráficos e visualização de dados, construída sobre a base de programação python.	import matplotlib.pyplot as plt
Sklearn	Pacote de ferramentas utilizadas para trabalhar com Machine Learning, com ela são importados diversos métodos algoritmos e técnicas para facilitar a codificação.	from sklearn. cluster import KMeans from sklearn.decomposition import PCA from sklearn.metrics import silhouette_score from sklearn.neighbors import NearestNeighbors
Plotly	Pacote de ferramentas para visualização de dados e compreensão dos dados.	import plotly import plotly.offline as py import plotly.graph_objs as go import plotly.io as pio from plotly.offline import plot, iplot"

3.2 Importação dos dados

Nesse ponto iremos realizar as importações dos quatro arquivos CSV e nomear da seguinte forma:

Na Figura 3 contém o código para importação do arquivo **BX-Books.csv** para criação do DataFrame “**books**”.

Figura 3: Obtendo os dados do arquivo BX-Books.csv

```
[ ] # 2. Importação do Dataset Books
books = pd.read_csv("BX-Books.csv", sep=';', encoding='latin-1', error_bad_lines=False)

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: FutureWarning:
The error_bad_lines argument has been deprecated and will be removed in a future version.

b'Skipping line 6452: expected 8 fields, saw 9\nSkipping line 43667: expected 8 fields, saw 10\nSkipping line 51751: expected 8 fields, saw
b'Skipping line 92038: expected 8 fields, saw 9\nSkipping line 104319: expected 8 fields, saw 9\nSkipping line 121768: expected 8 fields, :
b'Skipping line 144058: expected 8 fields, saw 9\nSkipping line 150789: expected 8 fields, saw 9\nSkipping line 157128: expected 8 fields,
b'Skipping line 209388: expected 8 fields, saw 9\nSkipping line 220626: expected 8 fields, saw 9\nSkipping line 227933: expected 8 fields,
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: DtypeWarning:
Columns (3) have mixed types.Specify dtype option on import or set low_memory=False.
```

Fonte: Autor

Na Figura 4 contém o código para importação do arquivo **books_data.csv** para criação do DataFrame “**inf_books**”.

Figura 4: Obtendo os dados do arquivo BX-Books.csv

```
inf_books = pd.read_csv("books_data.csv", sep=',', encoding='latin-1', error_bad_lines=False)

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: FutureWarning:
The error_bad_lines argument has been deprecated and will be removed in a future version.

b'Skipping line 2398: expected 8 fields, saw 13\n'
```

Fonte: Autor

Na Figura 5 contém o código para importação do arquivo **BX-Users.csv** para criação do DataFrame “**users**”.

Figura 5: Obtendo os dados do arquivo BX-Users.csv

```
users = pd.read_csv("BX-Users.csv", sep=';', encoding='latin-1', error_bad_lines=False)

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: FutureWarning:
The error_bad_lines argument has been deprecated and will be removed in a future version.
```

Fonte: Autor

Na Figura 6 contém o código para importação do arquivo **BX- Ratings.csv** para criação do DataFrame “**ratings**”

Figura 6: Obtendo os dados do arquivo BX- Ratings.csv

```
users = pd.read_csv("BX-Users.csv", sep=';', encoding='latin-1', error_bad_lines=False)

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: FutureWarning:
The error_bad_lines argument has been deprecated and will be removed in a future version.
```

Fonte: Autor

3.3 Visualizando informações dos conjuntos de dados

Para cada objeto DataFrame foi utilizado a função `info()`, para visualizarmos algumas informações como a quantidade de registros, quantidade de colunas, informações de cada coluna e o tipo dela.

Podemos observar na Figura abaixo, que no DataFrame `inf_books` foram encontrados 111436 de registros com um total de 8 colunas.

Figura 7: Exibindo as informações do DataFrame `inf_books`

```

# Visualizando informações sobre os dados antes da formatação
inf_books.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 111436 entries, 0 to 111435
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Unnamed: 0             111436 non-null int64
1   Name                   111436 non-null object
2   Authors                111436 non-null object
3   ISBN                   111436 non-null object
4   PublishYear            111436 non-null int64
5   Language               111436 non-null object
6   Description            111436 non-null object
7   Genres                 111436 non-null object
dtypes: int64(2), object(6)
memory usage: 6.8+ MB

```

Fonte: Autor

Para o DataFrame com a lista de livros (`books`), temos o seguinte resultado 271360 registros e 8 colunas.

Figura 8: Exibindo as informações do DataFrame `books`

```

# Visualizando informações sobre os dados antes da formatação
books.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271360 entries, 0 to 271359
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ISBN                   271360 non-null object
1   Book-Title             271360 non-null object
2   Book-Author            271359 non-null object
3   Year-Of-Publication    271360 non-null object
4   Publisher              271358 non-null object
5   Image-URL-S            271360 non-null object
6   Image-URL-M            271360 non-null object
7   Image-URL-L            271357 non-null object
dtypes: object(8)
memory usage: 16.6+ MB

```

Fonte: Autor

Para o DataFrame com a lista de usuários (users), temos o seguinte resultado 278858 registros e 3 colunas.

Figura 9: Exibindo as informações do DataFrame users

```
✓ [77] # Visualizando informações sobre os dados antes da formatação
0s users.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278858 entries, 0 to 278857
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   User-ID     278858 non-null  int64
1   Location    278858 non-null  object
2   Age         168096 non-null  float64
dtypes: float64(1), int64(1), object(1)
memory usage: 6.4+ MB
```

Fonte: Autor

Para o DataFrame com a lista de avaliações (ratings), temos o seguinte resultado 1149780 registros e 3 colunas.

Figura 10: Exibindo as informações do DataFrame ratings

```
✓ # Visualizando informações sobre os dados antes da formatação
0s ratings.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1149780 entries, 0 to 1149779
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   User-ID     1149780 non-null  int64
1   ISBN        1149780 non-null  object
2   Book-Rating 1149780 non-null  int64
dtypes: int64(2), object(1)
memory usage: 26.3+ MB
```

Fonte: Autor

3.4 Visualizando a aparência dos Resultados

A seguir é apresentado o resultado nos conjuntos de dados coletados, selecionando apenas os cinco primeiros registros encontrados através do comando head().

Figura 11: Screenshot exibindo os resultados do DataFrame `inf_books`

✓ [83] #Vamos dar uma olhada rápida na aparência dos dados de Livros:
inf_books.head(5)

Unnamed: 0		Name	Authors	ISBN	PublishYear	Language	Description	Genres
0	0	Haroun and the Sea of Stories	Salman Rushdie	0613495632	1991	eng	The author of The Satanic Verses returns with ...	[Fiction]
1	1	The Desire and Pursuit of the Whole: A Romance...	Frederick Rolfe	0306802589	1988	eng	<i>The Desire and Pursuit of the Whole</i> sta...	[Fiction]
2	2	Green Arrow, Vol. 2: Sounds of Violence	Kevin Smith	1401200451	2004	eng	The reinvention of a classic comics character ...	[Comic books, strips, etc]
3	3	Trojan Odyssey (Dirk Pitt, #17)	Clive Cussler	0399150803	2003	eng	Long hailed as the grand master of adventure f...	[Fiction]
4	4	Strontium Dog: Search/Destroy Agency Files, Vo...	John Wagner	1905437153	2007	eng	Earth, the late 22nd century. Following the at...	[Bounty hunters]

Fonte: Autor

Figura 12: Screenshot exibindo os resultados do DataFrame `books`

books.head(5)

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	Image-URL-M	Image-URL-L
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.com/images/P/0195153448.0...
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.com/images/P/0002005018.0...
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/images/P/0060973129.0...
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	http://images.amazon.com/images/P/0374157065.0...	http://images.amazon.com/images/P/0374157065.0...
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	http://images.amazon.com/images/P/0393045218.0...	http://images.amazon.com/images/P/0393045218.0...	http://images.amazon.com/images/P/0393045218.0...

Fonte: Autor

Figura 13: Screenshot exibindo os resultados do DataFrame `users`

users.head(5)

	User-ID	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	NaN
3	4	porto, v.n.gaila, portugal	17.0
4	5	farnborough, hants, united kingdom	NaN

Fonte: Autor

Figura 14: Screenshot exibindo os resultados do DataFrame `ratings`

ratings.head(5)

	User-ID	ISBN	Book-Rating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6

Fonte: Autor

3.5 Removendo as colunas

Após realizado análise em cima dos DataFrames, verificou-se que existem colunas que não serão necessárias e não terão impacto em análises futuras. Desta forma essas colunas foram removidas conforme a Figura 15.

Figura 15: Removendo colunas dos DataFrames books e inf_books

```
[166] #inf_books
      del inf_books['Unnamed: 0'] # Coluna de index

      #books
      del books['Image-URL-M'] # Coluna img
      del books['Image-URL-S'] # Coluna img
      del books['Image-URL-L'] # Coluna img
```

Fonte: Autor

3.6 Renomeando DataFrames

Para deixar nossos dados formatados e padronizados, algumas colunas foram renomeadas utilizando a função rename(), conforme as Figuras abaixo.

Figura 16: Renomeando o DataFrame books

```
#Renomeando colunas do df books
books.rename(columns = {'Book-Title':'title',
                        'Book-Author':'author',
                        'Year-Of-Publication':'year_publisher',
                        'Publisher':'publisher'
                        }, inplace=True)
```

Fonte: Autor

Figura 17: Validando a renomeação do DataFrame books

```
[168] books.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271360 entries, 0 to 271359
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   ISBN             271360 non-null object
1   title            271360 non-null object
2   author           271359 non-null object
3   year_publisher   271360 non-null object
4   publisher        271358 non-null object
dtypes: object(5)
memory usage: 10.4+ MB
```

Fonte: Autor

Figura 18: Renomeando o DataFrame inf_books

```
#Renomeando colunas do df inf_books
inf_books.rename(columns = {'Name': 'inf_title',
                           'Authors': 'inf_author',
                           'PublishYear': 'inf_year_publishe',
                           'Description': 'description',
                           'Language': 'language',
                           'Genres': 'genres'
                           }, inplace=True)
```

Fonte: Autor

Figura 19: Validando a renomeação do DataFrame inf_books

```
[170] inf_books.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 111436 entries, 0 to 111435
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   inf_title              111436 non-null object
1   inf_author             111436 non-null object
2   ISBN                   111436 non-null object
3   inf_year_publishe     111436 non-null int64
4   Language               111436 non-null object
5   description            111436 non-null object
6   genres                 111436 non-null object
dtypes: int64(1), object(6)
memory usage: 6.0+ MB
```

Fonte: Autor

Figura 20: Renomeando o DataFrame users

```
[171] #Renomeando colunas do df users
users.rename(columns = {'User-ID': 'user_id',
                       'Location': 'location',
                       'Age': 'age'}, inplace=True)
```

Fonte: Autor

Figura 21: Validando a renomeação do DataFrame users

```

✓ [172] users.info()
0 s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278858 entries, 0 to 278857
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   user_id     278858 non-null  int64
1   location    278858 non-null  object
2   age         168096 non-null  float64
dtypes: float64(1), int64(1), object(1)
memory usage: 6.4+ MB

```

Fonte: Autor

Figura 22: Renomeando o DataFrame ratings

```

[173] #Renomeando Colunas do df rating
ratings.rename(columns = {'User-ID': 'user_id',
                          'Book-Rating': 'rating'}, inplace=True)

```

Fonte: Autor

Figura 23: Validando a renomeação do DataFrame ratings

```

▶ ratings.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1149780 entries, 0 to 1149779
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   user_id     1149780 non-null  int64
1   ISBN        1149780 non-null  object
2   rating      1149780 non-null  int64
dtypes: int64(2), object(1)
memory usage: 26.3+ MB

```

Fonte: Autor

3.7 Associando DataFrames

Após realizado a remoção das colunas que não serão necessárias e não terão impacto em análises futuras, iremos unir as tabelas books e inf_books realizando um merge utilizando o identificador 'ISBN' para encontrar os livros e para não perder os dados do DataFrame books, foi feito um "left join" nos DataFrame books com inf_books.

Figura 20: Associando os DataFrames books e inf_books

```
✓ [175] books = books.merge(inf_books, on= 'ISBN', how='left')
```

Fonte: Autor

Figura 21: Visualizando como ficou os campos DataFrames books

```
✓ books.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 271360 entries, 0 to 271359
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ISBN                  271360 non-null object
1   title                 271360 non-null object
2   author                271359 non-null object
3   year_publisher        271360 non-null object
4   publisher             271358 non-null object
5   inf_title             20669 non-null object
6   inf_author            20669 non-null object
7   inf_year_publicatio  20669 non-null float64
8   language              20669 non-null object
9   description           20669 non-null object
10  genres                20669 non-null object
dtypes: float64(1), object(10)
memory usage: 24.8+ MB
```

Fonte: Autor

3.7.1 Campos duplicados

Após juntar os DataFrames de books, observamos que o conjunto de dados está com colunas duplicadas, iremos retirar a duplicidade conforme Figura 21. Mantendo somente ISBN, title, author, year_publisher, publisher, language, description, genres.

Figura 21: Mantendo os campos DataFrames books

```
books = books[['ISBN', 'title', 'author', 'year_publisher', 'publisher', 'language', 'description', 'genres']]
```

Fonte: Autor

4. Análise e Exploração dos Dados

Nessa seção será mostrado todas as análises e exploração dos dados tratados anteriormente. Analisaremos as ocorrências, padrões e informações importantes que levantamos dos DataFrames.

4.1 Pré-Análise para exploração

Iremos realizar uma pré-análise dos dados para um refinamento dos DataFrames buscando identificar anomalias e iniciarmos os tratamentos dos dados.

4.1.1 Análise coluna year_publisher (Ano de publicação)

Na coluna ano de publicação do DataFrame books, podemos observar os anos em comum de publicação nas Figura abaixo:

Figura 22: Todos os anos de publicação cadastrados

```
books['year_publisher'].unique()

array([2002, 2001, 1991, 1999, 2000, 1993, 1996, 1988, 2004, 1998, 1994,
       2003, 1997, 1983, 1979, 1995, 1982, 1985, 1992, 1986, 1978, 1980,
       1952, 1987, 1990, 1981, 1989, 1984, 0, 1968, 1961, 1958, 1974,
       1976, 1971, 1977, 1975, 1965, 1941, 1970, 1962, 1973, 1972, 1960,
       1966, 1920, 1956, 1959, 1953, 1951, 1942, 1963, 1964, 1969, 1954,
       1950, 1967, 2005, 1957, 1940, 1937, 1955, 1946, 1936, 1930, 2011,
       1925, 1948, 1943, 1947, 1945, 1923, 2020, 1939, 1926, 1938, 2030,
       1911, 1904, 1949, 1932, 1928, 1929, 1927, 1931, 1914, 2050, 1934,
       1910, 1933, 1902, 1924, 1921, 1900, 2038, 2026, 1944, 1917, 1901,
       2010, 1908, 1906, 1935, 1806, 2021, '2000', '1995', '1999', '2004',
       '2003', '1990', '1994', '1986', '1989', '2002', '1981', '1993',
       '1983', '1982', '1976', '1991', '1977', '1998', '1992', '1996',
       '0', '1997', '2001', '1974', '1968', '1987', '1984', '1988',
       '1963', '1956', '1970', '1985', '1978', '1973', '1980', '1979',
       '1975', '1969', '1961', '1965', '1939', '1958', '1950', '1953',
       '1966', '1971', '1959', '1972', '1955', '1957', '1945', '1960',
       '1967', '1932', '1924', '1964', '2012', '1911', '1927', '1948',
       '1962', '2006', '1952', '1940', '1951', '1931', '1954', '2005',
       '1930', '1941', '1944', 'DK Publishing Inc', '1943', '1938',
       '1900', '1942', '1923', '1920', '1933', 'Gallimard', '1909',
       '1946', '2008', '1378', '2030', '1936', '1947', '2011', '2020',
       '1919', '1949', '1922', '1897', '2024', '1376', '1926', '2037'],
      dtype=object)
```

Fonte: Autor

Como pode ser visto acima, existem algumas entradas incorretas neste campo. Parece que os nomes dos editores 'DK Publishing Inc' e 'Gallimard' foram carregados incorretamente como “ano de publicação”.

Figura 23: Visualizando linhas com ano de publicação incorreto

```
[37] books.loc[books.year_publisher == 'DK Publishing Inc',:]

   ISBN      title  author  year_publisher  publisher  language  description  genres
209538  078946697X  DK Readers: Creating the X-Men, How It All Beg...  2000  DK Publishing Inc  http://images.amazon.com/images/P/078946697X.0...  NaN  NaN  NaN
221678  0789466953  DK Readers: Creating the X-Men, How Comic Book...  2000  DK Publishing Inc  http://images.amazon.com/images/P/0789466953.0...  NaN  NaN  NaN

[38] books.loc[books.year_publisher == 'Gallimard',:]

   ISBN      title  author  year_publisher  publisher  language  description  genres
220731  2070426769  Peuple du ciel, suivi de 'Les Bergers', Jean-M...  2003  Gallimard  http://images.amazon.com/images/P/2070426769.0...  NaN  NaN  NaN
```

- Como não temos o ano de publicação em nenhuma linha da DK Publishing Inc e da Gallimard, descartaremos essas linhas do conjunto de dados original

Fonte: Autor

Como não temos o ano de publicação em nenhuma linha da DK Publishing Inc e da Gallimard, descartaremos essas linhas do conjunto de dados books conforme Figura 24.

Figura 24: Excluindo linhas com ano de publicação incorreto

```
[39] books = books[~books['year_publisher'].isin(['DK Publishing Inc', 'Gallimard'])]
```

Fonte: Autor

Agora iremos realizar a alteração do campo **year_publisher** para numérico.

Figura 25: Alterando para numérico year_publisher

```
books['year_publisher'] = pd.to_numeric(books['year_publisher'])
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 271357 entries, 0 to 271359
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   ISBN            271357 non-null object
1   title           271357 non-null object
2   author          271356 non-null object
3   year_publisher  271357 non-null int64
4   publisher       271355 non-null object
5   language        20669 non-null object
6   description     20669 non-null object
7   genres          20669 non-null object
dtypes: int64(1), object(7)
memory usage: 18.6+ MB
```

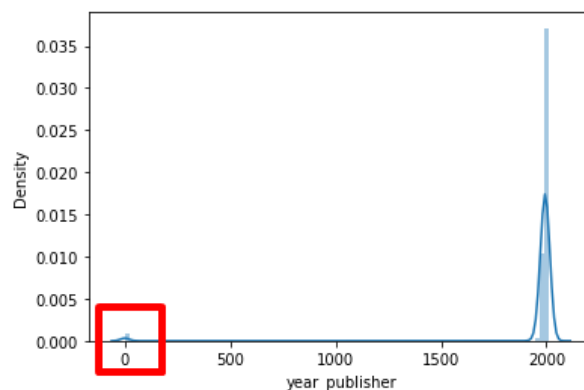
Fonte: Autor

Feito a alteração do tipo **year_publisher** para numérico, podemos observar no gráfico que existem anos fora do padrão.

Figura 26: Gráfico com os dados de ano de publicação

```
sns.distplot(books['year_publisher'], bins=100);
```

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.
`distplot` is a deprecated function and will be removed in a



Fonte: Autor

Note na Figura 27 que existem anos de publicação incorretos, com anos cadastrados de 0 a 2050.

Figura 27: Informações do ano de publicação

```
books['year_publisher'].describe()

count    271357.000000
mean      1959.760817
std       257.994226
min        0.000000
25%       1989.000000
50%       1995.000000
75%       2000.000000
max       2050.000000
Name: year_publisher, dtype: float64
```

Fonte: Autor

Com isso iremos filtrar os livros publicados de 1950 a 2016 retirando todos os dados que considere fora do padrão.

Figura 28: Filtrando os anos de publicação

```
books = books[(books['year_publisher']>=1950) & (books['year_publisher']<=2016)]

books['year_publisher'].describe()

count    266429.000000
mean      1993.759883
std        7.873026
min       1950.000000
25%       1989.000000
50%       1996.000000
75%       2000.000000
max       2012.000000
Name: year_publisher, dtype: float64
```

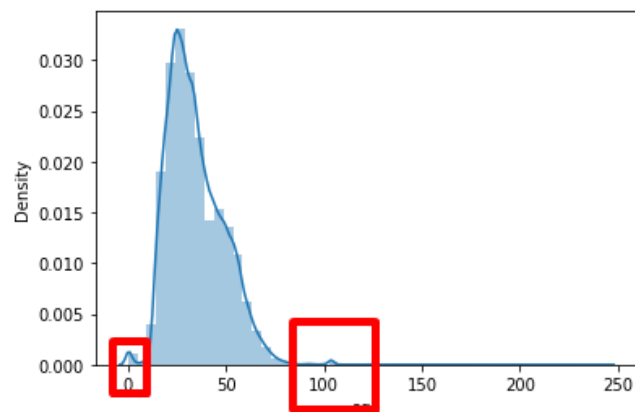
Fonte: Autor

4.1.2 Análise coluna Year (Idade do usuário)

Aqui avaliei as idades dos usuários cadastrados para investigar e tratar anomalias que podem existir. Como visto na Figura 29 mostra que temos idades cadastradas fora de um padrão.

Figura 29: Idades fora do padrão

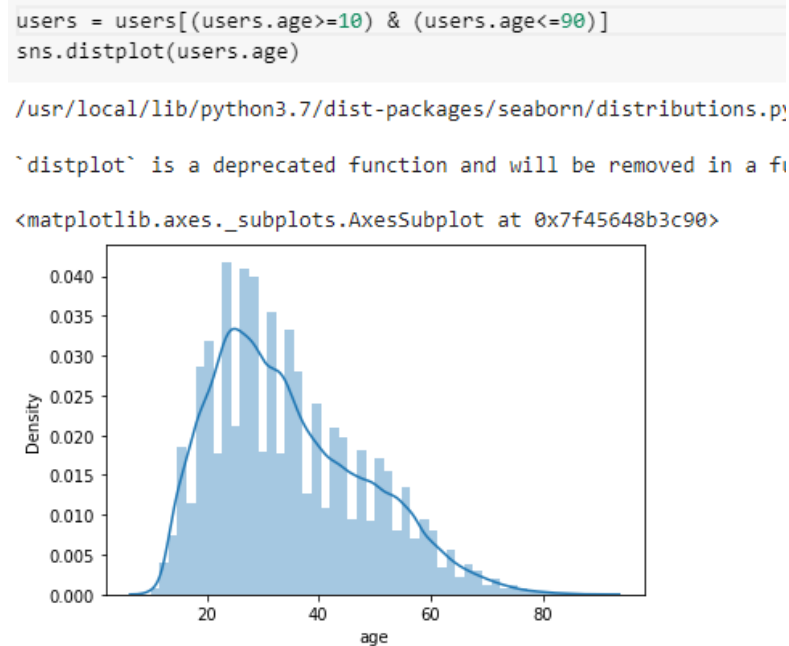
```
sns.distplot(users.age);  
  
/usr/local/lib/python3.7/dist-packages/seaborn/distributor  
`distplot` is a deprecated function and will be removed in
```



Fonte: Autor

Vamos limpar os usuários invalidados, para isso iremos filtrar os usuários com idade de 10 a 90 anos. Conforme a Figura 30.

Figura 30: Realizando filtragem no período estabelecido no campo Year.



Fonte: Autor

4.1.3 Limpando avaliações inoperantes

Foi realizado a limpeza das avaliações que não contém livros e usuários nos nossos conjuntos de dados, pois não há necessidade de manter os dados de avaliações de livros se não tivermos usuários e livros cadastrados. Conforme mostra na imagem abaixo, realizado limpeza das avaliações utilizando os identificadores ISBN e user_id para identificar os livros e usuários.

Podemos observar na Figura 31 que antes da limpeza tínhamos 1149780 registros e após limpeza das avaliações passou 734564 avaliações.

Figura 31: Limpando avaliações inoperantes

```
[381] print("Avaliações de livros antes da limpeza:", ratings.shape)

#Limpando avaliações pelo ISBN que não possuem Livros cadastrados
ratings = ratings[ratings['ISBN'].isin(list(books['ISBN'].unique()))]

#Limpando avaliações pelo user_id que não possuem usuários cadastrados
ratings = ratings[ratings['user_id'].isin(list(users['user_id'].unique()))]

print("Avaliações de livros após limpeza:", ratings.shape)
```

Avaliações de livros antes da limpeza: (1149780, 3)
Avaliações de livros após limpeza: (734564, 3)

_Fonte: Autor

4.2 Descrição e Exploração de informações

Obter dos dados a maior quantidade possível de informação, que indique modelos e comportamentos a serem utilizados na fase final do processo e realizando com isso alguns tratamentos dos dados.

4.2.1 Quantidade de avaliações por livro

Aqui iremos criar um campo que irá conter a quantidade de avaliações por livro. Ajudando a identificar os livros que possuem mais avaliações. Na Figura abaixo demonstro passo a passo para criação do campo **qtd_ratings** no DataFrame **ratings**.

Figura 32: Criando campo de quantidade de avaliações por livro

```
# Pegando a quantidade de avaliações por 'ISBN'
var_qtd_ratings = ratings.groupby('ISBN')['rating'].count()
# Alterei o index para mesclar a nova coluna com a tabela.
ratings = ratings.set_index('ISBN')
```

- Nesse ponto juntei o campo `qtd_ratings` com o nosso conjuntos de dados `ratings`

```
[383] #Criando coluna de quantidade de avaliações por livro
ratings['qtd_ratings'] = var_qtd_ratings
```

Fonte: Autor

4.2.2 Regra de Negócio nas avaliações

Não queremos livros que foi avaliado somente uma única vez, para isso iremos manter os usuários que avaliaram mais de 50 livros e menos de 250 livros, o limite de 250 foi usado para normalizar os dados para facilitar e explicitar os grupos (Remoção do outliers). Com isso iremos excluir o valor anômalo sem causar grandes prejuízos à análise de dados e observar a quantidade de avaliações após filtragem na Figura 33.

Figura 33: Mantendo os livros de 50 a 250 avaliações

```
[594] print("Antes de limpar o conjunto de dados ratings: ", ratings.shape)
      val = ratings['user_id'].value_counts()
      list_to_keep = list(val[(val>50) & (val<250)].index)
      ratings = ratings[ratings['user_id'].isin(list_to_keep)]
      print("Depois de limpar o conjunto de dados ratings: ", ratings.shape)

Antes de limpar o conjunto de dados ratings: (734564, 3)
Depois de limpar o conjunto de dados ratings: (193885, 3)
```

Fonte: Autor

4.2.3 Valores duplicados

Iremos tratar os valores duplicados, não queremos que o usuário tenha avaliado o livro mais de uma vez. Para isso iremos retirar as duplicidades das avaliações conforme figura abaixo.

Figura 34: Excluindo avaliações duplicadas

```
print("Quantidade antes de limpar os valores duplicados dos ratings: ", ratings.shape)
ratings.drop_duplicates(['user_id', 'ISBN'], inplace=True)
ratings.shape
print("Quantidade depois de limpar os valores duplicados dos ratings: ", ratings.shape)
```

Quantidade antes de limpar os valores duplicados dos ratings: (193885, 4)
 Quantidade depois de limpar os valores duplicados dos ratings: (193885, 4)

Fonte: Autor

4.2.4 Tratamento exclusivo para o primeiro problema (NearestNeighbors)

Vamos criar algumas condições exclusivas para tratar o primeiro problema proposto, pois a ideia é tentar encontrar duas soluções de recomendações de sistema de livros. Lembrando que para o primeiro problema iremos construir uma máquina que com base nas escolhas de leituras de outras pessoas, o livro seja recomendado a outras pessoas com interesse semelhante.

Para nossa primeira solução iremos recomendar o livro avaliado pela pessoa que avaliou também o livro escolhido por você.

4.2.4.1 Unindo a tabela ratings e books

Nessa etapa irei unir os DataFrames para relacionarmos os dados de avaliações com os dados dos livros. Na Figura abaixo demonstro como foi feito o Merge e exibo o resultado do conjunto de dados rating_with_books.

Figura 35: Unindo os dados ratings e books

```
rating_with_books = ratings.merge(books, on= 'ISBN', )
```

- Visualizando dados da nova tabela rating_with_books

```
rating_with_books.loc[rating_with_books.author == "Luther Blissett",:].head(5)
```

	ISBN	user_id	rating	qtd_ratings	title	author	year_publisher	publisher	language	description	genres
78	0151010633	276925	0	20	Q	Luther Blissett	2004	Harcourt	eng	<div>In 1517, Martin Luther nails his ninety-f...	['Fiction']
79	0151010633	3675	0	20	Q	Luther Blissett	2004	Harcourt	eng	<div>In 1517, Martin Luther nails his ninety-f...	['Fiction']
80	0151010633	13093	0	20	Q	Luther Blissett	2004	Harcourt	eng	<div>In 1517, Martin Luther nails his ninety-f...	['Fiction']
81	0151010633	14079	0	20	Q	Luther Blissett	2004	Harcourt	eng	<div>In 1517, Martin Luther nails his ninety-f...	['Fiction']
82	0151010633	38281	0	20	Q	Luther Blissett	2004	Harcourt	eng	<div>In 1517, Martin Luther nails his ninety-f...	['Fiction']

Fonte: Autor

4.2.4.2 Livros mais avaliados

Para nossa primeira solução queremos pegar os livros que foram avaliados no mínimo trinta vezes, para poder recomendar o livro e não simplesmente recomendar um livro que foi avaliado poucas vezes. Conforme Figura 36 estou filtrando os livros que tenham trinta ou mais avaliações e por fim mostro a quantidade de registros em nosso DataFrame.

Figura 36: Unindo os dados ratings e books

```
✓ [51] rating_with_books = rating_with_books[rating_with_books['qtd_ratings'] >= 30]
```

- Validando quantidade de linhas

```
✓ [52] rating_with_books.shape
```

```
(51379, 11)
```

Fonte: Autor

4.2.4.3 Tabela dinâmica para primeiro Problema

Nas linhas quero ter os livros e nas colunas cada nota de usuário por isso irei colocar os usuários em colunas, porque iremos usar o algoritmo NearestNeighbors que faz o cálculo da distância matemática e a melhor forma que posso colocar isso como uma previsão é colocar as notas dos usuários em coluna. Criei uma tabela dinâmica onde as colunas serão ids de usuário, o índice será o título do livro e o valor será classificações. E o id de usuário que não avaliou nenhum livro terá valor como NAN, então coloquei como zero.

Figura 37: Criando Tabela dinâmica

```
[102] %%time
book_pivot = rating_with_books.pivot_table(columns='user_id', index='title', values='rating')
#Transformando dados null em 0
book_pivot.fillna(0, inplace=True)

CPU times: user 2.53 s, sys: 224 ms, total: 2.76 s
Wall time: 2.91 s
```

Fonte: Autor

Figura 38: Visualizando a Tabela dinâmica

```
[103] book_pivot.head(4)
```

	user_id	638	643	741	882	929	1211	1435	1674	1733	1848	...
	title											
	A Light in the Storm: The Civil War Diary of Amelia Martin, Fenwick Island, Delaware, 1861 (Dear America)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
	Ask Lily (Young Women of Faith: Lily Series, Book 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
	Deceived	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
	Earth Prayers From around the World: 365 Prayers, Poems, and Invocations for Honoring the Earth	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

4 rows x 1792 columns

Fonte: Autor

4.2.5 Tratamento exclusivo para o segundo problema (PCA e Kmeas)

Vamos criar algumas condições exclusivas para tratar o segundo problema proposto. Lembrando que para o segundo problema iremos criar uma máquina de aprendizagem não-supervisionada utilizando K-means e PCA. A forma para identificar se o usuário gosta do livro é se ele avaliou o mesmo acima da média de suas avaliações.

Iremos agrupar os perfis semelhantes de comportamento de avaliações dos livros, para listar os livros com melhores avaliações dos usuários de acordo com o perfil de cada

usuário. Uma das condições é, se uma pessoa avaliar um livro mais do que sua classificação média, então ela gosta do livro.

4.2.5.1 Média de avaliações do usuário

Conforme Figura 39, estou calculando a média de avaliações por usuário para criar a coluna que conterá a média de avaliações por usuário.

Figura 39: Criando coluna de média de avaliações

```
# Criando média de avaliações por usuário
v_rating_mean = ratings.groupby('user_id')['rating'].mean()
# Alterei o index para mesclar a coluna com o df ratings.
ratings = ratings.set_index('user_id')
# Criando coluna de média de avaliações no df ratings.
ratings['mean_rating'] = v_rating_mean
#Retomado indice padrao
ratings.reset_index(inplace=True)
ratings.head(3)
```

	user_id	ISBN	rating	qtd_ratings	mean_rating
0	276925	002542730X	10	134	7.5
1	276925	0140154078	6	13	7.5
2	276925	0194216748	5	1	7.5

Fonte: Autor

4.2.5.2 Avaliações acima da média

Conforme foi dito, se uma pessoa avaliar um livro mais do que sua classificação média, então ela gosta do livro. Com isso filtrei as avaliações dos livros que possuem o valor acima da média da avaliação do usuário.

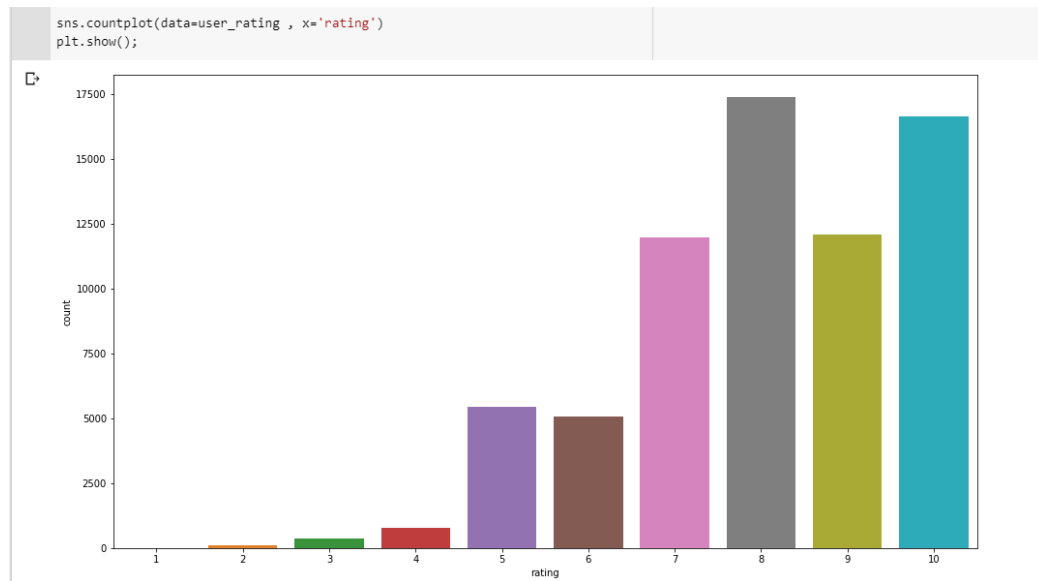
Figura 40: Filtrando avaliações acima da média

```
✓ [202] print("Qtd do Conjunto de dados ratings antes de limpar: ",ratings.shape)
0s ratings = ratings[ratings['rating'] > ratings['mean_rating']]
print("Depois de limpar o conjunto de dados ratings: ",ratings.shape)

Qtd do Conjunto de dados ratings antes de limpar: (193885, 5)
Depois de limpar o conjunto de dados ratings: (69855, 5)
```

Fonte: Autor

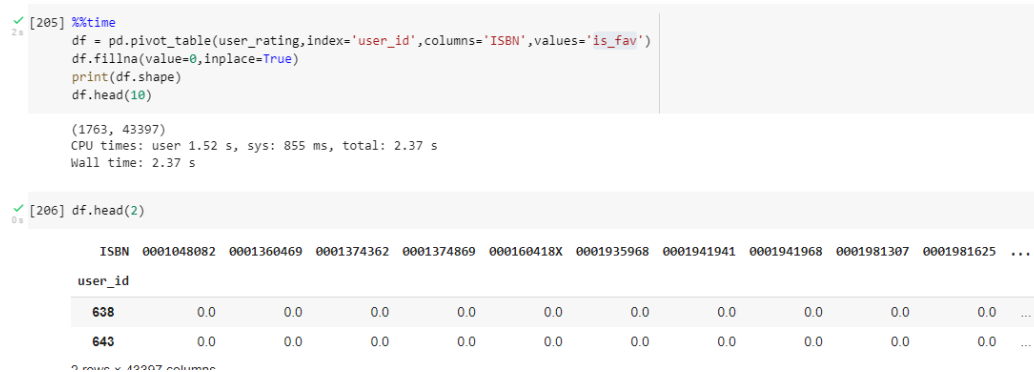
Após filtragem podemos observar a quantidade de avaliações por nota dos livros.

Figura 41: Gráfico de avaliações

Fonte: Autor

4.2.5.3 Tabela dinâmica para o segundo Problema

Para esse problema irei criar uma Tabela dinâmica, porque iremos criar uma máquina utilizando o Kmeans que será agrupado o perfil de avaliações de usuário para trazer os livros mais bem avaliados em um determinado perfil de avaliações de livros.

Figura 42: Criando segunda tabela dinâmica

Fonte: Autor

5. Criação de Modelos de Machine Learning

Antes de implementar o modelo vou falar um pouco de cada algoritmo e método escolhido:

- **NearestNeighbors:**

Para nosso primeiro problema irei utilizar o algoritmo de vizinhos mais próximo para criar uma máquina não supervisionada. Ele atua como uma interface uniforme para três algoritmos de vizinhos mais próximos diferentes: BallTree, KDTree e um algoritmo de brute-force baseado em rotinas em `sklearn.metrics.pairwise`. A escolha do algoritmo de busca de vizinhos é controlada através da palavra-chave 'algorithm'. Quando o valor padrão é passado, o algoritmo tenta determinar a melhor abordagem a partir dos dados.

- **PCA:**

A Análise de Componentes Principais ou PCA (Principal Component Analysis) é uma técnica de análise multivariada que pode ser usada para analisar inter-relações entre muitas variáveis e explicar essas variáveis em termos de suas dimensões inerentes (Componentes).

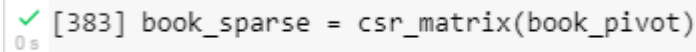
O objetivo é encontrar um meio de condensar a informação contida em várias variáveis originais em um conjunto menor de variáveis estatísticas (componentes) com uma perda mínima de informação. Os componentes principais em geral são extraídos via matriz de covariância, mas também podem ser extraídas via matriz de correlação.

- **K-Means:**

K-Means é um algoritmo de clusterização (ou agrupamento) disponível na biblioteca Scikit-Learn. O K-means é um algoritmo do tipo não supervisionado, ou seja, que não trabalha com dados rotulados. O objetivo desse algoritmo é encontrar similaridades entre os dados e agrupá-los conforme o número de cluster passado pelo argumento `k`. O algoritmo calcula a distância entre dois pontos, utilizando a distância euclidiana.

5.1 Filtragem colaborativa com NearestNeighbors

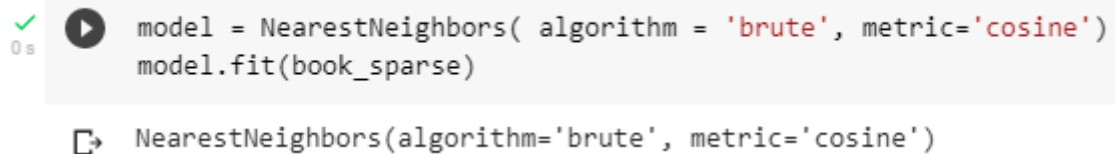
Como foi dito iremos utilizar para nosso primeiro problema o algoritmo de vizinhos mais próximos, essa técnica NearestNeighbors é utilizada para medir a distância. Então irei utilizar o algoritmo para gerar scores para sugestões de livros. Porém na tabela dinâmica, temos muitos valores zero e no agrupamento, esses valores aumentará a distância dos valores zero, então converteremos a tabela dinâmica na matriz esparsa e a alimentaremos no modelo.

Figura 43: Matriz esparsa


```
[383] book_sparse = csr_matrix(book_pivot)
```

Fonte: Autor

Irei treinar o algoritmo dos vizinhos mais próximos, aqui precisamos especificar um algoritmo que é 'brute' significa encontrar a distância de cada ponto a todos os outros pontos. A implementação de pesquisa de vizinhos mais ingênua envolve o cálculo de 'brute force' de distâncias entre todos os pares de pontos no conjunto de dados: por N amostras em D dimensões, esta abordagem escala como $O[DN^2]$. E especificamos a métrica como um cosseno, para que o algoritmo calcule a similaridade do cosseno entre os vetores de classificação.

Figura 44: Treinando Algoritmo


```
model = NearestNeighbors( algorithm = 'brute', metric='cosine')
model.fit(book_sparse)
```

```
NearestNeighbors(algorithm='brute', metric='cosine')
```

Fonte: Autor

5.1.2 Validando sugestões do problema 1

Vamos fazer duas previsões e ver se está sugerindo livros ou não, encontraremos os vizinhos mais próximos do id do livro de entrada e depois disso, imprimiremos os 5 principais livros que estão mais próximos desses livros, faço um loop para pegar os livros mais próximos. Vamos passar Harry Potter and the Goblet of Fire (Book 4) com índice 794.

Figura 45: Realizando previsão

```

08 #Testando nosso modelo de recomendação
09 ##query_index = np.random.choice(book_pivot.shape[0])
distances, indices = model.kneighbors(book_pivot.iloc[794, :].values.reshape(1, -1), n_neighbors = 6)
v_array_livro = []
v_array_dist = []

for i in range(0, len(distances.flatten())):
    if i == 0:
        print('Recomendações para {}: \n'.format(book_pivot.index[794]))
        livro_selecionado = (book_pivot.index[794])
    else:
        v_array_livro.append(book_pivot.index[indices.flatten()[i]])
        v_array_dist.append(distances.flatten()[i])
        print('{}: {}, com distância de {}.'.format(i, book_pivot.index[indices.flatten()[i]], distances.flatten()[i]))

```

Fonte: Autor

Na Figura abaixo mostra os 5 livros mais próximos do escolhido. Lembrando a análise dos resultados acontecerá no tópico 6, iremos também realizar a análise de todos os resultados para avaliarmos se existe coerência.

Figura 46: Resultado da primeira previsão

```

Recomendações para Harry Potter and the Goblet of Fire (Book 4):

1: Harry Potter and the Prisoner of Azkaban (Book 3)
2: Harry Potter and the Chamber of Secrets (Book 2)
3: Harry Potter and the Order of the Phoenix (Book 5)
4: Harry Potter and the Sorcerer's Stone (Book 1)
5: Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))

```

Fonte: Autor

Vamos fazer a segunda previsão e ver se está sugerindo livros ou não, encontraremos os 5 principais livros que estão mais próximos desses livros. Vamos passar Jurassic Park que está em 958 índices.

Figura 47: Realizando segunda previsão

```

08 #Testando nosso modelo de recomendação
09 ##query_index = np.random.choice(book_pivot.shape[0])
distances, indices = model.kneighbors(book_pivot.iloc[958, :].values.reshape(1, -1), n_neighbors = 11)
v_array_livro = []
v_array_dist = []

for i in range(0, len(distances.flatten())):
    if i == 0:
        print('Recomendações para {}: \n'.format(book_pivot.index[958]))
        livro_selecionado = (book_pivot.index[958])
    else:
        v_array_livro.append(book_pivot.index[indices.flatten()[i]])
        v_array_dist.append(distances.flatten()[i])
        print('{}: {}, com distância de {}.'.format(i, book_pivot.index[indices.flatten()[i]], distances.flatten()[i]))

```

Fonte: Autor

Na Figura abaixo mostra os 5 livros mais próximos do livro escolhido nesta previsão.

Figura 48: Resultado da segunda previsão

```

Recomendações para Jurassic Park:

1: Strange Highways
2: The Terminal Man
3: Red Dragon
4: Silence of the Lambs
5: Lord of Chaos (The Wheel of Time, Book 6)
  
```

Fonte: Autor

5.2 Filtragem colaborativa com PCA e K-Means

Para o segundo problema iremos utilizar o PCA, onde o mesmo é uma técnica de aprendizagem não supervisionada utilizada para reduzir a dimensionalidade dos dados em cada livro.

Então iremos reduzir todas nossas variáveis em 3 componentes principais, a transformação é realizada por meio de Matemática(Álgebra) e com isso juntamos as variáveis de acordo sua semelhança entre si, essa semelhança é medida por variância.

Figura 49: Aplicando PCA

```

[ ] # Aplica redução de dimensionalidade das variáveis
%%time
pca = PCA(n_components=3) # 3 componentes
pca.fit(df)
pca_fit = pca.transform(df) # Realiza o fitTransforme com nossa tabela

CPU times: user 11.8 s, sys: 2.13 s, total: 14 s
Wall time: 7.87 s
  
```

Fonte: Autor

Depois de ajustar um objeto PCA à matriz padronizada, podemos ver quanto da variação é explicada por cada uma das características

Figura 50: Visualizando os 3 componentes

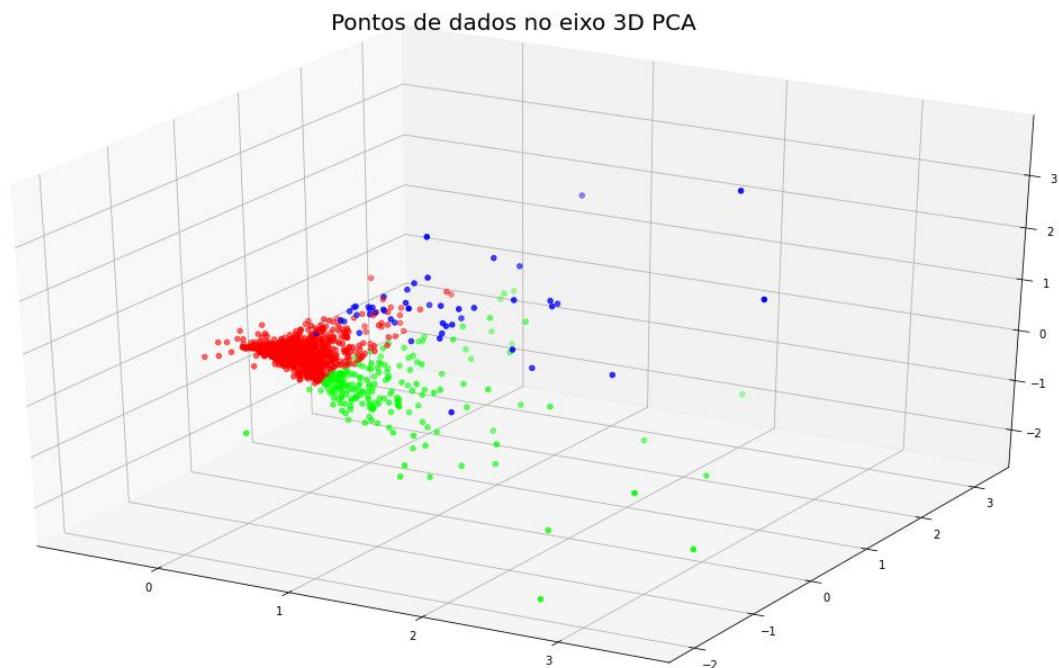

```
[ ] pca_fit = pd.DataFrame(pca_fit, index=df.index)
pca_fit.head(5)
```

	0	1	2
user_id			
638	0.782315	-0.405139	-0.175339
643	-0.271573	0.051084	-0.058399
741	-0.246465	0.030342	-0.061915
882	0.132996	0.015928	-0.059757
929	-0.162943	0.017567	-0.029452

Fonte: Autor

Nessa etapa usei o K-Means para criar os grupos com os nossos dados e dividir os livros. Dei início na criação de 3 cluster para visualizarmos os três componentes principais separados por grupos pelo K-means e ter uma visão dos dados de cada componente.

Figura 51: Criação de cluster para cada componente principal

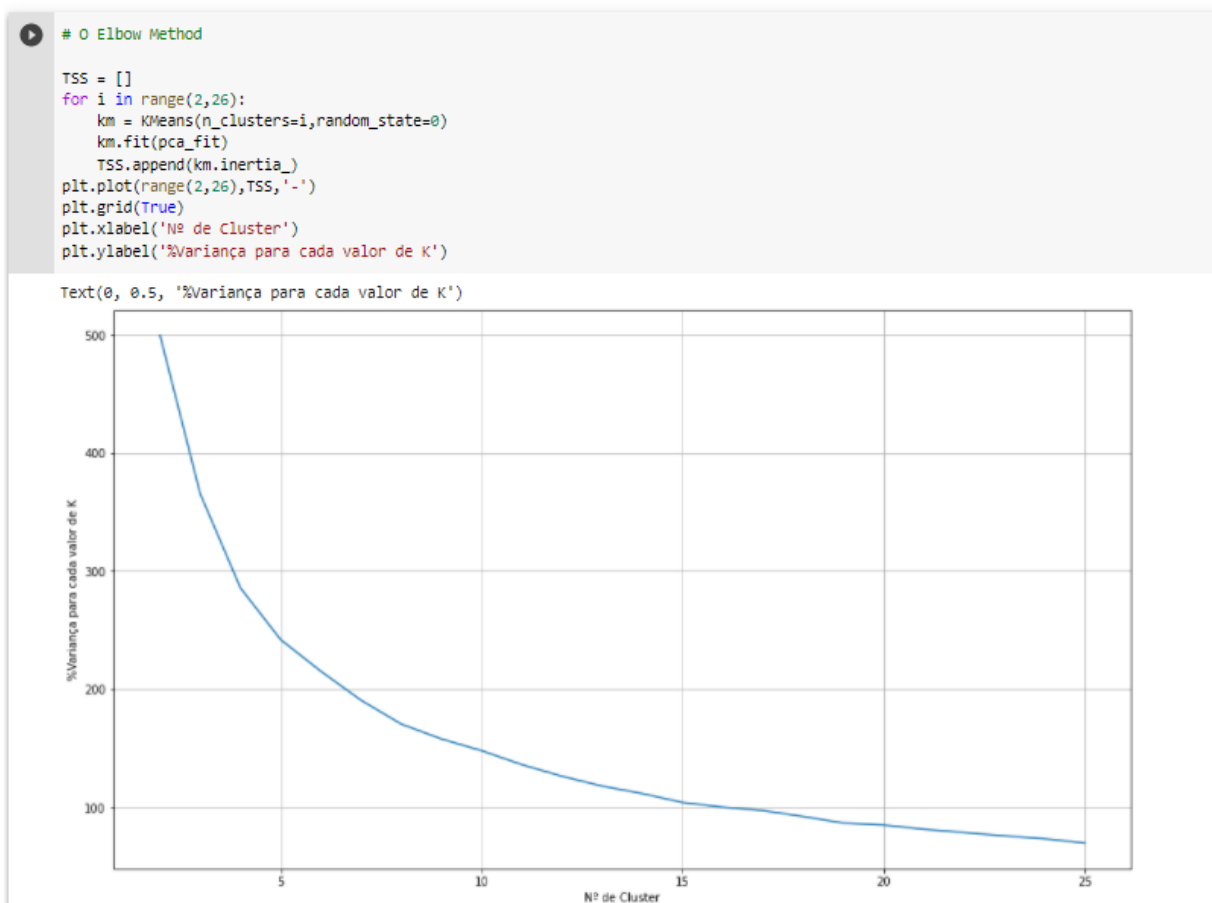


Fonte: Autor

No gráfico acima podemos observar bem cada grupo de cada componente com os dados e sua aproximação. Com isso agora iremos identificar a melhor quantidade de cluster.

Para nosso Insight escolhi uma quantidade de 25 K(Cluster) para testar com o Método de agrupamento Elbow, que irá testar a variância dos dados em relação ao número de clusters. A partir do valor indicado pelo “cotovelo” no gráfico significa que não existe ganho em relação ao aumento de clusters.

Figura 52: Método Elbow



Fonte: Autor

Não consegui um cotovelo claro, então resolvi utilizar o método Silhouette Analysis que faz o cálculo da distância dentro do cluster e a distância entre os clusters, para cada observação. Para isso verifiquei de 3 a 8 clusters, onde no gráfico acima havia mostrado um melhor ganho.

Figura 53: Aplicando método Silhouette

```

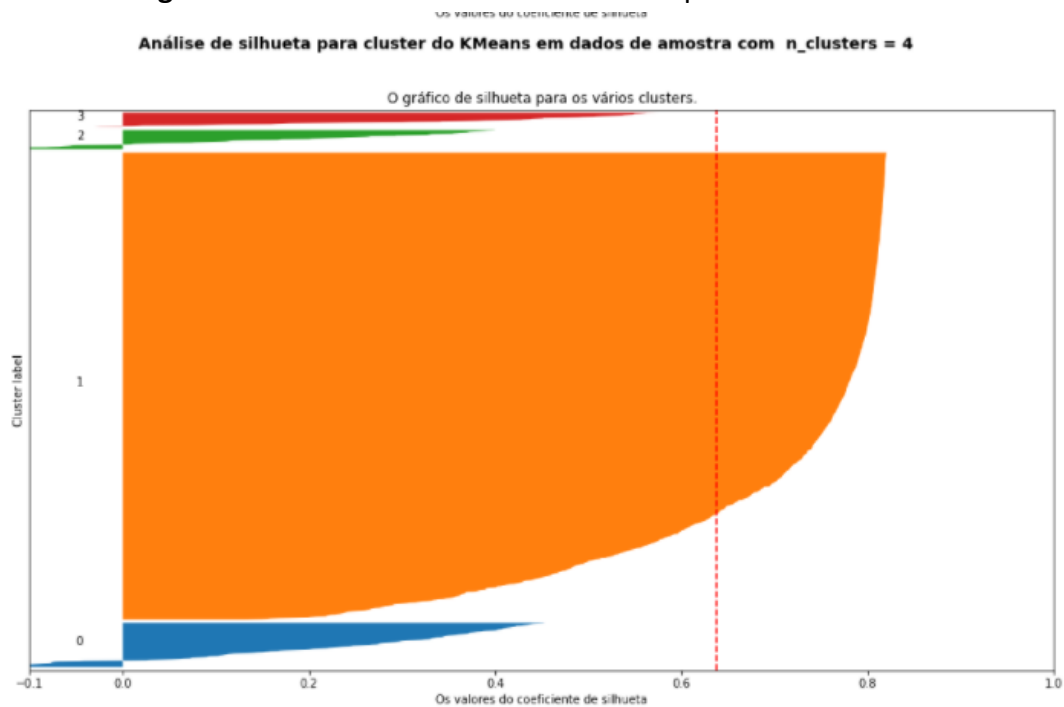
for n in [3,4,5,6,7,8]:
    ax1 = plt.figure().gca()
    ax1.set_xlim([-0.1, 1])
    ax1.set_ylim([0, len(pca_fit) + (n + 1) * 10])
    km = KMeans(n_clusters=n, random_state=0)
    clusters = km.fit_predict(pca_fit)
    silhouette_avg = silhouette_score(pca_fit, clusters)
    print("Para n_clusters =", n,
          "A média silhouette_score é :", silhouette_avg)
    silhouette_values = silhouette_samples(pca_fit, clusters)
    y_start = 10
    for i in range(n):
        ith_cluster = np.sort(silhouette_values[clusters==i])
        cluster_size = ith_cluster.shape[0]
        y_end = y_start + cluster_size
        ax1.fill_betweenx(np.arange(y_start, y_end),
                          0, ith_cluster)
        ax1.text(-0.05, y_start + 0.5 * cluster_size, str(i))
        y_start = y_end + 10
    ax1.set_title("O gráfico de silhueta para os vários clusters.")
    ax1.set_xlabel("Os valores do coeficiente de silhueta")
    ax1.set_ylabel("Cluster label")
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")
    ax1.set_yticks([])
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])
    plt.suptitle("Análise de silhueta para cluster do KMeans em dados de amostra "
                 "com n_clusters = %d" % n),
                 fontsize=14, fontweight='bold')
plt.show()

```

Fonte: Autor

Identificamos que K = 4 fornece o melhor clustering conforme as Figuras abaixo:

Figura 54: Gráficos com a melhor silhueta para cluster do KMeans



Fonte: Autor

Figura 55: Mostrando a melhor média de score da silhueta

```

Para n_clusters = 3 A média silhouette_score é : 0.6212642234280237
Para n_clusters = 4 A média silhouette_score é : 0.6380416619356795
Para n_clusters = 5 A média silhouette_score é : 0.537835975568687
Para n_clusters = 6 A média silhouette_score é : 0.4930894765506903
Para n_clusters = 7 A média silhouette_score é : 0.45087654812522665
Para n_clusters = 8 A média silhouette_score é : 0.4705297231626228

```

Fonte: Autor

Identificado a quantidade de cluster, iremos definir os quatro clusters no KMeans criei um gráfico para mostrar os quatro grupos e obtive os livros e usuários para cada cluster.

Figura 56: Definindo a quantidade de Cluster

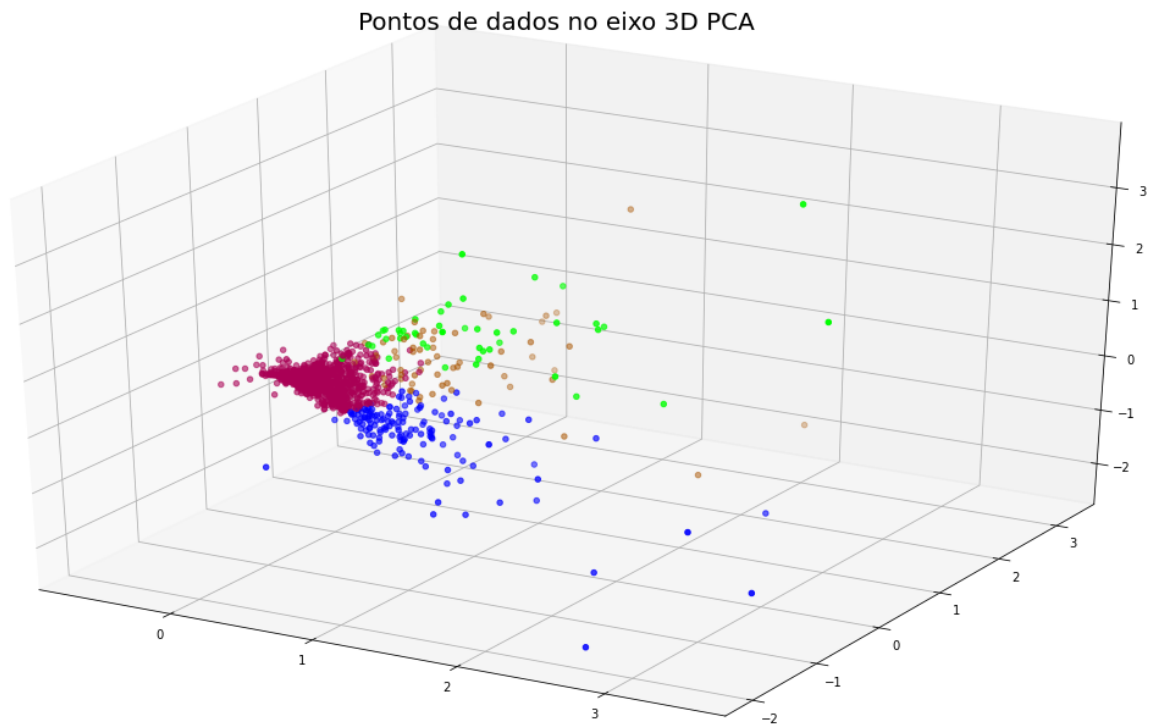
```

Kmeans_final = KMeans(n_clusters=4,random_state=0).fit(pca_fit)
df['cluster'] = Kmeans_final.labels_
fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(pca_fit[0], pca_fit[2], pca_fit[1],c=df['cluster'],cmap=cmhot)
plt.title('Pontos de dados no eixo 3D PCA', fontsize=20)
plt.show()
# Obtendo os livros para cada cluster
cl1_books = df[df.cluster == 0].mean()
cl2_books = df[df.cluster == 1].mean()
cl3_books = df[df.cluster == 2].mean()
cl4_books = df[df.cluster == 3].mean()
# Obtendo os usuarios para cada cluster
cl1_users = df[df.cluster == 0].index
cl2_users = df[df.cluster == 1].index
cl3_users = df[df.cluster == 2].index
cl4_users = df[df.cluster == 3].index

```

Na figura abaixo mostra o gráfico criado com a divisão dos quatro clusters criados.

Figura 57: Definindo a quantidade de Cluster



Fonte: Autor

Vamos fazer quatro previsões uma para cada cluster e ver se está sugerindo livros ou não, imprimiremos os 5 principais livros de cada grupo. E no tópico 6 iremos avaliar o resultado.

5.2.1 Validando sugestões do problema 2

Vamos fazer testes e ver se está sugerindo livros, começando pelo **cluster 1**, encontraremos os 5 principais livros do cluster selecionado. Note que estamos fazendo um loop e retornando o autor, título do livro e a média de publicação dos livros.

Figura 58: Realizando Teste para cluster 1

```

result_isbn = []
def cluster_books_des(Ser):
    bks = pd.DataFrame(Ser).merge(books, left_index=True, right_on='ISBN', how='left')
    bks.rename(columns={0: 'avg_score'}, inplace=True)
    bks.sort_values(by='avg_score', ascending=False, inplace=True)
    print('Media de ano de Publicação:', int(bks['year_publisher'].median()))
    print('\nTop 5 Livros\n')
    Top5_books = bks.index[:5]
    for i, isbn in enumerate(Top5_books):
        print(str(i+1)+'. ', bks.loc[isbn]['title'])
        result_isbn.append(bks.loc[isbn]['ISBN'])
    Top5_authors = bks['author'].unique()[:5]
    print('\n-----\n')
    print('\nTop 5 Autores\n')

    for i, auth in enumerate(Top5_authors):
        if auth is not "nan":
            print(str(i+1)+'. ', auth)
    cluster_books_des(c11_books)

df_result = pd.DataFrame(result_isbn, columns=['ISBN'])

```

Fonte: Autor

Podemos observar que retornou os melhores livros do cluster 1. Lembrando a análise do resultado acontecerá no tópico 6, iremos também realizar a análise de todos os resultados para avaliarmos se existe coerência.

Figura 59: Imprimindo Resultado cluster 1

```

➤ Media de ano de Publicação: 1997

Top 5 Livros

1. Red Dragon
2. Jurassic Park
3. Silence of the Lambs
4. Cat & Mouse (Alex Cross Novels)
5. The Vampire Lestat (Vampire Chronicles, Book II)

-----

Top 5 Autores

1. Thomas Harris
2. Michael Crichton
3. James Patterson
4. ANNE RICE
5. Anne Rice

```

Fonte: Autor

Para os demais clusters foi feito o mesmo teste para validarmos se retorna os melhores livros do cluster 2,3 e 4. Abaixo temos o resultado de cada cluster.

Figura 60: Resultado Cluster 2

```

↳ Média de ano de Publicação: 1997

Top 5 Livros

1. The Lovely Bones: A Novel
2. Where the Heart Is (Oprah's Book Club (Paperback))
3. The Secret Life of Bees
4. The Red Tent (Bestselling Backlist)
5. House of Sand and Fog

-----

Top 5 Autores

1. Alice Sebold
2. Billie Letts
3. Sue Monk Kidd
4. Anita Diamant
5. Andre Dubus III

```

Fonte: Autor

Figura 61: Resultado Cluster 3

```

↳ Média de ano de Publicação: 1997

Top 5 Livros

1. The Da Vinci Code
2. Interview with the Vampire
3. Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))
4. Life of Pi
5. The Lovely Bones: A Novel

-----

Top 5 Autores

1. Dan Brown
2. Anne Rice
3. J. K. Rowling
4. Yann Martel
5. Alice Sebold

```

Fonte: Autor

Figura 62: Resultado Cluster 4

↳ Média de ano de Publicação: 1997

Top 5 Livros

1. Harry Potter and the Prisoner of Azkaban (Book 3)
2. Harry Potter and the Chamber of Secrets (Book 2)
3. Harry Potter and the Sorcerer's Stone (Book 1)
4. Harry Potter and the Order of the Phoenix (Book 5)
5. Harry Potter and the Goblet of Fire (Book 4)

Top 5 Autores

1. J. K. Rowling
2. J.D. Salinger
3. Harper Lee
4. Dan Brown
5. C. S. Lewis

Fonte: Autor

6. Interpretação dos Resultados

Nessa seção você deve interpretar os resultados obtidos dos dados. Para análise das máquinas não supervisionadas identificamos padrões repetidos de avaliações e elegemos os livros por grupo ou por vizinhos mais próximos. Para análise do resultado como não fizemos e não temos testes que foram rotulados, classificados ou categorizados previamente o rótulo. Nossas máquinas reagem com base na presença ou ausência de tais semelhanças em cada novo dado.

O aprendizado de máquina não supervisionado descobre padrões previamente desconhecidos em dados. Como você não sabe quais devem ser os resultados, ainda não há como determinar a precisão deles. Isso ocorre porque, como não há uma variável específica a ser explicada (ou seja, não há um target), então não há sentido em treinar o conjunto de dados, pois também não será possível avaliar a assertividade do modelo. No aprendizado de máquina não supervisionado procuramos encontrar padrões, perfis, itens semelhantes.

Com isso iremos avaliar os dados para analisar se a uma coerência nos dois problemas propostos se existe padrões, perfis ou itens semelhantes para identificar semelhança de gênero, assuntos, validarmos as médias dos livros, quantidades de avaliações e distância dos dados coletados. Também analisaremos se a pessoa avaliou o livro “A” tenha avaliado o livro “B” sugerido.

Para iniciarmos podemos ver que no problema 1 a distância dos dados dos problemas que utilizamos é uma distância relativamente similar próxima do zero.

Figura 63: Distância da primeira previsão da nossa primeira máquina

Fonte: Autor

Figura 64: Distância da segunda previsão da nossa primeira máquina

Fonte: Autor

Conforme figura podemos observar que é até mais fácil de identificar que existe uma semelhança dos livros ao escolher o livro do Harry Potter, o algoritmo me sugeriu outros livros do Harry Potter. Onde as pessoas que avaliaram o livro do Harry Potter também avaliaram os outros livros da saga do tema.

Figura 65: Livros indicados ao escolher Harry Potter

Recomendações para Harry Potter and the Goblet of Fire (Book 4):

- 1: Harry Potter and the Prisoner of Azkaban (Book 3)
- 2: Harry Potter and the Chamber of Secrets (Book 2)
- 3: Harry Potter and the Order of the Phoenix (Book 5)
- 4: Harry Potter and the Sorcerer's Stone (Book 1)
- 5: Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))

Fonte: Autor

Podemos além disso comparar os dois problemas e perceber que existe uma coerência dos dados, por exemplo o cluster 4 pertence ao grupo que gostou dos livros do Harry.

Figura 66: Validando coerência dos dois resultados dos dois problemas

Recomendações para Harry Potter and the Goblet of Fire (Book 4):

- 1: Harry Potter and the Prisoner of Azkaban (Book 3)
- 2: Harry Potter and the Chamber of Secrets (Book 2)
- 3: Harry Potter and the Order of the Phoenix (Book 5)
- 4: Harry Potter and the Sorcerer's Stone (Book 1)
- 5: Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))

Fonte: Autor

Além disso para a primeira máquina onde nos sugere os livros avaliados pelos usuários que avaliou o mesmo livro que escolhi, investiguei os dados para validar se realmente os usuários que avaliou os livros sugeridos avaliou também o livro escolhido.

Tivemos êxito em todos os testes e peguei um exemplo feito nos testes escolhendo o livro “Jurassic Park”, resultado da nossa segunda previsão, foi a sugestão do livro “Strange Highways”. Obtive no exemplo abaixo o usuário 91633 avaliando os dois livros.

Figura 67: Validando se usuário 91633 avaliou Jurassic Park

estes recursos, salve-a em um formato de arquivo do Excel.

A46	:	X	✓	fx	13107,0345370775,91633,0,361,Jurassic Park,Ballantine Books,1999							
	A	B	C	D	E	F	G	H	I	J	K	
44	13105,0345370775,86242,0,361,Jurassic Park,Ballantine Books,1999											
45	13106,0345370775,87712,0,361,Jurassic Park,Ballantine Books,1999											
46	13107,0345370775,91633,0,361,Jurassic Park,Ballantine Books,1999											
47	13108,0345370775,94951,0,361,Jurassic Park,Ballantine Books,1999											

Fonte: Autor

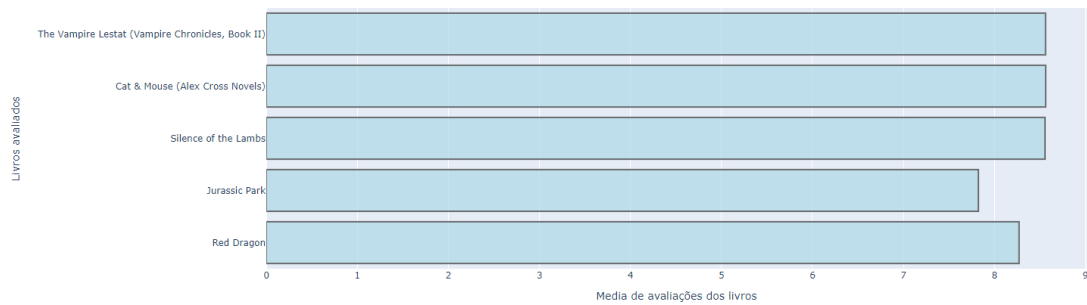
Figura 68: Realmente o usuário 91633 avaliou também Strange Highways

A5	:	X	✓	fx	42900,0446603392,91633,0,41,Strange Highways,Warner Books,1996							
	A	B	C	D	E	F	G	H	I			
1	,ISBN,user_id,rating,qtd_ratings,title,publisher,year_publisher											
2	42897,0446603392,8362,0,41,Strange Highways,Warner Books,1996											
3	42898,0446603392,8487,0,41,Strange Highways,Warner Books,1996											
4	42899,0446603392,41084,8,41,Strange Highways,Warner Books,1996											
5	42900,0446603392,91633,0,41,Strange Highways,Warner Books,1996											
6	42901,0446603392,105054,0,41,Strange Highways,Warner Books,1996											

Fonte: Autor

Analisei também as médias de livros de cada cluster para validar se realmente foi indicado os livros com uma avaliação relevante. Podemos notar que temos uma média de avaliações relevante em cada cluster.

Figura 69: Média Cluster 1



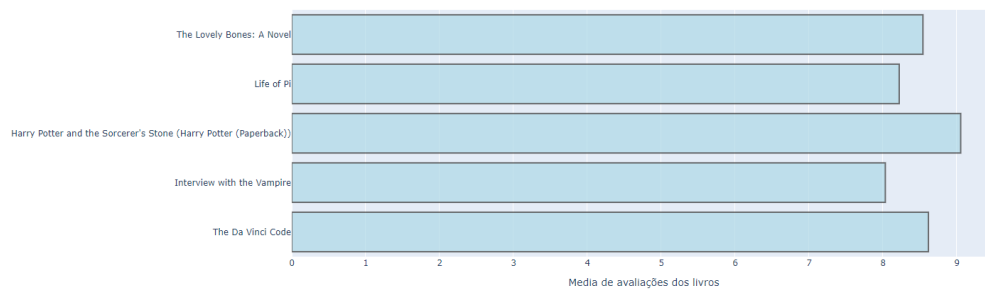
Fonte: Autor

Figura 70: Média Cluster 2



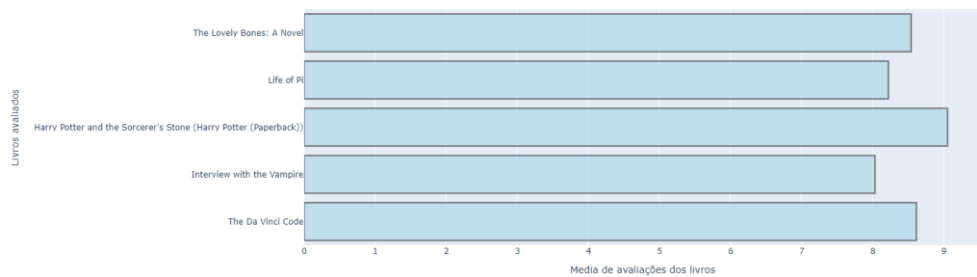
Fonte: Autor

Figura 71: Média Cluster 3



Fonte: Autor

Figura 72: Média Cluster 4



Fonte: Autor

Realizei análise para validar sobre os gêneros e podemos observar que existe uma semelhança nos livros e descrição, onde são livros que tenham semelhança no perfil de gosto por gênero separados em cada cluster. Por exemplo no cluster 1 podemos observar que existe opção para livros de Ficção científica e suspense.

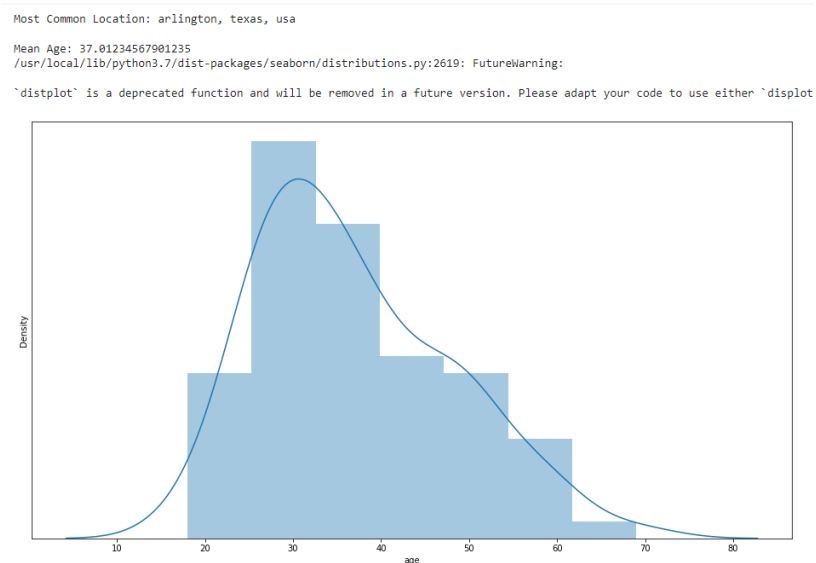
Figura 73: Validando gêneros

ISBN	title	author	year	publisher	language	genres
039912442X	Red Dragon	Thomas Harris	1981	Putnam Pub Group	eng	['Fiction', 'Thriller']
06794306	Mouse (Alex Cross Novels)	James Patterson	1998	Warner Books	eng	['Fiction', 'Thriller']
0446606189	Mouse (Alex Cross Novels)	James Patterson	1998	Warner Books	eng	['Fiction', 'Thriller']
0312924585	Silence of the Lambs	Thomas Harris	1991	St. Martin's Press	eng	['Thriller']
0345313860	The Vampire Lestat (Vampire Chronicles, Book II)	ANNE RICE	1986	Ballantine Books	eng	['Thriller']

Fonte: Autor

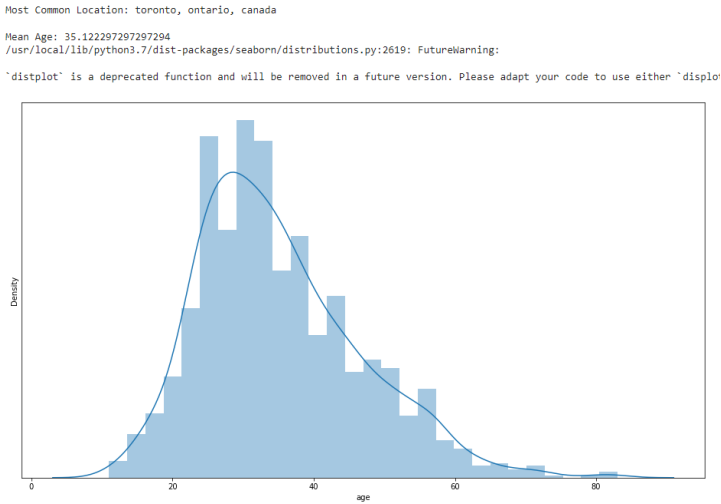
Realizado análise com as idades dos usuários de cada cluster para verificar se a alguma anomalia nas idades e se existe um padrão adequado, além disso validei as localidades mais comuns dos usuários de cada cluster, para validarmos os principais locais de cada cluster, e podemos observar que existem preferências identificados por idade e região.

Figura 74: Média de idade dos usuários e localidades cluster 1



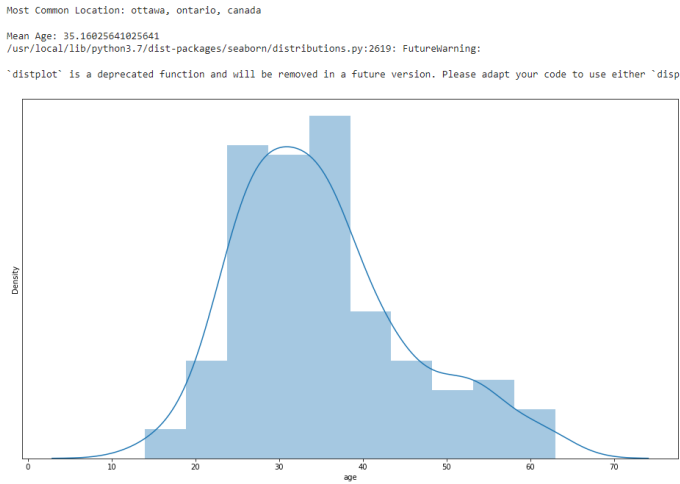
Fonte: Autor

Figura 75: Média de idade dos usuários e localidades cluster 2



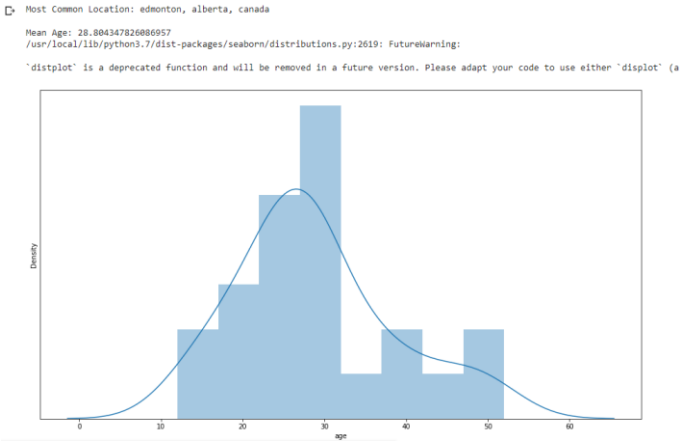
Fonte: Autor

Figura 76: Média de idade dos usuários e localidades cluster 3



Fonte: Autor

Figura 77: Média de idade dos usuários e localidades cluster 4



Fonte: Autor

Conclusão foi que os dois problemas propostos foram atendidos conforme proposta e foi possível identificar padrões nos dados e até mesmo itens semelhantes em nossas recomendações, obtemos comparações claras de que as duas máquinas não supervisionadas possuem padrões em recomendações que elegem livros com critérios relevantes.

7. Apresentação dos Resultados

Nessa seção você deve apresentar os resultados obtidos.

Problema	Resultado e previsões	Aquisição de dados
A proposta e atender dois requisitos o 1º problema com base nas escolhas de leitura de outras pessoas, o livro seja recomendado a outras pessoas com interesse semelhante. E o 2º problema além disso, iremos agrupar os perfis semelhantes de comportamento de avaliações dos livros, para recomendar os livros por perfil.	Avaliar os atributos de avaliações com a finalidade de tentar sugerir os livros com as melhores avaliações e com grande índice de avaliação.	Os dados do dataset books-data foi coletado do site-Kaggle e os demais conjuntos de dados foram coletados na comunidade Book-Crossing.
Modelagem	Avaliação do Modelo	Preparação dos dados
Realizados análises nos datasets coletados, tanto forma gráfica quanto análise descritiva dos dados, utilizando a biblioteca Pandas em Python. Desta forma foi possível identificar um dataset adequado para aplicar o modelo de classificação de ML	Para nossos modelos a forma que encontrei para avaliá-los foi procurar encontrar padrões, perfis, itens semelhantes.	Após a união dos datasets, os dados foram tratados, as colunas foram renomeadas, os dados duplicados foram removidos e dados desnecessários para a análise também foram removidos.

8. Links

Todos os códigos desenvolvidos e a documentação utilizada são disponibilizadas no repositório do GitHub.

Link para o vídeo: <https://www.youtube.com/watch?v=ugUZBTN-mAw>

Link para o repositório https://github.com/Tarcisioms23/TCC_RECOMENDAR_LIVROS

REFERÊNCIAS

Machine Learning — O que é, tipos de aprendizagem de máquina, algoritmos e aplicações.

Disponível em:

<https://medium.com/camilawaltrick/introducao-machine-learning-o-que-e-tipos-de-aprendizado-de-maquina-445dcfb708f0>

Acesso em: 01/03/2021.

MEDIUM. Por que usar Jupyter Notebook?

Disponível em:

<https://suzana-svm.medium.com/por-que-usar-jupyter-notebook-77d5a59b42a1>

Acesso em: 12/03/2022.

TRANSFORMAÇÃO DIGITAL. Quais as vantagens do big data em vendas?

Disponível em:

<http://www.mma.gov.br/sitio/index.php?ido=conteudo.monta&idEstrutura=18>

Acesso em: 12/02/2022.

PCA na mão e no Python.

Disponível em:

<https://leandrocruvinel.medium.com/pca-na-m%C3%A3o-e-no-python-d559e9c8f053>

Acesso em: 12/02/2022.

K-means: o que é, como funciona, aplicações e exemplo em Python,

Disponível em:

<https://medium.com/programadores-ajudando-programadores/k-means-o-que-%C3%A9-como-funciona-aplica%C3%A7%C3%B5es-e-exemplo-em-python-6021df6e2572>

Acesso em: 10/02/2022.

Nearest Neighbor Method.

Disponível em:

<https://www.sciencedirect.com/topics/mathematics/nearest-neighbor-method>

Acesso em: 18/01/2022.

Aprendizado de máquina: supervisionado e não supervisionado

Disponível em:

<https://cienciaenegocios.com/aprendizado-de-maquina-supervisionado-e-nao-supervisionado/>

Acesso em: 18/01/2022.