Relatório de Projeto – Mineração de Dados

Estudantes: Maria Eduarda Maciel e Renata Cavalheiro

1. Escolha da base de dados e definição do problema

A primeira etapa do projeto foi selecionar, no Kaggle, a base de dados que seria analisada e definir qual o problema, ou seja, o que gostaríamos de extrair a partir dos dados presentes na base. Inicialmente, a base escolhida foi a base Kepler, que contém dados sobre objetos do espaço que tem características compatíveis com as de um possível exoplaneta. O objetivo era extrair quais eram os dados mais relevantes dos objetos que foram confirmados exoplanetas para facilitar o processo de detecção. No entanto, para realizar essa análise, seriam necessários dados que refletem às características físicas dos exoplanetas, porém na base de amostra fornecida pelo Kaggle, esses dados não estavam presentes, portanto não seria possível dar continuidade à ideia inicial. Considerando isso, decidimos selecionar outra base e outro objetivo.

A outra base escolhida, também retirada do Kaggle, contém dados de usuários e os filmes que assistiram. Na base estão presentes informações de mais de 7.000 usuários e seus históricos de filmes assistidos, considerando mais de 200.000 títulos diferentes, pertencentes a diferentes gêneros cinematográficos e as notas dadas pelos usuários aos títulos. Considerando o grande crescimento de serviços de streaming de vídeos nos últimos anos e a aparição de algoritmos de recomendação de filmes, que muitas vezes não possuem um desempenho satisfatório, decidimos executar um algoritmo sobre a base para extrair regras de associação entre os filmes assistidos pelos usuários.

2. Pré-processamento e Algoritmo Apriori

Considerando o tempo de execução e o tempo máximo permitido pela ferramenta do Kaggle, reduzimos a base, utilizando os dados de 5.000 usuários em vez de 7.000. Além disso, algumas colunas foram removidas, como as notas de classificação dos usuários para cada filme, já que não seriam relevantes para a extração das regras de associação.

O algoritmo aplicado na base de dados para extrair as regras foi o Apriori. O algoritmo Apriori funciona de forma que busca todos os conjuntos de itens que aparecem com grande frequência dentro de uma base de dados. Além disso, ele utiliza duas funções para filtrar os itens que podem ser relevantes para as regras e excluir os que não são. No caso deste projeto, o algoritmo foi implementado em Python, com a utilização da biblioteca Apyori, que se encontra dento do Python Package Index.

Os passos para o desenvolvimento do código foram os seguintes:

1. Criação de uma lista para adicionar todos os filmes assistidos por cada usuário;
2. Iteração na base de dados de zero até 5.000, adicionando os filmes de acordo com o id do usuário;
3. Uma lista para adicionar cada lista de filmes de cada usuário;

4. Transformação dessa lista de listas em uma tabela de dados em que a primeira coluna corresponde ao id do usuário e as colunas subsequentes contém cada um título de um filme.

Após a transformação do data frame criado para representar a base de dados, o algoritmo Apriori foi executado, de forma a iterar cada linha da tabela e aplicar a lógica de funcionamento do algoritmo. O algoritmo Apriori utiliza como parâmetro as seguintes informações:

- min_support: 0,1; o que corresponde à análise de 500 usuários que assistiram determinado filme dentre os 5.000, chamado de suporte (ou seja, 500 5000 = 0,1);
- min_confidence: 0,4; que é a medida da proporção dos usuários que assistiram ao filme X e que também assistiram ao filme Y. A confiança entre dois itens (filme X e filme Y) é definida como o número total de usuários que assistiram tanto o filme X quanto o Y, sobre o número total de usuários que assistiram X. Isto é: número de usuários que assistiram X e Y sobre o número de usuários que assistiram X;
- min_lift: 2; que é a relação entre a confiança e o suporte, dado por: confiança (X →Y) sobre suporte(Y);
- min_length: 10; que é o número mínimo de filmes que devem estar na lista de filmes assistidos de cada usuário.

Os valores de cada parâmetro foram definidos de maneira que garantisse que a confiança fosse no mínimo de 40%, dessa forma é possível saber que considerando o total de pessoas que assistiram ao filme X, 40% desse total também assistiu o filme Y. Além disso, definimos esses valores para que fossem considerados pelo menos 10 filmes, já que com um número menor, as principais regras trariam como associação filmes que são feitos em sequência, como por exemplos Star Wars e Senhor dos Anéis, que possuem mais de um filme.

3. Regras de associação resultantes

Ao final da execução do algoritmo Apriori, ele retorna as regras de associação que foram extraídas da base. Cada regra retornada é exibida da seguinte forma: nome dos dois filmes que foram associados, valor de suporte, relação ordenada dos títulos (por exemplo, se X → Y ou Y → X, valor de confiança e valor de lift. Para algumas regras, ambas as relações X → Y e Y → X existem, portanto são mostradas na mesma regra (no entanto, os valores de confiança podem variar dependendo da ordem). No total, foram extraídas mais de 4.000 regras, a seguir são apresentadas as 10 primeiras e mais relevantes:

```
Association 1 is: RelationRecord(items=frozenset({'2001: A Space Odyssey (196
8)', 'Alien (1979)'}), support=0.10702140428085617, ordered_statistics=[OrderedS
tatistic(items_base=frozenset({'2001: A Space Odyssey (1968)'}), items_add=froze
nset({'Alien (1979)'}), confidence=0.6430288461538463, lift=2.9654070128441674),
OrderedStatistic(items_base=frozenset({'Alien (1979)'}), items_add=frozenset({'2
001: A Space Odyssey (1968)'}), confidence=0.4935424354243543, lift=2.9654070128
441674)])
------------------------
Association 2 is: RelationRecord(items=frozenset({'2001: A Space Odyssey (196
8)', 'Back to the Future (1985)'}), support=0.11002200440088018, ordered_statist
ics=[OrderedStatistic(items_base=frozenset({'2001: A Space Odyssey (1968)'}), it
ems_add=frozenset({'Back to the Future (1985)'}), confidence=0.6610576923076924,
lift=2.2480458529565674)])
------------------------
Association 3 is: RelationRecord(items=frozenset({'2001: A Space Odyssey (196
8)', 'Blade Runner (1982)'}), support=0.11302260452090418, ordered_statistics=[O
rderedStatistic(items_base=frozenset({'2001: A Space Odyssey (1968)'}), items_ad
d=frozenset({'Blade Runner (1982)'}), confidence=0.6790865384615385, lift=3.0945
794036182597), OrderedStatistic(items_base=frozenset({'Blade Runner (1982)'}), i
tems_add=frozenset({'2001: A Space Odyssey (1968)'}), confidence=0.5150410209662
716, lift=3.0945794036182592)])
------------------------
Association 4 is: RelationRecord(items=frozenset({'2001: A Space Odyssey (196
8)', 'Godfather, The (1972)'}), support=0.10802160432086418, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'2001: A Space Odyssey (1968)'}), items_
add=frozenset({'Godfather, The (1972)'}), confidence=0.6490384615384616, lift=2.
1054790845105575)])
```

```
Association 5 is: RelationRecord(items=frozenset({'2001: A Space Odyssey (196
8)', 'Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1
981)'}), support=0.11382276455291059, ordered_statistics=[OrderedStatistic(items
_base=frozenset({'2001: A Space Odyssey (1968)'}), items_add=frozenset({'Raiders
of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)'}), confi
dence=0.6838942307692308, lift=2.06573248315129)])
------------------------
Association 6 is: RelationRecord(items=frozenset({'2001: A Space Odyssey (196
8)', 'Star Wars: Episode V - The Empire Strikes Back (1980)'}), support=0.121824
3648729746, ordered_statistics=[OrderedStatistic(items_base=frozenset({'2001: A
Space Odyssey (1968)'}), items_add=frozenset({'Star Wars: Episode V - The Empire
Strikes Back (1980)'}), confidence=0.731971153846154, lift=2.0957180974094634)])
------------------------
Association 7 is: RelationRecord(items=frozenset({'2001: A Space Odyssey (196
8)', 'Star Wars: Episode VI - Return of the Jedi (1983)'}), support=0.1122224444
8889777, ordered_statistics=[OrderedStatistic(items_base=frozenset({'2001: A Spa
ce Odyssey (1968)'}), items_add=frozenset({'Star Wars: Episode VI - Return of th
e Jedi (1983)'}), confidence=0.6742788461538461, lift=2.0453397766523524)])
------------------------
Association 8 is: RelationRecord(items=frozenset({'2001: A Space Odyssey (196
8)', 'Terminator, The (1984)'}), support=0.10122024404880976, ordered_statistics
=[OrderedStatistic(items_base=frozenset({'2001: A Space Odyssey (1968)'}), items
_add=frozenset({'Terminator, The (1984)'}), confidence=0.608173076923077, lift=
2.673928945944118), OrderedStatistic(items_base=frozenset({'Terminator, The (198
4)'}), items_add=frozenset({'2001: A Space Odyssey (1968)'}), confidence=0.44503
07827616535, lift=2.673928945944118)])
```

```
Association 9 is: RelationRecord(items=frozenset({'Ace Ventura: Pet Detective (1
994)', 'Ace Ventura: When Nature Calls (1995)'}), support=0.10762152430486097, o
rdered_statistics=[OrderedStatistic(items_base=frozenset({'Ace Ventura: Pet Dete
ctive (1994)'}), items_add=frozenset({'Ace Ventura: When Nature Calls (1995)'}),
confidence=0.4487072560467055, lift=3.414136336343198), OrderedStatistic(items_b
ase=frozenset({'Ace Ventura: When Nature Calls (1995)'}), items_add=frozenset
({'Ace Ventura: Pet Detective (1994)'}), confidence=0.8188736681887366, lift=3.4
14136336343198)])
------------------------
Association 10 is: RelationRecord(items=frozenset({'Aladdin (1992)', 'Ace Ventur
a: Pet Detective (1994)'}), support=0.15043008601720345, ordered_statistics=[Ord
eredStatistic(items_base=frozenset({'Ace Ventura: Pet Detective (1994)'}), items
_add=frozenset({'Aladdin (1992)'}), confidence=0.6271893244370309, lift=2.324180
4543074256), OrderedStatistic(items_base=frozenset({'Aladdin (1992)'}), items_ad
d=frozenset({'Ace Ventura: Pet Detective (1994)'}), confidence=0.557449962935507
7, lift=2.3241804543074256)])
```

Abaixo, as mesmas regras representadas em formato de tabela para facilitar a exibição:

Regra	Suporte	Antecedente	Consequente	Confiança	Lift
Regra de associação 1	0.1070	2001: A Space Odyssey (1968)	Alien (1979)	0.6430	2.9654
		Alien (1979)	2001: A Space Odyssey (1968)	0.4935	2.9654
Regra de associação 2	0.1100	2001: A Space Odyssey (1968)	Back to the Future (1985)	0.6610	2.2480
	0.1130	2001: A Space Odyssey (1968)	Blade Runner (1982)	0.6790	3.0945
Regra de associação 3		Blade Runner (1982)	2001: A Space Odyssey (1968)	0.5150	3.0945
Regra de associação 4	0.1080	2001: A Space Odyssey (1968)	Godfather, The (1972)	0.6490	2.1054
Regra de associação 5	0.1138	2001: A Space Odyssey (1968)	Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	0.6838	2.0657

Regra de associação 6	0.1218	2001: A Space Odyssey (1968)	Star Wars: Episode V - The Empire Strikes Back (1980)	0.7319	2.0957
Regra de associação 7	0.1122	2001: A Space Odyssey (1968)	Star Wars: Episode VI - Return of the Jedi (1983)	0.6742	2.0453
Regra de associação 8	0.1012	2001: A Space Odyssey (1968)	Terminator, The (1984)	0.6081	2.6739
		Terminator, The (1984)	2001: A Space Odyssey (1968)	0.4450	2.6739
Regra de associação 9	0.1076	Ace Ventura: Pet Detective (1994)	Ace Ventura: When Nature Calls (1995)	0.4487	3.4141
		Ace Ventura: When Nature Calls (1995))	Ace Ventura: Pet Detective (1994)	0.8188	3.4141
Regra de associação 10	0.1504	Aladdin (1992)	Ace Ventura: Pet Detective (1994)	0.6271	2.3241
		Ace Ventura: Pet Detective (1994)	Aladdin (1992)	0.5574	2.3241

4. Conclusões

Considerando as regras extraídas a partir da base de filmes, é possível perceber a existência de padrões relevantes no consumo de filmes por parte dos usuários, ou seja, muitos dos que consumiram um filme X também consumiram Y e isso se repete de maneira frequente. Além disso, observando a necessidade de melhorias nos algoritmos de recomendação de filmes em serviços de streaming, mostra-se relevante o desenvolvimento de mais estudos para aperfeiçoar os sistemas de recomendação. A aplicação dessas técnicas é útil tanto para usuários quanto para as plataformas, já que dessa forma os usuários conseguirão encontrar mais filmes que lhe agradem e consequentemente, continuarão assinando os serviços.

5. Referências:

Hipp, J., Güntzer, U. e Nakhaeizadeh, G. (2000) "Algorithms for Association Rule Mining – A General Survey and Comparison." ACM SIGKDD Explorations Newsletter, 58–64.

Romão, W., Niederauer, C., Martins, A., Tcholakian, A., Pacheco, R. e Barcia, R. (1999) "Extração De Regras De Associação em C&t: O Algoritmo Apriori". Programa de Pós-

Graduação em Engenharia de Produção - PPGEP Universidade Federal de Santa Catarina, Centro Tecnológico, Florianópolis, Brasil.

Phorasim, P. e Yu, L. (2017) “Movies recommendation system using collaborative filtering and k-means". International Journal of Advanced Computer Research, Vol 7(29).

Bourreau, M. e Gaudin, G. (2018) “Streaming Platform and Strategic Recommendation Bias”. CESifo Working Paper No. 7390, Munique, Alemanha.

6. Anexos

Código desenvolvido durante o trabalho, disponível na plataforma *Kaggle:*

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
import matplotlib.pyplot as plt

filename = '../input/large-movie-dataset/movies_dataset.csv'

netflix = pd.read_csv(filename, na_filter=False)
columns = netflix.columns.tolist()
print("Colunas: ", columns)

newNetflix = netflix.drop(netflix[netflix.User_Id > 5000].index)
print(newNetflix)
```

```
newNetflix.groupby(['User_Id']).size()
```

```
numberOfViewers = len(newNetflix.groupby('User_Id').nunique())
print(numberOfViewers)

allMovies = []
for i in range(1, numberOfViewers+1):
    df_new = newNetflix.query('User_Id==@i')
    movies = []
    for movie in df_new['Movie_Name']:
        movies.append(movie)
    allMovies.append(movies)

df = pd.DataFrame(allMovies)
df.fillna(0,inplace=True)
df.head(20)

pip install apyori

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from apyori import apriori

data_list = []
```

```
for row in range(0, 4999):
    data_list.append([str(df.values[row,column]) for column in range(0, 4
226)])
algo = apriori(data_list, min_support=0.1, min_confidence=0.4, min_lift=2
, min_length=10)
results = list(algo)

print(len(results))
for i in range(0,13):
    print(f"Association {i+1} is: {results[i]}")
    print('-'*25)
```