Week 10

- Joshua Burden
- DSC630 Predictive Analytics
- Bellevue University
- Andrew Hua
- 11/06/2022

- https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109
- https://analyticsindiamag.com/how-to-build-your-first-recommender-system-using-python-movielens-dataset/

In [ ]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
#Get the datasets
tags_df = pd.read_csv('./ml-latest-small/tags.csv')
links_df = pd.read_csv('./ml-latest-small/links.csv')
movies_df = pd.read_csv('./ml-latest-small/movies.csv')
ratings_df = pd.read_csv('./ml-latest-small/ratings.csv')
```

Time to do some preprocesssing of the data

In [ ]:
```
movies_df.head()
```

Out[ ]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [ ]:
```
# Check unique movie title
movies_df["title"].unique()
```

Out[ ]:
```
array(['Toy Story (1995)', 'Jumanji (1995)', 'Grumpier Old Men (1995)',
       ..., 'Flint (2017)', 'Bungo Stray Dogs: Dead Apple (2018)',
       'Andrew Dice Clay: Dice Rules (1991)'], dtype=object)
```

Remove whitespaces from year

In [ ]:
```
# create column name as year from title
```

```
movies_df["Year"] = movies_df.title.str.extract("(\(\d\d\d\d\))" ,expand = True)
# Removing extra brackets
movies_df["Year"] = movies_df.title.str.extract("(\d\d\d\d)" ,expand = True)
# replace year and add whitespaces
movies_df["title"]= movies_df.title.str.replace("(\(\d\d\d\d\))","")
```

```
C:\Users\Joshu\AppData\Local\Temp\ipykernel_15340\3022570178.py:6: FutureWarning: The
default value of regex will change from True to False in a future version.
  movies_df["title"]= movies_df.title.str.replace("(\(\d\d\d\d\))","")
```

In [ ]:
```
movies_df["title"]= movies_df["title"].apply(lambda x: x.strip())
movies_df.head(3)
```

Out[ ]:

	movieId	title	genres	Year
0	1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy	1995
1	2	Jumanji	Adventure|Children|Fantasy	1995
2	3	Grumpier Old Men	Comedy|Romance	1995

In [ ]:
```
movies_df["genres"] = movies_df["genres"].apply(lambda x:x.lower())
movies_df
```

Out[ ]:

	movieId	title	genres	Year
0	1	Toy Story	adventure|animation|children|comedy|fantasy	1995
1	2	Jumanji	adventure|children|fantasy	1995
2	3	Grumpier Old Men	comedy|romance	1995
3	4	Waiting to Exhale	comedy|drama|romance	1995
4	5	Father of the Bride Part II	comedy	1995
...	...	...	...	...
9737	193581	Black Butler: Book of the Atlantic	action|animation|comedy|fantasy	2017
9738	193583	No Game No Life: Zero	animation|comedy|fantasy	2017
9739	193585	Flint	drama	2017
9740	193587	Bungo Stray Dogs: Dead Apple	action|animation	2018
9741	193609	Andrew Dice Clay: Dice Rules	comedy	1991

9742 rows × 4 columns

In [ ]:
```
# merge  movie and tag file
df = pd.merge(movies_df,tags_df,on="movieId",how="left")
df.head(3)
```

Out[ ]:

	movieId	title	genres	Year	userId	tag	timestamp
0	1	Toy Story	adventure|animation|children|comedy|fantasy	1995	336.0	pixar	1.139046e+09
1	1	Toy Story	adventure|animation|children|comedy|fantasy	1995	474.0	pixar	1.137207e+09
2	1	Toy Story	adventure|animation|children|comedy|fantasy	1995	567.0	fun	1.525286e+09

In [ ]:
```
# create metadata by adding genres and tag
df.fillna("", inplace =True)
df = pd.DataFrame(df.groupby("movieId")["tag"].apply(lambda x: "%s" % " ".join(x)))
df
```

Out[ ]:

movieId	tag
1	pixar pixar fun
2	fantasy magic board game Robin Williams game
3	moldy old
4	
5	pregnancy remake
...	...
193581	
193583	
193585	
193587	
193609	

9742 rows × 1 columns

In [ ]:
```
#merge movie and df dataset
new_df = pd.merge(movies_df,df ,on = "movieId" ,how="left")
new_df["metadata"] = new_df[["tag","genres"]].apply(lambda x: " ".join(x),axis=1)
new_df["metadata"]
```

Out[ ]:
```
0       pixar pixar fun adventure|animation|children|c...
1       fantasy magic board game Robin Williams game a...
2                                 moldy old comedy|romance
3                                     comedy|drama|romance
4                                  pregnancy remake comedy
                              ...
9737                       action|animation|comedy|fantasy
9738                              animation|comedy|fantasy
9739                                                 drama
9740                                      action|animation
9741                                                comedy
Name: metadata, Length: 9742, dtype: object
```

In [ ]:
```
# store movieId title metadata year in new_df
new_df = new_df[["movieId","title","metadata","Year"]]
new_df
```

Out[ ]:

	movieId	title	metadata	Year
0	1	Toy Story	pixar pixar fun adventure|animation|children|c...	1995
1	2	Jumanji	fantasy magic board game Robin Williams game a...	1995
2	3	Grumpier Old Men	moldy old comedy|romance	1995
3	4	Waiting to Exhale	comedy|drama|romance	1995
4	5	Father of the Bride Part II	pregnancy remake comedy	1995
...	...	...	...	...
9737	193581	Black Butler: Book of the Atlantic	action|animation|comedy|fantasy	2017
9738	193583	No Game No Life: Zero	animation|comedy|fantasy	2017
9739	193585	Flint	drama	2017
9740	193587	Bungo Stray Dogs: Dead Apple	action|animation	2018
9741	193609	Andrew Dice Clay: Dice Rules	comedy	1991

9742 rows × 4 columns

In [ ]:
```
# split |  from metadata
new_df["metadata"]=new_df.metadata.str.split("|")
new_df
```

Out[ ]:

	movieId	title	metadata	Year
0	1	Toy Story	[pixar pixar fun adventure, animation, childre...	1995
1	2	Jumanji	[fantasy magic board game Robin Williams game ...	1995
2	3	Grumpier Old Men	[moldy old comedy, romance]	1995
3	4	Waiting to Exhale	[comedy, drama, romance]	1995
4	5	Father of the Bride Part II	[pregnancy remake comedy]	1995
...	...	...	...	...
9737	193581	Black Butler: Book of the Atlantic	[action, animation, comedy, fantasy]	2017
9738	193583	No Game No Life: Zero	[animation, comedy, fantasy]	2017
9739	193585	Flint	[drama]	2017
9740	193587	Bungo Stray Dogs: Dead Apple	[action, animation]	2018
9741	193609	Andrew Dice Clay: Dice Rules	[comedy]	1991

9742 rows × 4 columns

In [ ]:
```
# Remove all extra spaces from metadata column
new_df['metadata']=new_df['metadata'].apply(lambda x:[i.replace(" ","") for i in x])
new_df
```

Out[ ]:

	movieId	title	metadata	Year
0	1	Toy Story	[pixarpixarfunadventure, animation, children, ...	1995
1	2	Jumanji	[fantasymagicboardgameRobinWilliamsgameadventu...	1995
2	3	Grumpier Old Men	[moldyoldcomedy, romance]	1995
3	4	Waiting to Exhale	[comedy, drama, romance]	1995
4	5	Father of the Bride Part II	[pregnancyremakecomedy]	1995
...	...	...	...	...
9737	193581	Black Butler: Book of the Atlantic	[action, animation, comedy, fantasy]	2017
9738	193583	No Game No Life: Zero	[animation, comedy, fantasy]	2017
9739	193585	Flint	[drama]	2017
9740	193587	Bungo Stray Dogs: Dead Apple	[action, animation]	2018
9741	193609	Andrew Dice Clay: Dice Rules	[comedy]	1991

9742 rows × 4 columns

In [ ]:
```
# join metadata spaces
new_df['metadata']=new_df['metadata'].apply(lambda x:" ".join(x))
new_df
```

Out[ ]:

	movieId	title	metadata	Year
0	1	Toy Story	pixarpixarfunadventure animation children come...	1995
1	2	Jumanji	fantasymagicboardgameRobinWilliamsgameadventur...	1995
2	3	Grumpier Old Men	moldyoldcomedy romance	1995
3	4	Waiting to Exhale	comedy drama romance	1995
4	5	Father of the Bride Part II	pregnancyremakecomedy	1995
...	...	...	...	...
9737	193581	Black Butler: Book of the Atlantic	action animation comedy fantasy	2017
9738	193583	No Game No Life: Zero	animation comedy fantasy	2017
9739	193585	Flint	drama	2017
9740	193587	Bungo Stray Dogs: Dead Apple	action animation	2018
9741	193609	Andrew Dice Clay: Dice Rules	comedy	1991

9742 rows × 4 columns

In [ ]:
```
# Find indexing position
x = new_df[new_df["title"]=="Jumanji"].index
x
```

Out[ ]:
```
Int64Index([1], dtype='int64')
```

In [ ]:
```
# check 0th location of new_df
new_df['metadata'][0]
```

Out[ ]:
```
'pixarpixarfunadventure animation children comedy fantasy'
```

In [ ]:
```
# check values of new_df
new_df.values
```

Out[ ]:
```
array([[1, 'Toy Story',
        'pixarpixarfunadventure animation children comedy fantasy',
        '1995'],
       [2, 'Jumanji',
        'fantasymagicboardgameRobinWilliamsgameadventure children fantasy',
        '1995'],
       [3, 'Grumpier Old Men', 'moldyoldcomedy romance', '1995'],
       ...,
       [193585, 'Flint', 'drama', '2017'],
       [193587, 'Bungo Stray Dogs: Dead Apple', 'action animation',
        '2018'],
       [193609, 'Andrew Dice Clay: Dice Rules', 'comedy', '1991']],
      dtype=object)
```

Convert a dataframe into sparse matrix and apply the meta data in a TfidVectorizer

In [ ]:
```
# for create vector using TfidfVectorizer library
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfid = TfidfVectorizer(stop_words='english')
new_df['metadata'] = new_df['metadata'].fillna('')
tfv_matrix = tfid.fit_transform(new_df['metadata'])
tfv_matrix
```

Out[ ]:
```
<9742x1360 sparse matrix of type '<class 'numpy.float64'>'
        with 23318 stored elements in Compressed Sparse Row format>
```

In [ ]:
```
# import linear_kernel from sklearn
from sklearn.metrics.pairwise import linear_kernel
### compute the sigmoid kernel
sig = linear_kernel(tfv_matrix, tfv_matrix)
```

In [ ]:
```
# store index of title
indices = pd.Series(new_df.index, index=new_df["title"])
```

In [ ]:
```
indices
```

Out[ ]:
```
title
Toy Story                                0
Jumanji                                  1
Grumpier Old Men                         2
Waiting to Exhale                        3
Father of the Bride Part II              4
                                      ...
Black Butler: Book of the Atlantic    9737
No Game No Life: Zero                 9738
Flint                                 9739
Bungo Stray Dogs: Dead Apple          9740
Andrew Dice Clay: Dice Rules          9741
Length: 9742, dtype: int64
```

Create a Recommendation function that finds the movies that are closely related based on similar scores and return 10 recommendations

In [ ]:
```
def recommendation(title,cosine_sin=sig):
    idx = indices[title]
    sim_scores = enumerate(cosine_sin[idx])
    sim_scores = sorted(sim_scores, key=lambda x : x[1], reverse=True)
    # how many movies will recomend is set here
    sim_scores = sim_scores[1:12]
    movies_indices = [i[0] for i in sim_scores]
    movie_rec = new_df['title'].iloc[movies_indices].reset_index
    print(movie_rec)
```

Test some recommendations

In [ ]:
```
recommendation("Toy Story")
```

```
<bound method Series.reset_index of 7184                                    Partl
y Cloudy
7917                                          Presto
8273                 Cloudy with a Chance of Meatballs 2
8674                   Stuart Little 3: Call of the Wild
9536                        Last Year's Snow Was Falling
9560                                Wow! A Talking Fish!
1584                               All Dogs Go to Heaven
2160                                           Thumbelina
3937                               Care Bears Movie, The
4208                                   Last Unicorn, The
7499    Secret World of Arrietty, The (Kari-gurashi no...
Name: title, dtype: object>
```

In [ ]:
```
recommendation("Die Hard")
```

```
<bound method Series.reset_index of 22                          Assassins
138     Die Hard: With a Vengeance
417                 Judgment Night
793                       Die Hard
1306                     Hard Rain
1315      Replacement Killers, The
1325                 U.S. Marshals
1693                         Ronin
2062                      No Mercy
2225      Someone to Watch Over Me
2802                          Shaft
Name: title, dtype: object>
```

In [ ]:
```
recommendation('Groundhog Day')
```

```
<bound method Series.reset_index of 2533           Date with an Angel
2630         Bell, Book and Candle
3093                  Down to Earth
3572                    Shallow Hal
4735    Midsummer Night's Dream, A
4923                 Ella Enchanted
5402                   Sherlock Jr.
5424         Here Comes Mr. Jordan
5565      Mannequin 2: On the Move
5871                 Boccaccio '70
5923                      Bewitched
Name: title, dtype: object>
```

In [ ]:
```
recommendation('Grease')
```

```
<bound method Series.reset_index of 1064                        Grease 2
1947               Never Been Kissed
2416    Fast Times at Ridgemont High
984                         Heathers
2195        Ferris Bueller's Day Off
5173               Napoleon Dynamite
2317               Stand and Deliver
4405            From Justin to Kelly
5432                   Easter Parade
3196                    Moulin Rouge
3494                         Glitter
Name: title, dtype: object>
```

In [ ]:
```
recommendation('True Lies')
```

```
<bound method Series.reset_index of 6693                                  1
0,000 BC
1383                               Six Days Seven Nights
2054                                               Vibes
2100    European Vacation (aka National Lampoon's Euro...
2209                                  Sullivan's Travels
2609                               Son of the Sheik, The
5810                                         Royal Flash
5921                                Herbie: Fully Loaded
7169                          Princess and the Pirate, The
7643                                         Monte Carlo
8136                   Perfect Plan, A (Plan parfait, Un)
Name: title, dtype: object>
```