Exercise 10.2: Recommender System

In [1]:

```python
# First I will import some needed libraries
import pandas as pd
from importlib import reload
import sys
import numpy as np
from imp import reload
import nltk
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
%matplotlib inline
import seaborn as sns
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.lancaster import LancasterStemmer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
import re
nltk.download('wordnet')
import string
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
import warnings
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
warnings.filterwarnings('ignore')
if sys.version[0] == '2':
    reload(sys)
    sys.setdefaultencoding("utf-8")
```

```
[nltk_data] Downloading package wordnet to /Users/Robyn/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Importing the Data

In [47]:

```python
# I will use pandas to pull the data to create a data frame to work from
Movies_Data = pd.read_csv('movies.csv')
Movies_Data
```

Out[47]:

	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
...	...	...	...
9737	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantasy
9738	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantasy
9739	193585	Flint (2017)	Drama
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animation
9741	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy

9742 rows × 3 columns

In [48]:

```python
# I will use pandas to pull the data to create a data frame to work from
Links_Data = pd.read_csv('links.csv')
Links_Data
```

Out[48]:

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

...	...	...	...
9737	193581	5476944	432131.0
9738	193583	5914996	445030.0
9739	193585	6397426	479308.0
9740	193587	8391976	483455.0
9741	193609	101726	37891.0

9742 rows × 3 columns

In [49]:
```
# I will use pandas to pull the data to create a data frame to work from
Ratings_Data = pd.read_csv('ratings.csv')
Ratings_Data
```

Out[49]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
...	...	...	...	...
100831	610	166534	4.0	1493848402
100832	610	168248	5.0	1493850091
100833	610	168250	5.0	1494273047
100834	610	168252	5.0	1493846352
100835	610	170875	3.0	1493846415

100836 rows × 4 columns

In [50]:
```
# I will use pandas to pull the data to create a data frame to work from
Tags_Data = pd.read_csv('tags.csv')
```

```
Tags_Data
```

Out[50]:

	userId	movieId	tag	timestamp
0	2	60756	funny	1445714994
1	2	60756	Highly quotable	1445714996
2	2	60756	will ferrell	1445714992
3	2	89774	Boxing story	1445715207
4	2	89774	MMA	1445715200
...	...	...	...	...
3678	606	7382	for katie	1171234019
3679	606	7936	austere	1173392334
3680	610	3265	gun fu	1493843984
3681	610	3265	heroic bloodshed	1493843978
3682	610	168248	Heroic Bloodshed	1493844270

3683 rows × 4 columns

Above we can see all the different data sets that come in the movie lens dowlnload but I believe the two dataset that are helpful are the Movies_Data and the Ratings_Data as the other variables from the other data set are not as useful.So I will only be using Movies_Data and Ratings_Data sets.

Update Data Set

In [51]:

```
# I will now pivot and combine the two data sets and fill NAs with zeros as we can see alot
Updated_Movies_Data = Ratings_Data.pivot(index='movieId',columns='userId',values='rating')
Updated_Movies_Data.fillna(0,inplace=True)
Updated_Movies_Data
```

Out[51]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
movieId																					
1	4.0	0.0	0.0	0.0	4.0	0.0	4.5	0.0	0.0	0.0	...	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0

```
     1  4.0  0.0  0.0  0.0  4.0  0.0  4.5  0.0  0.0  0.0  ...  4.0  0.0  4.0  3.0  4.0  2.5  4.0  2.5  3.0  5.0
     2  0.0  0.0  0.0  0.0  0.0  4.0  0.0  4.0  0.0  0.0  ...  0.0  4.0  0.0  5.0  3.5  0.0  0.0  2.0  0.0  0.0
     3  4.0  0.0  0.0  0.0  0.0  5.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0  0.0  0.0
     4  0.0  0.0  0.0  0.0  0.0  3.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
     5  0.0  0.0  0.0  0.0  0.0  5.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  3.0  0.0  0.0  0.0  0.0  0.0  0.0
   ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
193581  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
193583  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
193585  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
193587  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
193609  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

9724 rows × 610 columns
```

Modeling and Recommender

In [18]:

```
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
```

In [52]:

```
# I will first apply the csr_matix method to the Updated_Movies_Data
# As this will allow me to reset my index and for the values
Movies_csr_data = csr_matrix(Updated_Movies_Data.values)
Updated_Movies_Data.reset_index(inplace=True)
```

In [53]:

```
# Next I will create my NearestNeighbors algorithm and fit my data
# set to create a recommendation
movies_knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20, n_jobs=-1)
movies_knn.fit(Movies_csr_data)
```

Out[53]:

```
NearestNeighbors(algorithm='brute', metric='cosine', n_jobs=-1, n_neighbors=20)
```

In [81]:

```python
# Next I will define my function and create a recommendation from my datasets
def recommend_Movie(recommendation):
    # I will set movies_recommended to ten I am wanting to
    # receive ten recomendations
    movies_recommended = 10
    # I will set the movies equal to the titles of the movies
    # from the original Movies_Data set and use str.contains on the recommendation
    movies = Movies_Data[Movies_Data['title'].str.contains(recommendation)]
    # Next I set the if to the lenght of my movies above
    if len(movies):
        # I will now use iloc[0] on my movieId feature from my movies above
        index_movie = movies.iloc[0]['movieId']
        # Next I bring in the movieId from the Updated_Movies_Data and set it equal to index_movie
        index_movie = Updated_Movies_Data[Updated_Movies_Data['movieId'] == index_movie].index[0]
        # I am now ready to bring in the movies_knn algoritm from above along with the Movies_csr_data
        # and set the n_neighbor to movies_recommended+1
        distances , ind = movies_knn.kneighbors(Movies_csr_data[index_movie],
                                                n_neighbors=movies_recommended+1)
        # Next my indices will be sorted with squeeze().tolist() for both ind and distances
        ind_movie = sorted(list(zip(ind.squeeze().tolist(),
                                    distances.squeeze().tolist())),key=lambda x: x[1])[:0:-1]
        # I will now create an blank movie_rec to store my recommendations
        movie_rec = []
        # the use of a for statement will bring the values in the indices from above
        for val in ind_movie:
            # next I will locate values with a val of 0 in the feature movieId
            index_movie = Updated_Movies_Data.iloc[val[0]]['movieId']
            # I willnow set my feature movieId from my data frame Movies_Data
            # equal to index_movie and have it index
            movie_idex = Movies_Data[Movies_Data['movieId'] == index_movie].index
            # Use append on my movie_rec that adds in the title of the
            # movie and caculated the distance from match
            movie_rec.append({'Movie Title':Movies_Data.iloc[movie_idex]['title'].values[0],
                              'Distance from Match':val[1]})
        # I will now create my MovieData that will put my movie_rec and
        # add another until I get to 10 and have my return set to MovieData or return an auto answer
        MovieData = pd.DataFrame(movie_rec,index=range(1,movies_recommended+1))
        return MovieData
    else:
        return "No recommendations found please pick another movie"
```

In [82]:

```python
# I can now test the model to receive reommendations
recommend_Movie('Beauty and the Beast')
```

Out[82]:

	Movie Title	Distance from Match
1	Batman Forever (1995)	0.473620
2	Pretty Woman (1990)	0.472610
3	True Lies (1994)	0.465299
4	Mask, The (1994)	0.458271
5	Batman (1989)	0.456609
6	Jurassic Park (1993)	0.442575
7	Mrs. Doubtfire (1993)	0.423384
8	Snow White and the Seven Dwarfs (1937)	0.415539
9	Lion King, The (1994)	0.291639
10	Aladdin (1992)	0.252944

Reference

The sites I used to create this recommender system is seen below. I would have liked to explore this assignment more as it was very fun but I have had Covid all week and had a hard time. https://techvidvan.com/tutorials/movie-recommendation-system-python-machine-learning/

https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/

In [ ]: