Itu Mukherjee - Python for Data Science Week 6 Mini-Project

Import Modules

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

Import Data

In [2]:
```
movies = pd.read_csv('movies.csv', sep = ",")
print(movies.shape)
movies.head()
```

```
(27278, 3)
```

Out[2]:

	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 [3]:
```
ratings = pd.read_csv('ratings.csv', sep = ",")
print(ratings.shape)
ratings.head()
```

```
(20000263, 4)
```

Out[3]:

	userId	movieId	rating	timestamp
0	1	2	3.5	1112486027
1	1	29	3.5	1112484676
2	1	32	3.5	1112484819
3	1	47	3.5	1112484727
4	1	50	3.5	1112484580

In [4]:
```
tags = pd.read_csv('tags.csv', sep = ",")
print(tags.shape)
tags.head()
```

```
(465564, 4)
```

Out[4]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	1240597180
1	65	208	dark hero	1368150078
2	65	353	dark hero	1368150079
3	65	521	noir thriller	1368149983
4	65	592	dark hero	1368150078

Check for presence of NaN values in the dataframes

In [5]:
```
movies.isnull().any()
```

Out[5]:
```
movieId    False
title      False
genres     False
dtype: bool
```

In [6]:
```
ratings.isnull().any()
```

Out[6]:
```
userId       False
movieId      False
rating       False
timestamp    False
dtype: bool
```

In [7]:
```
tags.isnull().any()
```

Out[7]:
```
userId       False
movieId      False
tag           True
timestamp    False
dtype: bool
```

In [8]:
```
tags = tags.dropna()
```

In [9]:
```
tags.isnull().any()
```

Out[9]:
```
userId       False
movieId      False
tag          False
timestamp    False
dtype: bool
```

Extract year from title and store the information in a separate column

In [10]:
```
movies['year'] = movies['title'].str.extract('.*\((.*)\).*', expand=True)
```

In [11]:
```
movies.head()
```

Out[11]:

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

In [12]:
```
average_rating = ratings[['movieId','rating']].groupby('movieId', as_index=False).mean()
average_rating.head()
```

Out[12]:

	movieId	rating
0	1	3.921240
1	2	3.211977
2	3	3.151040
3	4	2.861393
4	5	3.064592

In [13]:
```
average_rating['movieId'].unique().shape
```

Out[13]: (26744,)

Merge dataframes to have yearwise average ratings for all genres

In [14]:
```
joined = movies.merge(average_rating, on='movieId', how='inner')
joined.head()
```

Out[14]:

	movieId	title	genres	year	rating
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.921240
1	2	Jumanji (1995)	Adventure|Children|Fantasy	1995	3.211977
2	3	Grumpier Old Men (1995)	Comedy|Romance	1995	3.151040
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	1995	2.861393
4	5	Father of the Bride Part II (1995)	Comedy	1995	3.064592

Merge all dataframes to have one dataframe with all information

In [15]:
```
joined_with_tag = joined.merge(tags, on='movieId', how='inner')
joined_with_tag.head()
```

Out[15]:

	movieId	title	genres	year	rating	userId	
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.92124	1644	W
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.92124	1741	co an
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.92124	1741	Di an fe
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.92124	1741	Pi an
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.92124	1741	TÂ Le do sta thi m

Extract the movies appearing in IMDb top 250 list

In [16]:
```
imdb_top_250 = joined_with_tag.loc[(joined_with_tag['tag'] == 'imdb top 250')]
print(imdb_top_250.shape)
imdb_top_250.head()
```

```
(1218, 8)
```

Out[16]:

	movieId	title	genres	year	rating	userId
139	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.92124	49307
172	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.92124	55313
292	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.92124	103125
312	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.92124	105214
381	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995	3.92124	120937

In [17]:
```
average_rating_top250 = imdb_top_250[['genres','rating']].groupby('genres', as_index=False).mean()
average_rating_top250.head()
```

Out[17]:

	genres	rating
0	Action|Adventure	4.169116
1	Action|Adventure|Animation	3.719764
2	Action|Adventure|Animation|Children|Comedy	3.908572
3	Action|Adventure|Animation|Children|Comedy|Fan...	3.750635
4	Action|Adventure|Animation|Drama|Fantasy	4.096299

Split the genre information and then merge them to remove the '|' and insert ',' instead

In [18]:
```
imdb_top250_genres = average_rating_top250['genres'].str.split('|', expand=True)
imdb_top250_genres.head()
```

Out[18]:

	0	1	2	3	4	5	6
0	Action	Adventure	None	None	None	None	None
1	Action	Adventure	Animation	None	None	None	None
2	Action	Adventure	Animation	Children	Comedy	None	None
3	Action	Adventure	Animation	Children	Comedy	Fantasy	None
4	Action	Adventure	Animation	Drama	Fantasy	None	None

In [19]:
```
imdb_top250_genres['genres'] = imdb_top250_genres[imdb_top250_genres.columns[0:]].apply(
                                    lambda x: ','.join(x.dropna().astype(str)), axis=1)
imdb_top250_genres.head()
```

Out[19]:

	0	1	2	3	4	5	6	
0	Action	Adventure	None	None	None	None	None	Action,Adventure
1	Action	Adventure	Animation	None	None	None	None	Action,Adventure,Animatio
2	Action	Adventure	Animation	Children	Comedy	None	None	Action,Adventure,Animatio
3	Action	Adventure	Animation	Children	Comedy	Fantasy	None	Action,Adventure,Animatio
4	Action	Adventure	Animation	Drama	Fantasy	None	None	Action,Adventure,Animatio

Create a wordcloud of movie genres appearing in IMDb top 250 list

In [20]:
```
text = ','.join(style for style in imdb_top250_genres.genres)
```

In [21]:
```
text
```

Out[21]:
```
'Action,Adventure,Action,Adventure,Animation,Action,Adventure,Animation,Child
ren,Comedy,Action,Adventure,Animation,Children,Comedy,Fantasy,Action,Adventur
e,Animation,Drama,Fantasy,Action,Adventure,Comedy,Fantasy,Action,Adventure,Co
medy,Fantasy,Romance,Action,Adventure,Crime,Drama,Thriller,Action,Adventure,C
rime,IMAX,Action,Adventure,Drama,Action,Adventure,Drama,Fantasy,Action,Advent
ure,Drama,Fantasy,Mystery,IMAX,Action,Adventure,Drama,Sci-Fi,Thriller,Action,
Adventure,Drama,Thriller,Action,Adventure,Drama,War,Action,Adventure,Drama,We
stern,Action,Adventure,Fantasy,Horror,Action,Adventure,Horror,Sci-Fi,Action,A
dventure,Mystery,Romance,Thriller,Action,Adventure,Romance,Action,Adventure,S
ci-Fi,Action,Adventure,Sci-Fi,IMAX,Action,Adventure,Sci-Fi,Thriller,War,Actio
n,Adventure,Thriller,IMAX,Action,Adventure,Western,Action,Comedy,Crime,Myster
y,Action,Comedy,Horror,Action,Crime,Action,Crime,Drama,Action,Crime,Drama,IMA
X,Action,Crime,Drama,Mystery,Sci-Fi,Thriller,IMAX,Action,Crime,Drama,Thrille
r,Action,Crime,Film-Noir,Mystery,Thriller,Action,Crime,IMAX,Action,Crime,Thri
ller,Action,Drama,Romance,Action,Drama,Romance,War,Action,Drama,Sci-Fi,Actio
n,Drama,Thriller,Action,Drama,Thriller,War,Action,Drama,Thriller,Western,Acti
on,Drama,War,Action,Drama,Western,Action,Horror,Action,Horror,Sci-Fi,Thrille
r,Action,Sci-Fi,Action,Sci-Fi,IMAX,Action,Sci-Fi,Thriller,Action,Sci-Fi,Thril
ler,IMAX,Action,Western,Adventure,Animation,Children,Comedy,Adventure,Animati
on,Children,Comedy,Fantasy,Adventure,Animation,Children,Comedy,Fantasy,Romanc
e,Adventure,Animation,Children,Romance,Sci-Fi,Adventure,Animation,Fantasy,Adv
enture,Animation,Fantasy,Romance,Adventure,Children,Fantasy,Musical,Adventur
e,Comedy,Drama,Adventure,Comedy,Fantasy,Adventure,Comedy,Romance,Adventure,Co
medy,Romance,War,Adventure,Comedy,Sci-Fi,Adventure,Drama,Adventure,Drama,IMA
X,Adventure,Drama,Sci-Fi,Adventure,Drama,Thriller,Adventure,Drama,War,Adventu
re,Fantasy,Adventure,Fantasy,IMAX,Adventure,Western,Animation,Children,Drama,
Animation,Drama,War,Children,Comedy,Comedy,Comedy,Crime,Comedy,Crime,Drama,Co
medy,Crime,Drama,Thriller,Comedy,Crime,Thriller,Comedy,Drama,Comedy,Drama,Rom
ance,Comedy,Drama,Romance,War,Comedy,Drama,War,Comedy,Fantasy,Comedy,Fantasy,
Romance,Comedy,Musical,Romance,Comedy,Musical,War,Comedy,Mystery,Thriller,Com
edy,Romance,Comedy,War,Crime,Drama,Crime,Drama,Film-Noir,Crime,Drama,Film-Noi
r,Thriller,Crime,Drama,Mystery,Crime,Drama,Mystery,Thriller,Crime,Drama,Roman
ce,Crime,Drama,Sci-Fi,Thriller,Crime,Drama,Thriller,Crime,Drama,Western,Crim
e,Fantasy,Horror,Crime,Film-Noir,Crime,Film-Noir,Mystery,Crime,Film-Noir,Myst
ery,Thriller,Crime,Film-Noir,Thriller,Crime,Horror,Crime,Horror,Thriller,Crim
e,Mystery,Thriller,Crime,Thriller,War,Drama,Drama,Fantasy,Drama,Fantasy,Horro
r,Romance,Drama,Fantasy,Romance,Drama,Fantasy,Thriller,Drama,Film-Noir,Romanc
e,Drama,Film-Noir,Thriller,Drama,Horror,Mystery,Drama,Horror,Sci-Fi,Drama,Hor
ror,Thriller,Drama,Musical,Romance,Drama,Mystery,Drama,Mystery,Romance,Thrill
er,Drama,Mystery,Sci-Fi,Drama,Mystery,Sci-Fi,Thriller,Drama,Mystery,Thriller,
Drama,Mystery,War,Drama,Romance,Drama,Romance,Sci-Fi,Drama,Romance,Thriller,D
rama,Romance,War,Drama,Sci-Fi,Drama,Sci-Fi,Thriller,Drama,Thriller,Drama,War,
Drama,Western,Fantasy,Sci-Fi,Film-Noir,Film-Noir,Mystery,Film-Noir,Mystery,Th
riller,Film-Noir,Romance,Thriller,Horror,Horror,Mystery,Horror,Mystery,Thrill
er,Horror,Sci-Fi,Mystery,Sci-Fi,Thriller,Mystery,Thriller,Sci-Fi'
```

In [22]:
```
wordcloud = WordCloud(width=800, height=500, colormap="jet",
                      background_color="white").generate(text)
```

In [23]:
```
plt.figure()
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
wordcloud.to_file("imdbtop250_wordcloud.png")
```

Out[23]: <wordcloud.wordcloud.WordCloud at 0x27c0000f828>

Plot temporal trends in ratings of movie genres since year 2000

Create a new dataframe with information only from year 2000

In [24]:
```
temp = joined.loc[(joined['year'] == '2000') | (joined['year'] == '2001') |
                  (joined['year'] == '2002') | (joined['year'] == '200
3') |
                  (joined['year'] == '2004') | (joined['year'] == '200
5') |
                  (joined['year'] == '2006') | (joined['year'] == '200
7') |
                  (joined['year'] == '2009') | (joined['year'] == '200
9') |
                  (joined['year'] == '2010') | (joined['year'] == '201
1') |
                  (joined['year'] == '2012') | (joined['year'] == '201
3') |
                  (joined['year'] == '2014') | (joined['year'] == '201
5') |
                  (joined['year'] == '2016') | (joined['year'] == '201
7') |
                  (joined['year'] == '2018')]
temp.head()
```

Out[24]:

	movieId	title	genres	year	rating
2683	2769	Yards, The (2000)	Crime|Drama	2000	3.129956
3090	3177	Next Friday (2000)	Comedy	2000	2.810680
3103	3190	Supernova (2000)	Adventure|Sci-Fi|Thriller	2000	2.280392
3138	3225	Down to You (2000)	Comedy|Romance	2000	2.644370
3141	3228	Wirey Spindell (2000)	Comedy	2000	2.104167

Create separate dataframes for each of the six genres to be visualized

In [25]:
```
value_list = ['Drama']
drama = temp[temp.genres.isin(value_list)]
drama.head()
```

Out[25]:

	movieId	title	genres	year	rating
3320	3408	Erin Brockovich (2000)	Drama	2000	3.590389
3424	3514	Joe Gould's Secret (2000)	Drama	2000	3.290000
3443	3534	28 Days (2000)	Drama	2000	3.092077
3488	3579	I Dreamed of Africa (2000)	Drama	2000	2.488333
3489	3580	Up at the Villa (2000)	Drama	2000	2.997368

In [26]:
```
value_list = ['Thriller']
thriller = temp[temp.genres.isin(value_list)]
thriller.head()
```

Out[26]:

	movieId	title	genres	year	rating
3204	3291	Trois (2000)	Thriller	2000	1.729167
3394	3484	Skulls, The (2000)	Thriller	2000	2.754047
3706	3797	In Crowd, The (2000)	Thriller	2000	2.272201
3764	3857	Bless the Child (2000)	Thriller	2000	2.614094
3926	4020	Gift, The (2000)	Thriller	2000	3.390297

In [27]:
```
value_list = ['Crime']
crime = temp[temp.genres.isin(value_list)]
crime.head()
```

Out[27]:

	movieId	title	genres	year	rating
6394	6504	Gasoline (Benzina) (2001)	Crime	2001	2.400000
12636	59418	American Crime, An (2007)	Crime	2007	3.686528
13479	66691	Thick as Thieves (a.k.a. Code, The) (2009)	Crime	2009	2.992593
15600	79536	Hellsinki (Rööperi) (2009)	Crime	2009	3.144068
18730	93490	Law of the Lawless (Brigada) (2002)	Crime	2002	3.875000

In [28]:
```
value_list = ['Comedy']
comedy = temp[temp.genres.isin(value_list)]
comedy.head()
```

Out[28]:

	movieId	title	genres	year	rating
3090	3177	Next Friday (2000)	Comedy	2000	2.810680
3141	3228	Wirey Spindell (2000)	Comedy	2000	2.104167
3152	3239	Isn't She Great? (2000)	Comedy	2000	2.270732
3189	3276	Gun Shy (2000)	Comedy	2000	2.925926
3199	3286	Snow Day (2000)	Comedy	2000	2.222868

In [29]:
```
value_list = ['Romance']
romance = temp[temp.genres.isin(value_list)]
romance.head()
```

Out[29]:

	movieId	title	genres	year	rating
3930	4024	House of Mirth, The (2000)	Romance	2000	3.418321
4796	4892	Maze (2000)	Romance	2000	3.190000
5565	5664	Brown Sugar (2002)	Romance	2002	3.053672
6093	6192	Open Hearts (Elsker dig for evigt) (2002)	Romance	2002	3.680556
6446	6556	Sea Is Watching, The (Umi wa miteita) (2002)	Romance	2002	3.000000

In [30]:
```
value_list = ['Sci-Fi']
scifi = temp[temp.genres.isin(value_list)]
scifi.head()
```

Out[30]:

	movieId	title	genres	year	rating
3267	3354	Mission to Mars (2000)	Sci-Fi	2000	2.625766
10920	44671	Wild Blue Yonder, The (2005)	Sci-Fi	2005	2.803571
13071	62834	Babylon 5: The Legend of the Rangers: To Live ...	Sci-Fi	2002	2.952991
13072	62836	Babylon 5: The Lost Tales - Voices in the Dark...	Sci-Fi	2007	3.218447
15217	77795	Cargo (2009)	Sci-Fi	2009	3.464286

Plot each of the dataframes on the same matplotlib axis object, thereby creating an overlaid time series plot

In [31]:
```
with plt.style.context('seaborn-whitegrid'):
    fig,ax=plt.subplots(figsize=(10,4))
    ax=sns.pointplot(x='year',y='rating',data=drama,ci=None,color='blue')
    ax=sns.pointplot(x='year',y='rating',data=thriller,ci=None,color='red')
    ax=sns.pointplot(x='year',y='rating',data=crime,ci=None,color='magenta')
    ax=sns.pointplot(x='year',y='rating',data=comedy,ci=None,color='forestgree
n')
    ax=sns.pointplot(x='year',y='rating',data=romance,ci=None,color='orange')
    ax=sns.pointplot(x='year',y='rating',data=scifi,ci=None,color='black')

    plt.xticks(rotation=90)
    plt.box(False)
    ax.grid('both','both',linestyle='--')
    ax.legend(['Drama','Thriller','Crime',
               'Comedy','Romance','Sci-Fi'], bbox_to_anchor=(1,0.6))
    plt.savefig('ratings_timeline_whitegrid.png',dpi=300,bbox_inches='tight')
```