Movie Recommender System

In this iPython Notebook, I have created basic Movie Recommender System with Python.

```
#Import all necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.html.widgets import *
sns.set_style('white')
%matplotlib inline
```

```
/usr/local/lib/python3.10/dist-packages/IPython/html.py:12: ShimWarning: The `IPython.html`
  warn("The `IPython.html` package has been deprecated since IPython 4.0. "
```

```
#Get the data into Pandas Dataframe object
column_names = ['user_id', 'item_id', 'rating', 'timestamp']
df = pd.read_csv('dataset.csv', sep = '\t', names = column_names)
df.head()
```

	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742

```
#Get the Movie Titles
movie_titles = pd.read_csv('movieIdTitles.csv')
movie_titles.head()
```

	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

```
#Merge the dataset with movie titles
df = pd.merge(df, movie_titles, on = 'item_id')
df.head()
```

	user_id	item_id	rating	timestamp	title
0	0	50	5	881250949	Star Wars (1977)
1	290	50	5	880473582	Star Wars (1977)
2	79	50	4	891271545	Star Wars (1977)
3	2	50	5	888552084	Star Wars (1977)
4	8	50	5	879362124	Star Wars (1977)

Do some Exploratory Data Analysis

```
df.groupby('title')['rating'].mean().sort_values(ascending = False).head()
```

```
title
They Made Me a Criminal (1939)                5.0
Marlene Dietrich: Shadow and Light (1996)     5.0
Saint of Fort Washington, The (1993)          5.0
Someone Else's America (1995)                 5.0
Star Kid (1997)                               5.0
Name: rating, dtype: float64
```

```
df.groupby('title')['rating'].count().sort_values(ascending = False).head()
```

```
title
Star Wars (1977)             584
Contact (1997)               509
Fargo (1996)                 508
Return of the Jedi (1983)    507
Liar Liar (1997)             485
Name: rating, dtype: int64
```

```
ratings = pd.DataFrame(df.groupby('title')['rating'].mean())
ratings.head()
```

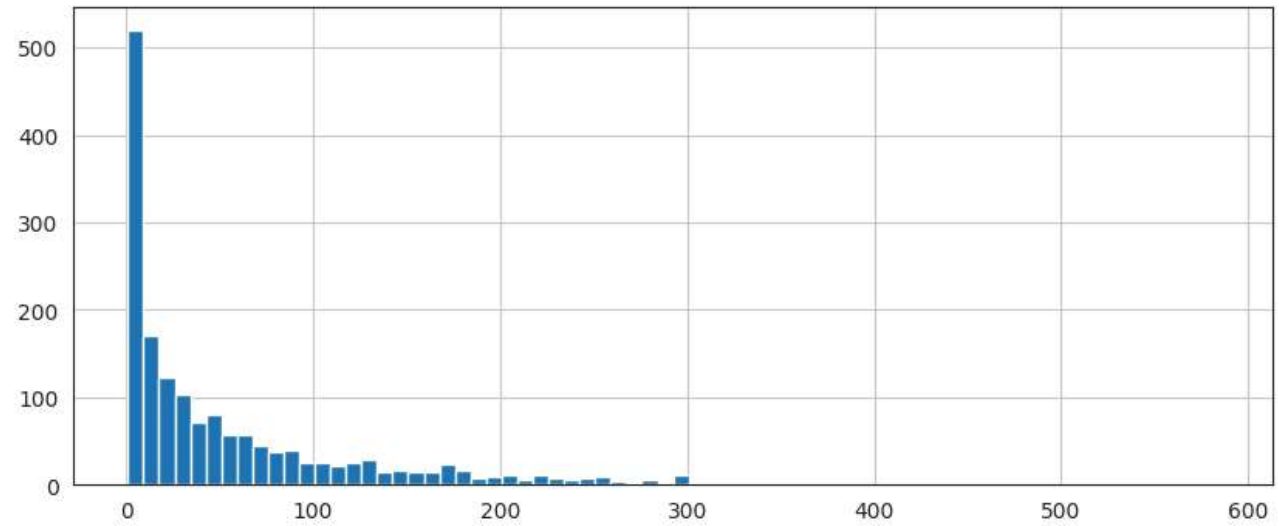
title	rating
'Til There Was You (1997)	2.333333
1-900 (1994)	2.600000
101 Dalmatians (1996)	2.908257
12 Angry Men (1957)	4.344000
187 (1997)	3.024390

```
ratings['numOfRatings'] = pd.DataFrame(df.groupby('title')['rating'].count())
ratings.head()
```

title	rating	numOfRatings
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

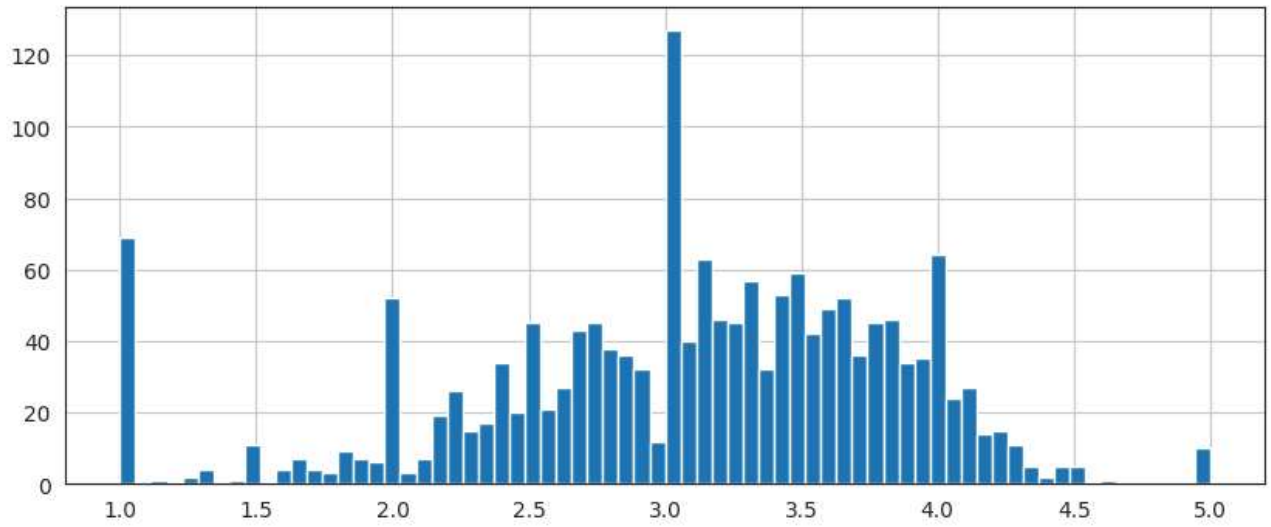
```
plt.figure(figsize = (10,4))
ratings['numOfRatings'].hist(bins = 70)
```

```
<Axes: >
```



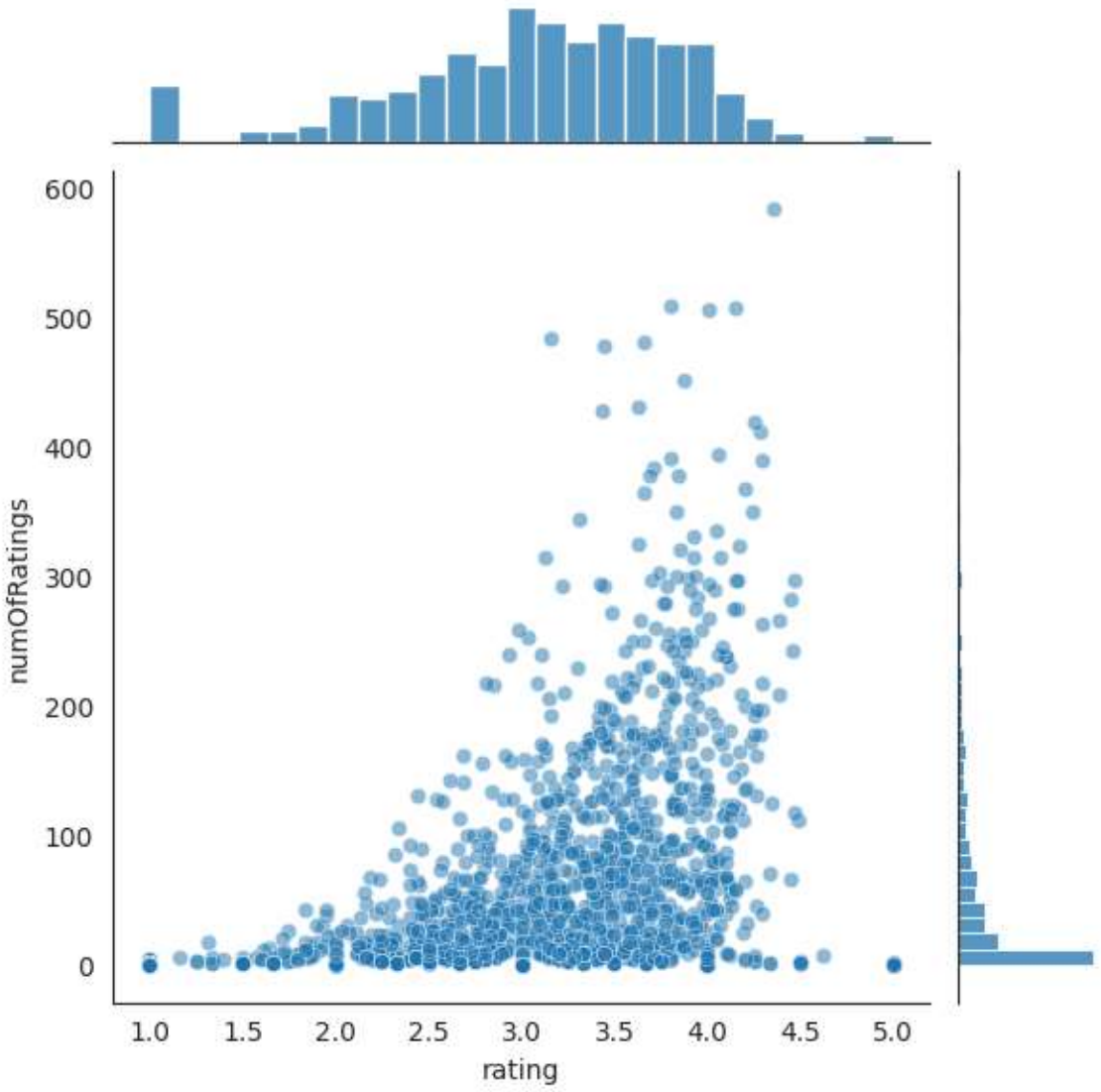
```
plt.figure(figsize = (10,4))
ratings['rating'].hist(bins = 70)
```

```
<Axes: >
```




```
sns.jointplot(x='rating', y='numOfRatings', data = ratings, alpha = 0.5)
```

```
<seaborn.axisgrid.JointGrid at 0x7bef20b37e50>
```



Create the Recommendation System

Create a matrix that has the user ids on one access and the movie title on another axis. Each cell will then consist of the rating the user gave to that movie. Note there will be a lot of NaN values, because most people have not seen most of the movies.

```
moviemat = df.pivot_table(index='user_id',columns='title',values='rating')
moviemat.head()
```

title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)
user_id									
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 1664 columns

```
#Most Rated Movies with their Average Ratings
ratings.sort_values('numOfRatings', ascending = False).head(10)
```

title	rating	numOfRatings
Star Wars (1977)	4.359589	584
Contact (1997)	3.803536	509
Fargo (1996)	4.155512	508
Return of the Jedi (1983)	4.007890	507
Liar Liar (1997)	3.156701	485
English Patient, The (1996)	3.656965	481
Scream (1996)	3.441423	478
Toy Story (1995)	3.878319	452
Air Force One (1997)	3.631090	431
Independence Day (ID4) (1996)	3.438228	429

Now we will create a correlation matrix of every movie with every other movie on user ratings. We will then use that correlation matrix to find top matches that relates the best for a particular movie (having atleast 100 ratings) and the result obtained (recommended movies) will then be added to the ratings dataframe of every movie. Those whose matches could not be obtained using correlation, their value will be converted to "-".

```
for i in ratings.index:
    movieUserRatings = moviemat[i]
    similarToThatMovie = moviemat.corrwith(movieUserRatings)
    corr_toMovie = pd.DataFrame(similarToThatMovie, columns = ['Correlation'])
    corr_toMovie.dropna(inplace = True)
    corr_toMovie = corr_toMovie.join(ratings['numOfRatings'])
    result = corr_toMovie[corr_toMovie['numOfRatings'] > 100].sort_values('Correlation', ascend
    if result['numOfRatings'].count() >= 5:
        print(i)
        ratings.loc[i, 'FirstMovieRecommendation'] = result.iloc[1:2].index.values[0]
        ratings.loc[i, 'SecondMovieRecommendation'] = result.iloc[2:3].index.values[0]
        ratings.loc[i, 'ThirdMovieRecommendation'] = result.iloc[3:4].index.values[0]
        ratings.loc[i, 'FourthMovieRecommendation'] = result.iloc[4:5].index.values[0]
```

```
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2889: RuntimeWarning:
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2748: RuntimeWarning:
  c *= np.true_divide(1, fact)
Age of Innocence, The (1993)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2889: RuntimeWarning:
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2748: RuntimeWarning:
  c *= np.true_divide(1, fact)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2889: RuntimeWarning:
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2748: RuntimeWarning:
  c *= np.true_divide(1, fact)
Air Bud (1997)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2889: RuntimeWarning:
```

```
#Check the result
ratings.head()
```

title	rating	numOfRatings	FirstMovieRecommendation	SecondMovieRecommendation	Thi
'Til There Was You (1997)	2.333333	9	Edge, The (1997)	William Shakespeare's Romeo and Juliet (1996)	Sta
1-900 (1994)	2.600000	5	Full Monty, The (1997)	Crow, The (1994)	
101 Dalmatians (1996)	2.908257	109	Murder at 1600 (1997)	Miracle on 34th Street (1994)	
12 Angry Men (1957)	4.344000	125	Ulee's Gold (1997)	Rear Window (1954)	S
187 (1997)	3.024390	41	Maverick (1994)	Conan the Barbarian (1981)	

```
ratings = ratings.fillna('-')
```

```
#Save the ratings data for later use
ratings.to_csv('MovieRecommendations.csv', encoding='utf-8')
```

Load the Saved Recommendation Data Generated for Reusability

```
#Load the dataset saved for reusability from this code block onwards
df_result = pd.read_csv('MovieRecommendations.csv')
df_result.head()
```

	title	rating	numOfRatings	FirstMovieRecommendation	SecondMovieRecommendatic
0	'Til There Was You (1997)	2.333333	9	Edge, The (1997)	William Shakespeare's Rome and Juliet (199
1	1-900 (1994)	2.600000	5	Full Monty, The (1997)	Crow, The (199
2	101 Dalmatians (1996)	2.908257	109	Murder at 1600 (1997)	Miracle on 34th Street (199
3	12 Angry Men (1957)	4.344000	125	Ulee's Gold (1997)	Rear Window (195
4	187 (1997)	3.024390	41	Maverick (1994)	Conan the Barbarian (198

```
#Load all the movie names
for i in df_result['title']:
    print(i)
```

```
'Til There Was You (1997)
1-900 (1994)
101 Dalmatians (1996)
12 Angry Men (1957)
187 (1997)
2 Days in the Valley (1996)
20,000 Leagues Under the Sea (1954)
2001: A Space Odyssey (1968)
3 Ninjas: High Noon At Mega Mountain (1998)
39 Steps, The (1935)
8 1/2 (1963)
8 Heads in a Duffel Bag (1997)
8 Seconds (1994)
A Chef in Love (1996)
Above the Rim (1994)
Absolute Power (1997)
Abyss, The (1989)
Ace Ventura: Pet Detective (1994)
Ace Ventura: When Nature Calls (1995)
Across the Sea of Time (1995)
Addams Family Values (1993)
Addicted to Love (1997)
Addiction, The (1995)
Adventures of Pinocchio, The (1996)
Adventures of Priscilla, Queen of the Desert, The (1994)
Adventures of Robin Hood, The (1938)
Affair to Remember, An (1957)
African Queen, The (1951)
```

```
Afterglow (1997)
Age of Innocence, The (1993)
Aiqing wansui (1994)
Air Bud (1997)
Air Force One (1997)
Air Up There, The (1994)
Airheads (1994)
Akira (1988)
Aladdin (1992)
Aladdin and the King of Thieves (1996)
Alaska (1996)
Albino Alligator (1996)
Alice in Wonderland (1951)
Alien (1979)
Alien 3 (1992)
Alien: Resurrection (1997)
Aliens (1986)
All About Eve (1950)
All Dogs Go to Heaven 2 (1996)
All Over Me (1997)
All Things Fair (1996)
Alphaville (1965)
Amadeus (1984)
Amateur (1994)
Amazing Panda Adventure, The (1995)
American Buffalo (1996)
American Dream (1990)
American President, The (1995)
American Strays (1996)
American Werewolf in London, An (1981)
```

```
inputMovieName = widgets.Text()

def getRecommendations(sender):
    searchMovie = inputMovieName.value
    list_result = df_result[df_result['title'] == searchMovie]
    fm = list_result['FirstMovieRecommendation'].values[0]
    sm = list_result['SecondMovieRecommendation'].values[0]
    tm = list_result['ThirdMovieRecommendation'].values[0]
    fourthm = list_result['FourthMovieRecommendation'].values[0]
    finalRecommendationText = '1:' + fm + ' \n2:' + sm + ' \n3:' + tm + ' \n4:' + fourthm
    print('Your Recommendations for the Movie ' + searchMovie + ' are:\n')
    print(finalRecommendationText)
```