simplilearnproject

May 30, 2019

In [261]:
```
import pandas as pd
import numpy as np
import time



# import plotting libraries

import matplotlib
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
%matplotlib inline
import seaborn as sns
plt.show()


sns.set(style="white", color_codes=True)
sns.set(font_scale=1.5)
```

In [262]:
```
location_movies  = r"/Users/vaseekaranrajagopal/movies.dat"
location_users   = r"/Users/vaseekaranrajagopal/users.dat"
location_ratings = r"/Users/vaseekaranrajagopal/ratings.dat"
```

In [263]:
```
col_movies  = ['movieId', "title",  'genres']
col_users   = ['userId',  'Gender', 'Age', 'Occupation', 'Zip-code']
col_ratings = ['userId',  'movieId','rating', 'timestamp']
```

In [264]:
```
df_movies  = pd.read_csv(location_movies, sep="::", engine='python', header=None, nar
df_users   = pd.read_csv(location_users, sep="::", engine='python', header=None, name
df_ratings = pd.read_csv(location_ratings, sep="::", engine='python', header=None, na
```

In [265]:
```
print('Number of rows in movies df :', df_movies.shape)
```

```
Number of rows in movies df : (3883, 3)
```

In [266]:
```
print('Number of rows in users df :', df_users.shape)
```

```
Number of rows in users df : (6040, 5)
```

```
In [267]: print('Number of rows in ratings df :', df_ratings.shape)
```

```
Number of rows in ratings df : (1000209, 4)
```

```
In [268]: #first five rows
          df_movies.head(5)
```

Out[268]:

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

```
In [269]: #First five user rows
          df_users.head(10)
```

Out[269]:

	userId	Gender	Age	Occupation	Zip-code
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455
5	6	F	50	9	55117
6	7	M	35	1	06810
7	8	M	25	12	11413
8	9	M	25	17	61614
9	10	F	35	1	95370

```
In [270]: #First five ratings rows
          df_ratings.head(3)
```

Out[270]:

	userId	movieId	rating	timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968

```
In [272]: #no numm

          df_movies.isnull().sum()
          df_users.isnull().sum()
          df_ratings.isnull().sum()
```

```
Out[272]: userId       0
          movieId      0
          rating       0
          timestamp    0
          dtype: int64
```

```
In [273]: #1  merge
          df_movie_ratings = pd.merge(df_movies, df_ratings)
          df_movie_ratings.head(3)
```

Out[273]:

	movieId	title	genres	userId	rating	timestamp
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268
1	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008
2	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496

```
In [274]: print('Movies columns   :', df_movies.columns)
```

```
Movies columns   : Index(['movieId', 'title', 'genres'], dtype='object')
```

```
In [275]: print('Ratings columns  :', df_ratings.columns)
```

```
Ratings columns  : Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

```
In [276]: print('Combined columns :', df_movie_ratings.columns)
```

```
Combined columns : Index(['movieId', 'title', 'genres', 'userId', 'rating', 'timestamp'], dtype
```

```
In [277]: #merge 2
          df_movie_ratings_users = pd.merge(df_movie_ratings, df_users)
          df_movie_ratings_users.head(3)
```

Out[277]:

	movieId	title	genres	userId	rating	timestamp	Gender	Age	Occupation	Zip-code
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	10	48067
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	10	48067

```
In [278]: print('Combined columns1 :', df_movie_ratings_users.columns)
```

```
Combined columns1 : Index(['movieId', 'title', 'genres', 'userId', 'rating', 'timestamp', 'Gend
       'Age', 'Occupation', 'Zip-code'],
      dtype='object')
```

In [279]:
```
#Visualize User Distribution
df_movie_ratings_users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 10 columns):
movieId       1000209 non-null int64
title         1000209 non-null object
genres        1000209 non-null object
userId        1000209 non-null int64
rating        1000209 non-null int64
timestamp     1000209 non-null int64
Gender        1000209 non-null object
Age           1000209 non-null int64
Occupation    1000209 non-null int64
Zip-code      1000209 non-null object
dtypes: int64(6), object(4)
memory usage: 83.9+ MB
```

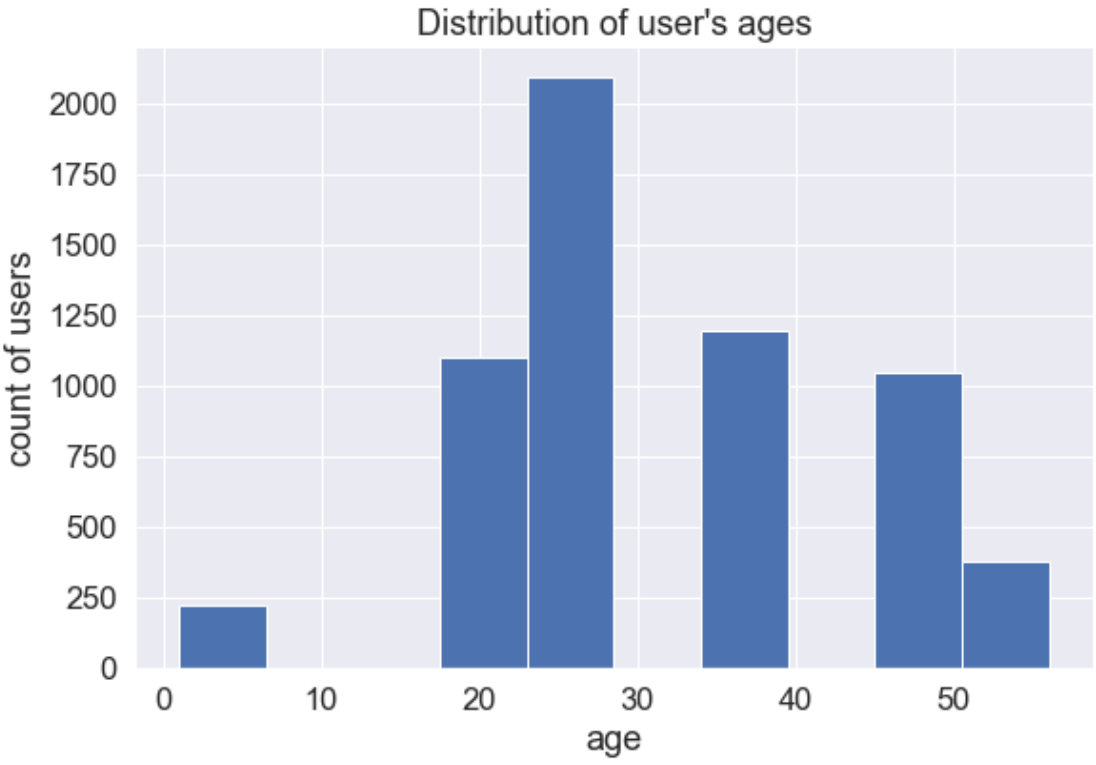
In [280]:
```
#basic stats
pd.options.display.float_format = '{:,.2f}'.format
df_movie_ratings_users.describe()
```

Out[280]:

	movieId	userId	rating	timestamp	Age	Occupation
count	1,000,209.00	1,000,209.00	1,000,209.00	1,000,209.00	1,000,209.00	1,000,209.00
mean	1,865.54	3,024.51	3.58	972,243,695.40	29.74	8.04
std	1,096.04	1,728.41	1.12	12,152,558.94	11.75	6.53
min	1.00	1.00	1.00	956,703,932.00	1.00	0.00
25%	1,030.00	1,506.00	3.00	965,302,637.00	25.00	2.00
50%	1,835.00	3,070.00	4.00	973,018,006.00	25.00	7.00
75%	2,770.00	4,476.00	4.00	975,220,939.00	35.00	14.00
max	3,952.00	6,040.00	5.00	1,046,454,590.00	56.00	20.00

In [281]:
```
#preprocessing

t = (2019,1,14,17,3,38,1,48,0)
t = time.mktime(t)
```

```
In [282]: print(time.strftime("%y", time.gmtime(t)))
          print(time.strftime("%m", time.gmtime(t)))
          print(time.strftime("%d", time.gmtime(t)))
          print(time.strftime("%H:%M:%S", time.gmtime(t)))
          print(time.strftime("%Z", time.gmtime(t)))

19
01
14
23:03:38
UTC


In [ ]:

In [283]: df_movie_ratings_users.timestamp.head()

Out[283]: 0    978824268
          1    978824351
          2    978301777
          3    978300760
          4    978824195
          Name: timestamp, dtype: int64

In [284]: #Visualization of users age

          plt.figure(figsize =(9,6))
          df_users.Age.plot.hist()
          plt.title("Distribution of user's ages")
          plt.ylabel('count of users')
          plt.xlabel('age');
```




```
In [285]: df_movie_ratings.columns

Out[285]: Index(['movieId', 'title', 'genres', 'userId', 'rating', 'timestamp'], dtype='object

In [ ]: df_movie_ratings.groupby('title')['rating']

In [286]: #1)Average rating of each movie
          df_movie_ratings.groupby('title')['rating'].mean().head()

Out[286]: title
          $1,000,000 Duck (1971)          3.03
          'Night Mother (1986)            3.37
          'Til There Was You (1997)       2.69
          'burbs, The (1989)              2.91
          ...And Justice for All (1979)   3.71
          Name: rating, dtype: float64

In [ ]:

In [287]: #sort the ratings in the descending order of 25 movies:
          df_movie_ratings.groupby('title')['rating'].mean().sort_values(ascending = False)[:25

Out[287]: title
          Gate of Heavenly Peace, The (1995)                                    5.00
```

```
Lured (1947)                                                    5.00
Ulysses (Ulisse) (1954)                                         5.00
Smashing Time (1967)                                            5.00
Follow the Bitch (1998)                                         5.00
Song of Freedom (1936)                                          5.00
Bittersweet Motel (2000)                                        5.00
Baby, The (1973)                                                5.00
One Little Indian (1973)                                        5.00
Schlafes Bruder (Brother of Sleep) (1995)                       5.00
I Am Cuba (Soy Cuba/Ya Kuba) (1964)                             4.80
Lamerica (1994)                                                 4.75
Apple, The (Sib) (1998)                                         4.67
Sanjuro (1962)                                                  4.61
Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)   4.56
Shawshank Redemption, The (1994)                                4.55
Godfather, The (1972)                                           4.52
Close Shave, A (1995)                                           4.52
Usual Suspects, The (1995)                                      4.52
Schindler's List (1993)                                         4.51
Wrong Trousers, The (1993)                                      4.51
Dangerous Game (1993)                                           4.50
Mamma Roma (1962)                                               4.50
Inheritors, The (Die Siebtelbauern) (1998)                      4.50
Hour of the Pig, The (1993)                                     4.50
Name: rating, dtype: float64
```

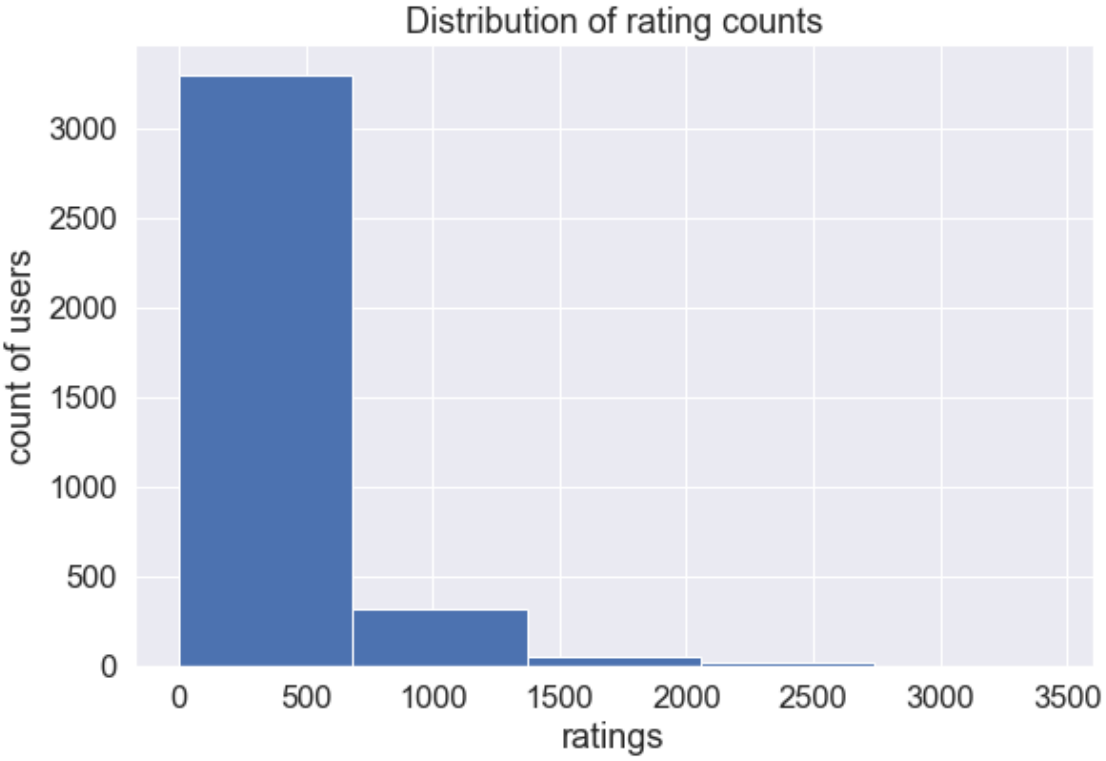
In [288]:
```
#get total number of ratings for a movie
df_movie_ratings.groupby('title')['rating'].count().sort_values(ascending = False)[:
```

Out[288]:
```
title
American Beauty (1999)                                3428
Star Wars: Episode IV - A New Hope (1977)             2991
Star Wars: Episode V - The Empire Strikes Back (1980) 2990
Star Wars: Episode VI - Return of the Jedi (1983)     2883
Jurassic Park (1993)                                  2672
Saving Private Ryan (1998)                            2653
Terminator 2: Judgment Day (1991)                     2649
Matrix, The (1999)                                    2590
Back to the Future (1985)                             2583
Silence of the Lambs, The (1991)                      2578
Name: rating, dtype: int64
```

In [289]:
```
#average ratings per movie and number of ratings per movie

ratings_mean_count = pd.DataFrame(data= df_movie_ratings.groupby('title')['rating'].
```

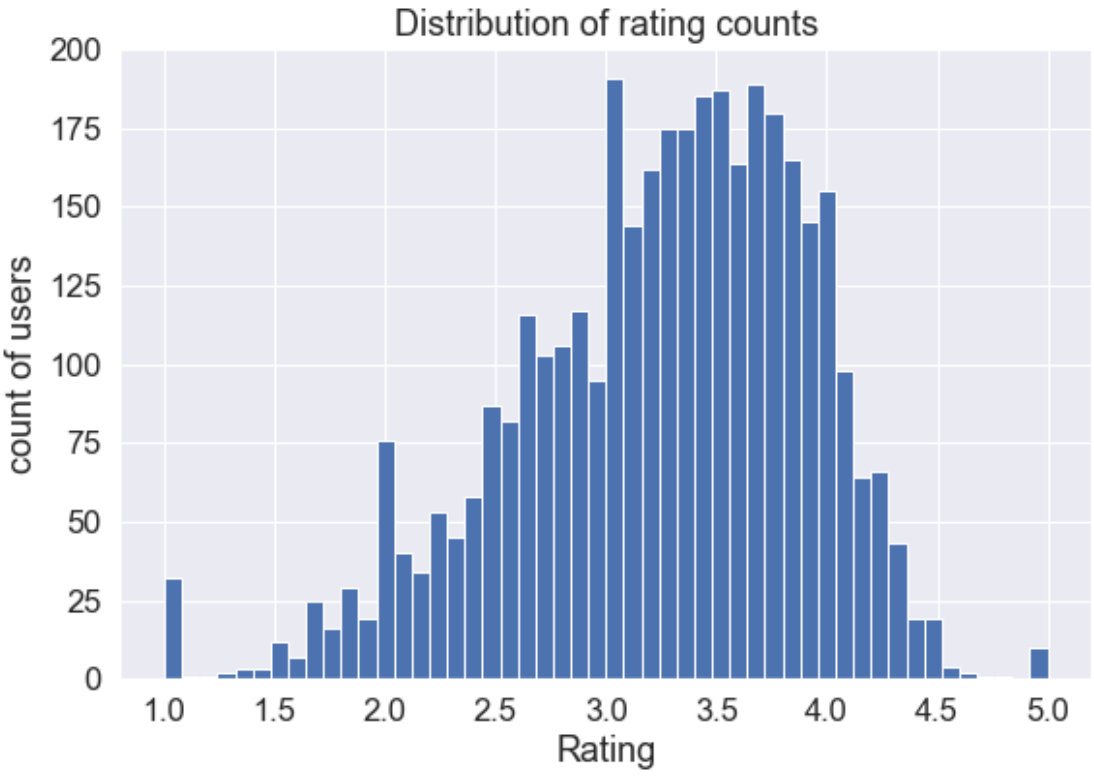
In [290]:
```
ratings_mean_count['rating_counts'] = pd.DataFrame(df_movie_ratings.groupby('title')
ratings_mean_count.columns
```

```
Out[290]: Index(['rating', 'rating_counts'], dtype='object')

In [291]: ratings_mean_count.reset_index(inplace =True)
          ratings_mean_count.columns

Out[291]: Index(['title', 'rating', 'rating_counts'], dtype='object')

In [292]:
          ratings_mean_count.head()

Out[292]:                              title  rating  rating_counts
          0         $1,000,000 Duck (1971)    3.03             37
          1           'Night Mother (1986)    3.37             70
          2      'Til There Was You (1997)    2.69             52
          3             'burbs, The (1989)    2.91            303
          4  ...And Justice for All (1979)    3.71            199

In [293]: plt.figure(figsize=(9,6))
          ratings_mean_count['rating_counts'].hist(bins=5)
          plt.title("Distribution of rating counts")
          plt.ylabel('count of users')
          plt.xlabel('ratings')

Out[293]: Text(0.5,0,'ratings')
```



In [294]:
```
plt.figure(figsize=(9,6))
ratings_mean_count['rating'].hist(bins=50)
plt.title("Distribution of rating counts")
plt.ylabel('count of users')
plt.xlabel('Rating')
```

Out[294]: Text(0.5,0,'Rating')

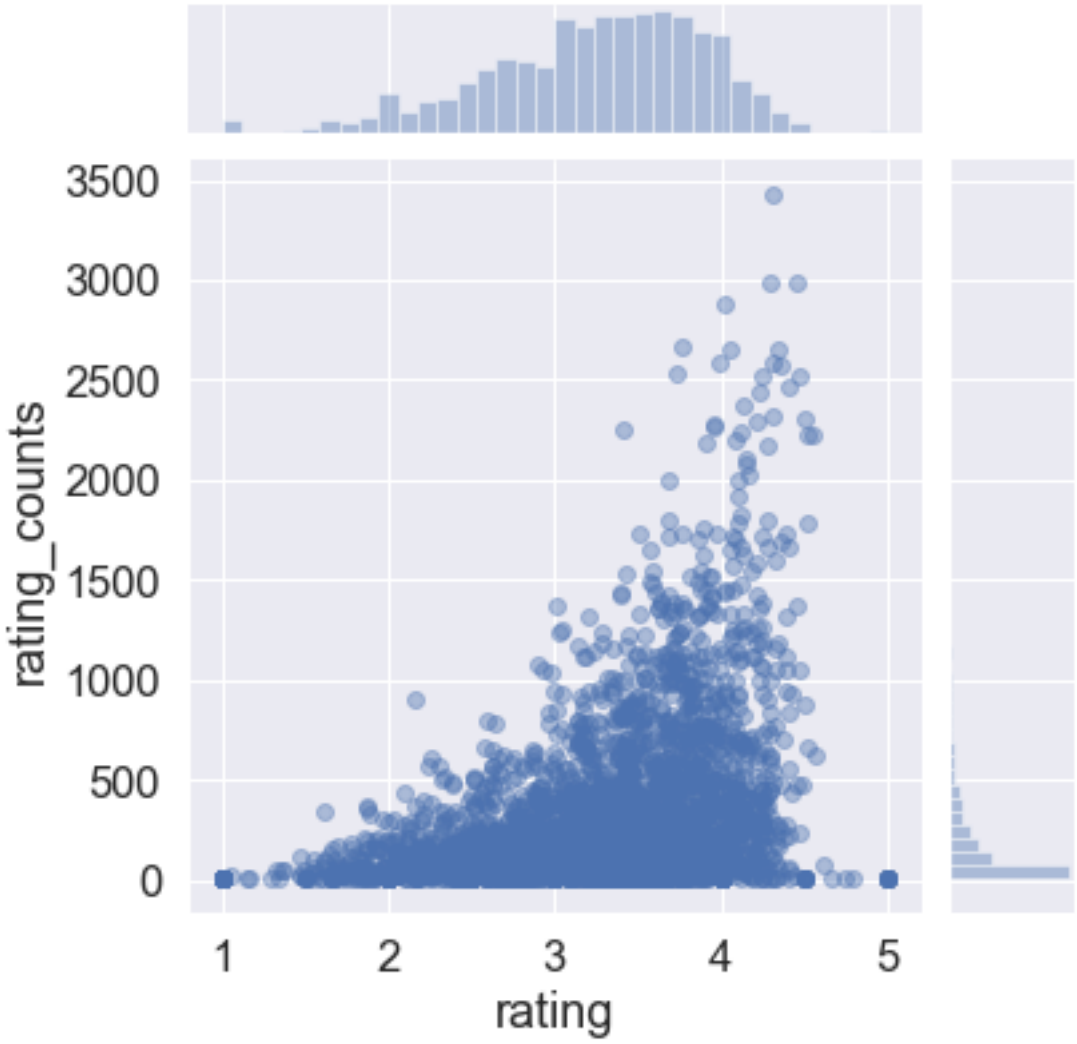


In [295]:
```
plt.figure(figsize=(10,6))
plt.rcParams['patch.force_edgecolor'] =True
sns.jointplot(x='rating', y= 'rating_counts', data= ratings_mean_count, alpha=0.4)
```

```
/Users/vaseekaranrajagopal/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: Fu
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[295]: <seaborn.axisgrid.JointGrid at 0x1a5342b080>

<Figure size 720x432 with 0 Axes>




```
In [ ]:
```

```
In [296]: df_movie_ratings.columns
```

```
Out[296]: Index(['movieId', 'title', 'genres', 'userId', 'rating', 'timestamp'], dtype='object
```

```
In [297]: df_movie_ratings.head(30)
```

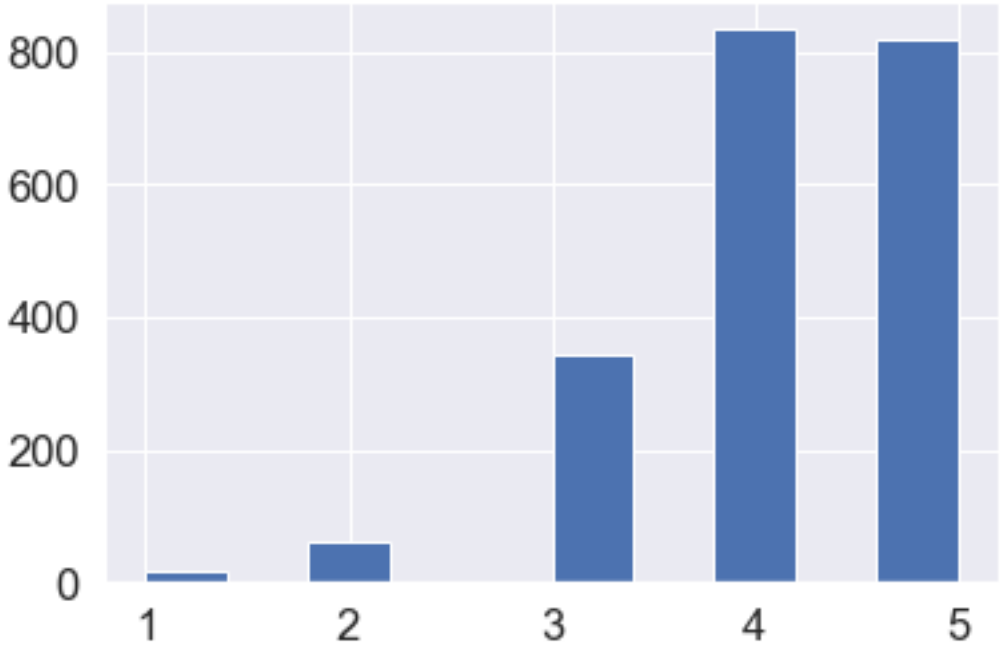
```
Out[297]:     movieId            title                        genres  userId  rating  \
          0         1  Toy Story (1995)  Animation|Children's|Comedy       1       5
          1         1  Toy Story (1995)  Animation|Children's|Comedy       6       4
          2         1  Toy Story (1995)  Animation|Children's|Comedy       8       4
          3         1  Toy Story (1995)  Animation|Children's|Comedy       9       5
          4         1  Toy Story (1995)  Animation|Children's|Comedy      10       5
          5         1  Toy Story (1995)  Animation|Children's|Comedy      18       4
```

6	1	Toy Story (1995)	Animation|Children's|Comedy	19	5
7	1	Toy Story (1995)	Animation|Children's|Comedy	21	3
8	1	Toy Story (1995)	Animation|Children's|Comedy	23	4
9	1	Toy Story (1995)	Animation|Children's|Comedy	26	3
10	1	Toy Story (1995)	Animation|Children's|Comedy	28	3
11	1	Toy Story (1995)	Animation|Children's|Comedy	34	5
12	1	Toy Story (1995)	Animation|Children's|Comedy	36	5
13	1	Toy Story (1995)	Animation|Children's|Comedy	38	5
14	1	Toy Story (1995)	Animation|Children's|Comedy	44	5
15	1	Toy Story (1995)	Animation|Children's|Comedy	45	4
16	1	Toy Story (1995)	Animation|Children's|Comedy	48	4
17	1	Toy Story (1995)	Animation|Children's|Comedy	49	5
18	1	Toy Story (1995)	Animation|Children's|Comedy	51	5
19	1	Toy Story (1995)	Animation|Children's|Comedy	56	5
20	1	Toy Story (1995)	Animation|Children's|Comedy	60	4
21	1	Toy Story (1995)	Animation|Children's|Comedy	65	5
22	1	Toy Story (1995)	Animation|Children's|Comedy	68	3
23	1	Toy Story (1995)	Animation|Children's|Comedy	73	3
24	1	Toy Story (1995)	Animation|Children's|Comedy	75	5
25	1	Toy Story (1995)	Animation|Children's|Comedy	76	5
26	1	Toy Story (1995)	Animation|Children's|Comedy	78	4
27	1	Toy Story (1995)	Animation|Children's|Comedy	80	3
28	1	Toy Story (1995)	Animation|Children's|Comedy	90	3
29	1	Toy Story (1995)	Animation|Children's|Comedy	92	4

	timestamp
0	978824268
1	978237008
2	978233496
3	978225952
4	978226474
5	978154768
6	978555994
7	978139347
8	978463614
9	978130703
10	978985309
11	978102970
12	978061285
13	978046225
14	978019369
15	977990044
16	977975909
17	977972501
18	977947828
19	977938855
20	977931983
21	991368774

```
22  991376026
23  977867812
24  977851099
25  977847069
26  978570648
27  977786904
28  993872933
29  977646817
```

In [298]:
```
#Find and visualize the user rating of the movie "Toy Story"

df_movie_ratings[df_movie_ratings['title'] == 'Toy Story (1995)'].rating.hist();
```



In [ ]:

In [299]:
```
#Find and visualize the viewership of the movie by age group

labels = ['0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79']
df_movie_ratings_users['age_group'] = pd.cut(df_movie_ratings_users.Age, range(0,81,

df_movie_ratings_users.head()

df_movie_ratings_users.groupby('age_group').agg({'rating': [np.size, np.mean]})
```

Out[299]:
```
              rating
                size mean
```

```
age_group
[0, 10)    27,211.00 3.55
[10, 20)  183,536.00 3.51
[20, 30)  395,556.00 3.55
[30, 40)  199,003.00 3.62
[40, 50)   83,633.00 3.64
[50, 60)  111,270.00 3.73
[60, 70)         nan  nan
[70, 80)         nan  nan
```

In [300]:
```
#Find and Visualize the top 25 movies by viewership rating

df_movie_ratings_users.groupby('title').size().sort_values(ascending=False)[:25]
```

Out[300]:
```
title
American Beauty (1999)                                   3428
Star Wars: Episode IV - A New Hope (1977)                2991
Star Wars: Episode V - The Empire Strikes Back (1980)    2990
Star Wars: Episode VI - Return of the Jedi (1983)        2883
Jurassic Park (1993)                                     2672
Saving Private Ryan (1998)                               2653
Terminator 2: Judgment Day (1991)                        2649
Matrix, The (1999)                                       2590
Back to the Future (1985)                                2583
Silence of the Lambs, The (1991)                         2578
Men in Black (1997)                                      2538
Raiders of the Lost Ark (1981)                           2514
Fargo (1996)                                             2513
Sixth Sense, The (1999)                                  2459
Braveheart (1995)                                        2443
Shakespeare in Love (1998)                               2369
Princess Bride, The (1987)                               2318
Schindler's List (1993)                                  2304
L.A. Confidential (1997)                                 2288
Groundhog Day (1993)                                     2278
E.T. the Extra-Terrestrial (1982)                        2269
Star Wars: Episode I - The Phantom Menace (1999)         2250
Being John Malkovich (1999)                              2241
Shawshank Redemption, The (1994)                         2227
Godfather, The (1972)                                    2223
dtype: int64
```

In [301]:
```
#Find a rating for a particular user id

df_movie_ratings_users[df_movie_ratings_users['userId'] == 2696]
```

Out[301]:
```
        movieId                                    title  \
991035      350                    Client, The (1994)
991036      800                      Lone Star (1996)
```

991037	1092	Basic Instinct (1992)
991038	1097	E.T. the Extra-Terrestrial (1982)
991039	1258	Shining, The (1980)
991040	1270	Back to the Future (1985)
991041	1589	Cop Land (1997)
991042	1617	L.A. Confidential (1997)
991043	1625	Game, The (1997)
991044	1644	I Know What You Did Last Summer (1997)
991045	1645	Devil's Advocate, The (1997)
991046	1711	Midnight in the Garden of Good and Evil (1997)
991047	1783	Palmetto (1998)
991048	1805	Wild Things (1998)
991049	1892	Perfect Murder, A (1998)
991050	2338	I Still Know What You Did Last Summer (1998)
991051	2389	Psycho (1998)
991052	2713	Lake Placid (1999)
991053	3176	Talented Mr. Ripley, The (1999)
991054	3386	JFK (1991)

	genres	userId	rating	timestamp	Gender	\
991035	Drama|Mystery|Thriller	2696	3	973308886	M	
991036	Drama|Mystery	2696	5	973308842	M	
991037	Mystery|Thriller	2696	4	973308886	M	
991038	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M	
991039	Horror	2696	4	973308710	M	
991040	Comedy|Sci-Fi	2696	2	973308676	M	
991041	Crime|Drama|Mystery	2696	3	973308865	M	
991042	Crime|Film-Noir|Mystery|Thriller	2696	4	973308842	M	
991043	Mystery|Thriller	2696	4	973308842	M	
991044	Horror|Mystery|Thriller	2696	2	973308920	M	
991045	Crime|Horror|Mystery|Thriller	2696	4	973308904	M	
991046	Comedy|Crime|Drama|Mystery	2696	4	973308904	M	
991047	Film-Noir|Mystery|Thriller	2696	4	973308865	M	
991048	Crime|Drama|Mystery|Thriller	2696	4	973308886	M	
991049	Mystery|Thriller	2696	4	973308904	M	
991050	Horror|Mystery|Thriller	2696	2	973308920	M	
991051	Crime|Horror|Thriller	2696	4	973308710	M	
991052	Horror|Thriller	2696	1	973308710	M	
991053	Drama|Mystery|Thriller	2696	4	973308865	M	
991054	Drama|Mystery	2696	1	973308842	M	

	Age	Occupation	Zip-code	age_group
991035	25	7	24210	[20, 30)
991036	25	7	24210	[20, 30)
991037	25	7	24210	[20, 30)
991038	25	7	24210	[20, 30)
991039	25	7	24210	[20, 30)
991040	25	7	24210	[20, 30)

```
991041   25          7    24210  [20, 30)
991042   25          7    24210  [20, 30)
991043   25          7    24210  [20, 30)
991044   25          7    24210  [20, 30)
991045   25          7    24210  [20, 30)
991046   25          7    24210  [20, 30)
991047   25          7    24210  [20, 30)
991048   25          7    24210  [20, 30)
991049   25          7    24210  [20, 30)
991050   25          7    24210  [20, 30)
991051   25          7    24210  [20, 30)
991052   25          7    24210  [20, 30)
991053   25          7    24210  [20, 30)
991054   25          7    24210  [20, 30)
```

In [302]:
```
#Machine Learning

#imort algorithm

from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

#import the ML algorithm


from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.cross_validation import LeaveOneOut
from sklearn.model_selection import train_test_split


#import libraries for metrices and reporting
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc
```

```
In [303]: df_movie_ratings_users.shape

Out[303]: (1000209, 11)

In [304]: feature_cols = [ 'movieId', 'Age', 'Occupation']
          response_col = ['rating']

In [234]: df_movie_ratings_users.rating.unique()

Out[234]: array([5, 4, 3, 2, 1])

In [305]: #class balance
          df_movie_ratings_users.rating.value_counts()

Out[305]: 4    348971
          3    261197
          5    226310
          2    107557
          1     56174
          Name: rating, dtype: int64

In [306]: df_movie_ratings_users_extract = df_movie_ratings_users.sample(n=500, random_state=0)

In [307]: #extract 500 random rows


          X = df_movie_ratings_users_extract[feature_cols].values
          y = df_movie_ratings_users_extract[response_col].values.ravel()

In [308]: #create objects of required models
          models = []
          models.append(('LR', LogisticRegression()))
          models.append(('Random Forest', RandomForestClassifier()))
          models.append(('KNN', KNeighborsClassifier()))
          models.append(('DecisionTree', DecisionTreeClassifier()))
          models.append(('NB', GaussianNB()))

In [309]: results = []
          for name, model in models:
              kfold = KFold(n_splits=2,random_state=0)

              cv_result = cross_val_score(model, X, y, cv = kfold, scoring = "accuracy")

              results.append(tuple([name, cv_result.mean(), cv_result.std()]))


In [310]: results.sort(key=lambda x: x[1], reverse = True)
          for i in range(len(results)):
              print('{:20s} {:2.2f}(+/-){:2.2f}'.format(results[i][0],results[i][1]*100, result
```

```
LR                  34.00(+/-)1.60
NB                  33.40(+/-)1.80
KNN                 29.80(+/-)0.60
Random Forest       27.80(+/-)1.00
DecisionTree        26.60(+/-)1.80
```

```
In [311]: df_movie_ratings_users.dtypes

Out[311]: movieId          int64
          title           object
          genres          object
          userId           int64
          rating           int64
          timestamp        int64
          Gender          object
          Age              int64
          Occupation       int64
          Zip-code        object
          age_group     category
          dtype: object

In [312]: feature_cols = ['genres','rating', 'Gender','Age','Occupation']
          df_movie_ratings_users_extract = df_movie_ratings_users[feature_cols].copy(deep=True)

In [313]: df_movie_ratings_users_extract.shape

Out[313]: (1000209, 5)

In [244]: df_movie_ratings_users_extract.dtypes

Out[244]: genres        object
          rating         int64
          Gender        object
          Age            int64
          Occupation     int64
          dtype: object

In [314]: # Random Forest

          def explode(df, lst_cols, fill_value=''):

              # make sure `lst_cols` is a list

              if lst_cols and not isinstance(lst_cols, list):
                  lst_cols = [lst_cols]

              # all columns except `lst_cols`
              idx_cols = df.columns.difference(lst_cols)
```

```
    # calculate lengths of lists
    lens = df[lst_cols[0]].str.len()

    if (lens > 0).all():

        # ALL lists in cells aren't empty
        return pd.DataFrame({
            col:np.repeat(df[col].values, lens)
            for col in idx_cols
        }).assign(**{col:np.concatenate(df[col].values) for col in lst_cols}) \
          .loc[:, df.columns]
    else:
        # at least one list in cells is empty
        return pd.DataFrame({
            col:np.repeat(df[col].values, lens)
            for col in idx_cols
        }).assign(**{col:np.concatenate(df[col].values) for col in lst_cols}) \
          .append(df.loc[lens==0, idx_cols]).fillna(fill_value) \
          .loc[:, df.columns]
```

In [ ]:

In [315]:
```
df_movie_ratings_users_extract.genres =     df_movie_ratings_users_extract.genres.str
df_movie_ratings_users_extract = explode(df_movie_ratings_users_extract,['genres'])
```

In [317]:
```
df_movie_ratings_users_extract.head()
```

Out[317]:

	genres	rating	Gender	Age	Occupation
0	Animation	5	F	1	10
1	Children's	5	F	1	10
2	Comedy	5	F	1	10
3	Animation	5	F	1	10
4	Children's	5	F	1	10

In [318]:
```
df_movie_ratings_users_extract.dtypes
```

Out[318]:
```
genres        object
rating         int64
Gender        object
Age            int64
Occupation     int64
dtype: object
```

In [319]:
```
df_movie_ratings_users_extract.genres.unique()
```

Out[319]:
```
array(['Animation', "Children's", 'Comedy', 'Musical', 'Romance', 'Drama',
       'Action', 'Adventure', 'Fantasy', 'Sci-Fi', 'War', 'Crime',
       'Thriller', 'Western', 'Horror', 'Mystery', 'Documentary',
       'Film-Noir'], dtype=object)
```

```
In [320]: df_movie_ratings_users_extract.shape

Out[320]: (2101815, 5)

In [321]: #convert the object datatypes to numeric
          df_movie_ratings_users_extract = pd.get_dummies(df_movie_ratings_users_extract, colu

          df_movie_ratings_users_extract.head()
```

Out[321]:

	rating	Age	Occupation	genres_Action	genres_Adventure	genres_Animation	\
0	5	1	10	0	0	1	
1	5	1	10	0	0	0	
2	5	1	10	0	0	0	
3	5	1	10	0	0	1	
4	5	1	10	0	0	0	

	genres_Children's	genres_Comedy	genres_Crime	genres_Documentary	\
0	0	0	0	0	
1	1	0	0	0	
2	0	1	0	0	
3	0	0	0	0	
4	1	0	0	0	

	...	genres_Horror	genres_Musical	genres_Mystery	genres_Romance	\
0	...	0	0	0	0	
1	...	0	0	0	0	
2	...	0	0	0	0	
3	...	0	0	0	0	
4	...	0	0	0	0	

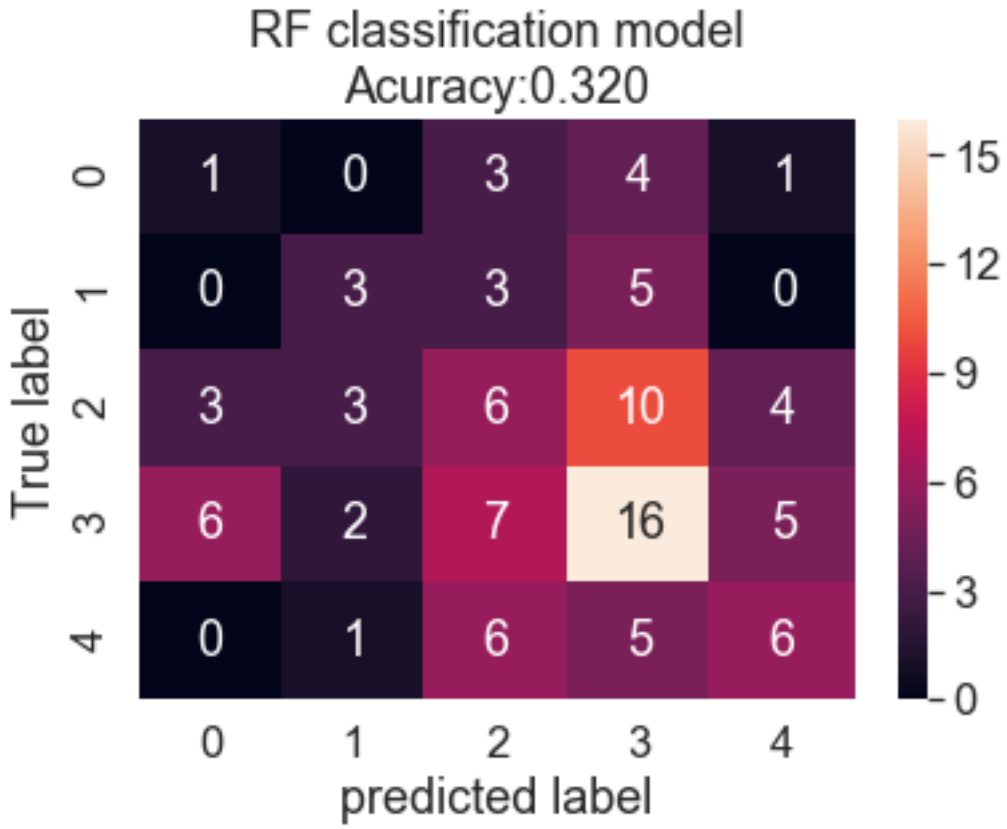
	genres_Sci-Fi	genres_Thriller	genres_War	genres_Western	Gender_F	\
0	0	0	0	0	1	
1	0	0	0	0	1	
2	0	0	0	0	1	
3	0	0	0	0	1	
4	0	0	0	0	1	

	Gender_M
0	0
1	0
2	0
3	0
4	0

[5 rows x 23 columns]

```
In [324]: df_movie_ratings_users_extract = df_movie_ratings_users_extract.sample(n=500, random

In [325]: response_col = ['rating']
```

```
          X = df_movie_ratings_users_extract.drop('rating', axis=1).values
          y = df_movie_ratings_users_extract[response_col].values.ravel()

In [326]: #split into train and test data

          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =0.20, random_sta

          print(X_train.shape)
          print(y_train.shape)
          print(X_test.shape)
          print(y_test.shape)

(400, 22)
(400,)
(100, 22)
(100,)


In [327]: #instatiate the RF classifier

          clf = RandomForestClassifier(n_estimators = 100)

In [329]: #train the classifier

          clf.fit(X_train, y_train)

Out[329]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
                      oob_score=False, random_state=None, verbose=0,
                      warm_start=False)

In [330]: y_pred = clf.predict(X_test)

In [ ]: #Create the confusion matrix

In [331]: cm = confusion_matrix(y_test,y_pred)

In [336]: plt.figure(figsize=(6,4))
          sns.heatmap(cm,annot= True)
          plt.title('RF classification model\nAcuracy:{0:.3f}'.format(accuracy_score(y_test, y_
          plt.ylabel('True label')
          plt.xlabel('predicted label')

Out[336]: Text(0.5,7.5,'predicted label')
```



In [338]:
```
#metric from confusion matrix

print(metrics.accuracy_score(y_test, y_pred))

print(1-metrics.accuracy_score(y_test, y_pred))
```

```
0.32
0.6799999999999999
```

In [342]:
```
#Feature importances

importances = list(clf.feature_importances_)
importances
```

Out[342]:
```
[0.22449715977895848,
 0.45190745229250884,
 0.025297003567072634,
 0.02108259912161549,
 0.014312866023511511,
 0.009239795277602903,
```

```
 0.025945667685944213,
 0.014714967844846532,
 0.0,
 0.026822816976099357,
 0.004458214286141736,
 0.009213053672162724,
 0.014852634431696352,
 0.011311192314261505,
 0.008799174329508172,
 0.01978418779652509,
 0.016614848335737786,
 0.023692817623262693,
 0.017056471417262745,
 0.0042689719145622505,
 0.028113513572577943,
 0.02801459173814101]
```

In [343]:
```
feature_list = df_movie_ratings_users_extract.drop('rating', axis=1).columns
feature_list
```

Out[343]:
```
Index(['Age', 'Occupation', 'genres_Action', 'genres_Adventure',
       'genres_Animation', 'genres_Children's', 'genres_Comedy',
       'genres_Crime', 'genres_Documentary', 'genres_Drama', 'genres_Fantasy',
       'genres_Film-Noir', 'genres_Horror', 'genres_Musical', 'genres_Mystery',
       'genres_Romance', 'genres_Sci-Fi', 'genres_Thriller', 'genres_War',
       'genres_Western', 'Gender_F', 'Gender_M'],
      dtype='object')
```

In [348]:
```
feature_importances =[(feature, round(importance, 2)) for feature, importance in zip
feature_importances
```

Out[348]:
```
[('Age', 0.22),
 ('Occupation', 0.45),
 ('genres_Action', 0.03),
 ('genres_Adventure', 0.02),
 ('genres_Animation', 0.01),
 ("genres_Children's", 0.01),
 ('genres_Comedy', 0.03),
 ('genres_Crime', 0.01),
 ('genres_Documentary', 0.0),
 ('genres_Drama', 0.03),
 ('genres_Fantasy', 0.0),
 ('genres_Film-Noir', 0.01),
 ('genres_Horror', 0.01),
 ('genres_Musical', 0.01),
 ('genres_Mystery', 0.01),
 ('genres_Romance', 0.02),
 ('genres_Sci-Fi', 0.02),
 ('genres_Thriller', 0.02),
```

```
 ('genres_War', 0.02),
 ('genres_Western', 0.0),
 ('Gender_F', 0.03),
 ('Gender_M', 0.03)]
```

In [350]:
```
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = Tru
feature_importances
```

Out[350]:
```
[('Occupation', 0.45),
 ('Age', 0.22),
 ('genres_Action', 0.03),
 ('genres_Comedy', 0.03),
 ('genres_Drama', 0.03),
 ('Gender_F', 0.03),
 ('Gender_M', 0.03),
 ('genres_Adventure', 0.02),
 ('genres_Romance', 0.02),
 ('genres_Sci-Fi', 0.02),
 ('genres_Thriller', 0.02),
 ('genres_War', 0.02),
 ('genres_Animation', 0.01),
 ("genres_Children's", 0.01),
 ('genres_Crime', 0.01),
 ('genres_Film-Noir', 0.01),
 ('genres_Horror', 0.01),
 ('genres_Musical', 0.01),
 ('genres_Mystery', 0.01),
 ('genres_Documentary', 0.0),
 ('genres_Fantasy', 0.0),
 ('genres_Western', 0.0)]
```

In [ ]: