In [1]:
```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
os.getcwd()
```

Out[2]:
```
'C:\\Users\\Preksha\\Simplilearn\\Data Science with Python\\MovieLens'
```

In [3]:
```
movie = ['MovieID', 'Title', 'Genres']
```

In [4]:
```
rating = ['UserID', 'MovieID', 'Rating', 'Timestamp']
```

In [5]:
```
user = ['UserID', 'Gender', 'Age', 'Occupation', 'Zip-code']
```

Import the three datasets

In [6]:
```
movies = pd.read_csv('movies.dat', delimiter = '::', names = movie)
```

```
<ipython-input-6-7f97736ab933>:1: ParserWarning: Falling back to the 'pytho
n' engine because the 'c' engine does not support regex separators (separat
ors > 1 char and different from '\s+' are interpreted as regex); you can av
oid this warning by specifying engine='python'.
  movies = pd.read_csv('movies.dat', delimiter = '::', names = movie)
```

In [7]:
```
ratings = pd.read_csv('ratings.csv', names = rating)
```

In [8]:
```
users = pd.read_csv('users.csv', names = user)
```

In [9]:
```
movies.head()
```

Out[9]:

	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 [10]:

```
ratings.head()
```

Out[10]:

	UserID	MovieID	Rating	Timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

In [11]:

```
users.head()
```

Out[11]:

	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

Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating. (Hint: (i) Merge two tables at a time. (ii) Merge the tables using two primary keys MovieID & UserId)

In [12]:

```
master_data = pd.merge(ratings, movies, how='outer', on='MovieID')
```

In [13]:

```
master_data.head()
```

Out[13]:

	UserID	MovieID	Rating	Timestamp	Title	Genres
0	1.0	1193	5.0	978300760.0	One Flew Over the Cuckoo's Nest (1975)	Drama
1	2.0	1193	5.0	978298413.0	One Flew Over the Cuckoo's Nest (1975)	Drama
2	12.0	1193	4.0	978220179.0	One Flew Over the Cuckoo's Nest (1975)	Drama
3	15.0	1193	4.0	978199279.0	One Flew Over the Cuckoo's Nest (1975)	Drama
4	17.0	1193	5.0	978158471.0	One Flew Over the Cuckoo's Nest (1975)	Drama

In [14]:
```
master_data1 = pd.merge(master_data, users, how = 'outer', on = 'UserID')
```

In [15]:
```
master_data1.head()
```

Out[15]:

	UserID	MovieID	Rating	Timestamp	Title	Genres	Gender	Age
0	1.0	1193	5.0	978300760.0	One Flew Over the Cuckoo's Nest (1975)	Drama	F	1.0
1	1.0	661	3.0	978302109.0	James and the Giant Peach (1996)	Animation|Children's|Musical	F	1.0
2	1.0	914	3.0	978301968.0	My Fair Lady (1964)	Musical|Romance	F	1.0
3	1.0	3408	4.0	978300275.0	Erin Brockovich (2000)	Drama	F	1.0
4	1.0	2355	5.0	978824291.0	Bug's Life, A (1998)	Animation|Children's|Comedy	F	1.0

In [16]:
```
master_data = master_data1.drop(['Timestamp', 'Genres', 'Zip-code'], axis = 1
```

In [17]:
```
master_data.head()
```

Out[17]:

	UserID	MovieID	Rating	Title	Gender	Age	Occupation
0	1.0	1193	5.0	One Flew Over the Cuckoo's Nest (1975)	F	1.0	10.0
1	1.0	661	3.0	James and the Giant Peach (1996)	F	1.0	10.0
2	1.0	914	3.0	My Fair Lady (1964)	F	1.0	10.0
3	1.0	3408	4.0	Erin Brockovich (2000)	F	1.0	10.0
4	1.0	2355	5.0	Bug's Life, A (1998)	F	1.0	10.0

In [18]:
```
master_data.isna().sum()
```

Out[18]:
```
UserID        177
MovieID         0
Rating        177
Title           0
Gender        177
Age           177
Occupation    177
dtype: int64
```

In [19]:
```
master_data.size
```

Out[19]:
```
7002702
```

In [20]:
```
master_data.shape
```

Out[20]:
```
(1000386, 7)
```

In [21]:
```
master_data.columns
```

Out[21]:
```
Index(['UserID', 'MovieID', 'Rating', 'Title', 'Gender', 'Age', 'Occupatio
n'], dtype='object')
```

In [22]:
```
master_data.dtypes
```

Out[22]:
```
UserID        float64
MovieID         int64
Rating        float64
Title          object
Gender         object
Age           float64
Occupation    float64
dtype: object
```

In [23]:
```
master_data.dropna(inplace = True)
```

In [24]:
```
master_data.isna().sum()
```

Out[24]:
```
UserID        0
MovieID       0
Rating        0
Title         0
Gender        0
Age           0
Occupation    0
dtype: int64
```

In [25]:
```
master_data.corr()
```

Out[25]:

	UserID	MovieID	Rating	Age	Occupation
UserID	1.000000	-0.017739	0.012303	0.034688	-0.026698
MovieID	-0.017739	1.000000	-0.064042	0.027575	0.008585
Rating	0.012303	-0.064042	1.000000	0.056869	0.006753
Age	0.034688	0.027575	0.056869	1.000000	0.078371
Occupation	-0.026698	0.008585	0.006753	0.078371	1.000000

In [26]:

```
master_data.nunique()
```

Out[26]:

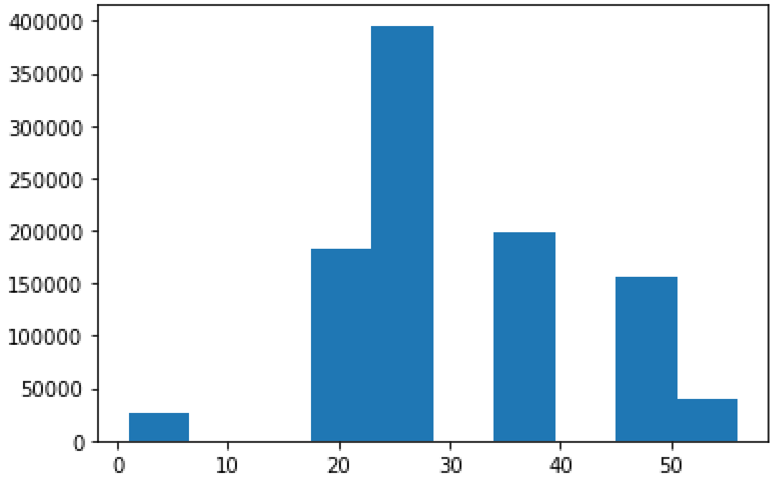
```
UserID        6040
MovieID       3706
Rating           5
Title         3706
Gender           2
Age              7
Occupation      21
dtype: int64
```

Explore the datasets using visual representations (graphs or tables), also include your comments on the following:

User Age Distribution

In [27]:

```
master_data.Age.hist(grid = False);
```



In [28]:

```
rating_movies= master_data.drop(['UserID', 'MovieID', 'Gender', 'Age', 'Occup
```

In [29]:

```
rating_movies.head()
```

Out[29]:

	Rating	Title
0	5.0	One Flew Over the Cuckoo's Nest (1975)
1	3.0	James and the Giant Peach (1996)
2	3.0	My Fair Lady (1964)
3	4.0	Erin Brockovich (2000)
4	5.0	Bug's Life, A (1998)

In [30]:

```
toy_story = rating_movies[rating_movies['Title'].str.startswith('Toy Story',
toy_story
```

Out[30]:

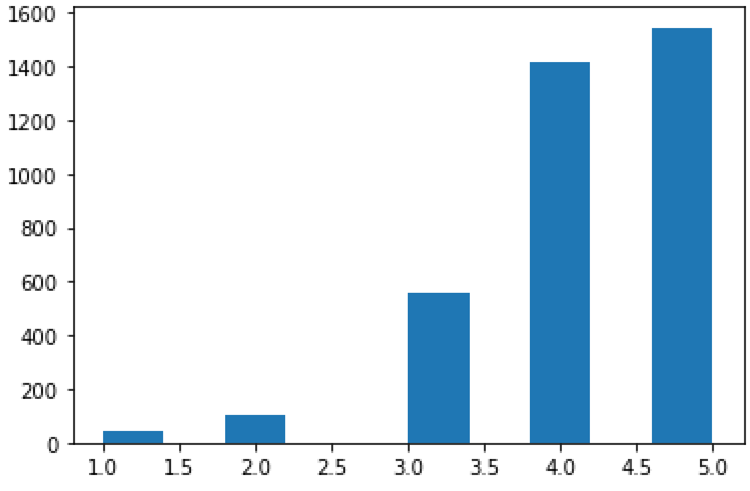
	Rating	Title
40	5.0	Toy Story (1995)
50	4.0	Toy Story 2 (1999)
417	5.0	Toy Story 2 (1999)
634	4.0	Toy Story (1995)
938	5.0	Toy Story (1995)
...	...	...
994256	5.0	Toy Story 2 (1999)
994289	5.0	Toy Story 2 (1999)
994315	4.0	Toy Story 2 (1999)
994367	4.0	Toy Story 2 (1999)
994389	4.0	Toy Story 2 (1999)

3662 rows × 2 columns

User rating of the movie “Toy Story”

In [31]:

```
toy_story.Rating.hist(grid=False);
```



Top 25 movies by viewership rating

In [32]:

```
top_25 = rating_movies.sort_values('Rating', ascending = False)
top_25 = top_25.head(25)
```

In [33]:
```
top_25
```

Out[33]:

	Rating	Title
0	5.0	One Flew Over the Cuckoo's Nest (1975)
327573	5.0	Crucible, The (1996)
327567	5.0	Blow-Out (La Grande Bouffe) (1973)
327564	5.0	Five Easy Pieces (1970)
830521	5.0	Black Cauldron, The (1985)
327559	5.0	Raging Bull (1980)
327554	5.0	Cook the Thief His Wife & Her Lover, The (1989)
830530	5.0	Great Escape, The (1963)
830539	5.0	Conan the Barbarian (1982)
830543	5.0	Sneakers (1992)
830547	5.0	Devil's Advocate, The (1997)
830552	5.0	Space Cowboys (2000)
327507	5.0	Conquest of the Planet of the Apes (1972)
830561	5.0	Secret of NIMH, The (1982)
327490	5.0	Manhattan Murder Mystery (1993)
327487	5.0	Love and Death (1975)
327479	5.0	Mother (1996)
830578	5.0	On Her Majesty's Secret Service (1969)
830583	5.0	Fox and the Hound, The (1981)
830520	5.0	Tron (1982)
327582	5.0	Lolita (1962)
327306	5.0	Jungle Fever (1991)
830514	5.0	Blade Runner (1982)
830468	5.0	Fantasia (1940)
830469	5.0	Dr. Strangelove or: How I Learned to Stop Worr...

In [34]:
```
master_data.columns
```

Out[34]:
```
Index(['UserID', 'MovieID', 'Rating', 'Title', 'Gender', 'Age', 'Occupatio
n'], dtype='object')
```

In [35]:
```
users_rating = master_data.drop(['MovieID','Gender', 'Age', 'Occupation'], ax
```

In [36]:

```
users_rating.head()
```

Out[36]:

	UserID	Rating	Title
0	1.0	5.0	One Flew Over the Cuckoo's Nest (1975)
1	1.0	3.0	James and the Giant Peach (1996)
2	1.0	3.0	My Fair Lady (1964)
3	1.0	4.0	Erin Brockovich (2000)
4	1.0	5.0	Bug's Life, A (1998)

Find the ratings for all the movies reviewed by for a particular user of user id = 2696

In [37]:

```
users_rating = users_rating[users_rating['UserID'] == 2696 ]
```

In [38]:

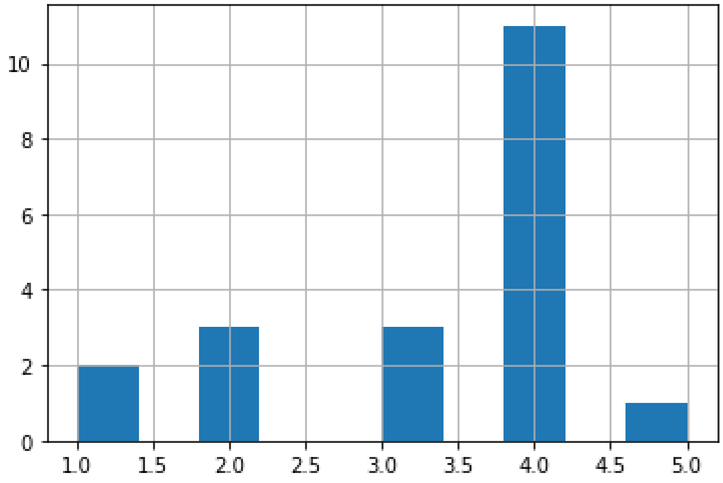
```
users_rating.sort_values('Rating', ascending = False)
```

Out[38]:

	UserID	Rating	Title
953850	2696.0	5.0	Lone Star (1996)
953857	2696.0	4.0	Game, The (1997)
953859	2696.0	4.0	Devil's Advocate, The (1997)
953849	2696.0	4.0	L.A. Confidential (1997)
953864	2696.0	4.0	Shining, The (1980)
953852	2696.0	4.0	Talented Mr. Ripley, The (1999)
953853	2696.0	4.0	Midnight in the Garden of Good and Evil (1997)
953862	2696.0	4.0	Basic Instinct (1992)
953855	2696.0	4.0	Palmetto (1998)
953856	2696.0	4.0	Perfect Murder, A (1998)
953861	2696.0	4.0	Wild Things (1998)
953860	2696.0	4.0	Psycho (1998)
953866	2696.0	3.0	Client, The (1994)
953848	2696.0	3.0	E.T. the Extra-Terrestrial (1982)
953854	2696.0	3.0	Cop Land (1997)
953858	2696.0	2.0	I Know What You Did Last Summer (1997)
953865	2696.0	2.0	I Still Know What You Did Last Summer (1998)
953847	2696.0	2.0	Back to the Future (1985)
953863	2696.0	1.0	Lake Placid (1999)
953851	2696.0	1.0	JFK (1991)

In [39]:

```
users_rating.Rating.hist();
```



In [40]:
```
master_data1.Genres.unique()
```

Out[40]:
```
array(['Drama', "Animation|Children's|Musical", 'Musical|Romance',
       "Animation|Children's|Comedy", 'Action|Adventure|Comedy|Romance',
       'Action|Adventure|Drama', 'Comedy|Drama',
       "Adventure|Children's|Drama|Musical", 'Musical', 'Comedy',
       "Animation|Children's", 'Comedy|Fantasy', 'Animation',
       'Comedy|Sci-Fi', 'Drama|War', 'Romance',
       "Animation|Children's|Musical|Romance",
       "Children's|Drama|Fantasy|Sci-Fi", 'Drama|Romance',
       'Animation|Comedy|Thriller',
       "Adventure|Animation|Children's|Comedy|Musical",
       "Animation|Children's|Comedy|Musical", 'Thriller',
       'Action|Crime|Romance', 'Action|Adventure|Fantasy|Sci-Fi',
       "Children's|Comedy|Musical", 'Action|Drama|War',
       "Children's|Drama", 'Crime|Drama|Thriller', 'Action|Crime|Drama',
       'Action|Adventure|Mystery', 'Crime|Drama',
       'Action|Adventure|Sci-Fi|Thriller',
       'Action|Adventure|Romance|Sci-Fi|War', 'Action|Thriller',
       'Action|Drama', 'Comedy|Drama|Western', 'Action|Adventure|Crime',
       'Action|Crime|Mystery|Thriller', 'Comedy|Drama|Romance',
       'Comedy|Drama|War', 'Drama|Sci-Fi', 'Action|Drama|Thriller',
```

In [41]:
```
master_data1.isna().sum()
```

Out[41]:
```
UserID        177
MovieID         0
Rating        177
Timestamp     177
Title           0
Genres          0
Gender        177
Age           177
Occupation    177
Zip-code      177
dtype: int64
```

In [42]:
```
master_data1.dropna(inplace = True)
```

In [43]:
```
master_data1.isna().any()
```

Out[43]:
```
UserID        False
MovieID       False
Rating        False
Timestamp     False
Title         False
Genres        False
Gender        False
Age           False
Occupation    False
Zip-code      False
dtype: bool
```

In [44]:
```
master_data1.head()
```

Out[44]:

	UserID	MovieID	Rating	Timestamp	Title	Genres	Gender	Age
0	1.0	1193	5.0	978300760.0	One Flew Over the Cuckoo's Nest (1975)	Drama	F	1.0
1	1.0	661	3.0	978302109.0	James and the Giant Peach (1996)	Animation|Children's|Musical	F	1.0
2	1.0	914	3.0	978301968.0	My Fair Lady (1964)	Musical|Romance	F	1.0
3	1.0	3408	4.0	978300275.0	Erin Brockovich (2000)	Drama	F	1.0
4	1.0	2355	5.0	978824291.0	Bug's Life, A (1998)	Animation|Children's|Comedy	F	1.0

Find out all the unique genres (Hint: split the data in column genre making a list and then process the data to find out only the unique categories of genres)

In [45]:
```
Genres = master_data1['Genres'].str.split('|')
```

In [46]:
```
list1 = Genres.tolist()
```

In [47]:
```
list1
```

Out[47]:
```
[['Drama'],
 ['Animation', "Children's", 'Musical'],
 ['Musical', 'Romance'],
 ['Drama'],
 ['Animation', "Children's", 'Comedy'],
 ['Action', 'Adventure', 'Comedy', 'Romance'],
 ['Action', 'Adventure', 'Drama'],
 ['Comedy', 'Drama'],
 ['Animation', "Children's", 'Musical'],
 ['Adventure', "Children's", 'Drama', 'Musical'],
 ['Animation', "Children's", 'Musical'],
 ['Musical'],
 ['Drama'],
 ['Comedy'],
 ['Musical'],
 ['Comedy'],
 ['Animation', "Children's"],
 ['Animation', "Children's"],
 ['Drama'],
```

In [48]:
```
list2 = [ item for elem in list1 for item in elem]
```

In [49]:
```
list2
```

Out[49]:
```
['Drama',
 'Animation',
 "Children's",
 'Musical',
 'Musical',
 'Romance',
 'Drama',
 'Animation',
 "Children's",
 'Comedy',
 'Action',
 'Adventure',
 'Comedy',
 'Romance',
 'Action',
 'Adventure',
 'Drama',
 'Comedy',
 'Drama',
```

In [50]:
```
np.unique(list2)
```

Out[50]:
```
array(['Action', 'Adventure', 'Animation', "Children's", 'Comedy',
       'Crime', 'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror',
       'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War',
       'Western'], dtype='<U11')
```

In [51]:

```
master_data2 = pd.concat([master_data1,master_data1.Genres.str.get_dummies()]
print(master_data2.columns)
```

```
Index(['UserID', 'MovieID', 'Rating', 'Timestamp', 'Title', 'Genres', 'Gend
er',
       'Age', 'Occupation', 'Zip-code', 'Action', 'Adventure', 'Animation',
       'Children's', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
       'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi',
       'Thriller', 'War', 'Western'],
      dtype='object')
```

In [52]:

```
master_data2.head()
```

Out[52]:

	UserID	MovieID	Rating	Timestamp	Title	Genres	Gender	Age
0	1.0	1193	5.0	978300760.0	One Flew Over the Cuckoo's Nest (1975)	Drama	F	1.0
1	1.0	661	3.0	978302109.0	James and the Giant Peach (1996)	Animation|Children's|Musical	F	1.0
2	1.0	914	3.0	978301968.0	My Fair Lady (1964)	Musical|Romance	F	1.0
3	1.0	3408	4.0	978300275.0	Erin Brockovich (2000)	Drama	F	1.0
4	1.0	2355	5.0	978824291.0	Bug's Life, A (1998)	Animation|Children's|Comedy	F	1.0

5 rows × 28 columns

In [53]:

```
master_data2.Occupation.unique()
```

Out[53]:

```
array([10., 16., 12.,  7.,  1.,  3.,  4.,  8., 17.,  0.,  2.,  9., 19.,
       18., 15., 11., 20., 13.,  5., 14.,  6.])
```

Create a separate column for each genre category with a one-hot encoding (1 and 0) whether or not the movie belongs to that genre.

In [54]:

```
master_data2 = master_data2.drop(['MovieID', 'UserID', 'Timestamp', 'Zip-code
```

In [55]:
```
master_data2 = pd.get_dummies(master_data2, columns=['Gender', 'Occupation'])
```

In [56]:
```
master_data2
```

Out[56]:

	Rating	Age	Action	Adventure	Animation	Children's	Comedy	Crime	Documentar
0	5.0	1.0	0	0	0	0	0	0	
1	3.0	1.0	0	0	1	1	0	0	
2	3.0	1.0	0	0	0	0	0	0	
3	4.0	1.0	0	0	0	0	0	0	
4	5.0	1.0	0	0	1	1	1	0	
...	...	...	...	...	...	...	...	...	
1000204	2.0	45.0	0	0	0	0	0	0	
1000205	3.0	45.0	0	0	0	0	0	0	
1000206	4.0	45.0	0	0	0	0	0	0	
1000207	2.0	45.0	1	0	0	0	0	0	
1000208	2.0	45.0	0	1	0	0	0	0	

1000209 rows × 43 columns

Determine the features affecting the ratings of any particular movie.

In [57]:
```
master_data2.columns
```

Out[57]:
```
Index(['Rating', 'Age', 'Action', 'Adventure', 'Animation', 'Children's',
       'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy', 'Film-Noir',
       'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'Wa
r',
       'Western', 'Gender_F', 'Gender_M', 'Occupation_0.0', 'Occupation_1.
0',
       'Occupation_2.0', 'Occupation_3.0', 'Occupation_4.0', 'Occupation_5.
0',
       'Occupation_6.0', 'Occupation_7.0', 'Occupation_8.0', 'Occupation_9.
0',
       'Occupation_10.0', 'Occupation_11.0', 'Occupation_12.0',
       'Occupation_13.0', 'Occupation_14.0', 'Occupation_15.0',
       'Occupation_16.0', 'Occupation_17.0', 'Occupation_18.0',
       'Occupation_19.0', 'Occupation_20.0'],
      dtype='object')
```

In [58]:
```
from sklearn.model_selection import train_test_split
```

In [59]:

```
X = master_data2[['Age', 'Action', 'Adventure', 'Animation',
       "Children's", 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
       'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi',
       'Thriller', 'War', 'Western', 'Gender_F', 'Gender_M', 'Occupation_0.0'
       'Occupation_1.0', 'Occupation_2.0', 'Occupation_3.0', 'Occupation_4.0'
       'Occupation_5.0', 'Occupation_6.0', 'Occupation_7.0', 'Occupation_8.0'
       'Occupation_9.0', 'Occupation_10.0', 'Occupation_11.0',
       'Occupation_12.0', 'Occupation_13.0', 'Occupation_14.0',
       'Occupation_15.0', 'Occupation_16.0', 'Occupation_17.0',
       'Occupation_18.0', 'Occupation_19.0', 'Occupation_20.0']]
```

In [60]:

```
y = master_data2[['Rating']]
```

In [61]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75,
```

In [62]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[62]:

```
((750156, 42), (250053, 42), (750156, 1), (250053, 1))
```

Develop an appropriate model to predict the movie ratings

In [63]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [64]:

```
dtc_model = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
```

In [65]:

```
dtc_model.fit(X_train, y_train)
```

Out[65]:

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [66]:

```
y_pred = dtc_model.predict(X_test)
```

In [67]:

```
from sklearn import metrics
```

In [68]:

```
np.sqrt(metrics.mean_squared_error(y_test, y_pred))
```

Out[68]:

```
1.2361860826434847
```