Project: Movielens Case Study

Table of Contents

Movielens Case Study

DESCRIPTION

Background of Problem Statement:

The GroupLens Research Project is a research group in the Department of Computer Science and Engineering at the University of Minnesota. Members of the GroupLens Research Project are involved in many research projects related to the fields of information filtering, collaborative filtering, and recommender systems. The project is led by professors John Riedl and Joseph Konstan. The project began to explore automated collaborative filtering in 1992 but is most well known for its worldwide trial of an automated collaborative filtering system for Usenet news in 1996. Since then the project has expanded its scope to research overall information by filtering solutions, integrating into content-based methods, as well as, improving current collaborative filtering technology.

Problem Objective:

Here, we ask you to perform the analysis using the Exploratory Data Analysis technique. You need to find features affecting the ratings of any particular movie and build a model to predict the movie ratings.

Domain: Entertainment

Analysis Tasks to be performed:

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

1. User Age Distribution
2. User rating of the movie “Toy Story”
3. Top 25 movies by viewership rating
4. Find the ratings for all the movies reviewed by for a particular user of user id = 2696

- Feature Engineering:
  Use column genres:

1. 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)
2. 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.
3. Determine the features affecting the ratings of any particular movie.
4. Develop an appropriate model to predict the movie ratings

Dataset Description:

These files contain 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 Movielens users who joined Movielens in 2000.

Ratings.dat

Format - UserID::MovieID::Rating::Timestamp

Field	Description
UserID	Unique identification for each user
MovieID	Unique identification for each movie

Rating	User rating for each movie
Timestamp	Timestamp generated while adding user review

- UserIDs range between 1 and 6040
- The MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only)
- A timestamp is represented in seconds since the epoch is returned by time(2)
- Each user has at least 20 ratings

Users.dat

Format - UserID::Gender::Age::Occupation::Zip-code

Field	Description
UserID	Unique identification for each user
Genere	Category of each movie
Age	User's age
Occupation	User's Occupation
Zip-code	Zip Code for the user's location

All demographic information is provided voluntarily by the users and is not checked for accuracy. Only users who have provided demographic information are included in this data set.

- Gender is denoted by an "M" for male and "F" for female
- Age is chosen from the following ranges:

Value	Description
1	"Under 18"
18	"18-24"
25	"25-34"
35	"35-44"
45	"45-49"
50	"50-55"
56	"56+"

- Occupation is chosen from the following choices:

Value	Description
0	"other" or not specified
1	"academic/educator"
2	"artist”
3	"clerical/admin"
4	"college/grad student"
5	"customer service"
6	"doctor/health care"
7	"executive/managerial"
8	"farmer"
9	"homemaker"
10	"K-12 student"
11	"lawyer"
12	"programmer"
13	"retired"
14	"sales/marketing"
15	"scientist"
16	"self-employed"
17	"technician/engineer"
18	"tradesman/craftsman"
19	"unemployed"
20	"writer”

Movies.dat
Format - MovieID::Title::Genres

Field	Description
MovieID	Unique identification for each movie
Title	A title for each movie
Genres	Category of each movie

- Titles are identical to titles provided by the IMDB (including year of release)

- Genres are pipe-separated and are selected from the following genres:

1. Action
2. Adventure
3. Animation
4. Children's
5. Comedy
6. Crime
7. Documentary
8. Drama
9. Fantasy
10. Film-Noir
11. Horror
12. Musical
13. Mystery
14. Romance
15. Sci-Fi
16. Thriller
17. War
18. Western

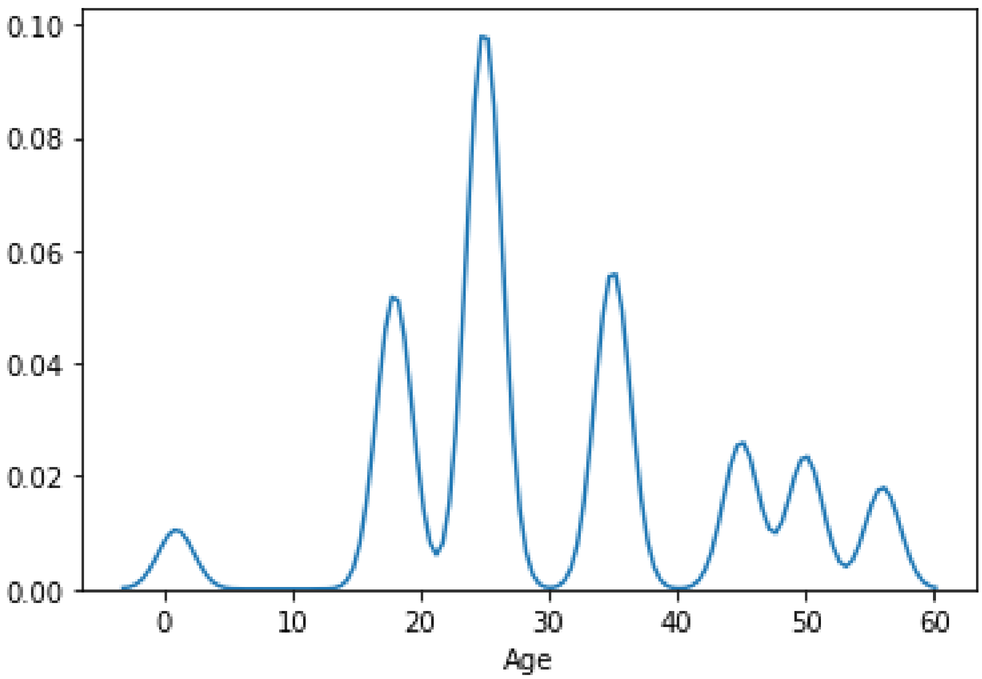
- Some Movie IDs do not correspond to a movie due to accidental duplicate entries and/or test entries
- Movies are mostly entered by hand, so errors and inconsistencies may exist

The Actual Project Code - Project_Movielens.ipynb

```
#import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings(action = "ignore")
```

```
#importing the data files
movies = pd.read_table("movies.dat", sep = "::", names = (["MovieID", "Title", "Genre"]))
ratings = pd.read_table("ratings.dat", sep = "::", names = (["UserID", "MovieID", "Rating", "TimeStamp"]))
users = pd.read_table("users.dat", sep = "::", names = (["UserID", "Gender", "Age", "Occupation", "ZipCode"]))

users.shape
type(users)
```

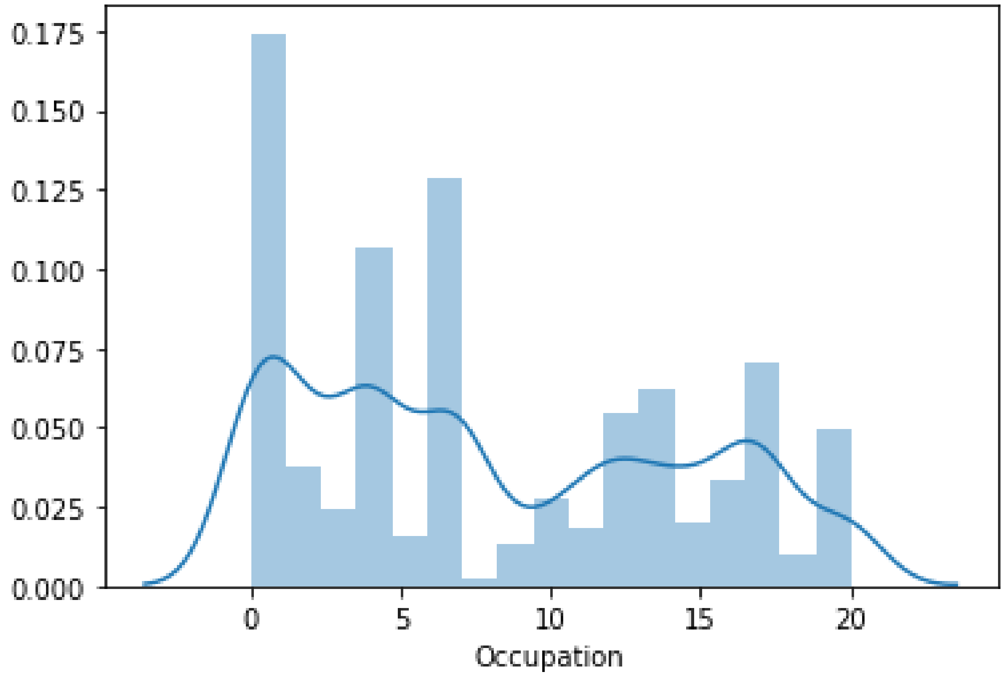
```
#Exploratory Data Analysis
#User Age distribution using histogram

sns.distplot(users["Age"], hist = False)

<matplotlib.axes._subplots.AxesSubplot at 0x1c97e3d2408>
```



#This shows most of the users are between 20 years and 30 years of age.

```
sns.distplot(users["Occupation"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x268ada71808>
```




```
movies.shape
```

```
 (3883, 3)
```

```
ratings.shape
```

```
 (1000209, 4)
```

```
movies.head(6)
```

	MovieID	Title	Genre
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
5	6	Heat (1995)	Action|Crime|Thriller

ratings.head(6)

	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
5	1	1197	3	978302268

#merging movies with ratings on MovieID
movie_ratings = pd.merge(movies, ratings, on = "MovieID")

movie_ratings.head(6)

	MovieID	Title	Genre	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
3	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952
4	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474
5	1	Toy Story (1995)	Animation|Children's|Comedy	18	4	978154768

#merging movies with ratings on UserID

combinedData = pd.merge(movie_ratings, users, on = "UserID") combinedData.head(6)

	MovieID	Title	Genre	UserID	Rating	TimeStamp	Gender	Age	Occupation	ZipCode
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
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067
5	531	Secret Garden, The (1993)	Children's|Drama	1	4	978302149	F	1	10	48067

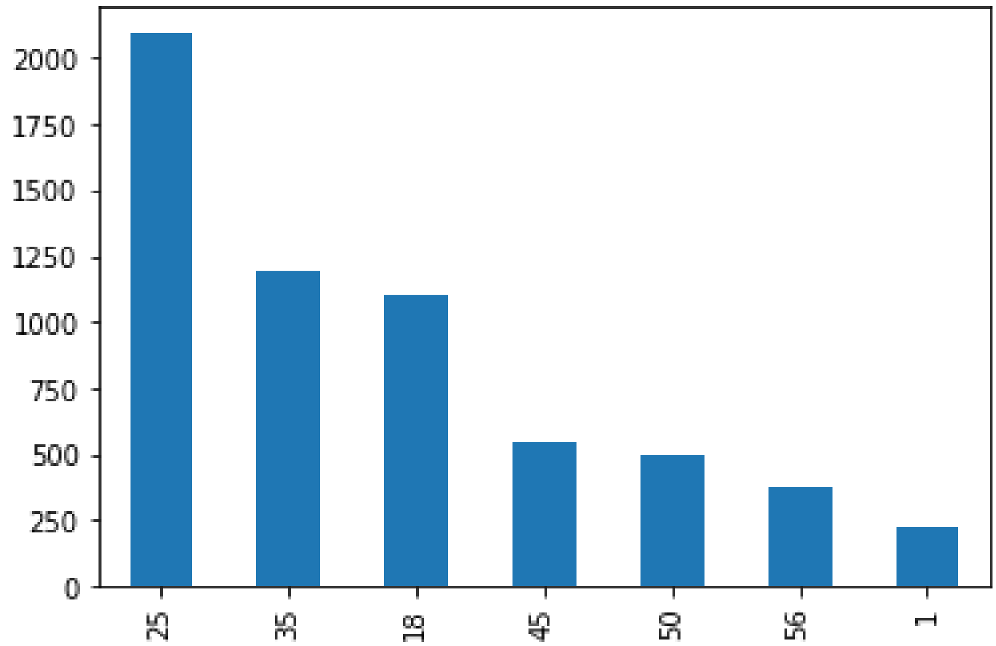
combinedData.shape

(1000209, 10)

df = users.Age.value_counts()

users["Age"].value_counts().plot(kind='bar')

<matplotlib.axes._subplots.AxesSubplot at 0x1c97e6e9888>



#df = pd.DataFrame(columns=["AgeGroup", "Freq"])
print(df)

#checking the Age Distribution.
#Age Group Under 18 - 222 people
#Age Group 18-24 - 1103 people
#Age Group 25-34 - 2096 people
#Age Group 35-44 - 1193 people
#Age Group 45-49 - 550 people
#Age Group 50-55 - 496 people
#Age Group 56+ - 380 people

```
25    2096
35    1193
18    1103
45     550
50     496
56     380
1      222
Name: Age, dtype: int64
```

type(df)

pandas.core.series.Series

combinedData.Rating.value_counts()

combinedData.head()

	MovieID	Title	Genre	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
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067

#group by movieID and sort then sort.
sortby_rating = combinedData.sort_values("Rating", ascending=**False**)

#extracting boolean to UserID 2696
filter_data_2696 = combinedData["UserID"]==2696

```
print(filter_data_2696.head(6))
0    False
1    False
2    False
3    False
4    False
5    False
Name: UserID, dtype: bool
```

#preparing the dataframe of the data
filter_data_2696_1 = combinedData[filter_data_2696]

filter_data_2696_1.shape

(20, 10)

#movies reviewed by UserID 2696
filter_data_2696_1

	MovieID	Title	Genre	UserID	Rating	TimeStamp	Gender	Age	Occupation	ZipCode
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	973308886	M	25	7	24210
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	973308842	M	25	7	24210
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	973308886	M	25	7	24210
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M	25	7	24210
99	1258	Shinin	Horror	26	4	9733087	M	25	7	24210

	MovieID	Title	Genre	UserID	Rating	TimeStamp	Gender	Age	Occupation	ZipCode
1039		g, The (1980)		96		10				
991040	1270	Back to the Future (1985)	Comedy|Sci-Fi	2696	2	973308676	M	25	7	24210
991041	1589	Cop Land (1997)	Crime|Drama|Mystery	2696	3	973308865	M	25	7	24210
991042	1617	L. A. Confidential (1997)	Crime|Film-Noir|Mystery|Thriller	2696	4	973308842	M	25	7	24210
991043	1625	Game, The (1997)	Mystery|Thriller	2696	4	973308842	M	25	7	24210
991	1644	I Know	Horror|Mystery|Thriller	2696	2	973308920	M	25	7	24210

	MovieID	Title	Genre	UserID	Rating	TimeStamp	Gender	Age	Occupation	ZipCode
044		What You Did Last Summer (1997)								
991045	1645	Devil's Advocate, The (1997)	Crime|Horror|Mystery|Thriller	2696	4	973308904	M	25	7	24210
991046	1711	Midnight in the Garden of Good and Evil (1997)	Comedy|Crime|Drama|Mystery	2696	4	973308904	M	25	7	24210

	MovieID	Title	Genre	UserID	Rating	TimeStamp	Gender	Age	Occupation	ZipCode
991047	1783	Palmetto (1998)	Film-Noir|Mystery|Thriller	2696	4	973308865	M	25	7	24210
991048	1805	Wild Things (1998)	Crime|Drama|Mystery|Thriller	2696	4	973308886	M	25	7	24210
991049	1892	Perfect Murder, A (1998)	Mystery|Thriller	2696	4	973308904	M	25	7	24210
991050	2338	I Still Know What You Did Last Summer (1998)	Horror|Mystery|Thriller	2696	2	973308920	M	25	7	24210

<table>
<tr><th></th><th>MovieID</th><th>Title</th><th>Genre</th><th>UserID</th><th>Rating</th><th>TimeStamp</th><th>Gender</th><th>Age</th><th>Occupation</th><th>ZipCode</th></tr>
<tr><td>991051</td><td>2389</td><td>Psycho (1998)</td><td>Crime|Horror|Thriller</td><td>2696</td><td>4</td><td>973308710</td><td>M</td><td>25</td><td>7</td><td>24210</td></tr>
<tr><td>991052</td><td>2713</td><td>Lake Placid (1999)</td><td>Horror|Thriller</td><td>2696</td><td>1</td><td>973308710</td><td>M</td><td>25</td><td>7</td><td>24210</td></tr>
<tr><td>991053</td><td>3176</td><td>Talented Mr. Ripley, The (1999)</td><td>Drama|Mystery|Thriller</td><td>2696</td><td>4</td><td>973308865</td><td>M</td><td>25</td><td>7</td><td>24210</td></tr>
<tr><td>991054</td><td>3386</td><td>JFK (1991)</td><td>Drama|Mystery</td><td>2696</td><td>1</td><td>973308842</td><td>M</td><td>25</td><td>7</td><td>24210</td></tr>
</table>

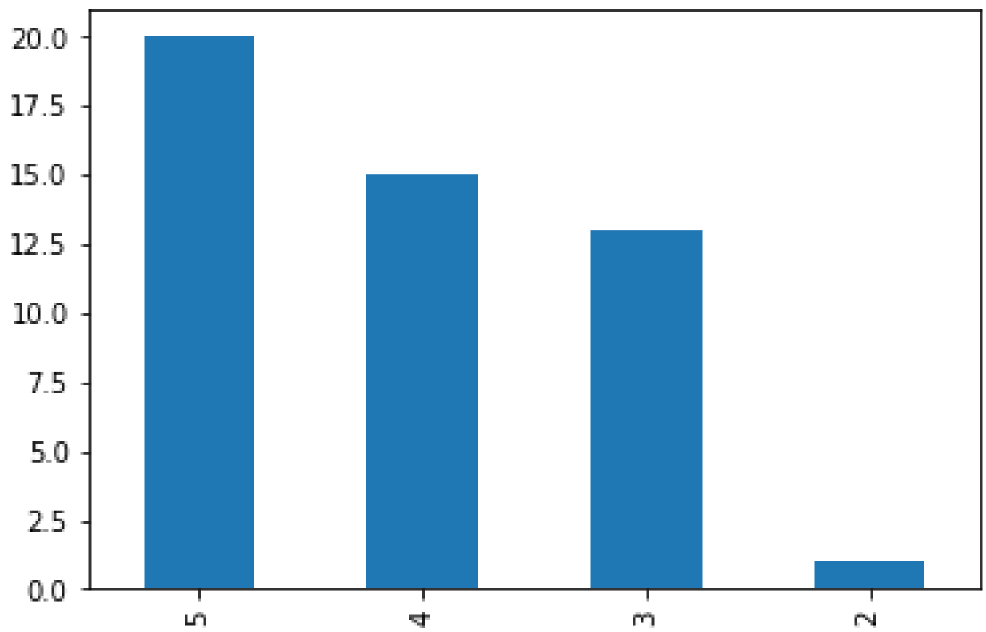
#User ratings for Toy Story

```
toystory = data10k[data10k.Title == "Toy Story (1995)"]

type(toystory)
toystory.shape

toystory
toystory["Rating"].value_counts().plot(kind = "bar")
```

#The ratings of Toy Story

#The below graph is showing the ratings of Toy Story given by the users.

<matplotlib.axes._subplots.AxesSubplot at 0x1c97e7ca388>



#extracting 10K data from Combined Data
data10k = combinedData[:10000]

#data10k.MovieID.value_counts()
data10k.shape
#groupby10k = data10k.groupby(["MovieID"], sort=True)

(10000, 10)

newdf = pd.DataFrame(data10k[["Title", "Rating", "MovieID", "Genre"]])

#creating a new DF
newdf.head()

	Title	Rating	MovieID	Genre
0	Toy Story (1995)	5	1	Animation|Children's|Comedy
1	Pocahontas (1995)	5	48	Animation|Children's|Musical|Romance
2	Apollo 13 (1995)	5	150	Drama
3	Star Wars: Episode IV - A New Hope (1977)	4	260	Action|Adventure|Fantasy|Sci-Fi

	Title	Rating	MovieID	Genre
4	Schindler's List (1993)	5	527	Drama|War

```
sum_rating = data10k.groupby(["Title", "Rating", "Genre"]).size().sort_values(ascending = False)

print(sum_rating[:25])
```

```
Title                                        Rating  Genre
Toy Story (1995)                             5       Animation|Children's|Comedy          20
Matrix, The (1999)                           5       Action|Sci-Fi|Thriller               17
Star Wars: Episode V - The Empire Strikes Back (1980)  5  Action|Adventure|Drama|Sci-Fi|War  16
E.T. the Extra-Terrestrial (1982)            4       Children's|Drama|Fantasy|Sci-Fi      15
Saving Private Ryan (1998)                   5       Action|Drama|War                     15
Toy Story (1995)                             4       Animation|Children's|Comedy          15
Raiders of the Lost Ark (1981)               5       Action|Adventure                     14
Princess Bride, The (1987)                   5       Action|Adventure|Comedy|Romance      14
Star Wars: Episode IV - A New Hope (1977)    5       Action|Adventure|Fantasy|Sci-Fi      14
Star Wars: Episode VI - Return of the Jedi (1983)  4  Action|Adventure|Romance|Sci-Fi|War  14
Bug's Life, A (1998)                         4       Animation|Children's|Comedy          13
American Beauty (1999)                       5       Comedy|Drama                         13
Forrest Gump (1994)                          5       Comedy|Romance|War                   13
X-Men (2000)                                 4       Action|Sci-Fi                        13
Toy Story (1995)                             3       Animation|Children's|Comedy          13
Schindler's List (1993)                      5       Drama|War                            13
Big (1988)                                   4       Comedy|Fantasy                       12
Shakespeare in Love (1998)                   5       Comedy|Romance                       12
Shawshank Redemption, The (1994)             5       Drama                                12
American Beauty (1999)                       4       Comedy|Drama                         12
Sixth Sense, The (1999)                      4       Thriller                             12
Jurassic Park (1993)                         4       Action|Adventure|Sci-Fi              12
Star Wars: Episode I - The Phantom Menace (1999)  3  Action|Adventure|Fantasy|Sci-Fi      11
Men in Black (1997)                          4       Action|Adventure|Comedy|Sci-Fi       11
Clueless (1995)                              4       Comedy|Romance                       11
dtype: int64
```

#creating an empty list & creating a list of unique genres

```
genres = []

uniquegenre = list(set(genres))

print(uniquegenre)
[]
```

#Eencoding for the genres present in the table.

```
for i in uniquegenre:
   data10k[i] = data10k["Genre"].str.contains(i)*1
```

data10k.head()

	MovieID	Title	Genre	UserID	Rating	TimeStamp	Gender	Age	Occupation	ZipCode
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
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067

data10k["Age"].value_counts()

```
25    3009
18    2316
35    2310
1     735
45     711
50     646
56     273
Name: Age, dtype: int64
```

```
#Making separate buckets for age groups.
data10k.loc[data10k['Age'] ==1, 'Age Group'] = 'Under 18'
data10k.loc[data10k['Age'] ==18, 'Age Group'] = '18-24'
data10k.loc[data10k['Age'] ==25, 'Age Group'] = '25-34'
data10k.loc[data10k['Age']==35, 'Age Group'] = '35-44'
data10k.loc[data10k['Age'] ==45, 'Age Group'] = '45-49'
```

```
data10k.loc[data10k['Age'] ==50, 'Age Group'] = '50-55'
data10k.loc[data10k['Age']==56, 'Age Group'] = '56+'

effect_rating = data10k[["Rating", "Gender", "Age Group", "Occupation"]]

# Here we are considering the data set of 500
working_dataset = effect_rating.iloc[:500, ]

X = working_dataset.iloc[:,[1, 2, 3]]
Y = working_dataset.iloc[:,0]

X.head()
```

	Gender	Age Group	Occupation
0	F	Under 18	10
1	F	Under 18	10
2	F	Under 18	10
3	F	Under 18	10
4	F	Under 18	10

```
Y.head()

0    5
1    5
2    5
3    4
4    5
Name: Rating, dtype: int64

X_dummy = pd.get_dummies(data=X)

X_dummy2 = pd.get_dummies(X['Occupation'], prefix='Occupation')

X_Concat = pd.concat([X_dummy, X_dummy2], axis = 1)

X_Concat.head()
```

	Occupation	Gender_F	Gender_M	Age Group_25-34	Age Group_35-44	Age Group_50-55	Age Group_Under 18	Occupation_1	Occupation_9	Occupation_10	Occupation_12	Occupation_17
0	10	1	0	0	0	0	1	0	0	1	0	0
1	10	1	0	0	0	0	1	0	0	1	0	0

	Occupation	Gender_F	Gender_M	Age Group_25-34	Age Group_35-44	Age Group_50-55	Age Group_Under 18	Occupation_1	Occupation_9	Occupation_10	Occupation_12	Occupation_17
2	10	1	0	0	0	0	1	0	0	1	0	0
3	10	1	0	0	0	0	1	0	0	1	0	0
4	10	1	0	0	0	0	1	0	0	1	0	0

X_Concat.columns

```
Index(['Occupation', 'Gender_F', 'Gender_M', 'Age Group_25-34',
       'Age Group_35-44', 'Age Group_50-55', 'Age Group_Under 18',
       'Occupation_1', 'Occupation_9', 'Occupation_10', 'Occupation_12',
       'Occupation_17'],
      dtype='object')
```

X_Concat.drop(['Occupation', "Gender_F"], axis = 1, inplace=**True**)

X_Concat.head()

	Gender_M	Age Group_25-34	Age Group_35-44	Age Group_50-55	Age Group_Under 18	Occupation_1	Occupation_9	Occupation_10	Occupation_12	Occupation_17
0	0	0	0	0	1	0	0	1	0	0
1	0	0	0	0	1	0	0	1	0	0
2	0	0	0	0	1	0	0	1	0	0
3	0	0	0	0	1	0	0	1	0	0
4	0	0	0	0	1	0	0	1	0	0

XY = pd.concat([X_Concat, Y], axis = 1)

XY.head(6)

	Gender_M	Age Group_25-34	Age Group_35-44	Age Group_50-55	Age Group_Under 18	Occupation_1	Occupation_9	Occupation_10	Occupation_12	Occupation_17	Rating
0	0	0	0	0	1	0	0	1	0	0	5

	Gender_M	Age Group_25-34	Age Group_35-44	Age Group_50-55	Age Group_Under 18	Occupation_1	Occupation_9	Occupation_10	Occupation_12	Occupation_17	Rating
1	0	0	0	0	1	0	0	1	0	0	5
2	0	0	0	0	1	0	0	1	0	0	5
3	0	0	0	0	1	0	0	1	0	0	4
4	0	0	0	0	1	0	0	1	0	0	5
5	0	0	0	0	1	0	0	1	0	0	4

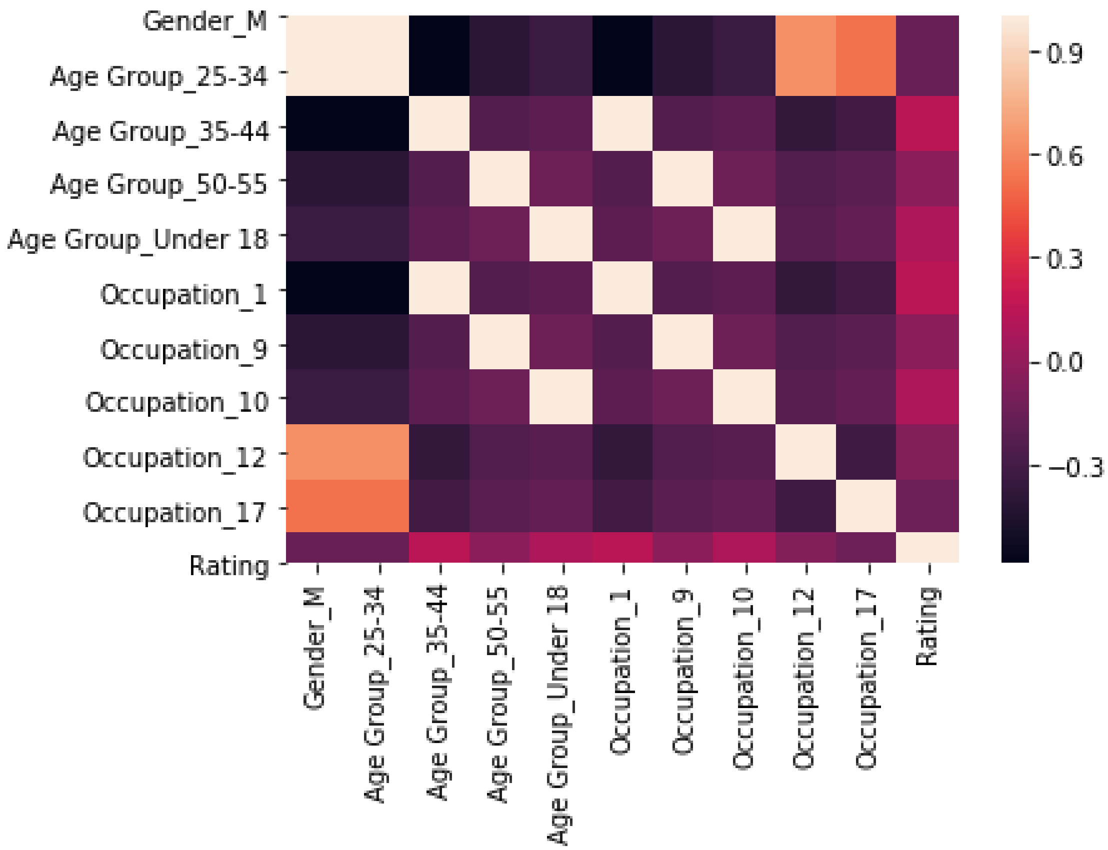
XY.corr()

	Gender_M	Age Group_25-34	Age Group_35-44	Age Group_50-55	Age Group_Under 18	Occupation_1	Occupation_9	Occupation_10	Occupation_12	Occupation_17	Rating
Gender_M	1.000000	1.000000	-0.584030	-0.398762	-0.337518	-0.584030	-0.398762	-0.337518	0.633054	0.529166	-0.161714
Age Group_25-34	1.000000	1.000000	-0.584030	-0.398762	-0.337518	-0.584030	-0.398762	-0.337518	0.633054	0.529166	-0.161714
Age Group_35-44	-0.584030	-0.584030	1.000000	-0.242395	-0.205167	1.000000	-0.242395	-0.205167	-0.369723	-0.309049	0.142974
Age Group_50-55	-0.398762	-0.398762	-0.242395	1.000000	-0.140083	-0.242395	1.000000	-0.140083	-0.252438	-0.211011	-0.028722
Age Group_Un	-0.3	-0.3	-0.2	-0.1	1.000000	-0.205	-0.140	1.000000	-0.213	-0.178	0.090

	Gender_M	**AgeGroup_25-34**	**AgeGroup_35-44**	**AgeGroup_50-55**	**AgeGroup_Under 18**	**Occupation_1**	**Occupation_9**	**Occupation_10**	**Occupation_12**	**Occupation_17**	**Rating**
der 18	37518	37518	05167	40083		167	083		667	603	948
Occupation_1	-0.584030	-0.584030	1.000000	-0.242395	-0.205167	1.000000	-0.242395	-0.205167	-0.369723	-0.309049	0.142974
Occupation_9	-0.398762	-0.398762	-0.242395	1.000000	-0.140083	-0.242395	1.000000	-0.140083	-0.252438	-0.211011	-0.028722
Occupation_10	-0.337518	-0.337518	-0.205167	-0.140083	1.000000	-0.205167	-0.140083	1.000000	-0.213667	-0.178603	0.090948
Occupation_12	0.633054	0.633054	-0.369723	-0.252438	-0.213667	-0.369723	-0.252438	-0.213667	1.000000	-0.321854	-0.055751
Occupation_17	0.529166	0.529166	-0.309049	-0.211011	-0.178603	-0.309049	-0.211011	-0.178603	-0.321854	1.000000	-0.136679
Rating	-0.161714	-0.161714	0.142974	-0.028722	0.090948	0.142974	-0.028722	0.090948	-0.055751	-0.136679	1.000000

sns.heatmap(XY.corr())

<matplotlib.axes._subplots.AxesSubplot at 0x1c97e7845c8>



#Building the model for predicting

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X_Concat,Y,test_size=0.3,random_state=102)

from sklearn.linear_model import LinearRegression
lm = LinearRegression()

[

lm.fit(x_train, y_train)

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

print(lm.intercept_)

-54426258464140.06

print(lm.coef_)

[-2.24506933e+13 7.16612361e+13 1.21833404e+14 -5.45248241e+13
  2.72131292e+13 -6.74071456e+13 1.08951083e+14 2.72131292e+13
  5.21571561e+12 5.21571561e+12]

coffecient=pd.DataFrame(data=lm.coef_,index=x_train.columns,columns=['coffecient'])

coffecient

	coffecient
Gender_M	-2.245069e+13
Age Group_25-34	7.166124e+13
Age Group_35-44	1.218334e+14
Age Group_50-55	-5.452482e+13
Age Group_Under 18	2.721313e+13
Occupation_1	-6.740715e+13
Occupation_9	1.089511e+14
Occupation_10	2.721313e+13
Occupation_12	5.215716e+12
Occupation_17	5.215716e+12

pred_y = lm.predict(X=x_test)

pred_y

```
array([3.8515625, 4.421875 , 3.6640625, 3.8515625, 3.8515625, 3.8359375,
       4.421875 , 4.1328125, 3.8515625, 4.421875 , 3.6640625, 3.8359375,
       3.6640625, 4.421875 , 3.6640625, 4.421875 , 4.1328125, 4.1328125,
       4.421875 , 4.1328125, 4.421875 , 3.6640625, 4.421875 , 3.8359375,
       3.8515625, 4.421875 , 4.421875 , 3.8515625, 3.6640625, 3.6640625,
       3.8515625, 3.8515625, 3.8515625, 4.421875 , 3.6640625, 3.8515625,
       4.421875 , 4.421875 , 4.1328125, 3.8359375, 4.421875 , 3.6640625,
       3.8515625, 3.6640625, 4.421875 , 4.421875 , 3.8359375, 4.1328125,
       3.8515625, 3.6640625, 4.1328125, 3.8515625, 4.421875 , 3.6640625,
       3.8515625, 3.8359375, 3.8515625, 3.8515625, 4.421875 , 4.421875 ,
       3.8515625, 4.421875 , 3.8515625, 3.8515625, 3.8359375, 3.8515625,
       4.1328125, 4.421875 , 3.8515625, 4.421875 , 3.6640625, 4.421875 ,
       3.8515625, 3.8359375, 4.1328125, 4.1328125, 3.8515625, 4.421875 ,
       3.8515625, 3.6640625, 4.421875 , 4.421875 , 3.8515625, 4.1328125,
       3.8359375, 3.6640625, 3.6640625, 4.421875 , 3.8359375, 3.6640625,
       3.8515625, 3.6640625, 4.1328125, 3.6640625, 3.6640625, 3.8515625,
       4.421875 , 4.421875 , 3.8515625, 4.421875 , 3.8515625, 3.6640625,
       3.8359375, 3.8515625, 3.8359375, 3.8515625, 4.1328125, 4.1328125,
       3.8515625, 4.421875 , 3.8359375, 3.6640625, 4.421875 , 3.8359375,
       3.8515625, 3.8359375, 4.421875 , 3.8515625, 3.6640625, 3.6640625,
       3.8359375, 3.8515625, 3.8515625, 3.8515625, 3.8515625, 3.8515625,
       3.8359375, 3.8359375, 3.6640625, 4.421875 , 4.1328125, 3.6640625,
       4.421875 , 3.6640625, 3.6640625, 4.421875 , 3.8515625, 4.1328125,
       3.6640625, 3.8515625, 3.8359375, 4.421875 , 4.421875 , 4.1328125,
```

3.6640625, 3.8515625, 3.8515625, 3.6640625, 3.8359375, 4.421875])

x_test.head()

	Gender_M	**AgeGroup_25-34**	**AgeGroup_35-44**	**AgeGroup_50-55**	**AgeGroup_Under 18**	**Occupation_1**	**Occupation_9**	**Occupation_10**	**Occupation_12**	**Occupation_17**
143	1	1	0	0	0	0	0	0	1	0
459	0	0	1	0	0	1	0	0	0	0
281	1	1	0	0	0	0	0	0	0	1
148	1	1	0	0	0	0	0	0	1	0
199	1	1	0	0	0	0	0	0	1	0

```
from sklearn import metrics

metrics.mean_absolute_error(y_test, pred_y)

0.7234895833333334

metrics.mean_squared_error(y_test, pred_y)

0.7921268717447917

np.sqrt(metrics.mean_squared_error(y_test, pred_y))

0.8900150963578043

lm.score(x_test, y_test)

-0.078277851912264
```

<end of document>