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.

Analysis Tasks to be performed:

Import the three datasets

Create a new dataset with the following columns MovieID Title UserID Age Gender Occupation Rating.

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

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

In [15]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.style import use
%matplotlib inline
import warnings
```

In [6]:
```
movies = pd.read_csv(r'movies.dat', sep = "::", names = ['MovieID', 'Title', 'Ger
movies.head()
```

Out[6]:

	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 [7]:
```
ratings = pd.read_csv(r'ratings.dat', sep = "::", names = ['UserID','MovieID','Ra
ratings.head()
```

Out[7]:

	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 [9]:
```
users = pd.read_csv(r'users.dat', sep = "::", names = ['UserID','Gender','Age','O
users.head()
```

Out[9]:

	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

In [10]:
```
movies.shape, users.shape, ratings.shape
```

Out[10]:
```
((3883, 3), (6040, 5), (1000209, 4))
```

Creating a new dataset with the following columns MovieID Title UserID Age Gender Occupation Rating.

In [13]:
```
movie_rat=pd.merge(movies,ratings,on='MovieID')
display(movie_rat.head())
movie_rat.shape
```

	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
3	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952
4	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474

Out[13]: (1000209, 6)

In [14]:
```
movie_user=pd.merge(movie_rat,users, on = "UserID")
display (movie_user.head())
movie_user.shape
```

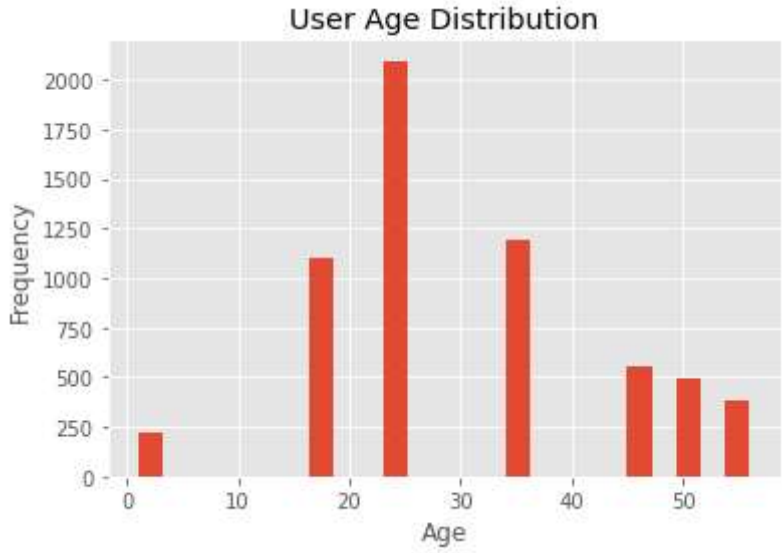
	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	A
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	

Out[14]: (1000209, 10)

Exploring the datasets using visual representations:

1. User Age Distribution

In [29]:
```
users.Age.plot.hist(bins=25)
plt.style.use('ggplot')
plt.title('User Age Distribution')
plt.xlabel('Age')
plt.show()
```



2. User rating of the movie “Toy Story”

In [41]:
```
rating_toystory=movie_user[movie_user['MovieID']==1]
rating_toystory.head(6)
```

Out[41]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occup
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	
53	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008	F	50	
124	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496	M	25	
263	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952	M	25	
369	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474	F	35	
770	1	Toy Story (1995)	Animation|Children's|Comedy	18	4	978154768	F	18	

3. Top 25 movies by viewership rating

In [48]:
```
top_count = movie_rat['Title'].value_counts()
top_count[:25]
```

Out[48]:
```
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
Name: Title, dtype: int64
```

4.Finding the ratings for all the movies reviewed by user of user id = 2696

In [69]:
```
user_2696_rat=movie_user[movie_user['UserID']==2696]
user_2696_rat
```

Out[69]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Ag
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	973308886	M	2
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	973308842	M	2
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	973308886	M	2
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M	2
991039	1258	Shining, The (1980)	Horror	2696	4	973308710	M	2
991040	1270	Back to the Future (1985)	Comedy|Sci-Fi	2696	2	973308676	M	2
991041	1589	Cop Land (1997)	Crime|Drama|Mystery	2696	3	973308865	M	2
991042	1617	L.A. Confidential (1997)	Crime|Film-Noir|Mystery|Thriller	2696	4	973308842	M	2
991043	1625	Game, The (1997)	Mystery|Thriller	2696	4	973308842	M	2
991044	1644	I Know What You Did Last Summer (1997)	Horror|Mystery|Thriller	2696	2	973308920	M	2
991045	1645	Devil's Advocate, The (1997)	Crime|Horror|Mystery|Thriller	2696	4	973308904	M	2
991046	1711	Midnight in the Garden of Good and Evil (1997)	Comedy|Crime|Drama|Mystery	2696	4	973308904	M	2
991047	1783	Palmetto (1998)	Film-Noir|Mystery|Thriller	2696	4	973308865	M	2
991048	1805	Wild Things (1998)	Crime|Drama|Mystery|Thriller	2696	4	973308886	M	2
991049	1892	Perfect Murder, A (1998)	Mystery|Thriller	2696	4	973308904	M	2
991050	2338	I Still Know What You Did Last Summer (1998)	Horror|Mystery|Thriller	2696	2	973308920	M	2

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Ag
991051	2389	Psycho (1998)	Crime|Horror|Thriller	2696	4	973308710	M	2
991052	2713	Lake Placid (1999)	Horror|Thriller	2696	1	973308710	M	2
991053	3176	Talented Mr. Ripley, The (1999)	Drama|Mystery|Thriller	2696	4	973308865	M	2
991054	3386	JFK (1991)	Drama|Mystery	2696	1	973308842	M	2

Feature Engineering:

1. Finding out all the unique genres

In [77]:
```
movie_user.Genres
```

Out[77]:
```
0                    Animation|Children's|Comedy
1          Animation|Children's|Musical|Romance
2                                         Drama
3               Action|Adventure|Fantasy|Sci-Fi
4                                    Drama|War
                          ...
1000204                          Drama|Thriller
1000205                 Comedy|Horror|Thriller
1000206                         Comedy|Romance
1000207                        Action|Thriller
1000208                           Action|Drama
Name: Genres, Length: 1000209, dtype: object
```

In [83]:
```
unique_gen=movie_user.Genres.str.split("|").values
unique_gen
```

Out[83]:
```
array([list(['Animation', "Children's", 'Comedy']),
       list(['Animation', "Children's", 'Musical', 'Romance']),
       list(['Drama']), ..., list(['Comedy', 'Romance']),
       list(['Action', 'Thriller']), list(['Action', 'Drama'])],
      dtype=object)
```

In [84]:
```
genre_labels = set()
for s in movie_user['Genres'].str.split('|').values:
    genre_labels = genre_labels.union(set(s))
genre_labels
```

Out[84]:
```
{'Action',
 'Adventure',
 'Animation',
 "Children's",
 'Comedy',
 'Crime',
 'Documentary',
 'Drama',
 'Fantasy',
 'Film-Noir',
 'Horror',
 'Musical',
 'Mystery',
 'Romance',
 'Sci-Fi',
 'Thriller',
 'War',
 'Western'}
```

2. Creating 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 [101]:
```
Genre_column=list(genre_labels)
values=np.array(Genre_column)
values
```

Out[101]:
```
array(['Sci-Fi', 'Musical', 'Horror', 'Documentary', 'Film-Noir',
       'Western', 'Drama', 'Adventure', 'War', "Children's", 'Comedy',
       'Romance', 'Action', 'Thriller', 'Crime', 'Animation', 'Mystery',
       'Fantasy'], dtype='<U11')
```

In [102]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values)
print(integer_encoded)
```

```
[14 11 10  6  9 17  7  1 16  3  4 13  0 15  5  2 12  8]
```

In [104]:
```
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
print(onehot_encoded)
```

```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

3. features affecting the ratings of any particular movie.

In [108]:
```
import seaborn as sns
%matplotlib inline
plt.figure(figsize=(20,8))
corr=movie_user[:5000].corr()
sns.heatmap(corr,xticklabels=corr.columns.values,yticklabels=corr.columns.values,
```

Out[108]: <AxesSubplot:>

4. Developing an appropriate model to predict the movie ratings

In [110]:
```
feature_cols = ['Age', 'Occupation']

X = movie_user[feature_cols]
y = movie_user.Rating

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

from sklearn.linear_model import LinearRegression
linreg=LinearRegression()
linreg.fit(X_train, y_train)

y_pred = linreg.predict(X_test)


from sklearn.metrics import mean_squared_error
print(np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
1.1153284258531615
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: