Movielens Case Study

Project 2

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:

```
#library imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

- Import the three datasets

```
movies_df = pd.read_csv('movies.dat', sep='::', names=['MovieID','Title','Genres'],
engine='python',header=None)
users_df = pd.read_csv('users.dat', sep='::', names=['UserID','Gender','Age', 'Occupation', 'zip-code'],
engine='python',header=None)
ratings_df = pd.read_csv('ratings.dat', sep='::', names=['UserID','MovieID','Rating', 'Timestamp'],
engine='python',header=None)

users_df.shape
users_df.info()
users_df.head(n=10)
movies_df.shape
movies_df.info()
movies_df.head(n=10)

ratings_df.shape
ratings_df.info()
```

```
ratings_df.head(n=10)
```

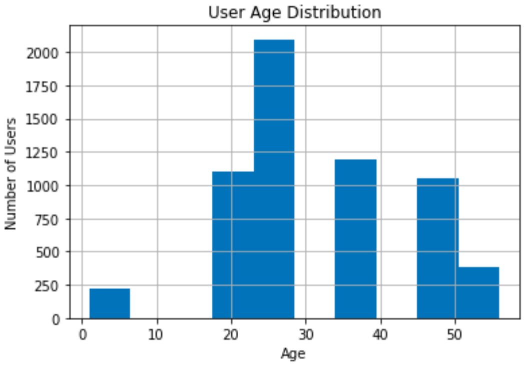
- Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating. (Hint: (i) Merge two tables at a time. (ii) Merge the tables using two primary keys MovieID & UserId)

```
MasterData = pd.merge(movies_df, ratings_df, on='MovieID')
MasterData = pd.merge(MasterData, users_df, on='UserID')
MasterData = MasterData.drop(['Genres', 'Timestamp', 'ZipCode'], axis=1)

MasterData.isnull().sum()
MasterData.describe()
```

- Explore the datasets using visual representations (graphs or tables), also include your comments on the following:
  1. User Age Distribution

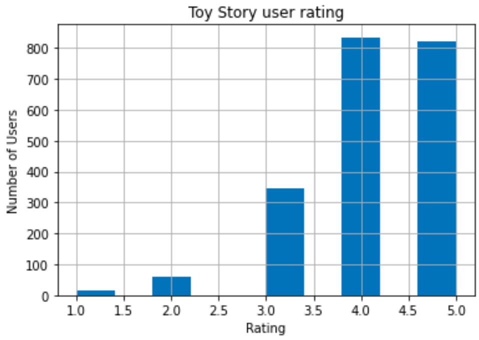
```
users_df.Age.hist()
plt.title('User Age Distribution')
plt.xlabel('Age')
plt.ylabel('Number of Users')
plt.show()
```



Most of the users are below 25-28 of age.

  2. User rating of the movie “Toy Story”

```
toystory_rating = ratings_df[ratings_df['MovieID']==1]
toystory_rating ['Rating'].hist()
plt.title('Toy Story user rating”')
plt.xlabel('Rating')
plt.ylabel('Number of Users')
```



Toy Story movie has an average user rating of 4.0

3. Top 25 movies by viewership rating

```
Top25 = ratings_df.groupby('MovieID')
Top25 = Top25.agg({'Rating':'mean'})
Top25 = Top25.sort_values('Rating',ascending=False).head(25)
```

	MovieID	Rating	Title
0	989	5.000000	Schlafes Bruder (Brother of Sleep) (1995)
1	3881	5.000000	Bittersweet Motel (2000)
2	1830	5.000000	Follow the Bitch (1998)
3	3382	5.000000	Song of Freedom (1936)
4	787	5.000000	Gate of Heavenly Peace, The (1995)
5	3280	5.000000	Baby, The (1973)
6	3607	5.000000	One Little Indian (1973)
7	3233	5.000000	Smashing Time (1967)
8	3172	5.000000	Ulysses (Ulisse) (1954)
9	3656	5.000000	Lured (1947)
10	3245	4.800000	I Am Cuba (Soy Cuba/Ya Kuba) (1964)
11	53	4.750000	Lamerica (1994)
12	2503	4.666667	Apple, The (Sib) (1998)
13	2905	4.608696	Sanjuro (1962)
14	2019	4.560510	Seven Samurai (The Magnificent Seven) (Shichin...
15	318	4.554558	Shawshank Redemption, The (1994)
16	858	4.524966	Godfather, The (1972)
17	745	4.520548	Close Shave, A (1995)
18	50	4.517106	Usual Suspects, The (1995)
19	527	4.510417	Schindler's List (1993)
20	1148	4.507937	Wrong Trousers, The (1993)
21	2309	4.500000	Inheritors, The (Die Siebtelbauern) (1998)
22	1795	4.500000	Callej�n de los milagros, El (1995)

	MovieID	Rating	Title
23	2480	4.500000	Dry Cleaning (Nettoyage � sec) (1997)
24	439	4.500000	Dangerous Game (1993)

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

```
user2696 = ratings_df [ratings_df['UserID']==2696]
user2696 = user2696.drop(['UserID','Timestamp'], axis=1)
user2696 = pd.merge(user2696, dataset_movies, on='MovieID')
```

	MovieID	Rating	Title	Genres
0	1258	4	Shining, The (1980)	Horror
1	1270	2	Back to the Future (1985)	Comedy|Sci-Fi
2	1617	4	L.A. Confidential (1997)	Crime|Film-Noir|Mystery|Thriller
3	1625	4	Game, The (1997)	Mystery|Thriller
4	1644	2	I Know What You Did Last Summer (1997)	Horror|Mystery|Thriller
5	1645	4	Devil's Advocate, The (1997)	Crime|Horror|Mystery|Thriller
6	1805	4	Wild Things (1998)	Crime|Drama|Mystery|Thriller
7	1892	4	Perfect Murder, A (1998)	Mystery|Thriller
8	800	5	Lone Star (1996)	Drama|Mystery
9	2338	2	I Still Know What You Did Last Summer (1998)	Horror|Mystery|Thriller
10	1711	4	Midnight in the Garden of Good and Evil (1997)	Comedy|Crime|Drama|Mystery
11	3176	4	Talented Mr. Ripley, The (1999)	Drama|Mystery|Thriller
12	2389	4	Psycho (1998)	Crime|Horror|Thriller
13	1589	3	Cop Land (1997)	Crime|Drama|Mystery
14	2713	1	Lake Placid (1999)	Horror|Thriller
15	3386	1	JFK (1991)	Drama|Mystery
16	1783	4	Palmetto (1998)	Film-Noir|Mystery|Thriller
17	350	3	Client, The (1994)	Drama|Mystery|Thriller
18	1092	4	Basic Instinct (1992)	Mystery|Thriller
19	1097	3	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi

- 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)

```
unique_genre = set()
for s in movies_df['Genres'].str.split('|').values:
    unique_genre = unique_genre.union(set(s))
```

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.

```
genres = pd.concat([movies_df, movies_df.Genres.str.get_dummies()], axis=1). drop('Genres', axis=1)
```

3. Determine the features affecting the ratings of any particular movie.

```
MasterData.corr()
```

#Age and occupation affect the ratings of the movie

4. Develop an appropriate model to predict the movie ratings

```
MasterData = MasterData[:500]
MasterData = MasterData.drop(['Title', 'Gender'], axis=1)
x= MasterData.iloc[:,:-1]
y= MasterData.iloc[:,-1]

x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.35, random_state= 21)
scaler = StandardScaler().fit_transform(x_train, x_test)

lr = LogisticRegression()
lr.fit(x_train, y_train)
y_pred_value = lr.predict(x_test)
accuracy = accuracy_score(y_test, y_pred_value)

accuracy = 98%

knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
y_pred_value = knn.predict(x_test)
accuracy = accuracy_score(y_test, y_pred_value)

accuracy = 58.2%

svc = SVC()
svc.fit(x_train, y_train)
y_pred_value = svc.predict(x_test)
accuracy = accuracy_score(y_test, y_pred_value)

accuracy = 39.7%

decisiontree = DecisionTreeClassifier()
decisiontree.fit(x_train, y_train)
y_pred_value = decisiontree.predict(x_test)
accuracy = accuracy_score(y_test, y_pred_value)

accuracy = 100%
```

Decision tree gave the best results