Assignment 1: Movielens case study.

Requirements

1. Import the three datasets
2. 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)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
movies_df = pd.read_csv('movies.dat', sep='::',names=['MovieID','Title','Genres'
user_df = pd.read_csv('users.dat', sep='::',names=['UserID','Gender','Age', 'Occ
ratings_df = pd.read_csv('ratings.dat', sep='::',names=['UserID','MovieID','Rati
```

```
movies_df.head()
```

	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

```
user_df.head()
```

	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

```
ratings_df.head()
```

	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

```
temp_df = pd.merge(movies_df, ratings_df,on='MovieID')
```

```
master_df = pd.merge(temp_df, user_df,on='UserID')
```

```
master_df.head()
```

	MovieID	Title	Genres	UserID	Rating	Timesta
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824
2	150	Apollo 13 (1995)	Drama	1	5	978301

```
master_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 10 columns):
 #   Column      Non-Null Count    Dtype 
---  ------      --------------    ----- 
 0   MovieID     1000209 non-null  int64 
 1   Title       1000209 non-null  object
 2   Genres      1000209 non-null  object
 3   UserID      1000209 non-null  int64 
 4   Rating      1000209 non-null  int64 
 5   Timestamp   1000209 non-null  int64 
 6   Gender      1000209 non-null  object
 7   Age         1000209 non-null  int64 
 8   Occupation  1000209 non-null  int64 
 9   zip-code    1000209 non-null  object
dtypes: int64(6), object(4)
memory usage: 83.9+ MB
```

```
master_data = master_df[['MovieID', 'Title',  'UserID', 'Age', 'Gender', 'Occupa
```

EDA

Requirements:

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

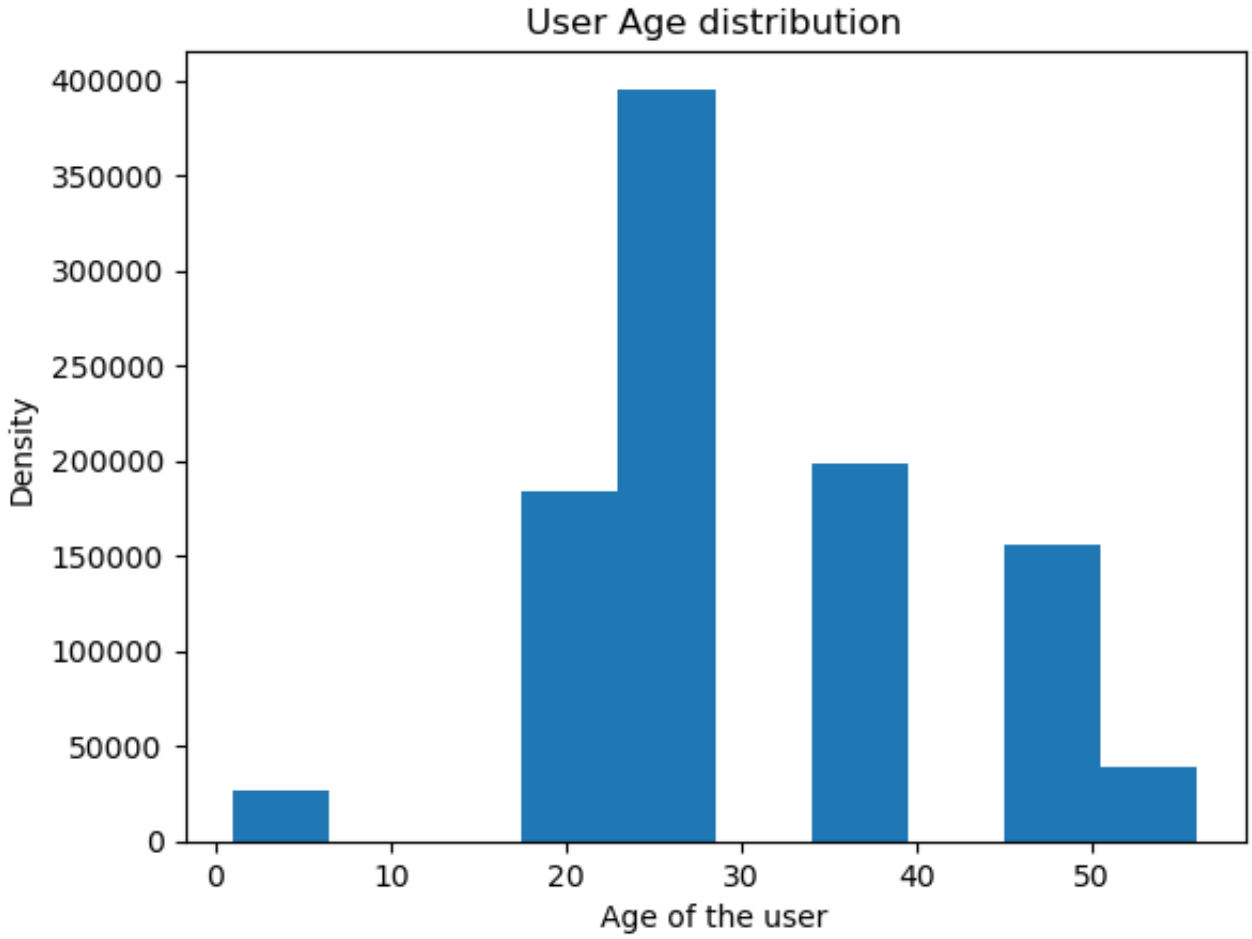
2) User rating of the movie “Toy Story”

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

```
plt.hist(master_df['Age'])
plt.title('User Age distribution')
plt.xlabel('Age of the user')
plt.ylabel('Density')
```

```
Text(0, 0.5, 'Density')
```

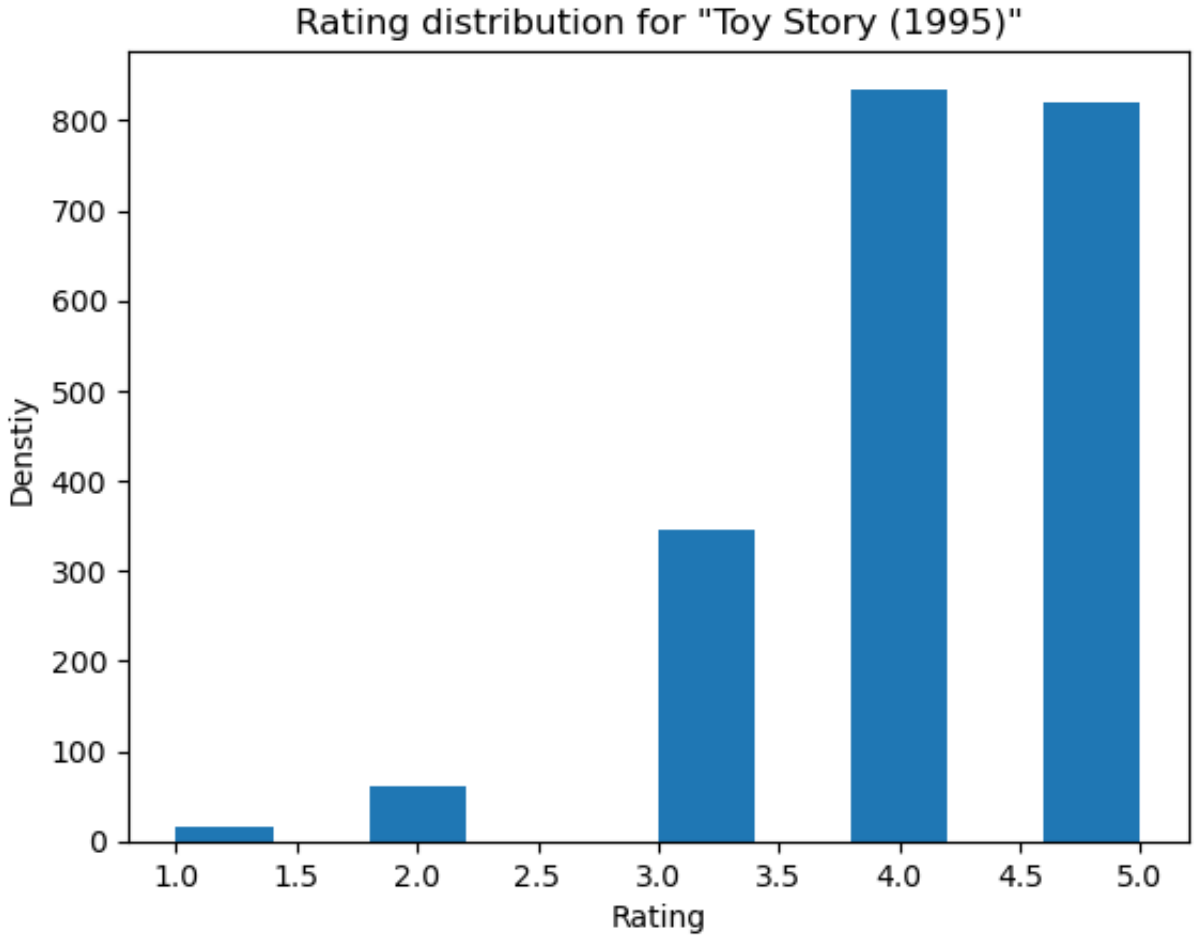


Key takeaway: The density for the ages between 18-28 is higher compared to any other age group. I.e, people between the age groups of 18-28 tend to watch more movies.

```
temp_df = master_df.loc[master_df['Title'] == 'Toy Story (1995)']
```

```
plt.hist(temp_df['Rating'])
plt.title('Rating distribution for "Toy Story (1995)"')
plt.xlabel('Rating')
plt.ylabel('Denstiy')
```

```
Text(0, 0.5, 'Denstiy')
```



Key takeaway: The ratings for the movie is on a positive side with the rating '4' being the highest density.

```
max_rating = master_df.groupby(['Title'])['Rating']
```

Top 25 movies by viewership rating

```
max_rating = max_rating.count().sort_values(ascending=False)
max_rating[0:25]
```

```
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
Name: Rating, dtype: int64
```

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

```
s_user = master_df.loc[master_df['UserID']== 2696]
```

s_user

	MovieID	Title	Genres	UserID	Rating	Timestam
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	97330888
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	97330884
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	97330888
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	97330869
991039	1258	Shining, The (1980)	Horror	2696	4	97330871
991040	1270	Back to the Future (1985)	Comedy|Sci-Fi	2696	2	97330867
991041	1589	Cop Land (1997)	Crime|Drama|Mystery	2696	3	97330886
991042	1617	L.A. Confidential (1997)	Crime|Film-Noir|Mystery|Thriller	2696	4	97330884
991043	1625	Game, The (1997)	Mystery|Thriller	2696	4	97330884
991044	1644	I Know What You Did Last Summer (1997)	Horror|Mystery|Thriller	2696	2	97330892

Feature engineering:

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

Unique genres in the dataset:

```
master_df['Genres'].unique()
```

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

```
'Comedy|Crime|Drama|Mystery', 'Action|Comedy|Drama',
'Action|Crime|Thriller', "Adventure|Children's|Drama",
'Drama|Mystery', 'Action|Comedy|Sci-Fi|Thriller',
'Action|Adventure|Sci-Fi|Thriller',
'Action|Drama|Romance|Thriller', 'Crime|Thriller', 'Documentary',
'Comedy|Crime|Fantasy', 'Animation|Comedy', 'Comedy|Crime',
'Crime|Film-Noir|Mystery|Thriller', 'Sci-Fi|Thriller',
'Action|Sci-Fi', 'Horror|Sci-Fi|Thriller',
"Adventure|Children's|Fantasy", 'Action|Adventure|Comedy|Crime',
'Action|Adventure', 'Action|Drama|Thriller',
"Children's|Comedy|Fantasy", 'Comedy|Romance|War',
'Film-Noir|Sci-Fi', 'Comedy|Romance|Thriller',
'Action|Adventure|Crime|Drama', 'Action|Adventure|Mystery',
'Action|Adventure|Fantasy', 'Sci-Fi|War', 'Action|Sci-Fi|War',
'Mystery|Thriller', 'Film-Noir|Mystery',
'Drama|Mystery|Sci-Fi|Thriller', 'Action|Adventure|Romance|War',
"Adventure|Children's", "Adventure|Children's|Fantasy|Sci-Fi",
"Adventure|Children's|Musical",
"Adventure|Children's|Comedy|Fantasy",
'Action|Adventure|Drama|Sci-Fi|War', 'Action|Sci-Fi|Thriller|War',
'Action|Western', 'Adventure|War', 'Action|Horror|Sci-Fi|Thriller',
'Action|Adventure|Comedy|Horror|Sci-Fi', 'Action|Comedy|Musical',
'Film-Noir|Mystery|Thriller', 'Adventure', 'Comedy|War',
'Adventure|Comedy|Drama', 'Comedy|Mystery|Thriller',
'Comedy|Horror', 'Horror|Romance', 'Horror', 'Action|Horror',
'Action|Romance|War', "Children's|Fantasy",
"Children's|Drama|Fantasy", 'Action|Adventure|Sci-Fi|War'
```

One hot encoder values (0,1)

```
df_dummies = master_data['Genres'].str.get_dummies(sep = '|')
df_f = pd.get_dummies(master_data['Gender'])
```

```
master_temp = master_data.drop(['Genres', 'Gender'], axis = 1)
master_temp = pd.concat([df_dummies, master_temp], axis = 1)
master_data = pd.concat([df_f, master_temp], axis = 1)
```

```
master_data.head()
```

	F	M	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary
0	1	0	0	0	1	1	1	0	0
1	1	0	0	0	1	1	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	1	1	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0

5 rows × 26 columns

```
main_df = master_data.drop(['MovieID', 'Title', 'UserID'], axis = 1)
main_df.head()
```

	F	M	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary
0	1	0	0	0	1	1	1	0	0
1	1	0	0	0	1	1	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	1	1	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0

5 rows × 23 columns

Features affecting Ratings of a movie:

We have eliminated features such as MovieID, UserID, Title as they are unique values which do not give any importance to predicting the dependent (Ratings) variable.

```
main_df.dtypes
```

```
F              uint8
M              uint8
Action         int64
Adventure      int64
Animation      int64
Children's     int64
Comedy         int64
Crime          int64
Documentary    int64
Drama          int64
Fantasy        int64
Film-Noir      int64
Horror         int64
Musical        int64
Mystery        int64
Romance        int64
Sci-Fi         int64
Thriller       int64
War            int64
Western        int64
Age            int64
Occupation     int64
Rating         int64
dtype: object
```

Building the model

- Now that the data is encoded, it is ready to be trained and tested.
- We first split the data into train and test (70-30).
- Train the data, and later go on to test the same.
- The measure to evaluate the model is MAE, MSE and RMSE

```
X = main_df.iloc[:,:-1].values
y = main_df.iloc[:, -1].values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, rando
```

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

```
LinearRegression()
```

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

```
prediction_df = pd.DataFrame({'Test': y_test, 'Prediction': y_pred})
prediction_df
```

	Test	Prediction
0	4	3.662307
1	3	3.682625
2	5	3.745125
3	4	3.656337
4	5	3.688505
...	...	...
300058	4	3.675135
300059	4	3.484614
300060	3	3.630937
300061	4	3.647404
300062	2	3.467056

300063 rows × 2 columns

```
print('MAE: ',metrics.mean_absolute_error(y_test, y_pred))
print('MSE: ',metrics.mean_squared_error(y_test, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
MAE:  0.9013901071913692
MSE:  1.201667512069826
RMSE: 1.0962059624312512
```