TASK 2 : MOVIE RATING PREDICTION

IMPORTING LIBRARIES

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

import warnings
warning.filterwarnings('ignore')
```

In [4]:
```
movie_df = pd.read_csv("movies.csv",sep ='::',engine='python',encoding=
movie_df.columns=['MovieID','MovieName','Genres']
movie_df.dropna(inplace=True)
movie_df.head()
```

Out[4]:

	MovieID	MovieName	Genres
0	2	Jumanji,(1995)	Adventure|Children's|Fantasy,,,,,,,,,,,,,,
1	3	Grumpier,Old,Men,(1995)	Comedy|Romance,,,,,,,,,,,,
2	4	Waiting,to,Exhale,(1995)	Comedy|Drama,,,,,,,,,,,,
3	5	Father,of,the,Bride,Part,II,(1995)	Comedy,,,,,,,,,
4	6	Heat,(1995)	Action|Crime|Thriller,,,,,,,,,,,,,,

To find out how many rows and columns are in our DataFrame "movies"

In [5]:
```
movie_df.shape
```

Out[5]:
```
(3882, 3)
```

To check the missing values in the dataframe

In [6]:
```
movie_df.isna().sum()
```

Out[6]:
```
MovieID      0
MovieName    0
Genres       0
dtype: int64
```

downloading Rating DataSet

In [9]:
```python
ratings_df = pd.read_csv('ratings.dat',sep='::',engine='python')
ratings_df.columns=['UserId','MovieID','Ratings','TimeStamp']
ratings_df.dropna(inplace=True)
ratings_df.head(10)
```

Out[9]:

	UserId	MovieID	Ratings	TimeStamp
0	1	661	3	978302109
1	1	914	3	978301968
2	1	3408	4	978300275
3	1	2355	5	978824291
4	1	1197	3	978302268
5	1	1287	5	978302039
6	1	2804	5	978300719
7	1	594	4	978302268
8	1	919	4	978301368
9	1	595	5	978824268

To find out how many rows and columns are in our DataFrame "ratings"

In [11]:
```python
ratings_df.shape
```

Out[11]:
```
(1000208, 4)
```

In [12]:
```python
ratings_df.isna().sum()
```

Out[12]:
```
UserId       0
MovieID      0
Ratings      0
TimeStamp    0
dtype: int64
```

Loading the User Data given by alfido-Tech

In [14]:
```
df_users=pd.read_csv("users.dat", sep="::", engine="python")
df_users.columns = ['userid','gender','age','occupation','zipcode']
df_users.dropna(inplace=True)
df_users.head(10)
```

Out[14]:

	userid	gender	age	occupation	zipcode
0	2	M	56	16	70072
1	3	M	25	15	55117
2	4	M	45	7	02460
3	5	M	25	20	55455
4	6	F	50	9	55117
5	7	M	35	1	06810
6	8	M	25	12	11413
7	9	M	25	17	61614
8	10	F	35	1	95370
9	11	F	25	1	04093

Concatenating the datasets for combining the 3 datasets of movies,users and rating

In [15]:
```
df_data=pd.concat([movie_df,ratings_df,df_users],axis=1)
df_data.head(10)
```

Out[15]:

	MovieID	MovieName	Genres	UserId	MovieID	Rati
0	2.0	Jumanji,(1995)	Adventure|Children's|Fantasy,,,,,,,,,,,,,,	1	661	
1	3.0	Grumpier,Old,Men,(1995)	Comedy|Romance,,,,,,,,,,,,	1	914	
2	4.0	Waiting,to,Exhale,(1995)	Comedy|Drama,,,,,,,,,,,,	1	3408	
3	5.0	Father,of,the,Bride,Part,II,(1995)	Comedy,,,,,,,,,	1	2355	
4	6.0	Heat,(1995)	Action|Crime|Thriller,,,,,,,,,,,,,,	1	1197	
5	7.0	Sabrina,(1995)	Comedy|Romance,,,,,,,,,,,,,,	1	1287	
6	8.0	Tom,and,Huck,(1995)	Adventure|Children's,,,,,,,,,,,,	1	2804	
7	9.0	Sudden,Death,(1995)	Action,,,,,,,,,,,,,	1	594	
8	10.0	GoldenEye,(1995)	Action|Adventure|Thriller,,,,,,,,,,,,,,	1	919	
9	11.0	American,President,The,(1995)	Comedy|Drama|Romance,,,,,,,,,,,,	1	595	

Removing the unwanted columns

In [20]:
```
df2=df_data.drop(["occupation","zipcode","TimeStamp"],axis=1)
df2.head()
```

Out[20]:

	MovieID	MovieName	Genres	UserId	MovieID	Rati
0	2.0	Jumanji,(1995)	Adventure|Children's|Fantasy,,,,,,,,,,,,,,	1	661	
1	3.0	Grumpier,Old,Men,(1995)	Comedy|Romance,,,,,,,,,,,,	1	914	
2	4.0	Waiting,to,Exhale,(1995)	Comedy|Drama,,,,,,,,,,,,	1	3408	
3	5.0	Father,of,the,Bride,Part,II,(1995)	Comedy,,,,,,,,,	1	2355	
4	6.0	Heat,(1995)	Action|Crime|Thriller,,,,,,,,,,,,,,	1	1197	

In [23]:
```
df2.isna().sum()
```

Out[23]:
```
MovieID      996326
MovieName    996326
Genres       996326
UserId            0
MovieID           0
Ratings           0
userid       994169
gender       994169
age          994169
dtype: int64
```

In [26]:
```
df_final = df2.dropna()
```

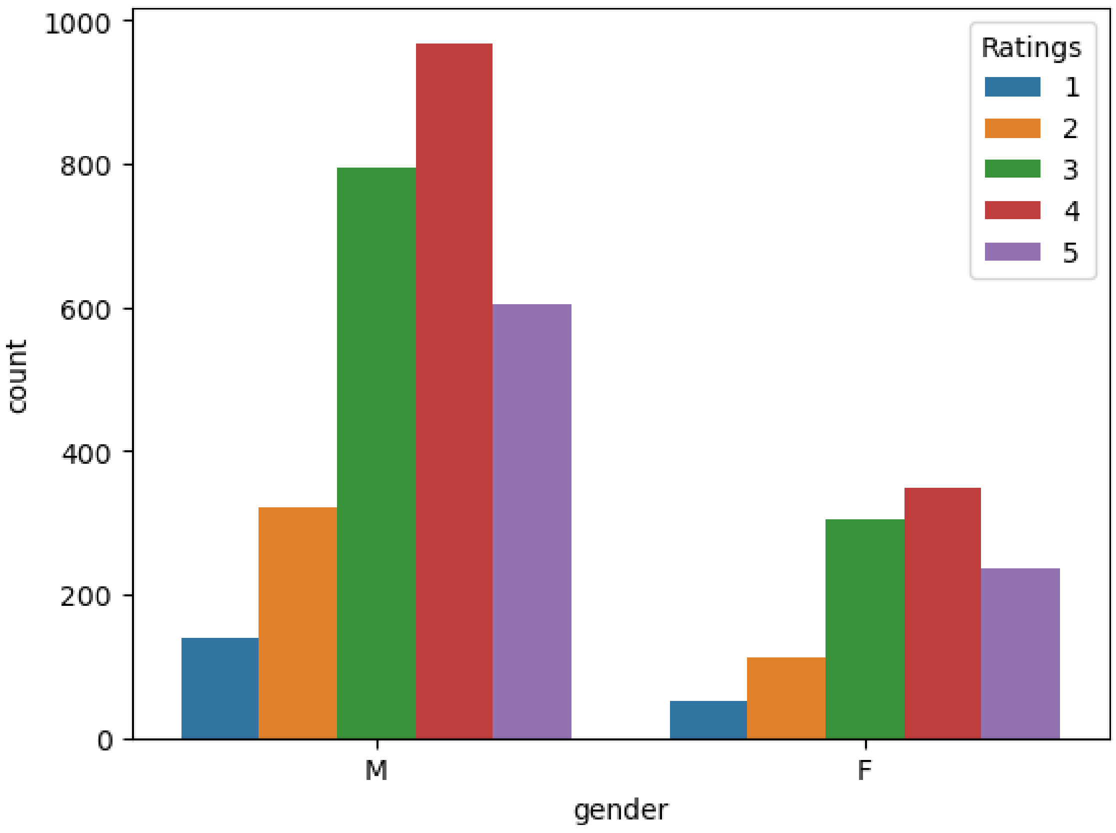
In [28]:
```
df_final.shape
```

Out[28]:
```
(3882, 9)
```

Using data visulaisation we can represent the processed data

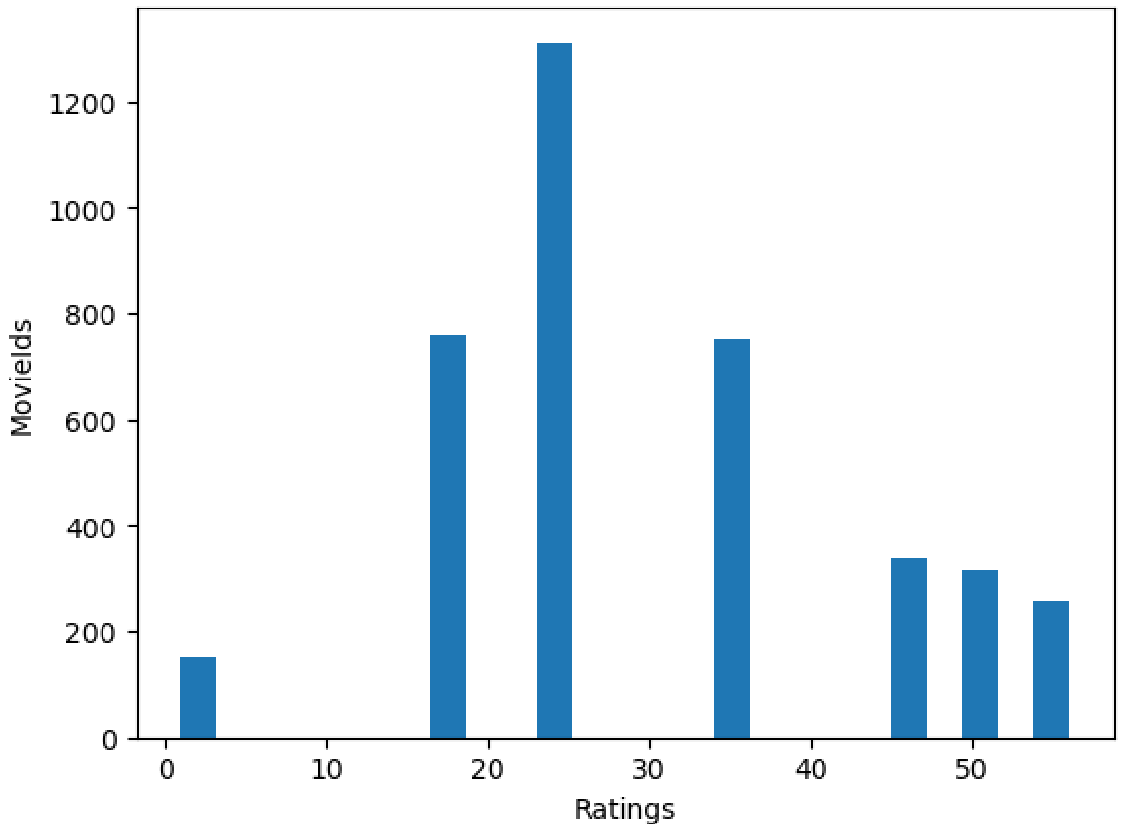
In [31]:
```
sns.countplot(x=df_final['gender'],hue=df_final['Ratings'])
```

Out[31]:
```
<AxesSubplot:xlabel='gender', ylabel='count'>
```

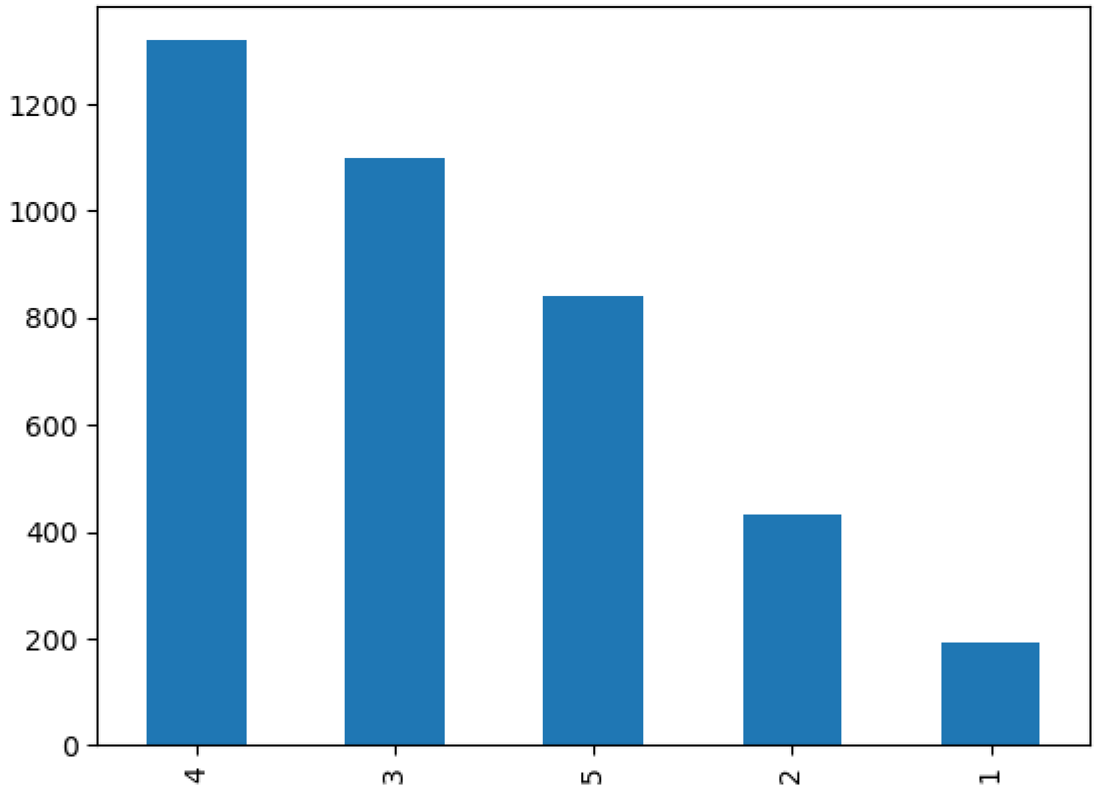


In [33]:
```
df_final.age.plot.hist(bins=25)
plt.ylabel("MovieIds")
plt.xlabel("Ratings")
```

Out[33]:
```
Text(0.5, 0, 'Ratings')
```

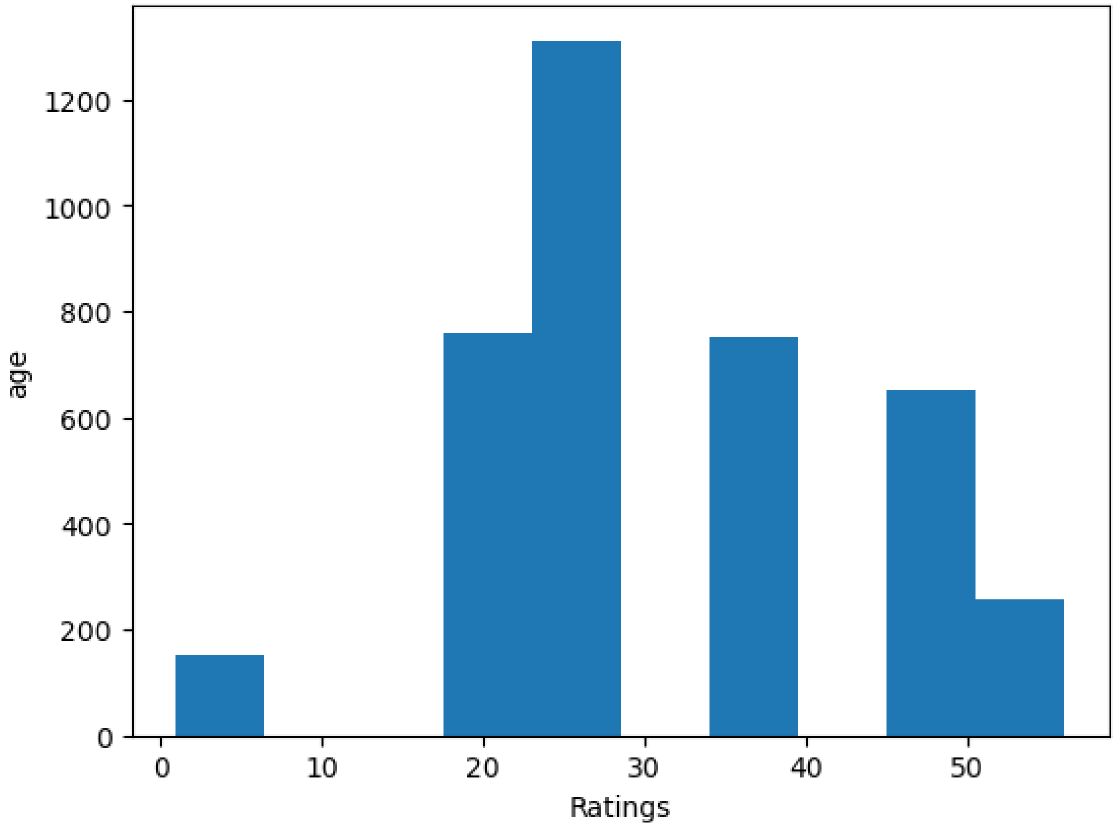


In [34]:
```
df_final['Ratings'].value_counts().plot(kind='bar')
plt.show()
```



In [39]:
```
df_final['age'].plot.hist(bins=10)
plt.xlabel("Ratings")
plt.ylabel("age")
```

Out[39]:
```
Text(0, 0.5, 'age')
```



FINAL DATASET

In [40]:
```
df_final.head()
```

Out[40]:

	MovieID	MovieName	Genres	UserId	MovieID	Rati
0	2.0	Jumanji,(1995)	Adventure|Children's|Fantasy,,,,,,,,,,,,,,	1	661	
1	3.0	Grumpier,Old,Men,(1995)	Comedy|Romance,,,,,,,,,,,,	1	914	
2	4.0	Waiting,to,Exhale,(1995)	Comedy|Drama,,,,,,,,,,,,	1	3408	
3	5.0	Father,of,the,Bride,Part,II,(1995)	Comedy,,,,,,,,,	1	2355	
4	6.0	Heat,(1995)	Action|Crime|Thriller,,,,,,,,,,,,,,	1	1197	

In [43]:
```
input = df_final.drop(['Ratings','MovieName','Genres','MovieID'], axis=
target=df_final['Ratings']
```

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

Out[44]:
```
0    3
1    3
2    4
3    5
4    3
Name: Ratings, dtype: int64
```

In [45]:
```
input.head()
```

Out[45]:

	UserId	userid	gender	age
0	1	2.0	M	56.0
1	1	3.0	M	25.0
2	1	4.0	M	45.0
3	1	5.0	M	25.0
4	1	6.0	F	50.0

Training the model using the logistic regression

In [68]:
```
X_train,X_test, Y_train, Y_test=train_test_split(input,target,test_size
```

In [69]:
```
print(Y_train)
print(Y_test)
```

```
3702    3
601     5
2869    4
3476    4
3213    4
       ..
1733    4
2831    4
1250    4
2761    4
3134    4
Name: Ratings, Length: 2717, dtype: int64
3798    5
3864    3
2325    1
411     5
3045    4
       ..
1217    5
1228    5
3505    1
2700    4
3653    2
Name: Ratings, Length: 1165, dtype: int64
```

In [74]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model
```

Out[74]:
```
LogisticRegression()
```

In [75]:
```
X_test = np.array(X_test)
```

In [77]:
```
X_test = np.array(X_test)
```

```
array([4., 4., 4., ..., 4., 4., 4.])
```