Movie lens 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.

Analysis Tasks to be performed:

Import the three datasets

1. 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)
2. 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

Output of the analysis:

Complete dataset: the dataset after merging all three dataset (Rating.dat, User.dat, movie.dat)

	UserID	MovieID	Rating	Timestamp	Title	Genre	Gender	Age	Occupation	Zip_code
0	1	1193	5	978300760	One Flew Over the Cuckoo's Nest (1975)	Drama	F	1	10	48067
1	1	661	3	978302109	James and the Giant Peach (1996)	Animation|Children's|Musical	F	1	10	48067
2	1	914	3	978301968	My Fair Lady (1964)	Musical|Romance	F	1	10	48067
3	1	3408	4	978300275	Erin Brockovich (2000)	Drama	F	1	10	48067
4	1	2355	5	978824291	Bug's Life, A (1998)	Animation|Children's|Comedy	F	1	10	48067

Some basic information about the merged dataset

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 10 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   UserID      1000209 non-null  int64
 1   MovieID     1000209 non-null  int64
 2   Rating      1000209 non-null  int64
 3   Timestamp   1000209 non-null  int64
 4   Title       1000209 non-null  object
 5   Genre       1000209 non-null  object
 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
```

As we see that occupation data type is integer, it is a Nominal variable so changed the dtype to o

Check for null value

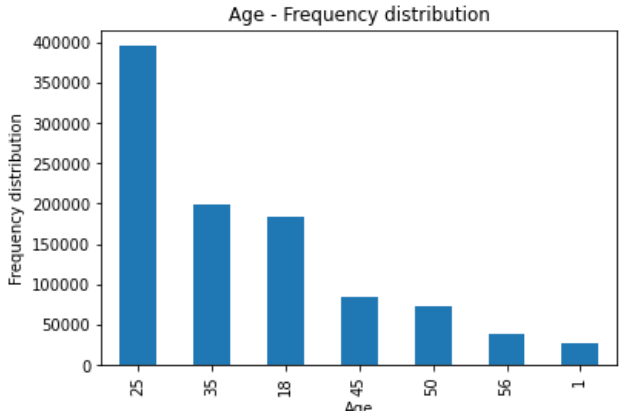
```
In [38]: master_data.isnull().sum()

Out[38]: UserID        0
         MovieID       0
         Rating        0
         Timestamp     0
         Title         0
         Genre         0
         Gender        0
         Age           0
         Occupation    0
         Zip_code      0
         dtype: int64
```

It has no null value so now we move towards explorative data analysis

User age Distribution

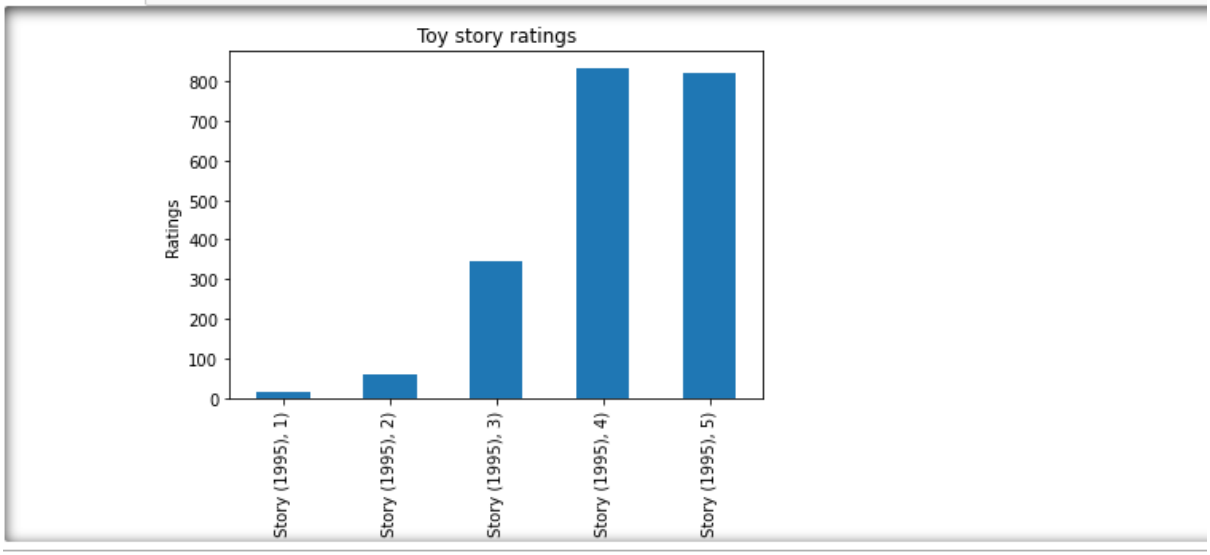
```
In [39]: master_data['Age'].value_counts().plot(kind='bar')
         plt.xlabel('Age')
         plt.ylabel('Frequency distribution')
         plt.title('Age - Frequency distribution')
         plt.show()
```



We can see from the above chart, according to the Age distribution we have most users in the range of “25-34” followed by “35-44” and least in the range of “Under 18”

User rating of the movie “Toy Story”

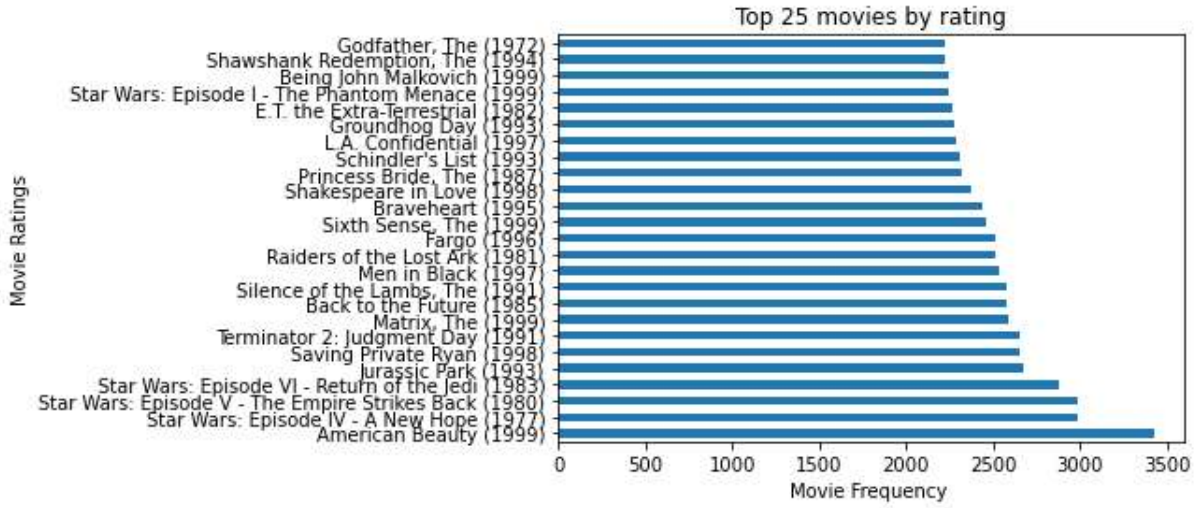
```
In [24]: toystory_ratings.groupby(['Title', 'Rating']).size().plot(kind = 'bar')
         plt.xlabel("Toy Story (1995) Ratings")
         plt.ylabel("Ratings")
         plt.title("Toy story ratings")
         plt.xticks(rotation=90)
         plt.show()
```



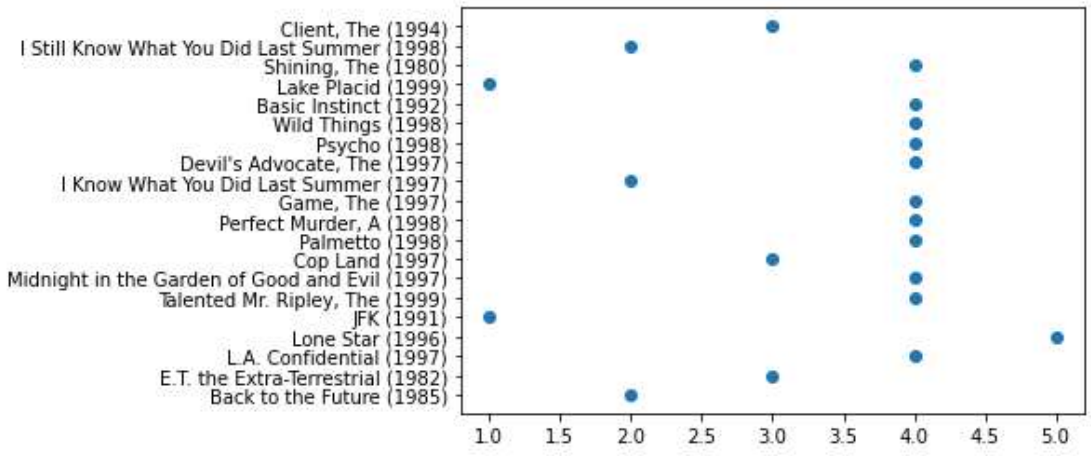
We can concur that most user i.e 835 have rated the movie ‘Toy Story” as 4 rating point and next 820 user have rated it as 5 rating point

Top 25 movies by viewership rating

```
In [26]: top25 = master_data.groupby('Title').size().sort_values(ascending=False)[0:25]
         top25.plot(kind='barh')
         plt.xlabel("Movie Frequency ")
         plt.ylabel("Movie Ratings")
         plt.title("Top 25 movies by rating")
         plt.show()
```



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



Feature Engineering:

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)

```
In [37]: list_genre = master_data['Genre'].str.split('|')

In [38]: list_genre.values

Out[38]: array([list(['Drama']), list(['Animation', "Children's", 'Musical']),
                list(['Musical', 'Romance']), ..., list(['Horror']),
                list(['Action', 'War']), list(['Adventure', 'Drama', 'Romance'])],
               dtype=object)

In [40]: # Find out all the unique genres

         listGenres = set()
         for genre in list_genre:
             listGenres = listGenres.union(set(genre))

In [42]: #type(listGenres)
         print(listGenres)

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

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.

```
In [50]: # 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.

         rating_onehotencoding = master_data['Genre'].str.get_dummies(sep='|')

In [51]: rating_onehotencoding.head()
```

Out[51]:

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Wes
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
1	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	
3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
4	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	

Develop an appropriate model to predict the movie ratings

To build the model we have to choose the feature and target (Ratings) selection

Create a smaller features dataframe of the movie data features that are relevant to rating outcome:

- Gender
- Age
- Occupation

```
In [77]: # Develop appropriate model to predict the ratings

         X_feature = df_final2[['Age','Gender','Occupation']].values
```

```
In [78]: X_feature

Out[78]: array([[1, '1', '10'],
                [1, '1', '10'],
                [1, '1', '10'],
                ...,
                [45, '0', '5'],
                [45, '0', '5'],
                [45, '0', '5']], dtype=object)
```

Pull the ratings column into target dataframe

```
In [79]: Y_target = master_data[['Rating']].values
```

```
In [80]: print(X_feature.shape)
         print(Y_target.shape)

         (1000209, 3)
         (1000209, 1)
```

Split the data into training and testing data. Set train – test split = 80-20%

```
In [81]: X_train, X_test, Y_train, Y_test = train_test_split(X_feature,Y_target,test_size=0.2, random_state=42)
```

```
In [82]: print(X_train.shape)
         print(X_test.shape)
         print(Y_train.shape)
         print(Y_test.shape)

         (800167, 3)
         (200042, 3)
         (800167, 1)
         (200042, 1)
```

Logistic Regression:

Logistics regression is used to predict the categorical variable. So we fit the training data into logreg estimator and try to predict the ratings for the test subject

```
In [83]: # fitting the logistics regression for rating based on Age, Gender, Occupation

         from sklearn import metrics
         logreg.fit(X_train,np.ravel(Y_train, order='c'))
         y_pred_log = logreg.predict(X_test)
         print(metrics.accuracy_score(y_pred_log,Y_test)*100)
```

```
34.768198678277564
```

KNN Classifier

Knn classifier used for the movie rating based on predictor

```
In [84]: # KNN classifier

         knn = KNeighborsClassifier(n_neighbors=3)
         knn.fit(X_train,np.ravel(Y_train, order='c'))
         y_pred_knn = knn.predict(X_test)
         print(metrics.accuracy_score(y_pred_knn,Y_test)*100)
```

```
24.460863218724068
```