In [1]:
```
# Import library
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
%matplotlib inline
```

1. Importing datasets

In [2]:
```
df_movie = pd.read_csv("/Users/rgm/DSwithPython/Data science with P
ython 1/movies.dat",sep="::", header=None, names=['MovieID','Title'
,'Genres'],
                        dtype={'MovieID': np.int32, 'Title': np.str,
'Genres': np.str}, engine='python')
df_user = pd.read_csv("/Users/rgm/DSwithPython/Data science with Py
thon 1/users.dat",sep="::", header=None, names=['UserID','Gender','
Age','Occupation','Zip-code'],
                        dtype={'UserID': np.int32, 'Gender': np.str,
'Age': np.int32, 'Occupation' : np.int32, 'Zip-code' : np.str}, eng
ine='python')
df_ratings = pd.read_csv("/Users/rgm/DSwithPython/Data science with
Python 1/ratings.dat",sep="::", header=None, names=['UserID','Movie
ID','Rating','Timestamp'],
                        dtype={'UserID': np.int32, 'MovieID': np.int
32, 'Rating': np.int32, 'Timestamp' : np.str}, engine='python')
```

Descriptive Analysis for Data

In [3]:
```
df_movie.shape
```

Out[3]: (3883, 3)

In [4]:
```
df_user.shape
```

Out[4]: (6040, 5)

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

Out[5]: (1000209, 4)

In [6]:
```
df_movie.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]:
```
df_user.head()
```

Out[7]:

	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 [8]:
```
df_ratings.head()
```

Out[8]:

	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

Checking for null values in all Datasets

In [9]:
```
df_movie.isnull().sum()
```

Out[9]:
```
MovieID    0
Title      0
Genres     0
dtype: int64
```

In [10]:
```
df_user.isnull().sum()
```

Out[10]:
```
UserID        0
Gender        0
Age           0
Occupation    0
Zip-code      0
dtype: int64
```

In [11]:
```
df_ratings.isnull().sum()
```

Out[11]:
```
UserID       0
MovieID      0
Rating       0
Timestamp    0
dtype: int64
```

No null or blank values in Datasets

In [12]:
```
df_movie.describe()
```

Out[12]:

	MovieID
count	3883.000000
mean	1986.049446
std	1146.778349
min	1.000000
25%	982.500000
50%	2010.000000
75%	2980.500000
max	3952.000000

In [13]:
```
df_user.describe()
```

Out[13]:

	UserID	Age	Occupation
count	6040.000000	6040.000000	6040.000000
mean	3020.500000	30.639238	8.146854
std	1743.742145	12.895962	6.329511
min	1.000000	1.000000	0.000000
25%	1510.750000	25.000000	3.000000
50%	3020.500000	25.000000	7.000000
75%	4530.250000	35.000000	14.000000
max	6040.000000	56.000000	20.000000

In [14]:
```
df_ratings.describe()
```

Out[14]:

	UserID	MovieID	Rating
count	1.000209e+06	1.000209e+06	1.000209e+06
mean	3.024512e+03	1.865540e+03	3.581564e+00
std	1.728413e+03	1.096041e+03	1.117102e+00
min	1.000000e+00	1.000000e+00	1.000000e+00
25%	1.506000e+03	1.030000e+03	3.000000e+00
50%	3.070000e+03	1.835000e+03	4.000000e+00
75%	4.476000e+03	2.770000e+03	4.000000e+00
max	6.040000e+03	3.952000e+03	5.000000e+00

In [15]:
```
df_movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3883 entries, 0 to 3882
Data columns (total 3 columns):
MovieID    3883 non-null int32
Title      3883 non-null object
Genres     3883 non-null object
dtypes: int32(1), object(2)
memory usage: 76.0+ KB
```

In [16]:
```
df_user.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6040 entries, 0 to 6039
Data columns (total 5 columns):
UserID        6040 non-null int32
Gender        6040 non-null object
Age           6040 non-null int32
Occupation    6040 non-null int32
Zip-code      6040 non-null object
dtypes: int32(3), object(2)
memory usage: 165.3+ KB
```

In [17]:
```
df_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000209 entries, 0 to 1000208
Data columns (total 4 columns):
UserID       1000209 non-null int32
MovieID      1000209 non-null int32
Rating       1000209 non-null int32
Timestamp    1000209 non-null object
dtypes: int32(3), object(1)
memory usage: 19.1+ MB
```

2. Merging Datasets to create Master_Data

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)

In [18]:
```
df_user_ratings = pd.merge(df_user,df_ratings, on='UserID')
```

In [19]:
```
df_user_ratings.head()
```

Out[19]:

	UserID	Gender	Age	Occupation	Zip-code	MovieID	Rating	Timestamp
0	1	F	1	10	48067	1193	5	978300760
1	1	F	1	10	48067	661	3	978302109
2	1	F	1	10	48067	914	3	978301968
3	1	F	1	10	48067	3408	4	978300275
4	1	F	1	10	48067	2355	5	978824291

In [20]:
```
Master_Data = pd.merge(df_user_ratings,df_movie, on='MovieID')
```

In [21]:
```
Master_Data.tail(10)
```

Out[21]:

	UserID	Gender	Age	Occupation	Zip-code	MovieID	Rating	Timestamp	Ti
1000199	5334	F	56	13	46140	3382	5	960796159	Song Freed (19
1000200	5420	F	1	19	14850	1843	3	960156505	Slappy a the Stink (19
1000201	5433	F	35	17	45014	286	3	960240881	Nemesis Neb (19
1000202	5494	F	35	17	94306	3530	4	959816296	Smoking/ Smok (19
1000203	5556	M	45	6	92103	2198	3	959445515	Modulatic (19
1000204	5949	M	18	17	47901	2198	5	958846401	Modulatic (19
1000205	5675	M	35	14	30030	2703	3	976029116	Brok Vess (19
1000206	5780	M	18	17	92886	2845	1	958153068	White Bc (19
1000207	5851	F	18	20	55410	3607	5	957756608	One Li Ind (19
1000208	5938	M	25	1	35401	2909	4	957273353	Five Wiv Th Secretar and (19

In [22]:
```
Master_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 10 columns):
UserID        1000209 non-null int32
Gender        1000209 non-null object
Age           1000209 non-null int32
Occupation    1000209 non-null int32
Zip-code      1000209 non-null object
MovieID       1000209 non-null int32
Rating        1000209 non-null int32
Timestamp     1000209 non-null object
Title         1000209 non-null object
Genres        1000209 non-null object
dtypes: int32(5), object(5)
memory usage: 64.9+ MB
```

In [23]:
```
Master_Data.shape
```

Out[23]:
```
(1000209, 10)
```

In [24]:
```
Master_Data.isnull().sum()
```

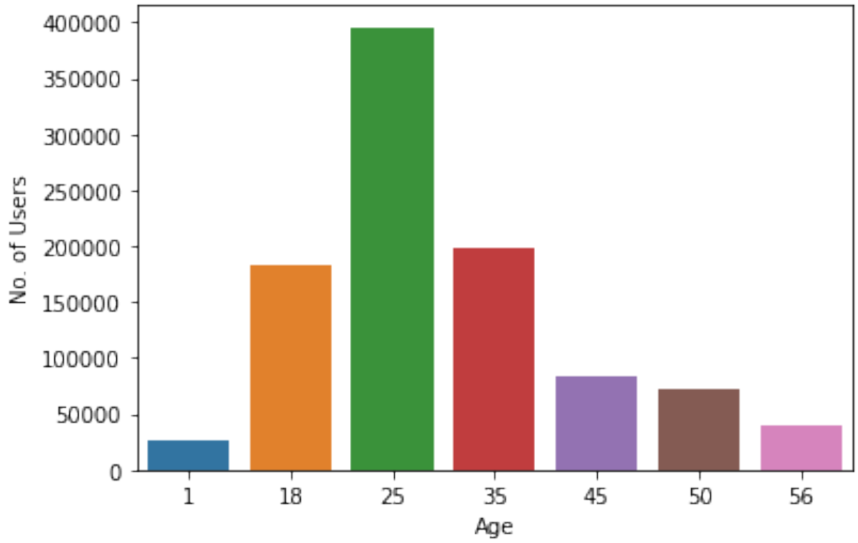
Out[24]:
```
UserID        0
Gender        0
Age           0
Occupation    0
Zip-code      0
MovieID       0
Rating        0
Timestamp     0
Title         0
Genres        0
dtype: int64
```

Master_Data does not have any blank values

3. Visual Representation of datasets: Data Visualisation

3(a). User Age Distribution

In [25]:
```
sns.barplot(x=Master_Data['Age'].value_counts().to_frame().index,y=
'Age', data = Master_Data['Age'].value_counts().to_frame())
plt.xlabel('Age')
plt.ylabel('No. of Users')
plt.show()
```



3(b). User rating of the movie “Toy Story”

In [26]:
```
Master_Data_TS = Master_Data[Master_Data['Title']=='Toy Story (1995
)']
```

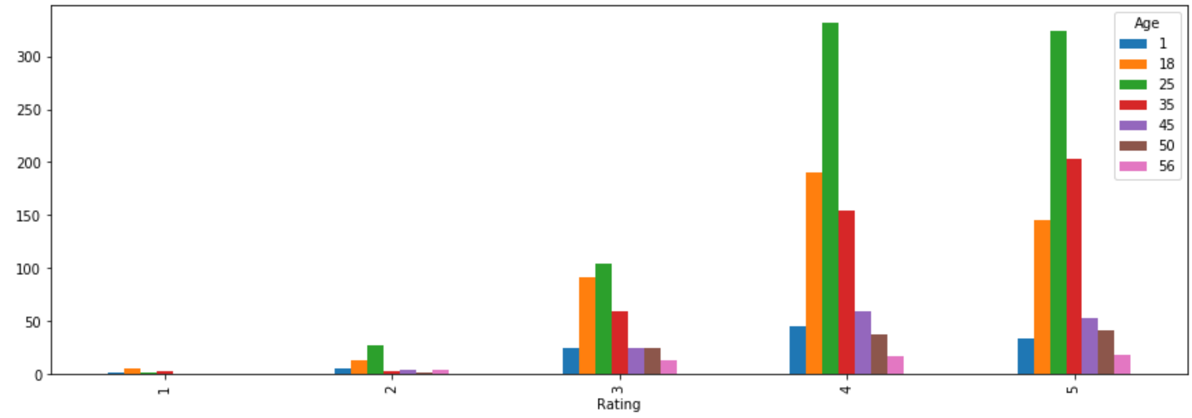
In [27]:
```
pd.crosstab(Master_Data_TS.Rating,Master_Data_TS.Age)
```

Out[27]:

Age	1	18	25	35	45	50	56
Rating							
1	2	6	2	3	1	1	1
2	6	14	27	3	5	2	4
3	25	92	105	60	25	25	13
4	45	190	332	154	59	38	17
5	34	146	324	203	53	42	18

In [28]:
```
pd.crosstab(Master_Data_TS.Rating,Master_Data_TS.Age).plot.bar(figsize=(15,5))
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1a283e4950>



3(c). Top 25 movies by viewership rating

In [29]:
```
Master_Data[Master_Data.Rating == 5].Title.value_counts().head(25)
```

Out[29]:
```
American Beauty (1999)                                    1963
Star Wars: Episode IV - A New Hope (1977)                 1826
Raiders of the Lost Ark (1981)                            1500
Star Wars: Episode V - The Empire Strikes Back (1980)     1483
Godfather, The (1972)                                     1475
Schindler's List (1993)                                   1475
Shawshank Redemption, The (1994)                          1457
Matrix, The (1999)                                        1430
Saving Private Ryan (1998)                                1405
Sixth Sense, The (1999)                                   1385
Silence of the Lambs, The (1991)                          1350
Fargo (1996)                                              1278
Braveheart (1995)                                         1206
Pulp Fiction (1994)                                       1193
Princess Bride, The (1987)                                1186
Usual Suspects, The (1995)                                1144
Star Wars: Episode VI - Return of the Jedi (1983)         1028
L.A. Confidential (1997)                                  1009
Being John Malkovich (1999)                               1007
Shakespeare in Love (1998)                                 987
Casablanca (1942)                                          984
Forrest Gump (1994)                                        945
Terminator 2: Judgment Day (1991)                          942
Godfather: Part II, The (1974)                             941
One Flew Over the Cuckoo's Nest (1975)                     937
Name: Title, dtype: int64
```

3(d). Find the ratings for all the movies reviewed by for a particular user of user id = 2696

In [30]:
```
Master_Data[Master_Data.UserID == 2696][['Title', 'Rating']]
```

Out[30]:

	Title	Rating
24345	Back to the Future (1985)	2
29848	E.T. the Extra-Terrestrial (1982)	3
244232	L.A. Confidential (1997)	4
250014	Lone Star (1996)	5
273633	JFK (1991)	1
277808	Talented Mr. Ripley, The (1999)	4
371178	Midnight in the Garden of Good and Evil (1997)	4
377250	Cop Land (1997)	3
598042	Palmetto (1998)	4
603189	Perfect Murder, A (1998)	4
609204	Game, The (1997)	4
611956	I Know What You Did Last Summer (1997)	2
612552	Devil's Advocate, The (1997)	4
613486	Psycho (1998)	4
616546	Wild Things (1998)	4
618708	Basic Instinct (1992)	4
621101	Lake Placid (1999)	1
689379	Shining, The (1980)	4
697451	I Still Know What You Did Last Summer (1998)	2
777089	Client, The (1994)	3

4(a). Feature Engineering

In [31]:
```
#Splitting data for Unique Genre
df_movie.Genres.str.get_dummies("|")
```

Out[31]:

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fa
0	0	0	1	1	1	0	0	0	
1	0	1	0	1	0	0	0	0	
2	0	0	0	0	1	0	0	0	
3	0	0	0	0	1	0	0	1	
4	0	0	0	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	
3878	0	0	0	0	1	0	0	0	
3879	0	0	0	0	0	0	0	1	
3880	0	0	0	0	0	0	0	1	
3881	0	0	0	0	0	0	0	1	
3882	0	0	0	0	0	0	0	1	

3883 rows × 18 columns

4(b). Adding each Genre as Column

In [32]:
```
modified_movie_df = pd.concat([df_movie,df_movie.Genres.str.get_dum
mies("|")], axis=1)
```

In [33]:
```
modified_movie_df.head()
```

Out[33]:

	MovieID	Title	Genres	Action	Adventure	Animation	Children's
0	1	Toy Story (1995)	Animation|Children's|Comedy	0	0	1	1
1	2	Jumanji (1995)	Adventure|Children's|Fantasy	0	1	0	1
2	3	Grumpier Old Men (1995)	Comedy|Romance	0	0	0	0
3	4	Waiting to Exhale (1995)	Comedy|Drama	0	0	0	0
4	5	Father of the Bride Part II (1995)	Comedy	0	0	0	0

5 rows × 21 columns

Converting Gender values to Numerical values

In [34]:
```
df_user_ratings['Gender'] = df_user_ratings['Gender'].replace(['M',
'F'], [1, 0])
```

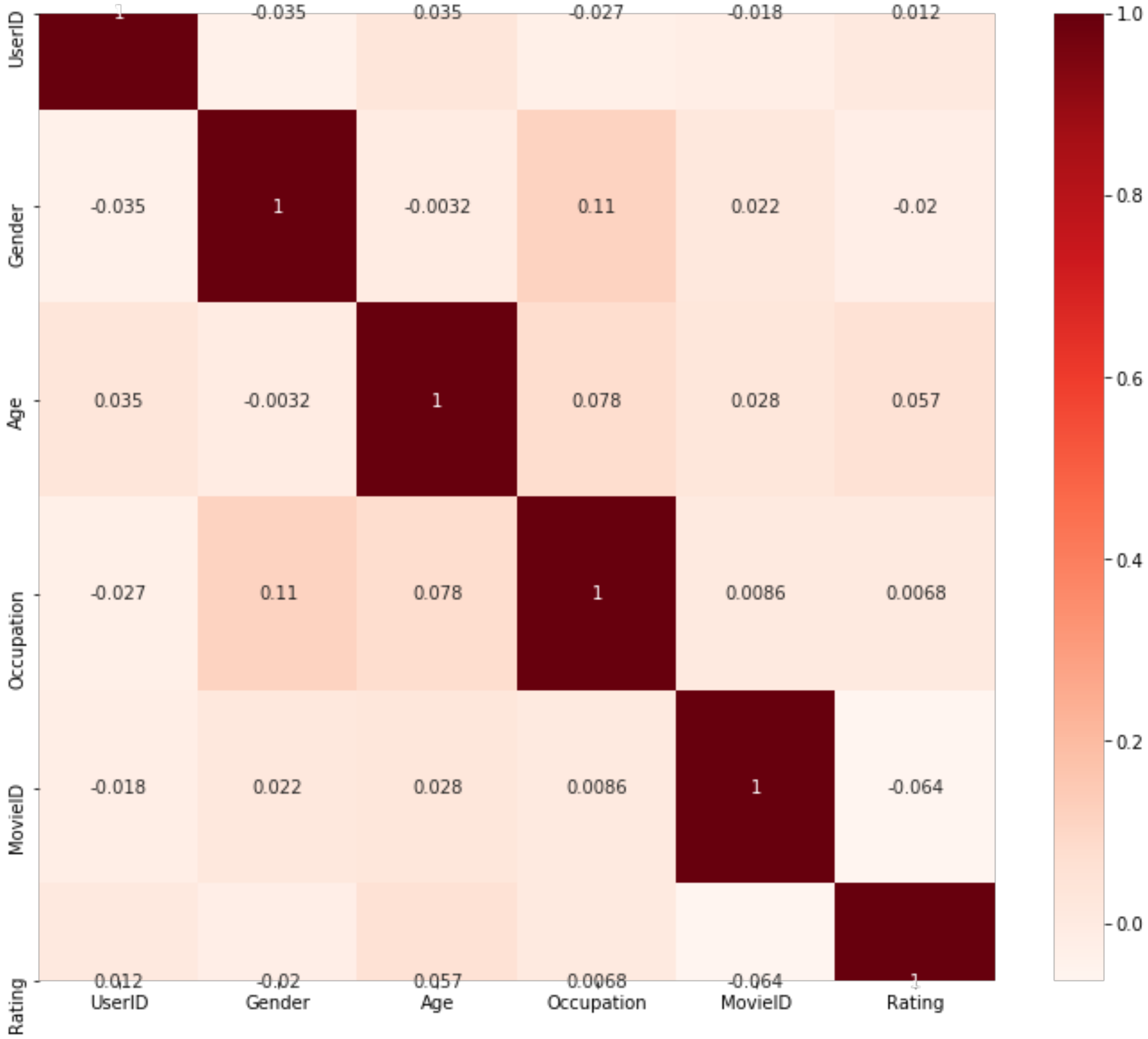
In [35]:
```
df_user_ratings.Gender.head()
```

Out[35]:
```
0    0
1    0
2    0
3    0
4    0
Name: Gender, dtype: int64
```

4(c). Feature affecting movie ratings

In [36]:
```
#Using Pearson Correlation

#include Genre

print(df_user_ratings.corr())

plt.figure(figsize=(12,10))
cor = df_user_ratings.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()

#Gender being in M & F, need to convert to 0 & 1.
```

```
                UserID    Gender       Age  Occupation   MovieID  
Rating
UserID      1.000000 -0.035042  0.034688   -0.026698 -0.017739  0.
012303
Gender     -0.035042  1.000000 -0.003189    0.114974  0.021626 -0.
019861
Age         0.034688 -0.003189  1.000000    0.078371  0.027575  0.
056869
Occupation -0.026698  0.114974  0.078371    1.000000  0.008585  0.
006753
MovieID    -0.017739  0.021626  0.027575    0.008585  1.000000 -0.
064042
Rating      0.012303 -0.019861  0.056869    0.006753 -0.064042  1.
000000
```



4(d). Model Fitting

In [37]:
```
#Merging User Data and Ratings Data for Model fitting
df_model = pd.merge(df_ratings, df_user, how='left', left_on=['User
ID'], right_on=['UserID'])
```

In [38]:
```
df_model['Gender'] = df_model['Gender'].replace(['M', 'F'], [1, 0])
```

In [39]:
```
df_model.head()
```

Out[39]:

	UserID	MovieID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
0	1	1193	5	978300760	0	1	10	48067
1	1	661	3	978302109	0	1	10	48067
2	1	914	3	978301968	0	1	10	48067
3	1	3408	4	978300275	0	1	10	48067
4	1	2355	5	978824291	0	1	10	48067

In [40]:
```
df_model.tail()
```

Out[40]:

	UserID	MovieID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
1000204	6040	1091	1	956716541	1	25	6	11106
1000205	6040	1094	5	956704887	1	25	6	11106
1000206	6040	562	5	956704746	1	25	6	11106
1000207	6040	1096	4	956715648	1	25	6	11106
1000208	6040	1097	4	956715569	1	25	6	11106

In [41]:
```
df_model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 8 columns):
UserID        1000209 non-null int32
MovieID       1000209 non-null int32
Rating        1000209 non-null int32
Timestamp     1000209 non-null object
Gender        1000209 non-null int64
Age           1000209 non-null int32
Occupation    1000209 non-null int32
Zip-code      1000209 non-null object
dtypes: int32(5), int64(1), object(2)
memory usage: 49.6+ MB
```

In [42]:
```
df_model.shape
```

Out[42]:
```
(1000209, 8)
```

Selecting first 10000 records for model fitting

In [43]:
```
# Select only few records of whole dataset if in case model fitting
takes time
df_model = df_model.head(10000)
```

In [44]:
```
#pre-process data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit(df_model['Age'])
x_age = le.transform(df_model['Age'])
x_age
```

Out[44]:
```
array([0, 0, 0, ..., 1, 1, 1])
```

In [45]:
```
le.fit(df_model['Occupation'])
x_occ = le.transform(df_model['Occupation'])
x_occ
```

Out[45]:
```
array([9, 9, 9, ..., 4, 4, 4])
```

In [46]:
```
set(x_occ)
```

Out[46]:
```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18}
```

In [47]:
```
le.fit(df_model['Gender'])
x_gender = le.transform(df_model['Gender'])
x_gender
```

Out[47]:
```
array([0, 0, 0, ..., 1, 1, 1])
```

In [48]:
```
df_model['New Age'] = x_age
df_model['New Occupation'] = x_occ
df_model['New Gender'] = x_gender
```

In [49]:
```
df_model.head()
```

Out[49]:

	UserID	MovieID	Rating	Timestamp	Gender	Age	Occupation	Zip-code	New Age	Ne Occupati
0	1	1193	5	978300760	0	1	10	48067	0	
1	1	661	3	978302109	0	1	10	48067	0	
2	1	914	3	978301968	0	1	10	48067	0	
3	1	3408	4	978300275	0	1	10	48067	0	
4	1	2355	5	978824291	0	1	10	48067	0	

Converting New Age, New Occupation, New Gender to Categorical variables

In [50]:
```
df_model['New Gender'] = df_model['New Gender'].astype('category')
df_model['New Occupation'] = df_model['New Occupation'].astype('cat
egory')
df_model['New Age'] = df_model['New Age'].astype('category')
df_model['Rating'] = df_model['Rating'].astype('category')
```

In [51]:
```
df_model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 0 to 9999
Data columns (total 11 columns):
UserID            10000 non-null int32
MovieID           10000 non-null int32
Rating            10000 non-null category
Timestamp         10000 non-null object
Gender            10000 non-null int64
Age               10000 non-null int32
Occupation        10000 non-null int32
Zip-code          10000 non-null object
New Age           10000 non-null category
New Occupation    10000 non-null category
New Gender        10000 non-null category
dtypes: category(4), int32(4), int64(1), object(2)
memory usage: 509.2+ KB
```

In [52]:
```
df_model = pd.merge(df_model, df_movie, how='left', left_on=['Movie
ID'], right_on=['MovieID'])
```

In [53]:
```
df_model = pd.concat([df_model,df_model.Genres.str.get_dummies("|")
], axis=1)
```

In [54]:
```
df_model.head()
```

Out[54]:

	UserID	MovieID	Rating	Timestamp	Gender	Age	Occupation	Zip-code	New Age	Ne Occupati
0	1	1193	5	978300760	0	1	10	48067	0	
1	1	661	3	978302109	0	1	10	48067	0	
2	1	914	3	978301968	0	1	10	48067	0	
3	1	3408	4	978300275	0	1	10	48067	0	
4	1	2355	5	978824291	0	1	10	48067	0	

5 rows × 31 columns

In [55]:
```
# df_model.head()
#df_model.info()
New_Genres = list(df_model.columns[13:])
```

In [56]:
```
df_model[New_Genres] = df_model[New_Genres].astype('category')
```

In [57]:
```
df_model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 0 to 9999
Data columns (total 31 columns):
UserID            10000 non-null int32
MovieID           10000 non-null int32
Rating            10000 non-null category
Timestamp         10000 non-null object
Gender            10000 non-null int64
Age               10000 non-null int32
Occupation        10000 non-null int32
Zip-code          10000 non-null object
New Age           10000 non-null category
New Occupation    10000 non-null category
New Gender        10000 non-null category
Title             10000 non-null object
Genres            10000 non-null object
Action            10000 non-null category
Adventure         10000 non-null category
Animation         10000 non-null category
Children's        10000 non-null category
Comedy            10000 non-null category
Crime             10000 non-null category
Documentary       10000 non-null category
Drama             10000 non-null category
Fantasy           10000 non-null category
Film-Noir         10000 non-null category
Horror            10000 non-null category
Musical           10000 non-null category
Mystery           10000 non-null category
Romance           10000 non-null category
Sci-Fi            10000 non-null category
Thriller          10000 non-null category
War               10000 non-null category
Western           10000 non-null category
dtypes: category(22), int32(4), int64(1), object(4)
memory usage: 843.0+ KB
```

In [58]:
```
x_input = df_model[['New Gender', 'New Age', 'New Occupation', 'Act
ion', 'Adventure', 'Animation', "Children's",
                    'Comedy', 'Crime', 'Documentary', 'Drama', 'Fan
tasy', 'Film-Noir', 'Horror', 'Musical',
                    'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'Wa
r', 'Western']]
y_target = df_model['Rating']
```

In [59]:
```
x_input.isnull().sum()
```

Out[59]:
```
New Gender        0
New Age           0
New Occupation    0
Action            0
Adventure         0
Animation         0
Children's        0
Comedy            0
Crime             0
Documentary       0
Drama             0
Fantasy           0
Film-Noir         0
Horror            0
Musical           0
Mystery           0
Romance           0
Sci-Fi            0
Thriller          0
War               0
Western           0
dtype: int64
```

Evaluate Algorithm

In [60]:
```
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysi
s
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

In [61]:
```
# Split-out validation dataset
x_train, x_test, y_train, y_test = train_test_split(x_input, y_targ
et, test_size=0.25)

x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[61]:
```
((7500, 21), (2500, 21), (7500,), (2500,))
```

In [62]:
```
# Fitting Logistic Regression
logitReg = LogisticRegression()
lm = logitReg.fit(x_train, y_train)
```

```
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
```

In [63]:
```
result = logitReg.predict(x_test)
estimated = pd.Series(result, name='Estimated Values')
final_result = pd.concat([y_test, estimated], axis=1)
```

In [64]:
```
# Test options and evaluation metric
print (accuracy_score(y_test, result))
print (confusion_matrix(y_test, result))
print (classification_report(y_test, result))
```

```
0.3472
[[  0   0  21  80   2]
 [  0   0  34 201  13]
 [  0   0  97 555  58]
 [  0   0  86 693  91]
 [  0   0  49 442  78]]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00       103
           2       0.00      0.00      0.00       248
           3       0.34      0.14      0.19       710
           4       0.35      0.80      0.49       870
           5       0.32      0.14      0.19       569

    accuracy                           0.35      2500
   macro avg       0.20      0.21      0.17      2500
weighted avg       0.29      0.35      0.27      2500


/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/me
trics/classification.py:1437: UndefinedMetricWarning: Precision an
d F-score are ill-defined and being set to 0.0 in labels with no p
redicted samples.
  'precision', 'predicted', average, warn_for)
```

In [65]:
```
# Checking other Algorithms fitting
seed = 7
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DTC', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=10, random_state=seed)
    cv_results = cross_val_score(model, x_train, y_train, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std(
))
    print(msg)
```

```
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
```

```
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
```

```
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
  FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/li
near_model/logistic.py:469: FutureWarning: Default multi_class wil
l be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
```

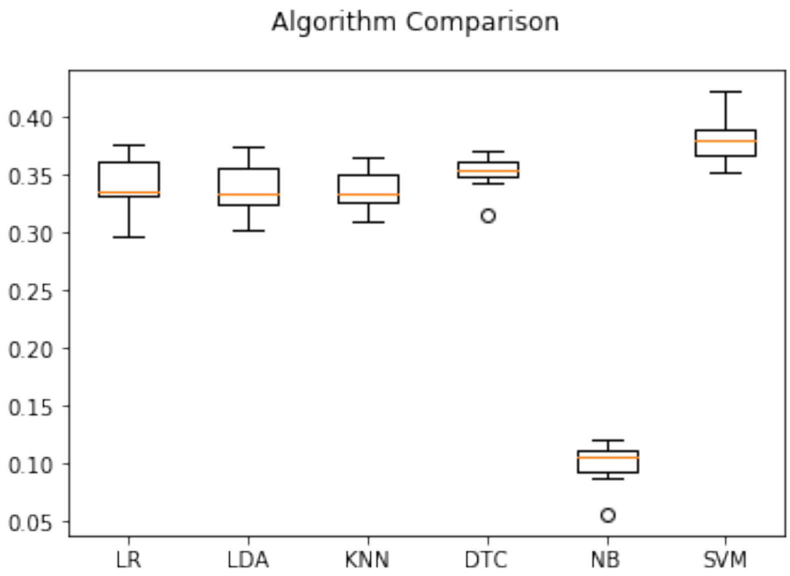
```
LR: 0.342133 (0.022096)
LDA: 0.339467 (0.020999)
KNN: 0.338667 (0.017127)
DTC: 0.352267 (0.014797)
NB: 0.100267 (0.017847)
```

```
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/sv
```

```
m/base.py:193: FutureWarning: The default value of gamma will chan
ge from 'auto' to 'scale' in version 0.22 to account better for un
scaled features. Set gamma explicitly to 'auto' or 'scale' to avoi
d this warning.
  "avoid this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/sv
m/base.py:193: FutureWarning: The default value of gamma will chan
ge from 'auto' to 'scale' in version 0.22 to account better for un
scaled features. Set gamma explicitly to 'auto' or 'scale' to avoi
d this warning.
  "avoid this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/sv
m/base.py:193: FutureWarning: The default value of gamma will chan
ge from 'auto' to 'scale' in version 0.22 to account better for un
scaled features. Set gamma explicitly to 'auto' or 'scale' to avoi
d this warning.
  "avoid this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/sv
m/base.py:193: FutureWarning: The default value of gamma will chan
ge from 'auto' to 'scale' in version 0.22 to account better for un
scaled features. Set gamma explicitly to 'auto' or 'scale' to avoi
d this warning.
  "avoid this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/sv
m/base.py:193: FutureWarning: The default value of gamma will chan
ge from 'auto' to 'scale' in version 0.22 to account better for un
scaled features. Set gamma explicitly to 'auto' or 'scale' to avoi
d this warning.
  "avoid this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/sv
m/base.py:193: FutureWarning: The default value of gamma will chan
ge from 'auto' to 'scale' in version 0.22 to account better for un
scaled features. Set gamma explicitly to 'auto' or 'scale' to avoi
d this warning.
  "avoid this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/sv
m/base.py:193: FutureWarning: The default value of gamma will chan
ge from 'auto' to 'scale' in version 0.22 to account better for un
scaled features. Set gamma explicitly to 'auto' or 'scale' to avoi
d this warning.
  "avoid this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/sv
m/base.py:193: FutureWarning: The default value of gamma will chan
ge from 'auto' to 'scale' in version 0.22 to account better for un
scaled features. Set gamma explicitly to 'auto' or 'scale' to avoi
d this warning.
  "avoid this warning.", FutureWarning)
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/sv
m/base.py:193: FutureWarning: The default value of gamma will chan
ge from 'auto' to 'scale' in version 0.22 to account better for un
scaled features. Set gamma explicitly to 'auto' or 'scale' to avoi
d this warning.
  "avoid this warning.", FutureWarning)
```

```
/Users/rgm/Python/anaconda3/lib/python3.7/site-packages/sklearn/sv
m/base.py:193: FutureWarning: The default value of gamma will chan
ge from 'auto' to 'scale' in version 0.22 to account better for un
scaled features. Set gamma explicitly to 'auto' or 'scale' to avoi
d this warning.
  "avoid this warning.", FutureWarning)

SVM: 0.381467 (0.018451)
```

In [66]:
```
# Compare Algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



From above analysis we can infer that SVM Model fits best with this data with value being 0.38

Adding Model to Predict Movie Rating

In [67]:
```
#recommendation model
modified_movie_df.head()
```

Out[67]:

	MovieID	Title	Genres	Action	Adventure	Animation	Children's
0	1	Toy Story (1995)	Animation|Children's|Comedy	0	0	1	1
1	2	Jumanji (1995)	Adventure|Children's|Fantasy	0	1	0	1
2	3	Grumpier Old Men (1995)	Comedy|Romance	0	0	0	0
3	4	Waiting to Exhale (1995)	Comedy|Drama	0	0	0	0
4	5	Father of the Bride Part II (1995)	Comedy	0	0	0	0

5 rows × 21 columns

In [68]:
```
#Using regular expressions to find a year stored between parenthese
s
#We specify the parantheses so we don't conflict with movies that h
ave years in their titles
modified_movie_df['Year'] = modified_movie_df.Title.str.extract('(\
(\d\d\d\d\))',expand=False)
#Removing the parentheses
modified_movie_df['Year'] = modified_movie_df.Year.str.extract('(\d
\d\d\d)',expand=False)
#Removing the years from the 'title' column
modified_movie_df['Title'] = modified_movie_df.Title.str.replace('(
\(\d\d\d\d\))', '')
#Applying the strip function to get rid of any ending whitespace ch
aracters that may have appeared
modified_movie_df['Title'] = modified_movie_df['Title'].apply(lambd
a x: x.strip())
modified_movie_df.head()
```

Out[68]:

	MovieID	Title	Genres	Action	Adventure	Animation	Children's
0	1	Toy Story	Animation|Children's|Comedy	0	0	1	1
1	2	Jumanji	Adventure|Children's|Fantasy	0	1	0	1
2	3	Grumpier Old Men	Comedy|Romance	0	0	0	0
3	4	Waiting to Exhale	Comedy|Drama	0	0	0	0
4	5	Father of the Bride Part II	Comedy	0	0	0	0

5 rows × 22 columns

In [69]:
```
df_ratings.head()
```

Out[69]:

	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 [70]:
```
#Drop removes a specified row or column from a dataframe
df_ratings = df_ratings.drop('Timestamp', 1)
df_ratings.head()
```

Out[70]:

	UserID	MovieID	Rating
0	1	1193	5
1	1	661	3
2	1	914	3
3	1	3408	4
4	1	2355	5

In [71]:
```
modified_movie_df.head()
# modified_movie_df = modified_movie_df.drop('Year',1)
```

Out[71]:

	MovieID	Title	Genres	Action	Adventure	Animation	Children's
0	1	Toy Story	Animation|Children's|Comedy	0	0	1	1
1	2	Jumanji	Adventure|Children's|Fantasy	0	1	0	1
2	3	Grumpier Old Men	Comedy|Romance	0	0	0	0
3	4	Waiting to Exhale	Comedy|Drama	0	0	0	0
4	5	Father of the Bride Part II	Comedy	0	0	0	0

5 rows × 22 columns

In [80]:
```
userInput = [
            {'Title':'Breakfast Club, The', 'rating':5},
            {'Title':'Toy Story', 'rating':3.5},
            {'Title':'Jumanji', 'rating':2},
            {'Title':"Pulp Fiction", 'rating':5},
            {'Title':'Akira', 'rating':4.5}
         ]
inputMovies = pd.DataFrame(userInput)
inputMovies
```

Out[80]:

	Title	rating
0	Breakfast Club, The	5.0
1	Toy Story	3.5
2	Jumanji	2.0
3	Pulp Fiction	5.0
4	Akira	4.5

In [81]:
```
#Filtering out the movies by title
inputId = modified_movie_df[modified_movie_df['Title'].isin(inputMo
vies['Title'].tolist())]
#Then merging it so we can get the movieId. It's implicitly merging
it by title.
inputMovies = pd.merge(inputId, inputMovies)
#Dropping information we won't use from the input dataframe
# inputMovies = inputMovies.drop('genres', 1).drop('year', 1)
#Final input dataframe
#If a movie you added in above isn't here, then it might not be in
the original
#dataframe or it might spelled differently, please check capitalisa
tion.
inputMovies
```

Out[81]:

	MovieID	Title	Genres	Action	Adventure	Animation	Children's
0	1	Toy Story	Animation|Children's|Comedy	0	0	1	1
1	2	Jumanji	Adventure|Children's|Fantasy	0	1	0	1
2	296	Pulp Fiction	Crime|Drama	0	0	0	0
3	1274	Akira	Adventure|Animation|Sci-Fi|Thriller	0	1	1	0
4	1968	Breakfast Club, The	Comedy|Drama	0	0	0	0

5 rows × 23 columns

In [82]:
```
#Filtering out the movies from the input
userMovies = modified_movie_df[modified_movie_df['MovieID'].isin(in
putMovies['MovieID'].tolist())]
userMovies
```

Out[82]:

	MovieID	Title	Genres	Action	Adventure	Animation	Childr
0	1	Toy Story	Animation|Children's|Comedy	0	0	1	
1	2	Jumanji	Adventure|Children's|Fantasy	0	1	0	
293	296	Pulp Fiction	Crime|Drama	0	0	0	
1254	1274	Akira	Adventure|Animation|Sci-Fi|Thriller	0	1	1	
1899	1968	Breakfast Club, The	Comedy|Drama	0	0	0	

5 rows × 22 columns

In [84]:
```
#Resetting the index to avoid future issues
userMovies = userMovies.reset_index(drop=True)
#Dropping unnecessary issues due to save memory and to avoid issues
userGenreTable = userMovies.drop('MovieID', 1).drop('Title', 1).dro
p('Genres', 1).drop('Year', 1)
userGenreTable
```

Out[84]:

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fanta
0	0	0	1	1	1	0	0	0	
1	0	1	0	1	0	0	0	0	
2	0	0	0	0	0	1	0	1	
3	0	1	1	0	0	0	0	0	
4	0	0	0	0	1	0	0	1	

In [85]:
```
inputMovies['rating']
```

Out[85]:
```
0    3.5
1    2.0
2    5.0
3    4.5
4    5.0
Name: rating, dtype: float64
```

In [86]:
```
#Dot produt to get weights
userProfile = userGenreTable.transpose().dot(inputMovies['rating'])
#The user profile
userProfile
```

Out[86]:
```
Action          0.0
Adventure       6.5
Animation       8.0
Children's      5.5
Comedy          8.5
Crime           5.0
Documentary     0.0
Drama          10.0
Fantasy         2.0
Film-Noir       0.0
Horror          0.0
Musical         0.0
Mystery         0.0
Romance         0.0
Sci-Fi          4.5
Thriller        4.5
War             0.0
Western         0.0
dtype: float64
```

In [88]:
```
#Now let's get the genres of every movie in our original dataframe
genreTable = modified_movie_df.set_index(modified_movie_df['MovieID
'])
#And drop the unnecessary information
genreTable = genreTable.drop('MovieID', 1).drop('Title', 1).drop('G
enres', 1).drop('Year', 1)
genreTable.head()
```

Out[88]:

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama
MovieID								
1	0	0	1	1	1	0	0	0
2	0	1	0	1	0	0	0	0
3	0	0	0	0	1	0	0	0
4	0	0	0	0	1	0	0	1
5	0	0	0	0	1	0	0	0

In [89]:
```
genreTable.shape
```

Out[89]:
```
(3883, 18)
```

In [90]:
```
#Multiply the genres by the weights and then take the weighted aver
age
recommendationTable_df = ((genreTable*userProfile).sum(axis=1))/(us
erProfile.sum())
recommendationTable_df.head()
```

Out[90]:
```
MovieID
1    0.403670
2    0.256881
3    0.155963
4    0.339450
5    0.155963
dtype: float64
```

In [91]:
```
#Sort our recommendations in descending order
recommendationTable_df = recommendationTable_df.sort_values(ascendi
ng=False)
#Just a peek at the values
recommendationTable_df.head()
```

Out[91]:
```
MovieID
673     0.559633
1566    0.522936
2054    0.495413
2138    0.467890
1259    0.458716
dtype: float64
```

In [92]:
```
#The final recommendation table
df_movie.loc[df_movie['MovieID'].isin(recommendationTable_df.head(2
0).keys())]
```

Out[92]:

	MovieID	Title	Genres
33	34	Babe (1995)	Children's|Comedy|Drama
399	403	Two Crimes (1995)	Comedy|Crime|Drama
667	673	Space Jam (1996)	Adventure|Animation|Children's|Comedy|Fantasy
1001	1014	Pollyanna (1960)	Children's|Comedy|Drama
1239	1259	Stand by Me (1986)	Adventure|Comedy|Drama
1445	1473	Best Men (1997)	Action|Comedy|Crime|Drama
1526	1566	Hercules (1997)	Adventure|Animation|Children's|Comedy|Musical
1663	1711	Midnight in the Garden of Good and Evil (1997)	Comedy|Crime|Drama|Mystery
1745	1809	Hana-bi (1997)	Comedy|Crime|Drama
1748	1812	Wide Awake (1998)	Children's|Comedy|Drama
1849	1918	Lethal Weapon 4 (1998)	Action|Comedy|Crime|Drama
1931	2000	Lethal Weapon (1987)	Action|Comedy|Crime|Drama
1932	2001	Lethal Weapon 2 (1989)	Action|Comedy|Crime|Drama
1933	2002	Lethal Weapon 3 (1992)	Action|Comedy|Crime|Drama
1985	2054	Honey, I Shrunk the Kids (1989)	Adventure|Children's|Comedy|Fantasy|Sci-Fi
2047	2116	Lord of the Rings, The (1978)	Adventure|Animation|Children's|Sci-Fi
2069	2138	Watership Down (1978)	Animation|Children's|Drama|Fantasy
3115	3184	Montana (1998)	Action|Comedy|Crime|Drama
3197	3266	Man Bites Dog (C'est arriv� pr�s de chez vous)...	Action|Comedy|Crime|Drama
3452	3521	Mystery Train (1989)	Comedy|Crime|Drama

END