SI671 Homework3

Name: Junqi Chen

Uniqname: junqich

UMID: 03846505

In [1]:
```
import networkx as nx
import numpy as np
import pandas as pd
```

Part 1: Exploratory Social Network Analysis

In [2]:
```
# a)
df = pd.read_csv('amazonNetwork.csv')
G = nx.from_pandas_edgelist(df, 'FromNodeId', 'ToNodeId', create_using=nx.DiGraph())
print(nx.info(G))
```

```
DiGraph with 2647 nodes and 10841 edges
```

In [3]:
```
# b)
print(f'There are {G.number_of_nodes()} items (nodes) present in the network.')
print(f'There are {G.number_of_edges()} co-purchases happened.')
```

```
There are 2647 items (nodes) present in the network.
There are 10841 co-purchases happened.
```

In [4]:
```
# c)
print(f'The average shortest distance is {nx.average_shortest_path_length(G)}')
```

```
The average shortest distance is 9.592795477759587
```

Answer for (c): The average shortest distance between the nodes inidicate the average numbers of steps need to take to walk from one node to another (both nodes are taken randomly). So in the amazonNetwork, it takes around 9.6 steps to get to another node in average.

In [5]:
```
# d)
print(f'The transitivity is {nx.transitivity(G)}')
print(f'The average clustering coefficient is {nx.average_clustering(G)}')
```

```
The transitivity is 0.4339169154480595
The average clustering coefficient is 0.4086089178720651
```

Answer for (d): According to the definition of transitivity and average clustering coefficient as shown below:

$$T = 3\frac{\#triangles}{\#triads}$$

$$C = \frac{1}{n}\sum_{v \in G} c_v$$

we can find out that both transitivity and average clustering coefficient are inidicators of the degree of association (or can alse be called degree of clustering) in the graph. In this graph, the numerical numbers of transitivity and average clustering coefficient are quiet close (both around 0.4), indicating that this graph is not a highly clustered graph.

In [6]:

```
# e)
pr = nx.pagerank(G, alpha=0.5)
pr_sorted = sorted(pr.items(), key=lambda x:x[1], reverse=True)
pr_top10 = pd.DataFrame(pr_sorted[:10], columns=["id", "PageRankScore"])
print(f'The top 10 nodes with the highest PageRank are:')
display(pr_top10)
```

```
The top 10 nodes with the highest PageRank are:
```

	id	PageRankScore
0	8	0.003625
1	481	0.002434
2	33	0.002297
3	18	0.002103
4	23	0.002079
5	30	0.001882
6	346	0.001863
7	99	0.001820
8	93	0.001792
9	21	0.001659

Answer for (e): From the table of top 10 nodes with the highest PageRank, we can find that these 10 nodes have the most importance in the graph G according to PageRank model. In other words, these 10 nodes are considered as the most important pages in the graph, indicating that they may be referred most frequently by other pages.

Part 2: Predicting Review-Rating using Features derived from network properties

2.1 Data Preprocessing

In [72]:

```
# load dataset
train = pd.read_csv('reviewTrain.csv')
train["group"] = [s.strip() for s in train["group"]]
# train = train.drop(columns=['title'])
train.head()
```

	id	title	group	review
0	3	World War II Allied Fighter Planes Trading Cards	Book	5.0
1	5	Prayers That Avail Much for Business: Executive	Book	0.0
2	7	Batik	Music	4.5
3	10	The Edward Said Reader	Book	4.0
4	11	Resetting the Clock : Five Anti-Aging Hormone...	Book	5.0

In [73]:
```python
test = pd.read_csv('reviewTest.csv')
test["group"] = [s.strip() for s in test["group"]]
# test = test.drop(columns=['title'])
test.head()
```

	id	title	group	review
0	90	The Eagle Has Landed	Book	NaN
1	1372	Che in Africa: Che Guevara's Congo Diary	Book	NaN
2	1382	The Darwin Awards II : Unnatural Selection	Book	NaN
3	253	Celtic Glory	Music	NaN
4	671	Sublte Aromatherapy	Book	NaN

In [32]:
```python
# compute features
clustering_coef = nx.clustering(G)
pr = nx.pagerank(G, alpha=0.5)
degree_cent = nx.degree_centrality(G)
closeness_cent = nx.closeness_centrality(G)
betweenness_cent = nx.betweenness_centrality(G)
```

In [74]:
```python
# add features
feature_list = []
def addFeature(feature_df, feature_name):
    feature_list.append(feature_name)
    train[feature_name] = [feature_df[i] if i in feature_df else 0 for i in train["id"]]
    test[feature_name] = [feature_df[i] if i in feature_df else 0 for i in test["id"]]

addFeature(clustering_coef, 'clustering_coef')
addFeature(pr, "pr")
addFeature(degree_cent, "degree_cent")
addFeature(closeness_cent, "closeness_cent")
addFeature(betweenness_cent, "betweenness_cent")

feature_list += list(pd.get_dummies(train['group'], prefix='group').keys())
train = pd.get_dummies(train, columns=["group"])
test = pd.get_dummies(test, columns=["group"])
test['group_Toy'] = 0

print(f'Here is the list of the features I used: {feature_list}')
```

```
Here is the list of the features I used: ['clustering_coef', 'pr', 'degree_cent', 'closeness_cent', 'bet
weenness_cent', 'group_Book', 'group_DVD', 'group_Music', 'group_Toy', 'group_Video']
```

In [59]:
```
train.head()
```

	id	title	review	clustering_coef	pr	degree_cent	closeness_cent	betweenness_cent	group_Bool
0	3	World War II Allied Fighter Planes Trading Cards	5.0	0.450000	0.000197	0.001890	0.000000	0.000000	1
1	5	Prayers That Avail Much for Business: Executive	0.0	0.142157	0.000774	0.005669	0.133688	0.004032	1
2	7	Batik	4.5	0.109562	0.001263	0.008692	0.150353	0.018768	0
3	10	The Edward Said Reader	4.0	0.285714	0.000424	0.003779	0.116834	0.003049	1
4	11	Resetting the Clock : Five Anti-Aging Hormone...	5.0	0.120344	0.000906	0.010204	0.008231	0.008756	1

In [60]:
```
test.head()
```

	id	title	review	clustering_coef	pr	degree_cent	closeness_cent	betweenness_cent	group_
0	90	The Eagle Has Landed	NaN	0.250000	0.000347	0.003779	0.116428	3.048563e-02	1
1	1372	Che in Africa: Che Guevara's Congo Diary	NaN	0.288462	0.000300	0.003023	0.080232	9.535202e-03	1
2	1382	The Darwin Awards II : Unnatural Selection	NaN	0.750000	0.000338	0.003401	0.063412	3.095826e-07	1
3	253	Celtic Glory	NaN	0.750000	0.000268	0.002268	0.072458	1.031101e-04	0
4	671	Sublte Aromatherapy	NaN	0.562500	0.000358	0.003401	0.093620	7.927132e-04	1

2.2 Model Building

In [96]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
```

In [93]:
```
X_train_val = train[feature_list]
y_train_val = train['review']
X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val, random_state=0)
X_test = test[feature_list]
```

In [97]:

```
# Support Vector Machine (SVM)
svr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2)).fit(X_train, y_train)
y_pred = svr.predict(X_val)
mean_absolute_error(y_val, y_pred)
```

```
1.4508816819598904
```

In [98]:

```
# Multi-layer perceptron
mlp = MLPRegressor(random_state=1, max_iter=500).fit(X_train, y_train)
y_pred = mlp.predict(X_val)
mean_absolute_error(y_val, y_pred)
```

```
1.6536110258690715
```

Comparing two models above, we find out that SVM has a better performance than Multi-layer perceptron on validation set, so we decide to adopt SVM as our final model and predict the result.

In [102]:

```
# make the prediction
svr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2)).fit(X_train_val, y_train_val)
y_pred = svr.predict(X_test)
```

In [103]:

```
# output result
test_result = pd.read_csv('reviewTest.csv')
test_result['review'] = y_pred
test_result.to_csv('reviewTest.csv')
```

In [ ]: