KNN (K-Nearest_Neighbors)

In [20]:
```
import pandas as pd
import numpy as np
from scipy import spatial
```

In [12]:
```
r_cols = ['user_id', 'movie_id', 'rating']
ratings = pd.read_csv('u.data', sep='\t', names=r_cols, usecols=range(3))
ratings.head()
```

Out[12]:

	user_id	movie_id	rating
0	0	50	5
1	0	172	5
2	0	133	1
3	196	242	3
4	186	302	3

In [15]:
```
movieProperties = ratings.groupby('movie_id').agg({'rating': [np.size, np.mean]})
movieProperties.head()
```

Out[15]:

	rating	
	size	mean
movie_id		
1	452	3.878319
2	131	3.206107
3	90	3.033333
4	209	3.550239
5	86	3.302326

In [17]:
```
movieNumRatings = pd.DataFrame(movieProperties['rating']['size'])
movieNormalizedNumRatings = movieNumRatings.apply(lambda x:
                                                   (x - np.min(x)) / (np.max(x) -
np.min(x)))
movieNormalizedNumRatings.head()
```

Out[17]:

	size
movie_id	
1	0.773585
2	0.222985
3	0.152659
4	0.356775
5	0.145798

Include genre information

In [18]:
```
movieDict = {}
with open(r'u.item', encoding = "ISO-8859-1") as f:
    temp = ''
    for line in f:
        #line.decode("ISO-8859-1")
        fields = line.rstrip('\n').split('|')
        movieID = int(fields[0])
        name = fields[1]
        genres = fields[5:25]
        genres = map(int, genres)
        movieDict[movieID] = (name, np.array(list(genres)),
                              movieNormalizedNumRatings.loc[movieID].get('size'),
                              movieProperties.loc[movieID].rating.get('mean'))
```

In [19]:
```
print(movieDict[1])
```

```
('Toy Story (1995)', array([0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0]), 0.7735849056603774, 3.8783185840707963)
```

Compute distance between two movies

In [21]:
```
from scipy import spatial

def ComputeDistance(a, b):
    genresA = a[1]
    genresB = b[1]
    genreDistance = spatial.distance.cosine(genresA, genresB)
    popularityA = a[2]
    popularityB = b[2]
    popularityDistance = abs(popularityA - popularityB)
    return genreDistance + popularityDistance

ComputeDistance(movieDict[2], movieDict[4])
```

Out[21]:
```
0.8004574042309892
```

In [22]:
```
print(movieDict[2])
print(movieDict[4])
```

```
('GoldenEye (1995)', array([0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0]), 0.22298456260720412, 3.2061068702290076)
('Get Shorty (1995)', array([0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0]), 0.3567753001715266, 3.550239234449761)
```

K Nearest Neighbors

- compute distance
- sort by distance
- print K nearest neighbor

In [23]:
```
import operator

def getNeighbors(movieID, K):
    distances = []
    for movie in movieDict:
        if (movie != movieID):
            dist = ComputeDistance(movieDict[movieID], movieDict[movie])
            distances.append((movie, dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(K):
        neighbors.append(distances[x][0])
    return neighbors

K = 10
avgRating = 0
neighbors = getNeighbors(1, K)
for neighbor in neighbors:
    avgRating += movieDict[neighbor][3]
    print (movieDict[neighbor][0] + " " + str(movieDict[neighbor][3]))

avgRating /= K
```

```
Liar Liar (1997) 3.156701030927835
Aladdin (1992) 3.8127853881278537
Willy Wonka and the Chocolate Factory (1971) 3.6319018404907975
Monty Python and the Holy Grail (1974) 4.0664556962025316
Full Monty, The (1997) 3.926984126984127
George of the Jungle (1997) 2.685185185185185
Beavis and Butt-head Do America (1996) 2.7884615384615383
Birdcage, The (1996) 3.4436860068259385
Home Alone (1990) 3.0875912408759123
Aladdin and the King of Thieves (1996) 2.8461538461538463
```

In [24]:
```
avgRating
```

Out[24]:
```
3.3445905900235564
```

Compare to its actual rating

In [25]:
```
movieDict[1]
```

Out[25]:
```
('Toy Story (1995)',
 array([0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]),
 0.7735849056603774,
 3.8783185840707963)
```

In [ ]:
```
```