Northeastern University, Khoury College of Computer Science

CS 6220 Data Mining — Assignment 2

Due: (100 points)

YOUR NAME
YOUR GIT USERNAME
YOUR E-MAIL

1 Getting Started

1.1 Using Docker

2 Identifying All Sets

2.1

In this part, we use data structure set in python to select all different items. We can read all lines or line by line and split the line, strip the line. Then we can use union operation in set to filter all duplicated items. And we use the data set basket_data.csv for testing and the corresponding result is 21.

2.2

For each item, you have two choice select or not. And for N different items, the total combination number is 2^N. When we ignore the null set and the final result is

$$2^N - 1$$

2.3

Use the function cardinality_items, we can obtain all different items. Then we can use function combinations in itertools to list all possible combinations.

2.4

The main step is to count the total number of S repeated in D.

```
## the input file is .csv,so the corresponding delimeter is ','.
def cardinality_items(file):
    with open(file,'r') as f:
        lines = f.readlines()
        col_name = set([])
        for l in lines:
            res = l.split(',')
            for i in range(len(res)):
                res[i] = res[i].strip()
            col_name = col_name | set(res)
    f.close()
    return len(col_name),col_name
print(cardinality_items('sample data/basket_data.csv')[0])
```

```
21
```

Figure 2.1: cardinality_items

```
def all_itemsets(filename):
    M,col_name = cardinality_items(filename)
    res = []
    for i in range(1,M+1):
        comb = combinations(col_name,i)
        comb = [list(val) for val in comb]
        res.append(comb)
    return res
all_itemsets('sample data/basket_data.csv')[1]
```

```
[['pork', 'leeks'],
 ['pork', 'sausages'],
 ['pork', 'tomotes'],
 ['pork', 'macaroni'],
 ['pork', 'squid'],
 ['pork', 'asparagus'],
 ['pork', 'beer'],
 ['pork', 'chips'],
 ['pork', 'butter'],
 ['pork', 'spaghetti'],
```

Figure 2.2: all_itemsets

```
]: def prob_S(S,D):
       count = 0
       for val in D:
           if set(S)==(set(val)):
               count = count + 1
       return count/len(D)
```

Figure 2.3: prob_S

3 The Netflix Challenge

3.1 Data Verification

From readme file, description of this data set are shown as below: Therefore, the following code

The file "training_set.tar" is a tar of a directory containing 17770 files, one per movie. The first line of each file contains the movie id followed by a colon. Each subsequent line in the file corresponds to a rating from a customer and its date in the following format:

CustomerID,Rating,Date

- MovieIDs range from 1 to 17770 sequentially.
- CustomerIDs range from 1 to 2649429, with gaps. There are 480189 users.
- Ratings are on a five star (integral) scale from 1 to 5.
- Dates have the format YYYY-MM-DD.

Figure 3.1: dataset description

mainly verify these three aspects. Firstly, the four training dataset will be merged. From Figure 3.2, we can see that the column is right. Then verify the columns' range. Finally, from Figure 3.3, it can be found that verification passed.

3.2 Data Analysis

3.2.1

There are total 100480507 records. From Figure 3.2, after dataset merge, examining its shape can count the number of records.

```
def data_merge(file_list):
    res = pd.DataFrame()
    for file in file_list:
        df = pd.read_csv(file,names=['user_id', 'rating', 'date'])
        res = pd.concat([res,df])
    res['movie_id']= res['user_id'].apply(lambda x:x[:-1] if x[-1]==':' else None)
    res['movie_id'] = res['movie_id'].fillna(method='ffill')
    res.dropna(subset=['date'],inplace=True)
    return res
```

```
file_list = ['combined_data_1.txt','combined_data_2.txt',\
             'combined_data_3.txt','combined_data_4.txt']
df = data_merge(file_list)
```

```
df.head()
```

	user_id	rating	date	movie_id
1	1488844	3.0	2005-09-06	1
2	822109	5.0	2005-05-13	1
3	885013	4.0	2005-10-19	1
4	30878	4.0	2005-12-26	1
5	823519	3.0	2004-05-03	1

Figure 3.2: dataset merge

```
df['user_id'].astype(int).min(),df['user_id'].astype(int).max()
```

```
(6, 2649429)
```

```
df['rating'].min(),df['rating'].max()
```

```
(1.0, 5.0)
```

```
df['movie_id'].astype(int).min(),df['movie_id'].astype(int).max()
```

```
(1, 17770)
```

Figure 3.3: range verification

```
df.shape
```

```
(100480507, 4)
```

Figure 3.4: dataset shape examination

3.2.2

Towards this problem, all customers gave ratings in the same day will be computed their mean values. From the Figure 3.5, we can see there were great change around 1999. At that time, very excellent or bad movies occurred. Or there were some controversial movies. After that, the mean rating value tend to decrease slightly. But after 2004, the mean value became to increase.

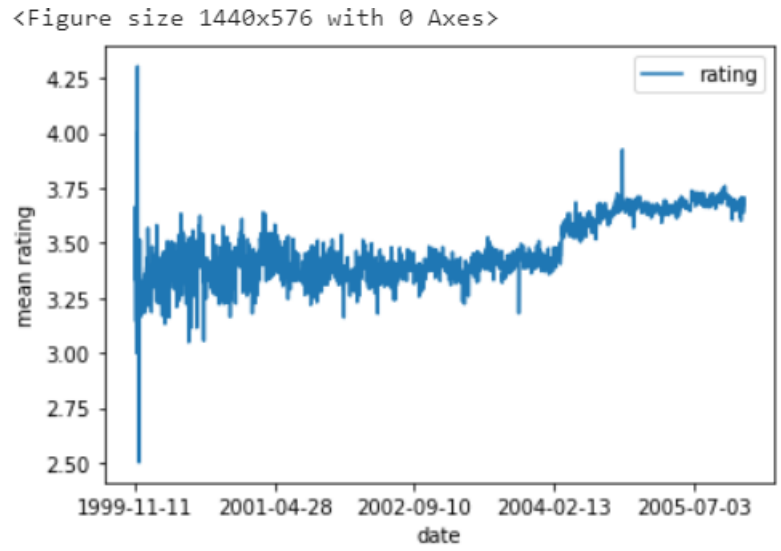

Figure 3.5: mean rating vs. date

3.2.3

Towards this problem, I will firstly give the concept of getting popular. In this experiment, I compute the mean rating stars of each year as corresponding value for that year. And then continue to compute their mean value as total mean value. Finally, using the total mean value subtract the first year rating as the improvement value. If improvement is positive, we can evaluate the movie become popular. According to the definition, the percentage of the films have gotten more popular over time is 59.76

3.2.4

For this problem, films re-released can be seen as file title occurring at least twice in movie_title.csv. To simplify this step, we use duplicate function in pandas to count the result and 473 was obtained from Figure 3.7.

```
res = 0
cnt = 0
for idx in tmp.index.levels[0]:
    rating_df = tmp.loc[idx]
    mean_rating = rating_df.mean()
    first_year_rating = rating_df.iloc[0]
    res += (mean_rating > first_year_rating).astype(int)
    cnt += 1
res/cnt
```

```
rating    0.59758
dtype: float64
```

Figure 3.6: percentage of the films become popular

```
df_movie = pd.read_csv('movie_titles.csv',encoding = "latin",header=None,names=['movie_id','year'
df_movie['name'].duplicated().sum()
```

```
473
```

Figure 3.7: percentage of the films become popular