Introduction to Data Science Report

Instructor: Than Quang Khoat

Nguyen Ngoc Lam, Phan Dinh Tuong,
Nguyen Minh Hieu, Le Xuan Thanh,
Le Bao Khanh

ICT.02-K61

Hanoi University of Science and Technology
School of Information and Communication Technology

Date: December 21, 2019

Contents

Chapter 1

Problem Statement

1.1 Description

Analyze the preference or interest film genre of user online user on MovieLens. Criteria like gender, age group, occupation and zipcode, which showed where the person live were used to group people into separate group. After that, we used summary statistics [1] on the rating of each group for each movies to recommend the genre that a lot of people on that group might like to watch.

1.2 Dataset

We will use the ml-1m for this project. According to the README file of the dataset, these files contain 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000. In the scope of this project, we will use all three files [2]. The structure of each file as followed:

1. **File** ratings.dat
   - **Line Structure:** UserID::MovieID::Rating::Timestamp
   - UserIDs range between 1 and 6040
   - MovieIDs range between 1 and 3952
   - Ratings are made on a 5-star scale (whole-star ratings only)
   - Timestamp is represented in seconds since the epoch
   - Each user has at least 20 ratings
2. **File** users.dat
   - **Line Structure:** UserID::Gender::Age::Occupation::Zip-code
   - Data was provided voluntarily by the users and was not checked for accuracy
   - Gender is denoted by a "M" for male and "F" for female
   - Age is chosen from the following ranges:
     - ∗ 1: "Under 18"
     - ∗ 18: "18-24"
     - ∗ 25: "25-34"
     - ∗ 35: "35-44"

[1] used to summarize the set observations

[2] ratings.dat, users.dat, movies.dat

  - ∗ 45: ”45-49”
  - ∗ 50: ”50-55”
  - ∗ 56: ”56+”
- Occupation is chosen from the following choices:
  - ∗ 0: ”other” or not specified
  - ∗ 1: ”academic/educator”
  - ∗ 2: ”artist”
  - ∗ 3: ”clerical/admin”
  - ∗ 4: ”college/grad student”
  - ∗ 5: ”customer service”
  - ∗ 6: ”doctor/health care”
  - ∗ 7: ”executive/managerial”
  - ∗ 8: ”farmer”
  - ∗ 9: ”homemaker”
  - ∗ 10: ”K-12 student”
  - ∗ 11: ”lawyer”
  - ∗ 12: ”programmer”
  - ∗ 13: ”retired”
  - ∗ 14: ”sales/marketing”
  - ∗ 15: ”scientist”
  - ∗ 16: ”self-employed”
  - ∗ 17: ”technician/engineer”
  - ∗ 18: ”tradesman/craftsman”
  - ∗ 19: ”unemployed”
  - ∗ 20: ”writer”

3. **File** movies.dat
   - **Line Structure:** MovieID::Title::Genres
   - Titles are identical to titles provided by the IMDB (including year of release)
   - Genres are pipe-separated and are selected from the following genres:
     - ∗ Action
     - ∗ Adventure
     - ∗ Animation

        - Children's
        - Comedy
        - Crime
        - Documentary
        - Drama
        - Fantasy
        - Film-Noir
        - Horror
        - Musical
        - Mystery
        - Romance
        - Sci-Fi
        - Thriller
        - War
        - Western
    - Some MovieIDs do not correspond to a movie due to accidental duplicate entries and/or test entries
    - Movies are mostly entered by hand, so errors and inconsistencies may exist

From the original dataset, we try to preprocess them into better structure and make it easier to work with.

1. **File** processed_movies.dat
    - **File structure:** MovieID|Title|Action|Adventure|Animation|Children's|Comedy|Crim
      Noir|Horror|Musical|Mystery| Romance|Sci-Fi|Thriller|War|Western
    - Each row from each column from Action to Western is either 1 (the film has that genre) or 0 (does not have)
2. **File** processed_users.dat
    - **File structure:** UserID|Gender|Age|Zip-code|other|academic|artist|clerical|college|cu
      care|managerial|farmer|homemaker|K-12 student|lawyer|programmer|retired|sales|scie
      employed|technician|tradesman|unemployed|writer
    - Gender has been encoded as 0 for female and 1 for male
    - Each row from each column from other to writer is either 1 or 0 represented the occupation status of that users

Chapter 2

Theoretical Background

2.1 Clustering

2.1.1 Overview

Cluster analysis or clustering is the task of grouping a set of similar objects into a single group (cluster). It is a main task of exploratory data mining, and a common technique for statistical data analysis

2.1.2 Algorithms

1. **Connectivity-based clustering (hierarchical clustering):** Connectivity-based clustering, also known as hierarchical clustering, is based on the core idea of objects being more related to nearby objects than to objects farther away. These algorithms connect "objects" to form "clusters" based on their distance. A cluster can be described largely by the maximum distance needed to connect parts of the cluster.

2. **Centroid-based clustering:** In centroid-based clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set.

3. **Distribution-based clustering:** The clustering model most closely related to statistics is based on distribution models. Clusters can then easily be defined as objects belonging most likely to the same distribution. A convenient property of this approach is that this closely resembles the way artificial data sets are generated: by sampling random objects from a distribution.

4. **Density-based clustering:** In density-based clustering, clusters are defined as areas of higher density than the remainder of the data set. Objects in these sparse areas - that are required to separate clusters - are usually considered to be noise and border points.

In this problem, we will use K-means clustering.

2.2 K-means Clustering

2.2.1 Overview

k-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

2.2.2 Pseudo Code

Algorithm 1 k-means

```
Require: $K$                                        ▷ Number of cluster
Require: $\{x_1, x_2, ..., x_n\}$                    ▷ List of observations collected
Ensure: $K < n$
procedure K-MEANS
    Place centroids $c_1, ...c_K$ at random locations
    while not convergence do  ▷ Repeat until all new centroids stay the same as last loop
        for each point $x_i$ do
            find the nearest centroid $c_j$
            assign the point $x_i$ to cluster $j$
        for for each cluster $j = 1..K$ do
            new centroid $c_j$ = mean of all points $x_i$ assigned to cluster $j$
```

2.3 Summary Statistics

2.3.1 Means

- The central value of a discrete set of numbers
- Denoted: μ
- Formular:

$$\mu = \frac{\sum allratings}{numberofratings}$$

2.3.2 Standard Deviation

- The square root of the average of the squared differences (variance)
- Denoted: σ
- Formular:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(x_i - \mu)^2}$$

Chapter 3

Our Implementation

We divided the problem into four (4) sub tasks, namely: data preprocessing, elbow method and k-means, data visualization (both user data and movie genre) and statistical approach (for catch genre trend of each user cluster).

3.1 Data Preprocessing

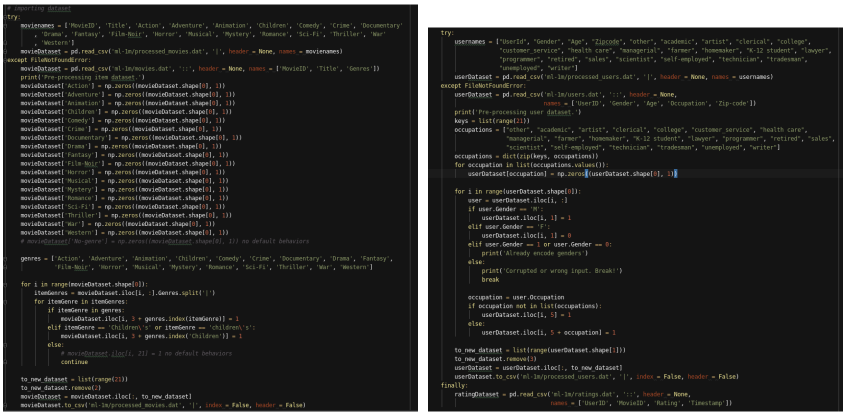
3.1.1 File movies.dat

- Treating data from genre column as catergorical data. Creating dummy variables for occupation column.
- Output: **movies:** a ndarray contains values of processed *movieDataset*

3.1.2 File users.dat

- Encoding the gender columns from 'M' and 'F' into 1 and 0 respectively
- Treating data from occupation column as catergorical data. Creating dummy variables for occupation column.
- If cannot find processed_users.dat, write out processed dataset to file, else load that file into *userDataset* variable.
- Using OneHotEncoder from sklearn.preprocessing to encode the zip-code columns (since this column's data is also catergorical data)
- Output: **users:** a ndarray contains values of processed *userDataset* plus already encoded zip-code

3.1.3 Actual Implementation



```
# importing dataset
try:
    movienames = ['MovieID', 'Title', 'Action', 'Adventure', 'Animation', 'Children', 'Comedy', 'Crime', 'Documentary'
        , 'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War'
        , 'Western']
    movieDataset = pd.read_csv('ml-1m/processed_movies.dat', '|', header = None, names = movienames)
except FileNotFoundError:
    movieDataset = pd.read_csv('ml-1m/movies.dat', '::', header = None, names = ['MovieID', 'Title', 'Genres'])
    print('Pre-processing item dataset.')
    movieDataset['Action'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Adventure'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Animation'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Children'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Comedy'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Crime'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Documentary'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Drama'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Fantasy'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Film-Noir'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Horror'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Musical'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Mystery'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Romance'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Sci-Fi'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Thriller'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['War'] = np.zeros((movieDataset.shape[0], 1))
    movieDataset['Western'] = np.zeros((movieDataset.shape[0], 1))
    # movieDataset['No-genre'] = np.zeros((movieDataset.shape[0], 1)) no default behaviors

    genres = ['Action', 'Adventure', 'Animation', 'Children', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
              'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']

    for i in range(movieDataset.shape[0]):
        itemGenres = movieDataset.iloc[i, :].Genres.split('|')
        for itemGenre in itemGenres:
            if itemGenre in genres:
                movieDataset.iloc[i, 3 + genres.index(itemGenre)] = 1
            elif itemGenre == 'Children\'s' or itemGenre == 'children\'s':
                movieDataset.iloc[i, 3 + genres.index('Children')] = 1
            else:
                # movieDataset.iloc[i, 21] = 1 no default behaviors
                continue

    to_new_dataset = list(range(21))
    to_new_dataset.remove(2)
    movieDataset = movieDataset.iloc[:, to_new_dataset]
    movieDataset.to_csv('ml-1m/processed_movies.dat', '|', index = False, header = False)
```

(a) Movies preprocessing

```
try:
    usernames = ["UserId", "Gender", "Age", "Zipcode", "other", "academic", "artist", "clerical", "college",
                 "customer_service", "health care", "managerial", "farmer", "homemaker", "K-12 student", "lawyer",
                 "programmer", "retired", "sales", "scientist", "self-employed", "technician", "tradesman",
                 "unemployed", "writer"]
    userDataset = pd.read_csv('ml-1m/processed_users.dat', '|', header = None, names = usernames)
except FileNotFoundError:
    userDataset = pd.read_csv('ml-1m/users.dat', '::', header = None,
                              names = ['UserID', 'Gender', 'Age', 'Occupation', 'Zip-code'])
    print('Pre-processing user dataset.')
    keys = list(range(21))
    occupations = ["other", "academic", "artist", "clerical", "college", "customer_service", "health care",
                   "managerial", "farmer", "homemaker", "K-12 student", "lawyer", "programmer", "retired", "sales",
                   "scientist", "self-employed", "technician", "tradesman", "unemployed", "writer"]
    occupations = dict(zip(keys, occupations))
    for occupation in list(occupations.values()):
        userDataset[occupation] = np.zeros((userDataset.shape[0], 1))

    for i in range(userDataset.shape[0]):
        user = userDataset.iloc[i, :]
        if user.Gender == 'M':
            userDataset.iloc[i, 1] = 1
        elif user.Gender == 'F':
            userDataset.iloc[i, 1] = 0
        elif user.Gender == 1 or user.Gender == 0:
            print('Already encode genders')
        else:
            print('Corrupted or wrong input. Break!')
            break

        occupation = user.Occupation
        if occupation not in list(occupations):
            userDataset.iloc[i, 5] = 1
        else:
            userDataset.iloc[i, 5 + occupation] = 1

    to_new_dataset = list(range(userDataset.shape[1]))
    to_new_dataset.remove(3)
    userDataset = userDataset.iloc[:, to_new_dataset]
    userDataset.to_csv('ml-1m/processed_users.dat', '|', index = False, header = False)
finally:
    ratingDataset = pd.read_csv('ml-1m/ratings.dat', '::', header = None,
                                names = ['UserID', 'MovieID', 'Rating', 'Timestamp'])
```

(b) Users preprocessing

Figure 3.1: Actual implementation of data preprocessing

3.2 Elbow Method and k-means

3.2.1 Elbow Method

- Used in determine the number of clusters for this particular problems.
- Pseudo code:

Algorithm 2 Elbow method

```
Require: $n$                                ▷ number of maximum clusters (n = 10)
procedure ELBOWMETHOD
    $i \leftarrow 0$
    $wcss \leftarrow []$
    for $i = 0$; $i < n$; $i++$ do
        $model \leftarrow Kmean(\text{n-clusters} = i)$
        $wcss[i] \leftarrow wcss(model)$
    Draw graph of $wcss$
    $K \leftarrow k$ with $k$ is the index of the point in array $wcss$ at which $wcss$ go for steep declined (greatly improve) to steady (small improve)
    return K
```

- Result: choose k = 3

3.2.2 k-means

We used the implementation of k-means [1] from sklearn.cluster with parameters:

- n_clusters = 3
- init = k-means++
- n_init = 20
- max_iter = 500
- random_state = 0
- everything else are set to default parameter

3.3 Data Visualization

Data Visualization use matplotlib.pyplot library to draw graph

3.4 Statistical Approach

After we get the groups of users, for each group, consider each movie, get the number of user rated that movies, mean score, standard deviation of that movies. We will choose the movies with more than *10 percent* users has rated, mean no smaller than *4*, standard deviation no more than *0.8*.

[1] You can see documentation at k-means

Chapter 4

Result

4.1 Data Preprocessing

Output into 2 files processed_movies.dat and processed_users.dat

4.2 Elbow Method

After calculate WCSS we get:

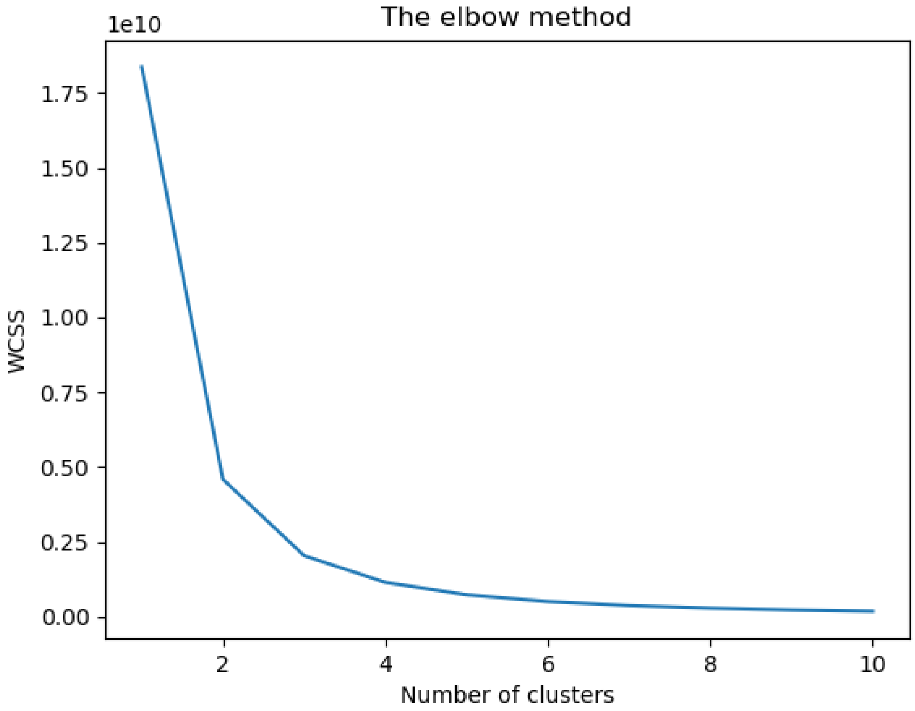
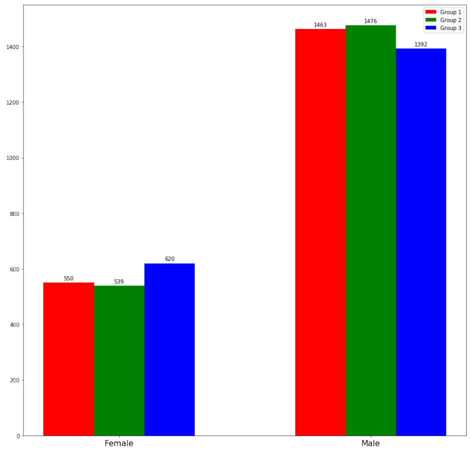
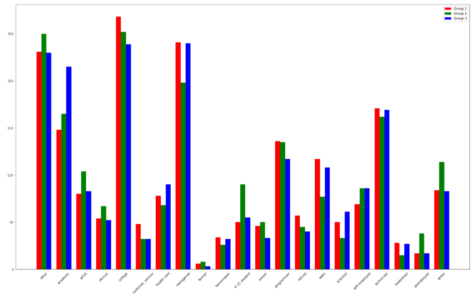


Figure 4.1: Elbow method

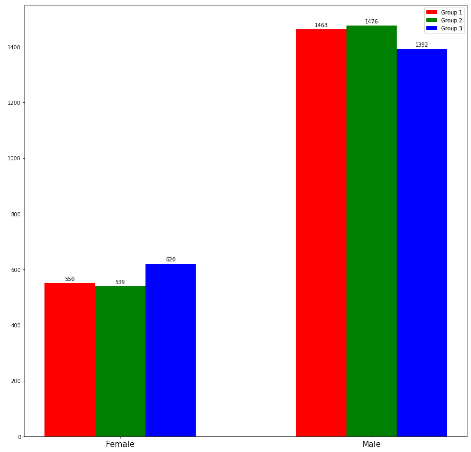
4.3 Data Visualization



(a) Representation of gender on each group



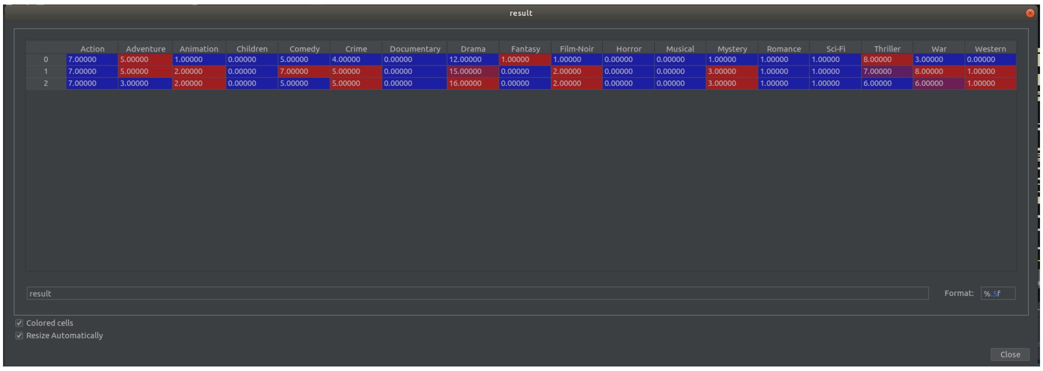
(b) Representation of occupation on each group



(c) Representation of gender on each group

4.4 Final Result

Ouput into result.dat contains number of movies rated highly for each genre on each group of user



	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	7.00000	5.00000	1.00000	0.00000	5.00000	4.00000	0.00000	12.00000	1.00000	1.00000	0.00000	0.00000	1.00000	1.00000	1.00000	8.00000	3.00000	0.00000
1	7.00000	5.00000	2.00000	0.00000	7.00000	5.00000	0.00000	15.00000	0.00000	2.00000	0.00000	0.00000	3.00000	1.00000	1.00000	7.00000	8.00000	1.00000
2	7.00000	3.00000	2.00000	0.00000	5.00000	5.00000	0.00000	16.00000	0.00000	2.00000	0.00000	0.00000	3.00000	1.00000	1.00000	6.00000	6.00000	1.00000

result

Format: %.5f

☑ Colored cells

☑ Resize Automatically

Figure 4.3: Representation of genre for each group

Chapter 5

Problems, Limitations and Future Development

5.1 Problems

- In the beginning, we want to get real user data from youtube to train. But after google updated user agreement, we cannot get the dataset freely anymore. Because of that, we used the ml-1m. The data is not guaranteed to be corrected and it is most likely to be synthetic data.
- Due to the limited time of the course, we cannot researched on all aspects of clustering and recommendation system. This project is a brief understanding of the problem for learning purpose.

5.2 Limitations

- Our work based on the rating of users so we will face the cold start problem.
- On cold start problem, if a new item added to the database and had genre already, we can recommend using the genre
- else if it had not have genre yet or wrong genre, we cannot recommend the right movie to group user.
- If a new user registered to the system, we would have to fit the clustering algorithm again to assign that user to the correct cluster (possibly have to run the elbow method again to re-select the number of clusters)

5.3 Future Development

- Implement a content-based algorithm
- Improve this algorithm to better match the preference of user.