Case Study 4.1 - Movies

Note: If you close this notebook at any time, you will have to run all cells again upon re-opening it.

BEGINNER R

As this is a beginner version, we include a lot of code here to help you along the way.

Identification Information

In [12]:

```
# YOUR NAME               = Ramesh Neupane
# YOUR MITX PRO USERNAME  = ramesh2018
# YOUR MITX PRO E-MAIL    = rneupane@dallasisd.org (which is miktakely noted as
"rneupane@dalllasisd.org")
```

Setup

Run these cells to install all the packages you need to complete the remainder of the case study. This may take a few minutes, so please be patient.

If you see red messages, don't worry! See this FAQ.

In [1]:

```
install.packages('recommenderlab')
print('Installation successful!')
```

```
Installing package into ‘/home/nbuser/R’
(as ‘lib’ is unspecified)
```

```
[1] "Installation successful!"
```

Import

Import the required tools into the notebook.

In [2]:

```
library('recommenderlab')
print('Import successful!')
```

```
Loading required package: Matrix
Loading required package: arules

Attaching package: ‘arules’

The following objects are masked from ‘package:base’:

    abbreviate, write

Loading required package: proxy

Attaching package: ‘proxy’
```

```
The following object is masked from ‘package:Matrix’:

    as.matrix

The following objects are masked from ‘package:stats’:

    as.dist, dist

The following object is masked from ‘package:base’:

    as.matrix

Loading required package: registry
```

```
[1] "Import successful!"
```

Data

Load the MovieLens data.

In [3]:

```
data = read.table('u.data.txt')
colnames(data) = c("user_id", "item_id", "rating", "timestamp")
data = data[ , -which(names(data) %in% c("timestamp"))]
print('Data loading successful!')
```

```
[1] "Data loading successful!"
```

We also want to get a sense of what the data looks like. Let's create a histogram of all the ratings we have in the dataset.

In [4]:

```
hist(data$rating)
print('Histogram generation successful!')
```

```
[1] "Histogram generation successful!"
```

We must now subset the data into train and test sets to use for our different models. Here, we split the entire dataset into 70% train and 30% test.

In [5]:

```
# First, we convert our data type to a recommenderlab compatible type
data_new = as(data, "realRatingMatrix")

# Now, we apply a split of 0.7 train, 0.3 test
# You can ignore the given parameter
data_split = evaluationScheme(data_new, method="split", train=0.7, goodRating=3, given=-1)
train = getData(data_split, "train")
test_X = getData(data_split, "known")
test_Y = getData(data_split, "unknown")

print('Data subset successful!')
```

```
[1] "Data subset successful!"
```

QUESTION 1: DATA ANALYSIS

Describe the dataset. How many ratings are in the dataset? How would you describe the distribution of ratings? Is there anything else we should observe? Make sure the histogram is visible in the notebook.

There are 100000 ratings in the data set. The data is right skiewed. Because almost 83% (82.5% to exact)of people rated 3 or more ratings. Only around 17000 people (17%) rated either 1 or 2.The largest number of people rated 4 which consist almost 34% of the data. From this distribution, we could say that most of the people look happy with the movie that had watched.

Model 1: Random

In [6]:

```
# Create model object on our training set
model_random = Recommender(train, method="RANDOM")
print('Model creation successful!')
```

```
[1] "Model creation successful!"
```

In [7]:

```
# Evaluate RMSE on test set
predicted_random = predict(model_random, test_X, type="ratings")
error_random = calcPredictionAccuracy(predicted_random, test_Y)
error_random
```

RMSE
1.42851749566273
MSE
2.04066223541452
MAE
1.1296271294781

Model 2: User-Based Collaborative Filtering

In [8]:

```
# Create model object on our training set
model_user = Recommender(train, method="UBCF")
print('Model creation successful!')
```

```
[1] "Model creation successful!"
```

In [9]:

```
# Evaluate RMSE on test set
predicted_user = predict(model_user, test_X, type="ratings")
error_user = calcPredictionAccuracy(predicted_user, test_Y)
error_user
```

RMSE
1.02115489946866
MSE
1.04275732870885
MAE
0.823394193314014

Model 3: Item-Based Collaborative Filtering

In [ ]:

```
# Create model object on our training set
# Note that this may take a while to train
model_item = Recommender(train, method="IBCF")
print('Model creation successful!')
```

In [ ]:

```
# Evaluate RMSE on test set
predicted_item = predict(model_item, test_X, type="ratings")
error_item = calcPredictionAccuracy(predicted_item, test_Y)
error_item
```

QUESTION 2: COLLABORATIVE FILTERING MODELS

Compare the results from the user-user and item-item models. How do they compare to each other? How do they compare to our original "random" model? Can you provide any intuition as to why the results came out the way they did?

*The result from three algorithms shows that the least RMSE is from user-user based collaborative filtering(1.02) which is smaller RMSE then item based collaborative filtering and ramdom algorithm. The main idea here is less the root square mean error better the the performance.

Model 4: Matrix Factorization

In [ ]:

```
# Create model object on our training set
model_matrix = Recommender(train, method="SVD")
print('Model creation successful!')
```

In [ ]:

```
# Evaluate RMSE on test set
predicted_matrix = predict(model_matrix, test_X, type="ratings")
error_matrix = calcPredictionAccuracy(predicted_matrix, test_Y)
error_matrix
```

QUESTION 3: MATRIX FACTORIZATION MODEL

The matrix factorization model is different from the collaborative filtering models. Briefly describe this difference. Also, compare the RMSE again. Does it improve? Can you offer any reasoning as to why that might be?

Metirx factorization model supposed be best for sparce data set. However, in both movie and songs datasets, Metrix factorization did not work bettern than collaborative filtering as it did not have less RMSE then collaborative filtering. There is no one silver bullet to solve problems. We need to run different alrorithms and compaire which predicts the best for perticular situation and choose best performer to solve that datasets.

Type your response here...

Precision and Recall @ `k`

We now want to compute the precision and recall for 2 values of `k`: 5 and 10. The `recommenderlab` library makes this super easy!

Note that some of these values may take some time (a few minutes) to compute.

QUESTION 4: PRECISION/RECALL

Compute the precision and recall, for each of the 4 models, at `k` = 5 and 10. This is 2 x 2 x 4 = 16 numerical values. Do you note anything interesting about these values? Anything differerent from the RMSE values you computed above?

By comaring its RMSE, item-item basex collaborative filtering performs better then other.

In [ ]:

```
results_random = evaluate(data_split, method="RANDOM", type="topNList", n=c(5, 10))
results_random@results
```

In [ ]:

```
results_user = evaluate(data_split, method="UBCF", type="topNList", n=c(5, 10))
results_user@results
```

In [ ]:

```
results_item = evaluate(data_split, method="IBCF", type="topNList", n=c(5, 10))
results_item@results
```

In [ ]:

```
results_matrix = evaluate(data_split, method="SVD", type="topNList", n=c(5, 10))
results_matrix@results
```

Top-n Predictions

Finally, we can see what some of the actual movie ratings are for particular users, as outputs of our model.

QUESTION 5: TOP N PREDICTIONS

Do the top n predictions that you received make sense? What is the rating value (1-5) of these predictions? How could you use these predictions in the real-world if you were trying to build a generic content recommender system for a company?

Type your response here...

In [ ]:

```
# Pick a user and value of n
user_id = 5
n = 10
print('Variable update successful!')
```

In [ ]:

```
# Repeat this process for the all models
for (model in c(model_random, model_user, model_item, model_matrix)) {
    # Get recs for model
    user_recs = predict(model, test_X, n=n, type="ratings")

    # Print the top n recs for that user
    print(model)
    print(user_recs@data[user_id,][order(-user_recs@data[user_id,])][c(1:n)])
}

print('Top N predictions successful!')
```

Great job! Now, make sure you check out the **Conclusion** section of the instruction manual to wrap up this case study properly.