PERIYAR MANIAMMAI INSTITUTE OF SCIENCE & TECHNOLOGY

PERIYAR MANIAMMAI
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University)
Established Under Sec. 3 of UGC Act, 1956 • NAAC Accredited
think • innovate • transform

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MINI PROJECT REPORT

By: *M.SABAREESWARAN- 121012012759*

B.TECH-CSE (2 ND YEAR)

TOPIC : Movie Recommendation System

TEAM MEMBERS :

- **V.KAMALESAKUMAR – 121012012740**
- **S.SATHISH – 121012012764**
- **M.SABAREESWARAN – 121012012759**

How to build a Movie Recommendation System using Machine Learning?

Dataset

In order to build our recommendation system, we have used the MovieLens Dataset. You can find the movies.csv and ratings.csv file that we have used in our Recommendation System Project . This data consists of 105339 ratings applied over 10329 movies**.**

Importing Essential Libraries

In our Data Science project, we will make use of these four packages – 'recommenderlab', 'ggplot2', 'data.table' and 'reshape2'.

Retrieving the Data

We will now retrieve our data from movies.csv into movie_data dataframe and ratings.csv into rating_data. We will use the str() function to display information about the movie_data dataframe.

Data Pre-processing

From the above table, we observe that the userId column, as well as the movieId column, consist of integers. Furthermore, we need to convert the genres present in the movie_data dataframe into a more usable format by the users. In order to do so, we will first create a one-hot encoding to create a matrix that comprises of corresponding genres for each of the films.

Exploring Similar

Data Collaborative Filtering involves suggesting movies to the users that are based on collecting preferences from many other users. For example, if a user A likes to watch action films and so does user B, then the movies that the user B will watch

in the future will be recommended to A and viceversa. Therefore, recommending movies is dependent on creating a relationship of similarity between the two users. With the help of recommenderlab, we can compute similarities using various operators like cosine, pearson as well as jaccard.

Most Viewed Movies Visualization

In this section of the machine learning project, we will explore the most viewed movies in our dataset. We will first count the number of views in a film and then organize them in a table that would group them in descending order.

Performing Data Preparation

We will conduct data preparation in the following three steps – • Selecting useful data. • Normalizing data. • Binarizing the data. For finding useful data in our dataset, we have set the threshold for the minimum number of users who have rated a film as 50. This is also same for minimum number of views that are per film. This way, we have filtered a list of watched films from least-watched ones.

Data Normalization

In the case of some users, there can be high ratings or low ratings provided to all of the watched films. This will act as a bias while implementing our model. In order to remove this, we normalize our data. Normalization is a data preparation procedure to standardize the numerical values in a column to a common scale value. This is done in such a way that there is no distortion in the range of values. Normalization transforms the average value of our ratings column to 0. We then plot a heatmap that delineates our normalized ratings

Collaborative Filtering System

In this section of data science project, we will develop our very own Item Based Collaborative Filtering System. This type of collaborative filtering finds similarity in the items based on the people's ratings of them. The algorithm first builds a similar-items table of the customers who have purchased them into a combination of similar items. This is then fed into the recommendation system. The similarity between single products and related products can be determined with the following algorithm – • For each Item i1 present in the product catalog, purchased by customer C. • And, for each item i2 also purchased by the customer C. • Create record that the customer purchased items i1 and i2. • Calculate the similarity between i1 and i2.

Building the Recommendation System using R

We will now explore the various parameters of our Item Based Collaborative Filter. These parameters are default in nature. In the first step, k denotes the number of items for computing their similarities. Here, k is equal to 30. Therefore, the algorithm will now identify the k most similar items and store their number. We use the cosine method which is the default one but you can also use pearson method.

CODE:

```
library(recommenderlab)
library(ggplot2)
library(data.table)
library(reshape2)
```

```
#2
setwd("D:\\openhouse\\MY Project")
movie_data <- read.csv("movies.csv",stringsAsFactors=FALSE)
rating_data <- read.csv("ratings.csv")
str(movie_data)

#3
summary(movie_data)
head(movie_data)

#4
summary(rating_data)
head(rating_data)

#5
movie_genre <- as.data.frame(movie_data$genres, stringsAsFactors=FALSE)
library(data.table)
movie_genre2 <- as.data.frame(tstrsplit(movie_genre[,1], '[|]',
                                type.convert=TRUE),
                        stringsAsFactors=FALSE) #DataFlair
colnames(movie_genre2) <- c(1:10)

list_genre <- c("Action", "Adventure", "Animation", "Children",
          "Comedy", "Crime","Documentary", "Drama", "Fantasy",
          "Film-Noir", "Horror", "Musical", "Mystery","Romance",
          "Sci-Fi", "Thriller", "War", "Western")
genre_mat1 <- matrix(0,10330,18)
genre_mat1[1,] <- list_genre
colnames(genre_mat1) <- list_genre
```

```
for (index in 1:nrow(movie_genre2)) {
  for (col in 1:ncol(movie_genre2)) {
    gen_col = which(genre_mat1[1,] == movie_genre2[index,col]) #Author DataFlair
    genre_mat1[index+1,gen_col] <- 1
  }
}
genre_mat2 <- as.data.frame(genre_mat1[-1,], stringsAsFactors=FALSE) #remove first
row, which was the genre list
for (col in 1:ncol(genre_mat2)) {
  genre_mat2[,col] <- as.integer(genre_mat2[,col]) #convert from characters to integers
}
str(genre_mat2)

#6
SearchMatrix <- cbind(movie_data[,1:2], genre_mat2[])
head(SearchMatrix)

#7
library(reshape2)
library(arules)
library(recommenderlab)
ratingMatrix <- dcast(rating_data, userId~movieId, value.var = "rating", na.rm=FALSE)
ratingMatrix <- as.matrix(ratingMatrix[,-1]) #remove userIds
#Convert rating matrix into a recommenderlab sparse matrix
ratingMatrix <- as(ratingMatrix,"realRatingMatrix")
ratingMatrix
#8
recommendation_model <- recommenderRegistry$get_entries(dataType =
"realRatingMatrix")
names(recommendation_model)
```

```
#9
lapply(recommendation_model, "[[", "description")

#10
recommendation_model$IBCF_realRatingMatrix$parameters

#11
similarity_mat <- similarity(ratingMatrix[1:4, ],
                             method = "cosine",
                             which = "users")
as.matrix(similarity_mat)

#12
image(as.matrix(similarity_mat), main = "User's Similarities")

#13
movie_similarity <- similarity(ratingMatrix[, 1:4], method =
                               "cosine", which = "items")
as.matrix(movie_similarity)

#14
image(as.matrix(movie_similarity), main = "Movies similarity")

#15
rating_values <- as.vector(ratingMatrix@data)
unique(rating_values) # extracting unique ratings

#16
Table_of_Ratings <- table(rating_values) # creating a count of movie ratings
```

Table_of_Ratings

```
#17
library(ggplot2)
movie_views <- colCounts(ratingMatrix) # count views for each movie
table_views <- data.frame(movie = names(movie_views),
                          views = movie_views) # create dataframe of views
table_views <- table_views[order(table_views$views,
                                 decreasing = TRUE), ] # sort by number of views
table_views$title <- NA
for (index in 1:10325){
  table_views[index,3] <- as.character(subset(movie_data,
                                              movie_data$movieId == table_views[index,1])$title)
}
table_views[1:6,]



#18
ggplot(table_views[1:6, ], aes(x = title, y = views)) +
  geom_bar(stat="identity", fill = 'steelblue') +
  geom_text(aes(label=views), vjust=-0.3, size=3.5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +

  ggtitle("Total Views of the Top Films")



#19
image(ratingMatrix[1:20, 1:25], axes = FALSE, main = "Heatmap of the first 25 rows
and 25 columns")
```

```
#20
movie_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,
                              colCounts(ratingMatrix) > 50]
movie_ratings

#21
minimum_movies<- quantile(rowCounts(movie_ratings), 0.98)
minimum_users <- quantile(colCounts(movie_ratings), 0.98)
image(movie_ratings[rowCounts(movie_ratings) > minimum_movies,
                    colCounts(movie_ratings) > minimum_users],
    main = "Heatmap of the top users and movies")

#22
average_ratings <- rowMeans(movie_ratings)
qplot(average_ratings, fill=I("steelblue"), col=I("red")) +
 ggtitle("Distribution of the average rating per user")

#23
normalized_ratings <- normalize(movie_ratings)
sum(rowMeans(normalized_ratings) > 0.00001)

#24
image(normalized_ratings[rowCounts(normalized_ratings) > minimum_movies,
                         colCounts(normalized_ratings) > minimum_users],
    main = "Normalized Ratings of the Top Users")

#25
binary_minimum_movies <- quantile(rowCounts(movie_ratings), 0.95)
binary_minimum_users <- quantile(colCounts(movie_ratings), 0.95)
```

```
#movies_watched <- binarize(movie_ratings, minRating = 1)

good_rated_films <- binarize(movie_ratings, minRating = 3)
image(good_rated_films[rowCounts(movie_ratings) > binary_minimum_movies,
                       colCounts(movie_ratings) > binary_minimum_users],
    main = "Heatmap of the top users and movies")

#26
sampled_data<- sample(x = c(TRUE, FALSE),
                      size = nrow(movie_ratings),
                      replace = TRUE,
                      prob = c(0.8, 0.2))
training_data <- movie_ratings[sampled_data, ]
testing_data <- movie_ratings[!sampled_data, ]

#27
recommendation_system <- recommenderRegistry$get_entries(dataType
="realRatingMatrix")
recommendation_system$IBCF_realRatingMatrix$parameters

#28
recommen_model <- Recommender(data = training_data,
                              method = "IBCF",
                              parameter = list(k = 30))
recommen_model

#29
class(recommen_model)

#30
```

```
model_info <- getModel(recommen_model)
class(model_info$sim)
dim(model_info$sim)
top_items <- 20
image(model_info$sim[1:top_items, 1:top_items],
    main = "Heatmap of the first rows and columns")

#31
sum_rows <- rowSums(model_info$sim > 0)
table(sum_rows)

sum_cols <- colSums(model_info$sim > 0)
qplot(sum_cols, fill=I("steelblue"), col=I("red"))+ ggtitle("Distribution of the column
count")

#32
top_recommendations <- 10 # the number of items to recommend to each user
predicted_recommendations <- predict(object = recommen_model,
                        newdata = testing_data,
                        n = top_recommendations)
predicted_recommendations

#33
user1 <- predicted_recommendations@items[[1]] # recommendation for the first user
movies_user1 <- predicted_recommendations@itemLabels[user1]
movies_user2 <- movies_user1
for (index in 1:10){
  movies_user2[index] <- as.character(subset(movie_data,
                           movie_data$movieId == movies_user1[index])$title)
}
```

```
movies_user2

#34
recommendation_matrix <- sapply(predicted_recommendations@items,
                                function(x){ as.integer(colnames(movie_ratings)[x]) }) # matrix with
the recommendations for each user
#dim(recc_matrix)
recommendation_matrix[,1:4]
```

Summary

Recommendation Systems are the most popular type of machine learning applications that are used in all sectors. They are an improvement over the traditional classification algorithms as they can take many classes of input and provide similarity ranking based algorithms to provide the user with accurate results. These recommendation systems have evolved over time and have incorporated many advanced machine learning techniques to provide the users with the content that they want.