# seventhlab

2025-11-13

```
#Load Dataset and keep only users which have greater than 5 ratings
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
X=read.csv('MovieLens.csv')
count_users=count(X,User)
keep_users=count_users[count_users$n>=5,]
X=filter(X,User %in% keep_users$User)# %in% is used to check set membership
```

```
#Form train and test data
Xm=mutate(X,timestamp2=as.POSIXct(X$timestamp,origin="1970-01-01",tz="UTC"))
Xm_group=group_by(Xm,User)
Xm_increase=arrange(Xm_group,Xm_group$timestamp2,by_group=TRUE)
nr=nrow(count_users)
train_list=list()
test_list=list()
#i=1
for(i in 1:nr){
  userid=keep_users[i,1]
  usercount=keep_users[i,2]
  temp=Xm_increase[Xm_increase$User==userid,]
  train_list[[i]]=temp[1:usercount-1,]
  test_list[[i]]=temp[usercount,]
}
trainset=bind_rows(train_list)
testset=bind_rows(test_list)
m=data.frame(sort(unique(trainset$item)))
```

```
#EDA:Useful in A/B Recommendar Testing
#EDA1:
summary(trainset$rating)#To ensure dataset is not biased
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   4.000   3.531   4.000   5.000
```

```
#EDA2:
keep_user2=arrange(keep_users,keep_users$n)#To identify most active users
```

```
#EDA3:
movie_count=count(X,item)
movie_count=arrange(movie_count,desc(movie_count$n))#To identify popular movies
```

```
#Model A
movie_count=arrange(movie_count,movie_count$item)
nr=nrow(movie_count)
C=20
bayes_score=numeric(nr)
for(j in 1:nr){
bayes_score[j]=(C*mean(trainset$rating)+movie_count[j,2]*mean(trainset[trainset$item==movi
e_count[j,1],]$rating))/(C+movie_count[j,2])
}
movie_count2=cbind(movie_count,bayes_score)
movie_count2d=arrange(movie_count2,desc(movie_count2$bayes_score))#Max Bayes score at the
top
```

```
#Model B
library(reshape2)
ui_mat=acast(trainset,User~item,value.var = "rating")#To get user item matrix
#Normalize Rows (Subtract the user mean rating)
rmeans=rowMeans(ui_mat,na.rm = TRUE)
ui_norm=ui_mat
for(i in 1:nrow(ui_mat)){
  ui_norm[i,]=ui_norm[i,]-rmeans[i]
}
ui_norm[is.na(ui_norm)]=0
#Evaluate cosine similarity between items
sim=cor(ui_norm)
```

```
calculaterecommendationB=function(u){
  #sim: similarity matrix
  #ui_norm: user item matrix
  #Returns a Dataframe containing recommendation and rank using bayes score
  r=data.frame(matrix(0,nrow=10,ncol=2))
  r1=data.frame(matrix(0,nrow=nrow(sim),ncol=2))
  sm=trainset[trainset$User==u,2]#Movies that user has rated
  ind=numeric(length(sm))
  for(i in 1:nrow(m)){
    if(m[i,1] %in% sm){
      ind[i]=i
    }
  }
  scores=rowSums(sim[,ind])#Sum of row scores(size(sm) x 1679)

  scores[ind]=-Inf#Removing already rated movies from consideration
  r1[,1]=scores
  r1[,2]=m
  r2 <- arrange(r1,desc(r1[,1]))
  h=1
  g=1
  while(g<=10){
    if(r2[h,2] %in% sm){
      h=h+1
    }else{
      r[g,1]=r2[h,2]
      r[g,2]=g
      g=g+1
      h=h+1
    }
  }
  return(r)
}

calculaterecommendationA=function(u){
  #Returns a Dataframe containing recommendation and rank using similarity score
  r=data.frame(matrix(0,nrow=10,ncol=2))
  h=1
  g=1
  while(g<=10){
    if(movie_count2d[h,1] %in% trainset[trainset$User==u,2]){
      h=h+1
    }else{
      r[g,1]=movie_count2d[h,1]
      r[g,2]=g
      g=g+1
      h=h+1
    }
  }
  return(r)
}
```

```
#Make predictions on test set and identify metrics
#HitRate@10
n=nrow(testset)
hitA=numeric(n)
hitB=numeric(n)
K=10
for(i in 1:n){
  ui=testset[i,1]#user index
  tm=testset[i,2]#target movie
  nA=calculaterecommendationA(ui)#Data frame consisting of recommendation and rank
  nB=calculaterecommendationB(ui)#Data frame consisting of recommendation and rank
  if(tm %in% nA[,1]){
    hitA[i]=1
  }
  if(tm %in% nB[,1]){
    hitB[i]=1
  }
}
```

```
hitA=data.frame(hitA)
hitB=data.frame(hitB)
dp0=0
alpha=0.05
size=nrow(testset)
#form 2 sets
s1=hitA
s2=hitB
sam1_mean=mean(s1[,1])
sam2_mean=mean(s2[,1])
p=mean(c(s1[,1],s2[,1]))#Combined Mean
sam_sd=sqrt(p*(1-p)*(2/size))
z=(sam1_mean-sam2_mean)/sam_sd
p_value=2*pnorm(abs(z),mean=0,sd=1,lower.tail = FALSE)
print(p_value)
```

```
## [1] 0.5008697
```

```
if(p_value<alpha){
  sprintf("Reject Null Hypothesis that p0=p1")
}else{
  sprintf("Accept Null Hypothesis that p0=p1")
}
```

```
## [1] "Accept Null Hypothesis that p0=p1"
```