**My Approach** for *HackerRank : Challenge Recommendation* -   
  
Important aspects of dataset -   
Challenges.csv :  
Rows – 2865  
Unique challenge Ids – 2228  
Unique Contest Ids – 256  
Unique domain – 17  
Unique sub domain – 98  
[Same challenge can belong to more than one contests.]  
  
Submissions.csv:  
Rows – 2,86,064  
Unique hacker Ids – 10,000  
Unique contest Ids – 254  
Unique challenge Ids – 2156  
Unique Languages - 47

My approach is an Item based collaborative filtering.  
Talking simply, recommending the similar challenges to the hacker based on the challenges he has already been a part of. This similarity is calculated based on the features corresponding to every challenge. More these features match, more they are similar challenges and they are more likely to be liked by the hacker.  
  
To get the similarity, I used the cosine function and it results in a square matrix of size 2228.  
On the other hand, I converted the submissions.csv such that instead of the original columns now it has the columns corresponding to the 2156 challenges and every time a hacker attempts the challenge the row corresponding to the hacker gets filled with number of times the challenge was attempted by the hacker. Thus it has 10,000 rows instead of 2,86,064 rows which were corresponding to each submission originally in the dataset.  
Now it was difficult to convert the submissions.csv in this format. This is how I achieved it -   
I listed all the unique hacker Ids in one column, every time a challenge which a hacker attempted is seen a new column corresponding to that challenge is created and is filled with value 1, and if the challenge column already exists, then no need of creating a new column but just fill the value as 1. Thus if any hacker doesn’t attempt any challenge the column corresponding it will be NA. Now that we know which challenges are attempted by the hacker as they are filled with 1 ,still there are 2,86,064 rows as every hacker has done many submissions. We want the total number of submissions hacker has done for a particular challenge. To achieve this, I grouped the similar hackers and their corresponding challenge columns were the sum of the total submissions for that challenge.

This was called as ratings!  
I am not using all the 2156 challenges, rather for fast processing only 445 popular challenges as these are those challenges which were attempted more than mean of total submissions.  
  
Now we have two data sets ready, namely – Item\_sim (Challenge similarity matrix) and ratings.  
  
Now we wish to predict the submissions by a hacker for an un-attempted challenge.  
To do this , a different function is made ‘’**rec\_itm\_for\_user(hacker)’’ 🡪** please check **recommender.R file**  
For every hacker, we create two lists : Attempted challenges and Un-attempted challenges.  
If the ratings matrix has NA then include the challenge in the Un-attempted challenges list else in the Attempted challenges list.  
Now using the item\_sim, get the similarity of un-attempted challenge and all the other challenges.  
Now get a weighted matrix which is a product of ratings and the chosen item\_sim challenge row.  
Using the weighted matrix, calculate the prediction score for all the challenges. This is done by adding the rows of weighted matrix and dividing it by the sum of item\_sim corresponding to the attempted challenges.  
  
for(i in 1:length(attempted\_challenges)){

temp\_sum = temp\_sum+ df[which(colnames(df)==attempted\_challenges[i])]

}

# Multiply the ratings and similarity values to get the weighted matrix.

weight\_mat = df\*ratings[hacker,2:ncol(ratings)]

# Calculated the prediction score for all the non attempted challenges.

non\_attempted\_pred\_score = c(non\_attempted\_pred\_score,rowSums(weight\_mat,na.rm=T)/temp\_sum)

}  
  
  
  
Now we got the prediction score for all the challenges and we need to recommend 10 challenges only.  
To do this, the challenges were sorted in decreasing order according to their prediction score and top 10 challenges were selected for recommendation.