Singular Value Decomposition:

Objective :-

The project is to predict ratings of users for movies using a sparse matrix of user-movie ratings.

Architecture:-

The database consists of 6041 users and 3953 movies and their ratings for some of the movies.

Flow of code :-

1) Creating the input matrix of user-movie(A) containing ratings.
2) Calculate U from AAT where columns of U are the eigenvectors which is a orthonormal and orthogonal matrix

3) Calculate S from the eigenvalues obtained. S will be a diagonal matrix containing eigenvalues as diagonals sorted in descending order.

4)Calculate V from ATA where columns of V are the eigenvectors which is a orthonormal and orthogonal matrix. Now, calculate V transpose.

4) Now, multiply U *S *V.T to obtain the predicted rating matrix.
5) Finally rmse and mae is calculated from the 2 matrices.

CUR

Objective :-

The project is to predict ratings of users for movies using a sparse matrix of user-movie ratings.

Architecture:-

The database consists of 6041 users and 3953 movies and their ratings for some of the movies.

Flow of code :-

6) Creating the input matrix of user-movie(A).

7) Calculate probabilities of rows and columns to create C and R matrix.

8) C and R matrix are created by randomly sampling rows and columns according to their respective probabilities. Here, 2000 rows and columns are sampled.
9) U matrix is created by taking the intersection of the C and R matrix.

10) SVD is applied on the U matrix and pseudoinverse is taken thereafter.

11) Final U matrix is prepared.

12) CUR product is calculated and predicted ratings are compared.

Recommender System Using Latent Factor Model Technique

Objective :-

The project is to predict the ratings of to users and recommend the movies
to them using the latent factor model from the database that contains
contain 1,000,209 anonymous ratings of approximately 3,900 movies made
by 6,040 MovieLens users who joined MovieLens in 2000.

Architecture:-

The database consists of three parts.
1)User database containing the information regarding the users
2)Movie database containing the information regarding the movies
3)ratings database containing the ratings given by the users to the movies
For this model we consider only the Ratings database and from this
database we create two dictionary of dictionaries user_rate and movie_rate
Primary structure of user_rate and movie_rate:
user_rate:- {user1:{movie1:rating1,movie2:rating2...},user2:{...}...}
where the keys are all the users of the database and the values are

the dictionaries where the key is the movie and the rating of that movie
the movie is part of the values of that user iff the user has given a rating to
that Movie

movie_rate:{movie1:{user1:rating1,user2:rating2...},movie2:{...}...}
where the keys are all the movies of the database and the values are
the dictionaries where the key is the user and his/her rating to that movie
the user is part of the values of that movie iff he has given a rating to that
movie

Lists and dictionaries were used at multiple occasions as required in the code

Flow of code :-

1) Import all the files
2)Create user_rate and movie_rate dictionaries
3)Set the number of latent factors,value of learning rate and the regularization parameter
4)Now we have to find 2 matrices Q_matrix and P_matrix such that
A=(Q_matrix)*((P_matrix)(Tra)) where A be the given ratings matrix.Therefore we use gradient descent
5)we calculate delta_Q and delta_P matrix such that
delta_Q=sigma(user)(-2*((movie_rate[movie][user]-np.dot(Q_matrix[movie],

P_matrix[user]))*P_matrix[user][latent_factor]))+2*lamda*P_matrix[movie][l
atent_factor]
delta_Q=sigma(user)(-2*((movie_rate[movie][user]-np.dot(Q_matrix[movie],
P_matrix[user]))*P_matrix[user][latent_factor]))+2*lamda*P_matrix[movie][l
atent_factor]
for a particular movie and a latent_factor
Where
(M)(Tra)=transpose of matrix M
sigma(par)=sum of all the elements around that parameter
6) After that
Q_matrix(time=t+1)=Q_matrix(time=t)-eta*delta_Q
P_matrix(time=t+1)=P_matrix(time=t)-eta*delta_P
7)repeat step 5 and 6 for 200 iterations
8)Then predict the ratings on the test data and calculate RMSE and MAE

predict[user][movie]= np.dot(Q_matrix[movie],P_matrix[user]
RMSE=(sse/count)**0.5
MAE=abs_error/count
sse=((sigma(user,movie))(predict[user][movie]-user_rate[user][movie]))**2
count=total number of ratings
abs_error=(sigma(1 to count)) abs((predict[user][movie]-user_rate[user][movie]))

Collaborative Model

Objective :-

The project is to predict ratings of users for movies using a sparse matrix of user-movie ratings using Collaborative Model.

Architecture:-

The database consists of 6041 users and 3953 movies and their ratings for some of the movies.

Flow of code :-

13) Creating the input matrix of user-movie(A) containing ratings.
14) Taking the transpose of user-movie matrix to obtain movie-user matrix.
15) Calculate the similarity matrix which is a movie-movie matrix. Now, for every movie subtract its row mean to normalize that row (so that the strict and loose raters are handled appropriately.) and divide by its norm so that the row becomes a unit vector .Now, take dot product of rows so as to find similarity between 2 movie vectors and store the row indices of top 3 similarities. This will give the most similar movies to that particular movie.

16) Now, calculate the predicted rating matrix from the similarity matrix by multiplying similarities with the corresponding original ratings.
17) Finally rmse and mae is calculated from the 2 matrices(i,e, the predicted matrix and the item user matrix).

Collaborative Model with Baseline Approach:-

Objective :-

The project is to predict ratings of users for movies using a sparse matrix of user-movie ratings using Collaborative Model.

Architecture:-

The database consists of 6041 users and 3953 movies and their ratings for some of the movies.

Flow of code :-

18) Creating the input matrix of user-movie(A) containing ratings.

19) Taking the transpose of user-movie matrix to obtain movie-user matrix.
20) Calculate the similarity matrix which is a movie-movie matrix. Now, for every movie subtract its row mean to normalize that row (so that the strict and loose raters are handled appropriately.) and divide by its norm so that the row becomes a unit vector .Now, take dot product of rows so as to find similarity between 2 movie vectors and store the row indices of top 3 similarities. This will give the most similar movies to that particular movie.
21) Now, calculate the predicted rating matrix from the similarity matrix by multiplying similarities with the corresponding original ratings and applying the baseline approach formula.
22) Finally rmse and mae is calculated from the 2 matrices(i,e, the predicted matrix and the item user matrix).

Recommender System Technique	**RMSE**	**MAE**	**Time taken for prediction**
Singular Value Decomposition	0.91	0.723	28 min
SVD with 90% energy	0.93	0.73	28 min
CUR	2.20	1.34	2 min
CUR with 90% energy	2.25	1.38	2 min
Latent factor model	0.90	0.71	32 min
Collaborative	1.33	0.62	16 min
Collaborative with Baseline Approach	1.14	0.53	16 min