1. PROBLEM STATEMENT

To build a movie recommendation system for users.

2. BUSINESS OBJECTIVE

The objective is to recommend movies to all the users in our dataset. Two collaborative filtering techniques have been implemented - namely the item-item based neighborhood-based and model-based. The idea is to find out what kind of samples and hyper parameters build a model having good accuracy and then recommend movies for the users using this model.

The focus was on giving the top 10 recommendations to the users in the dataset. The focus was on popular movies and active users by removing unpopular movies and inactive users. We focused on optimizing recommendation quality i.e high accuracy and sacrificed on the running time.

The system was built to serve the **users**.

3. COLLABORATIVE FILTERING

Recommender systems use the past preferences of users to predict what the user might prefer and make recommendations that should interest the user. Recommender systems are used in industries like E-commerce, online advertisements, entertainment industry, etc.
Collaborative filtering is the method of making predictions about the interests of a user using his past preferences. Collaborative filtering uses the technique of collaborating the ratings of different users and then using that gained information to filter out the products which might interest the user.

4. DATASET

The “20Ml” New Research Dataset (20M ratings; 190 MB). The dataset can be downloaded from https://grouplens.org/datasets/movielens/.Ratings and movies dataset is used to build a recommendation engine. Rating dataset contains UserId, MovieId, Ratings, and TimeStamp, while Movies dataset contains MovieId, Title of movie and Ratings.

Dataset Sampling

The dataset contains around 20 million ratings. There are 138493 unique users and 27278 unique movies. Dataset has explicit ratings on the scale of 1-5. Around 99% of user-

movie pairs have no ratings at all. That's why small samples are used to build a recommendation engine.

SAMPLE 1

Sample 1 is obtained by removing unpopular movies (movies with less than 1000 ratings) and then removing inactive users (users who have rated less than 1000 movies). From this most popular 100 movies and corresponding users are selected.
Sample 1 has 88008 user-item pairs and sparsity of User-Item matrix is 75.75%.

SAMPLE 2

Sample 2 is obtained by removing unpopular movies (movies with less than 1000 ratings) and then removing inactive users (users who have rated less than 1000 movies). From this 10% data is randomly selected.
Sample 2 has 120577 user-item pairs and sparsity of User-Item matrix is 95.94%

SAMPLE 3

Sample 3 is obtained by removing unpopular and most popular movies (removing movies with less than 1000 and greater than 20000 ratings) and then removing inactive users (users who have rated less than 1000 movies). From this 40% data is randomly selected.
Sample 3 has 44784 user-item pairs and sparsity of User-Item matrix is 98.30%

5. DESIGN CHOICES

- The variables considered for the two models were - movieID, userID, movie title, and rating.
- Neighborhood based - KNN algorithm and model based - ALS model was used.

6. MODEL 1

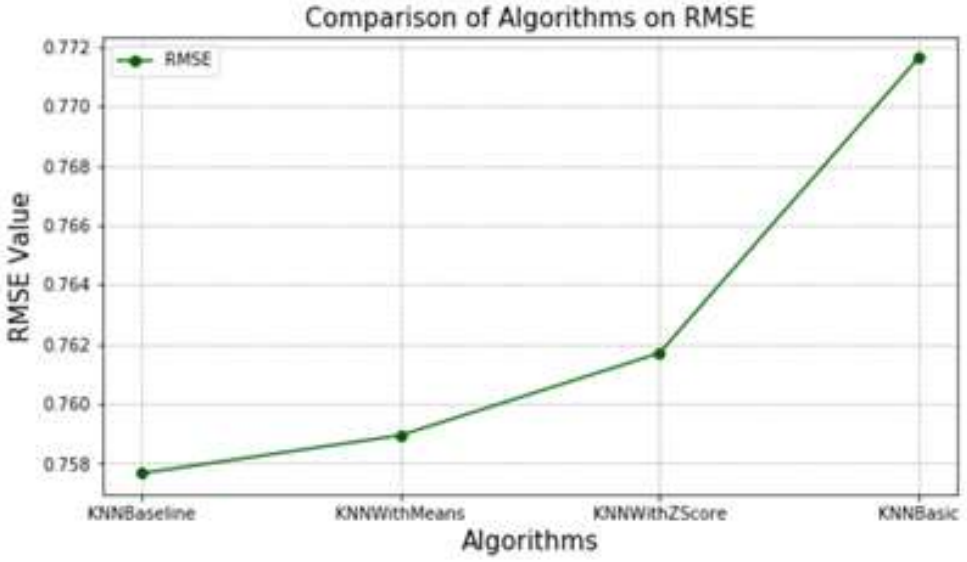
The first model implemented was **neighborhood-based collaborative filtering**. The main idea behind neighborhood-based filtering is that the user-item ratings which are stored are directly used to predict ratings for new items. This can be done in two ways known as user-based or **item-based recommendations**. User-based models evaluate the interest of a user u for an item I using the ratings for this item by other users, called neighbors, which have similar rating patterns. The neighbors of user u are typically the users v whose ratings on the items rated by both u and v, are most correlated to those of u. Item-based approaches, on the other hand, predict the rating of a user u for an item i based on the ratings of u for items similar to i. In such approaches, two items are similar if several users of the system have rated these items in a similar fashion. The user-based

approach is often harder to scale because of the dynamic nature of users, whereas items usually don't change much, and hence implemented item-based recommendation.

KNN algorithm was used out of all the neighborhood-based models available**.** KNN does not make any assumptions on the underlying data distribution but it relies on item feature similarity. When KNN makes inference about a movie, KNN calculates the "distance" between the target movie and every other movie in its database, then it ranks its distances and returns the top K (in our case 10) nearest neighbor movies as the most similar movie recommendations. To implement the KNN algorithm, perform all the evaluations and choose the hyperparameters, **surprise** package was used. The model was implemented on 3 samples. The same steps were followed for 3 samples and they have been given below for a particular sample:

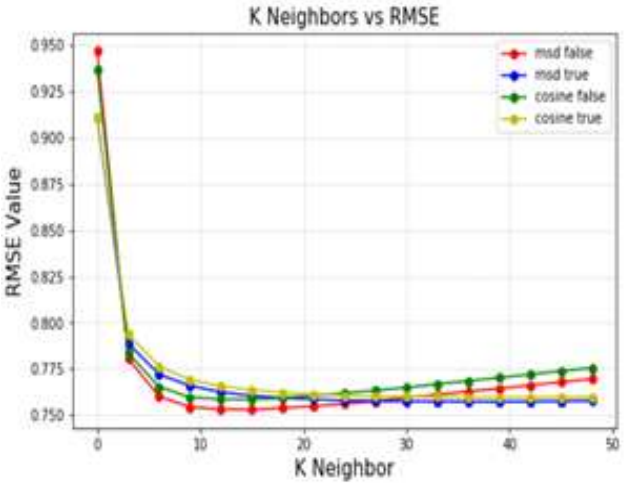
1. Loading the sample.

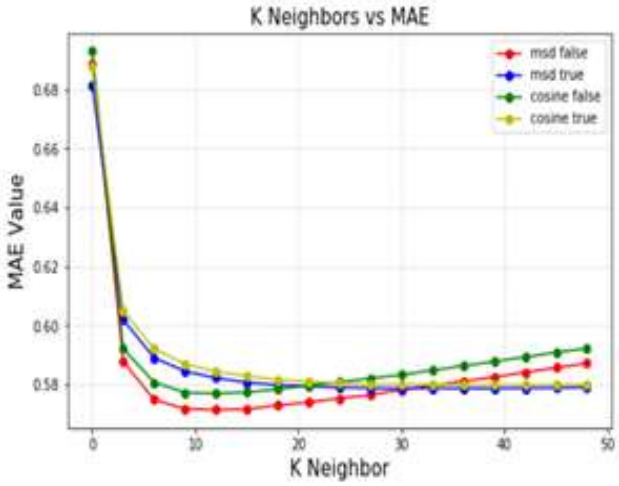
2. The surprise package allows performing **KNN using 4 methods namely KNNBasic, KNNBaseline, KNNwithMeans, and KNNwithZScore**. To choose the best KNN for the sample, cross-validation was done and the best algorithm was chosen based on the rmse value. For all three samples, **KNNBaseline** outperformed others and this can be seen below (output for medium-sized sample).



3. Next the **hyperparameters (Similarity function, User/Item based, neighborhood size K)** were chosen for KNNBaseline by using GridSearchCV function. The assumption

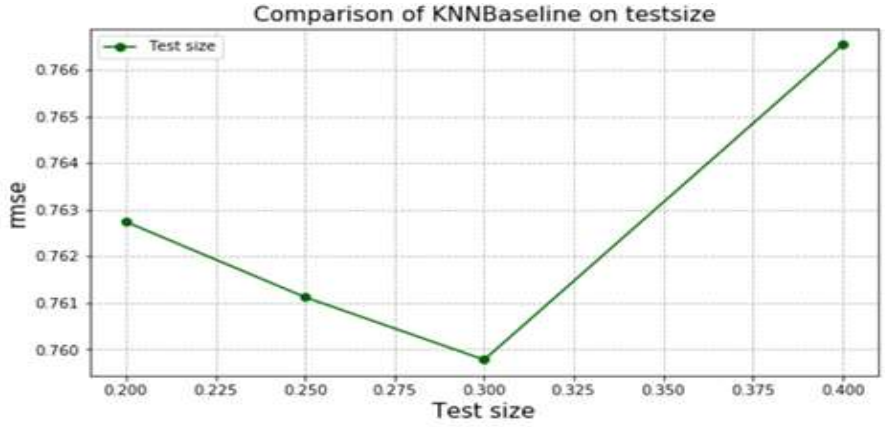
was to use item-based collaborative filtering , the hyperparameter tuning results supported the assumption and hence justified the choice by giving user-based- False for 2 out of the 3 samples. Out of the 2 available similarity functions – msd and cosine, MSD was considered best for all samples. MSD computes the Mean Squared Difference similarity between all pairs of users (or items). Finally based on sample size different values of neighborhood size K were chosen.





4. The primary evaluation metric was **rmse** and the secondary metric was **mae**. RMSE computes root mean square error and MAE computes mean absolute error.

5. Also, the best train test split ratio was computed using the best hyperparameters and the one with the least rmse was chosen.



6. The final set of hyperparameters was passed to KNNBaseline and the model was fitted on the train set, which was constructed using the best train/test split ratio. The predictions i.e- the top 10 movies were recommended for users in the test set. As the final objective is to recommend top n movies to a particular user, the output was also given for a particular user as can be seen below:

```
Movies recommended for userid 63071 are:

['Godfather, The (1972)',
 'Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)',
 'Memento (2000)',
 "Schindler's List (1993)",
 'Monty Python and the Holy Grail (1975)',
 'Blade Runner (1982)',
 'Matrix, The (1999)',
 'Die Hard (1988)',
 'Shining, The (1980)',
 'Aliens (1986)']
```

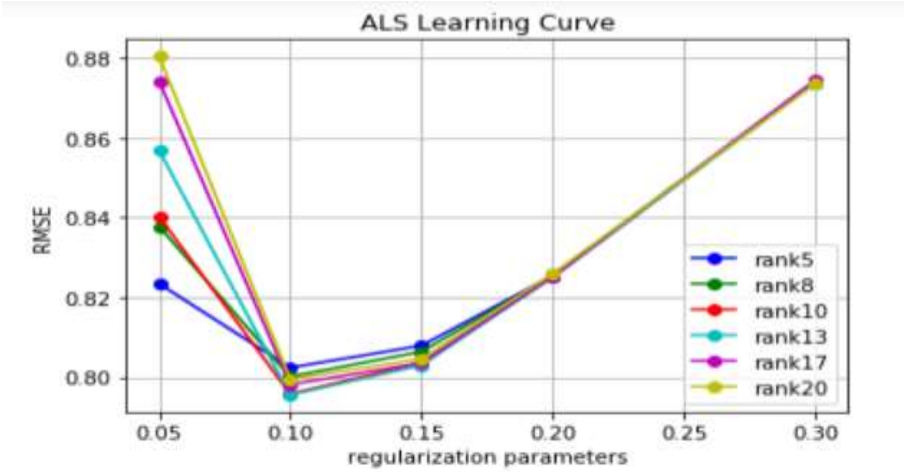
7. MODEL 2

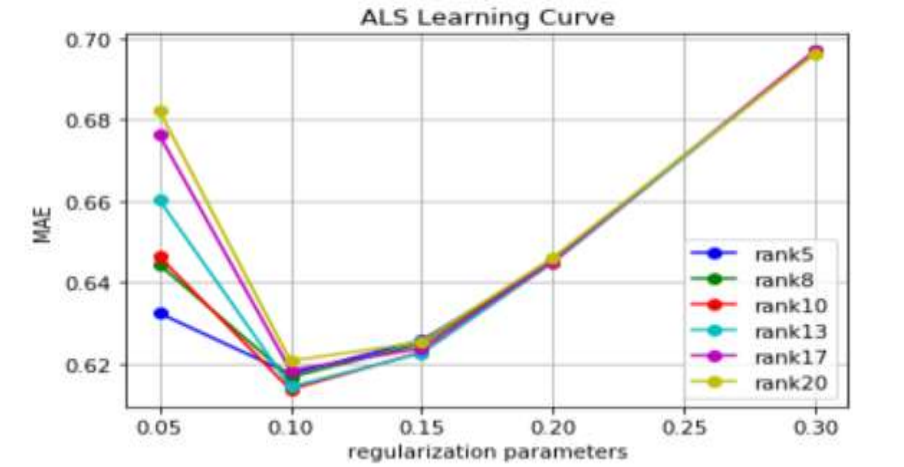
Second Implementation was performed using **Model-based collaborative filtering**. In model-based methods, machine learning and data mining methods are used in the context of predictive models. **Matrix Factorization algorithm** is used to reduce latent space of the algorithm. Matrix factorization algorithm factorizes a huge user-item interaction matrix R into two smaller rank matrices U and V sharing a joint latent vector space
(U: User feature matrix and V:item feature matrix),
such that R=UV^T , R has dimensions m*n (m: number of users, n: number of items)
U has dimensions m*r and V has dimensions n*r. (r: number of latent factors)
ALS (Alternating Least Squares) method is used to find matrix factorization. In ALS, 1st U is kept as constant and the optimization equation is solved for V. Then V is kept as constant and the optimization equation is solved for U. These two steps are iterated to convergence.

Drawback of this method
The drawback of ALS is that it is not quite as efficient as stochastic-gradient descent in large-scale settings with explicit ratings. Since the very sparse matrix R is factored into two low dimension matrices, quality of recommendations cannot be guaranteed.

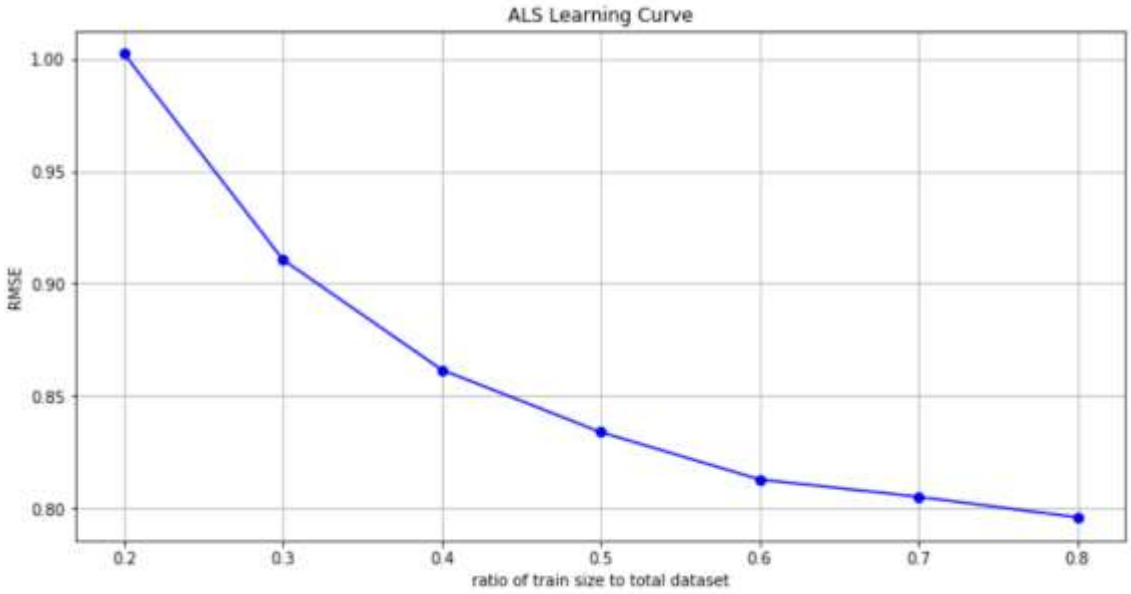
Implementation

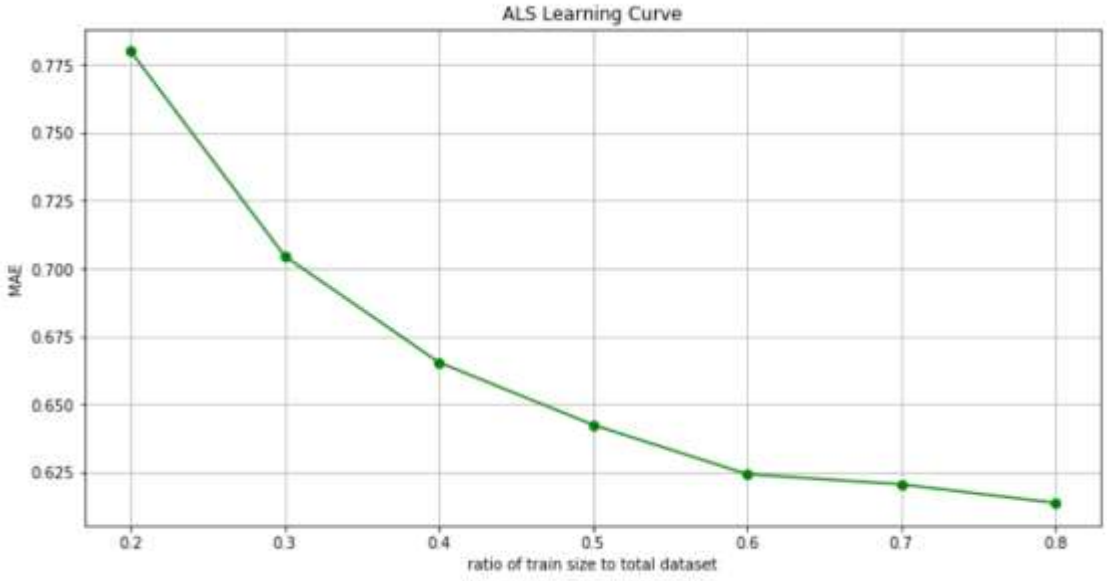
1. Loading the sample
2. ALS function from pyspark library in python is used for implementation.
3. **Hyperparameters** tuned for the model are: **number of latent factors and regularization parameter.** Since RMSE is primary evaluation metric, the set of hyperparameters for which model gives lowest RMSE values for test dataset is considered as the best model. Attached below is the plot of RMSE vs number of latent factors(rank) and regularization parameter for sample 1. For sample1 number of latent factors=13 and regularization parameter=0.1 are selected to build the final model.





4. The primary evaluation metric was **rmse** and the secondary metric was **mae**. RMSE computes root mean square error and MAE computes mean absolute error.
5. Also, the best train test split ratio was computed using the best hyperparameters and the one with the least rmse was chosen. Training size 0f 80% is chosen to build the final model on sample 1.





6. The final set of hyperparameters was passed to ALS and the model was fitted on the train set, which was constructed using the best train/test split ratio. The predictions i.e- the top 10 movies were recommended for users in the test set. As the final objective is to recommend top n movies to a particular user, the output was also given for a particular user as can be seen below:

```
make_recommendation(156,10,model1)
```

```
['Shawshank Redemption, The (1994)',
 'Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)',
 "Schindler's List (1993)",
 'Godfather, The (1972)',
 'Silence of the Lambs, The (1991)',
 'Star Wars: Episode IV - A New Hope (1977)',
 'Usual Suspects, The (1995)',
 'Matrix, The (1999)',
 'Saving Private Ryan (1998)',
 'Sixth Sense, The (1999)']
```

```
make_recommendation(143,10,model1)
```

```
['Shawshank Redemption, The (1994)',
 'Star Wars: Episode IV - A New Hope (1977)',
 'Usual Suspects, The (1995)',
 "Schindler's List (1993)",
 'Godfather, The (1972)',
 'Star Wars: Episode V - The Empire Strikes Back (1980)',
 'Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)',
 'Fugitive, The (1993)',
 'Matrix, The (1999)',
 'Silence of the Lambs, The (1991)']
```

8. EVALUATION

- **Accuracy**

  Accuracy is used to check how the model performs on the test data. It compares the predicted ratings of the movies with the actual ratings and gives an estimation of how much the model is accurate. The 2 accuracy metrics used here are *MAE*(Mean Absolute Error) and *RMSE*(Root mean square error).

  Mean Absolute Error (MAE):

  It is the average of all absolute errors i.e the absolute difference between the actual rating of a movie for a particular user and the predicted rating for the same.

$$MAE = \frac{\Sigma^{n}_{j=1}|r_{uj} - o_{uj}|}{n}$$

where r_{uj} is the predicted rating of the movie' j' given by user 'u'

And o_{uj} is the observed rating of the movie 'j' given by user 'u'

And n is the number of ratings in the test data.

Root Mean Squared Error(RMSE):

$$RMSE = \sqrt{\frac{\Sigma^{n}_{J=1}(r_{uj}-o_{uj})^2}{n}}$$

using the same above notations.

- **Coverage Ratio**

  Coverage is the distribution of accuracy over users and/or items.
  A good recommendation considered here is the one having rating 3.5 or more here.

  a. **User Coverage**

  It is defined as the fraction of users for which atleast k items can be recommended *well*. Here recommended well has been defined as a recommendation whose predicted rating by the model is more than or equal to 3.5. The value of k considered is 10.

  $$User\ Coverage = \frac{Number\ of\ users\ which\ have\ atleast\ k\ movies\ with\ rating\ >=\ 3.5\ recommended}{Total\ number\ of\ users\ in\ the\ testset}$$

  b. **Catalogue Coverage**

  It is defined as the fraction of items that in the top k recommendations for atleast one user. The value of k is considered as 10 here.

  $$Catalogue\ Coverage = \frac{|\cup_{u=1}^{m} T_u|}{n}$$

  if T_u is the set of top k items recommended to user u and n is the total number of items in the testset.

  c. **Item Coverage**

  It is defined as the fraction of items that can be recommended to atleast k users well. Similarly as above, recommended well is defined as the predicted rating of a movie being greater than or equal to 3.5.The value of k considered as 10.

  $$Item\ Coverage = \frac{Number\ of\ movies\ which\ have\ predicted\ rating\ as\ atleast\ 3.5\ by\ atleast\ k\ users}{Total\ number\ of\ movies\ in\ the\ testset}$$

9. RESULTS

The two models – neighbourhood based, KNNBaseline and model base, ALS algorithm were run on 3 samples of different sizes and the final results of both can be seen below:

KNNBaseline

	sample number	rmse	mae	user coverage	catalogue coverage ratio	sample size
0	1.0	0.764640	0.581822	0.866100	0.770000	88008
1	2.0	0.822772	0.634599	0.546227	0.502710	120577
2	3.0	0.915257	0.696392	0.251816	0.605743	44784

ALS

	sample_number	rmse	mae	user_coverage	item_coverage	catalog_coverage
0	1	0.795800	0.613681	0.917657	1.000000	0.990000
1	2	0.827519	0.652238	0.937301	0.753403	0.668132
2	3	0.946029	0.743810	0.965517	0.816920	0.718522

The results from both the models were concordant and indicated that the sampling size of 88008 gave the best results. This means that the sample which took care of popular movies i.e- those were rated the most and active users, gave the best results. Thus, we were successful in meeting the business objective and are comfortable in putting these solutions into production at a real company given that the dataset size is not humungous.

10. FUTURE WORK AND POTENTIAL WATCH OUTS

Potential watch outs

- The scalability is an issue. As the sample size increases, the time to fit the model increases for both the models.
- Sparsity is high i.e- the percentage of people who have rated the movies is quite low.
- Cold start problem occurred in the neighborhood-based model i.e- recommending movies to a completely new user can be a problem as there is no previous information.
- Both the models are inflexible i.e- for every new addition in data; both the models have to be fitted again.

Future work

In terms of design choices, we can consider context-aware features like time and location. Taking timestamp, for example, we could study how old are the ratings provided by a user before assigning them equal importance to a recent rating.