Advanced Data Science Capstone IBM

Ammar Shaheen

3 Dec 2022

OUTLINES

- Dataset - Use case
- Data exploration
- Data cleansing - Data aggregation
- Model definition and training
- Model evaluation
- Hyperparameters tuning

DATASET

Dataset published by Audioscrobbler - a music recommendation system for last.fm

Contains:

- 140.000 unique users
- 1.6 million unique artists

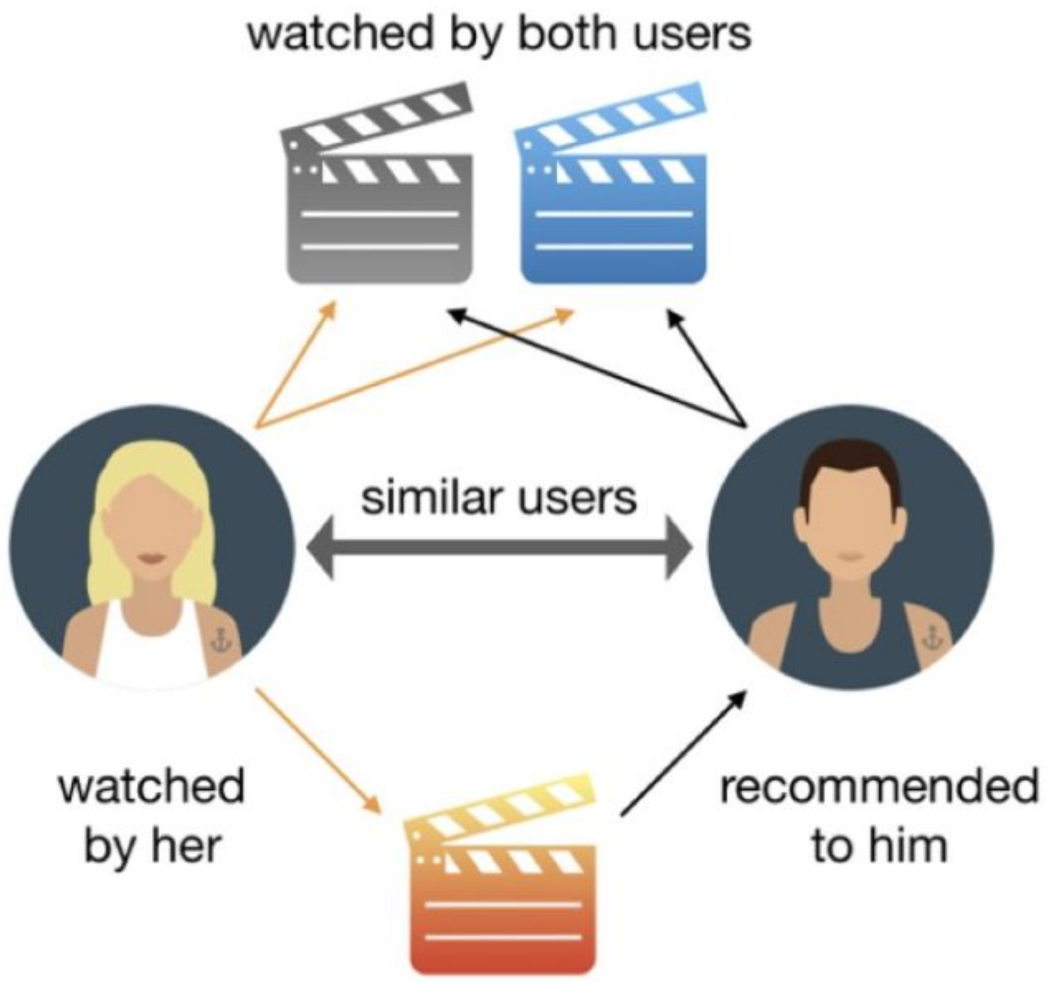
userID	artistID	playCount
113186	46843	56
745456	84646	118
...	...	...
...	...	...

misspelledID	standardID
84646	184528
148328	435846
...	...

artistID	artist_name
84646	Ed Sheeran
148328	Coldplay
...	...

USE CASE

Develop music
Recommendation system



DATA EXPLORATION

```
allusers = userArtistDF.count()
print("All rows in database: ", allusers )
uniqueUsers = userArtistDF.select('userID').distinct().count()
print("Total n. of distinct users: ", uniqueUsers)
```

```
All rows in database:  24296858
Total n. of distinct users:  148111
```

```
uniqueArtists = userArtistDF.select('artistID').distinct().count()
print("Total n. of artists: ", uniqueArtists)
```

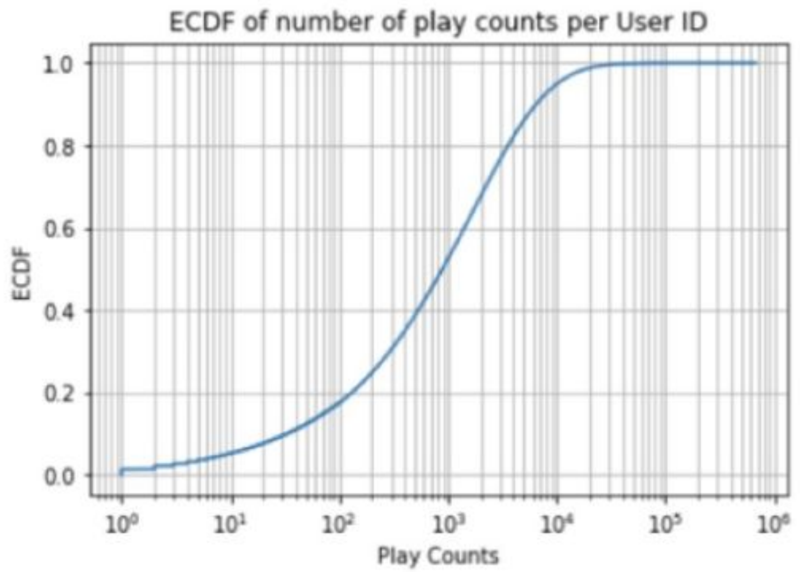
```
Total n. of artists:  1631028
```

DATA EXPLORATION

A user played 2509 times on average

50% of the users have the play counts less than or equal to 892 times.
75% of the users have the play counts less than or equal to 2800 times.
95% of the users have the play counts less than or equal to 10120 times.
About 7746 users (5.23%) have the play counts less than or equal to 10 times

These users have very little interaction with the system, so there is more difficult for recommending for these users.




```
Total = 371638969
Mean = 2509.1922207
Min = 1
Max = 674412
Percentile 25% :204.0
Percentile 50% :892.0
Percentile 75% :2800.0
Percentile 90% :6484.0
Percentile 95% :10120.0
Percentile 99% :21569.2
The percentage of user playing less than 10 times P(Y<=10) = 0.05228511049145573
```

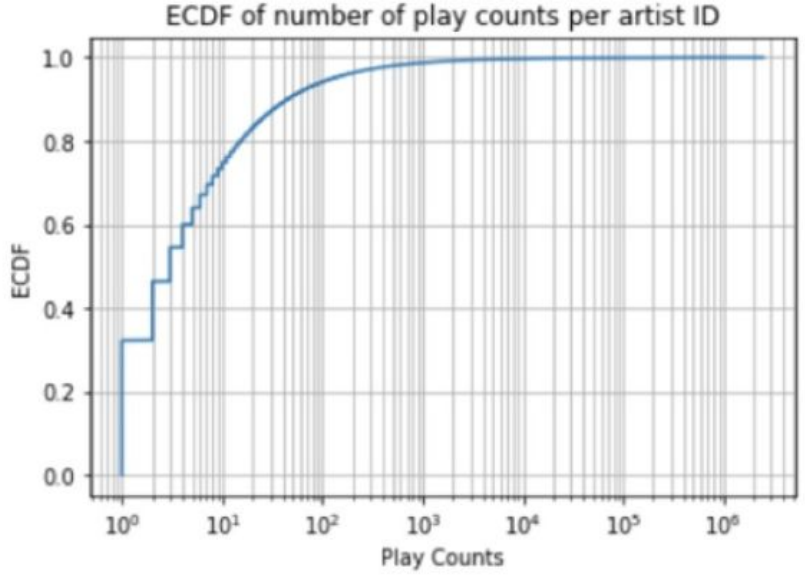
DATA EXPLORATION

In average, play count per artist is 227 times

Only 74.87% of the artists is played less than or equal to 10 times. And 98.74% of the artists is played less than or equal to 1000 times.

Top 5 artists play counts: 1425942 1542806 1930592 2259185 2502130.

These accounts for 2.6% on overall number of playCount. Moreover, the play count of top 5 is much higher than the mean. So we can infer that we can recommend most played artists to every user with this stop 5 artists. And still get high performance.




```
Sum = 371638969
Mean = 227.855664648
Min = 1
Max = 2502130
Top 5 play counts: [1425942 1542806 1930592 2259185 2502130]
Sum top 5 artist play counts: 9660655
Percentage of top 5 artist play counts: 0.0259947309239
P(playCount<=10) = 0.7486793605014751
P(playCount<=1000) = 0.987435531456235
```

DATA CLEANSING

Same artists but different IDs =>

artistID	name
1000010	Aerosmith

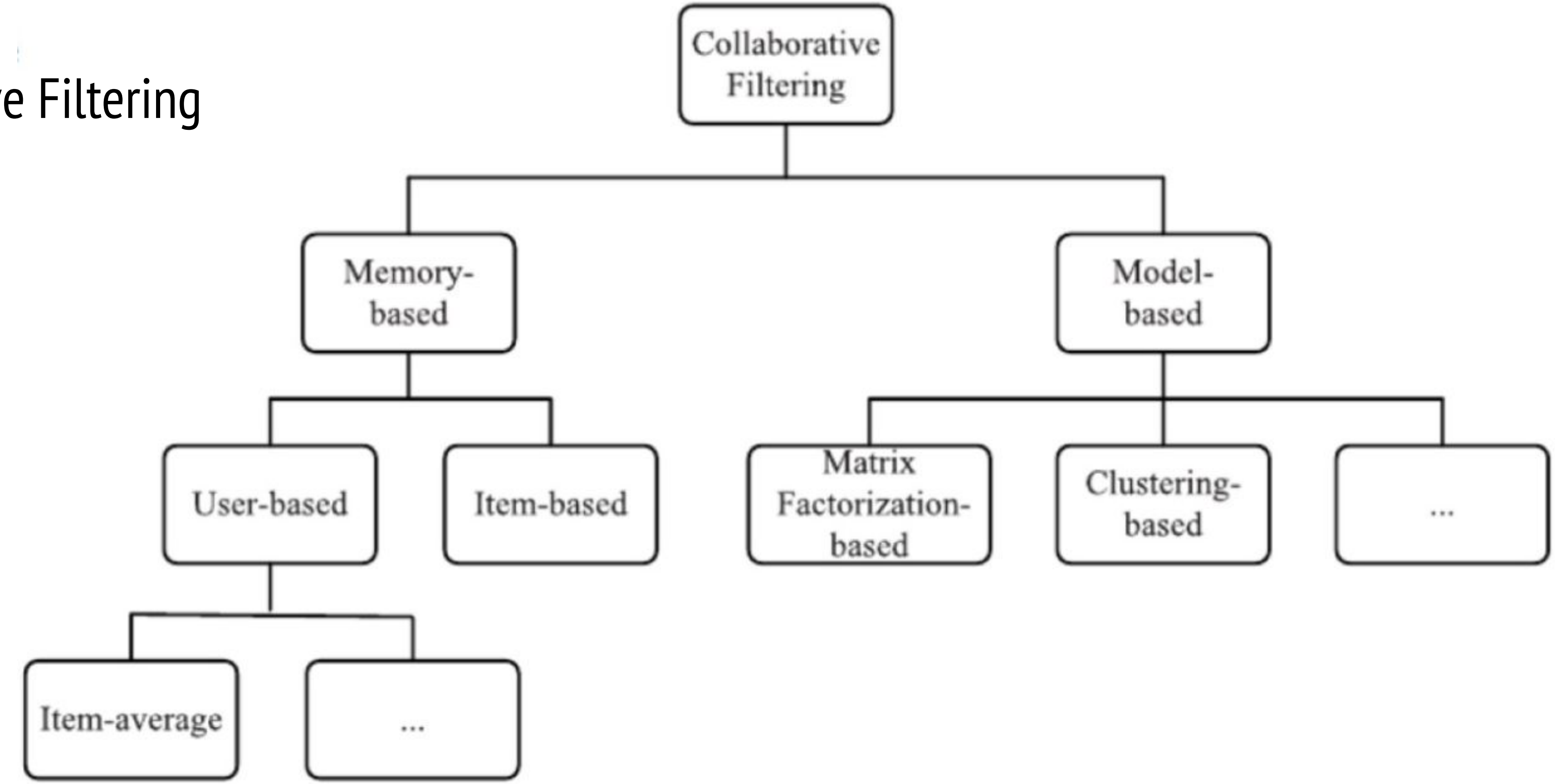
artistID	name
2082323	01 Aerosmith

mispelledID	standardID
2082323	1000010

Solved: Replace misspelledID by standardID

MODEL DEFINITION

Collaborative Filtering



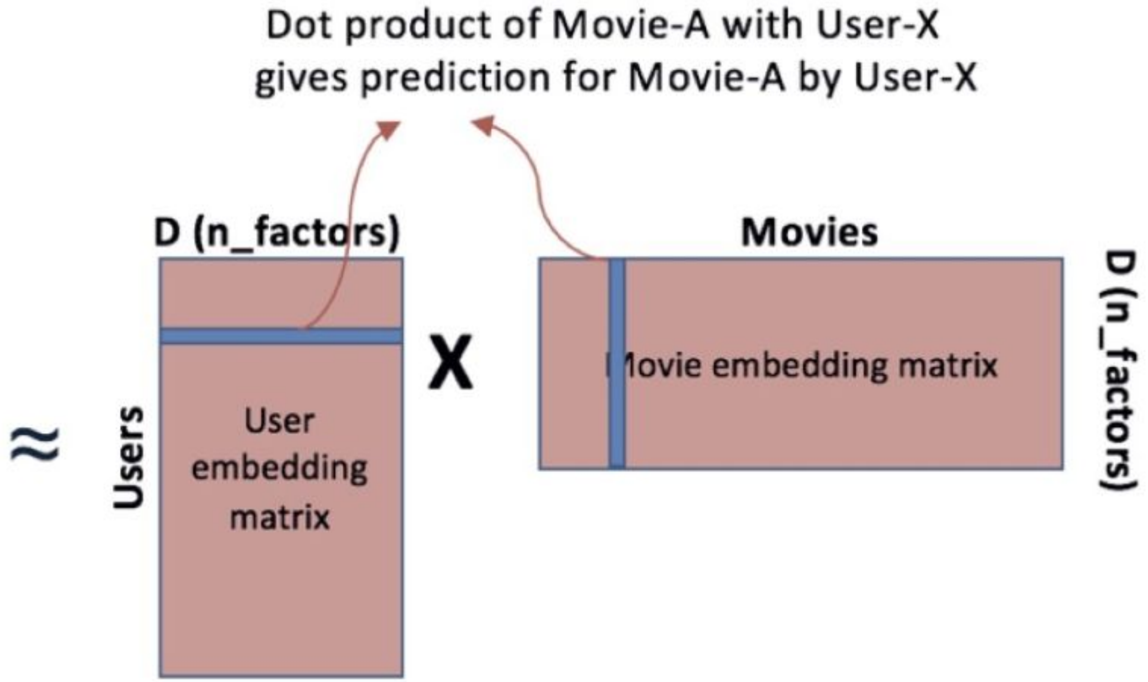
MODEL DEFINITION

Matrix Factorisation

Movies

(Sparse matrix)	Waking Life	Boyhood	Before Sunset
Jesse	4.5		4.0
Celine		3.5	5.0

Users



MODEL TRAINING

A model can be trained by using ALS.trainImplicit(<training data>,<rank>)
Split data to 70% for training and 30% for testing
We can also use some additional parameters to adjust the quality of the model. Currently, let's see:

Param	Value
Rank	10
Iterations	5
Lambda	0.01
Alpha	1.0

```
#setting parameters
rank=10
iterations=5
lambda_=0.01
alpha=1.0

#training
t0 = time()
model = ALS.trainImplicit(allData, rank)
t1 = time()
print("finish training model in %f secs" % (t1 - t0))
```

```
finish training model in 83.877863 secs
```

MODEL TRAINING

Let's predict the top 5 artists which user has ID = 2093760 may find interesting

```
userID = 2093760
recommendations = model.recommendProducts(userID,5)

recArtist = set(rating[1] for rating in recommendations)

# Filter in those artists, get just artist, and print
def artistNames(line):
#     [artistID, name]
    if (line[0] in recArtist):
        return True
    else:
        return False

recList = artistByID.filter(artistNames).values().collect()
print(recList)
```

```
=> ['Kent', 'Oasis', 'The Killers', 'Kaiser Chiefs', 'Unknown']
```

MODEL EVALUATION

```
t0 = time()
auc = calculateAUC( cvData,bAllItemIDs, model.predictAll)
t1 = time()
print("auc=",auc)
print("finish in %f seconds" % (t1 - t0))
```

```
auc= 0.96070668941573
finish in 79.103230 seconds
```

HYPERPARAMETER TUNING

```
evaluations = []

for rank in [10, 50]:
    for lambda_ in [1.0, 0.0001]:
        for alpha in [1.0, 40.0]:
            print("Train model with rank=%d lambda_=%f alpha=%f" % (rank, lambda_, alpha))
            # with each combination of params, we should run multiple times and get avg
            # for simple, we only run one time.
            model = ALS.trainImplicit(ratings=trainData,rank=rank,iterations=5,lambda_=lambda_ ,alpha=alpha)
            auc = calculateAUC(cvData,bListenCount,model.predictAll)

            evaluations.append(((rank, lambda_, alpha), auc))

            unpersist(model)
```

HYPERPARAMETER TUNING

Grid search

Rank	Lambda	Alpha	AUC
10	1.0	40.0	0.9738
10	0.0001	40.0	0.9718
50	1.0	40.0	0.9715
50	0.0001	40.0	0.97
10	1.0	1.0	0.9644
50	1.0	1.0	0.9592
10	0.0001	1.0	0.9584
50	0.0001	1.0	0.9427

The model with the largest AUC score has the combination of Rank = 10.Lambda = 1.0 and Alpha = 40.0

CONCLUSION

- The model is highly biased on the artists who have large number of play counts
- We may not need to include lambda and alpha parameters to the model if we only retrieve the top result less than 10 artists.