Million Songs-Output#1

January 23, 2021

[3]:
```
# %% [markdown]
# # Million Songs Problem Statement
#
# ### Context
# With the advent of technology, societies have become more efficient with␣
↪their lives. But at the same time, individual human lives have become much␣
↪more fast paced and distracted by leaving little time to explore artistic␣
↪pursuits. Also, the technology has made significant advancements in the␣
↪ability to coexist with art and general entertainment. In fact, it has made␣
↪it easier for humans with shortage of time to find and consume good content.␣
↪Therefore, one of the key challenges for the companies is to be able to␣
↪figure out what kind of content their customers are most likely to consume.␣
↪Almost every internet based company's revenue relies on the time consumers␣
↪spend on their platforms. These companies need to be able to figure out what␣
↪kind of content is needed in order to increase the time spent by customers␣
↪on their platform and make their experience better.
# Spotify is one such audio content provider who has got a huge market base␣
↪across the world. It has grown significantly because of its ability to␣
↪recommend the 'best' next song to each and every customer based on the huge␣
↪preference database they have gathered over time like millions of customers␣
↪and billions of songs. This is done by using smart recommendation systems␣
↪that can recommend songs based on the users' likes/dislikes
#
# ### Problem Statement
#
# Build a recommendation system to propose the top 10 songs for a user based on␣
↪the likelihood of listening to those songs.
#
# ### Data Dictionary
# The core data is the Taste Profile Subset released by The Echo Nest as part␣
↪of the Million Song Dataset. There are two files in this dataset. One␣
↪contains the details about the song id, titles, release, artist name and the␣
↪year of release. Second file contains the user id, song id and the play␣
↪count of users.
#
# #### song_data
# 1. song_id - A unique id given to every song
```

```
# 2. title - Title of the song
# 3. Release - Name of the released album
# 4. Artist_name - Name of the artist
# 5. year - Year of release
#
# #### count_data
# 1. user _id - A unique id given to the user
# 2. song_id - A unique id given to the song
# 3. play_count - Number of times the song was played
#
# #### Data Source
#  http://millionsongdataset.com/
#
```

[4]:

```
from Functions_Million_Songs import *
from Functions_Million_Songs import tmp_pivot_table, sparse_matrix, dense_matrix
from Functions_Million_Songs import interactions_matrix     # made from␣
↪dfu_small


%load_ext autoreload
%autoreload 2
#%reload_ext autoreload
```

[5]:

```
init_datsets()
from Functions_Million_Songs import dfs, dfu, dfus, dfu_small  #song, user and␣
↪joint datasets
dfs.info(), dfu.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 999056 entries, 0 to 999999
Data columns (total 5 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   song_id      999056 non-null  object
 1   title        999041 non-null  object
 2   release      999051 non-null  object
 3   artist_name  999056 non-null  object
 4   year         999056 non-null  int64
dtypes: int64(1), object(4)
memory usage: 45.7+ MB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2000000 entries, 0 to 1999999
Data columns (total 3 columns):
 #   Column     Dtype
---  ------     -----
 0   user_id    object
```

```
 1   song_id     object
 2   play_count  int64
dtypes: int64(1), object(2)
memory usage: 61.0+ MB
```

[5]: (None, None)

[6]:
```
print(f'\n\n Only a few songs have been listened to a lot by a specific user,␣
 ↪so we keep these "outliers": \n')
dfu[['user_id','play_count']].nlargest(10,'play_count'), dfs.count()

print_stats()

#Some Song titles appear in more than one song
print('\n\n Duplicate Titles example: "Intro"')
dfs.loc[dfs.title=='Intro']

print('\n\nSample data for "Kings Of Leon": \n ')
dfus.loc[dfus.artist_name=='Kings Of Leon'].sample(n=5)

top_n = 10
show_top_lists(top_n)

# surprise library -- Data Distributions and
# model evaluation (need more time to finish)
X = surprise_distribution_model_evaluation(dfus, dfu_small)
print(X)
# print(results)
```

```
 Only a few songs have been listened to a lot by a specific user, so we keep
these "outliers":



 Some Listner and Song Stats:
          Total Users:  76353
        Total Artists:  72652
          Total Songs: 999056
    Total Song Titles: 702350
Total Albums Released: 149211
     Oldest Song Year:   1922
     Newest Song Year:   2011
    User-Song Density: 0.0026%


 Duplicate Titles example: "Intro"
```

```
Sample data for "Kings Of Leon":



Top 10 Most Actie Listeners
                                         user_id  play_count
Rank
1     6d625c6557df84b60d90426c0116138b617b9449         711
2     fbee1c8ce1a346fa07d2ef648cec81117438b91f         643
3     4e11f45d732f4861772b2906f81a7d384552ad12         556
4     24b98f8ab023f6e7a1c37c7729c623f7b821eb95         540
5     1aa4fd215aadb160965110ed8a829745cde319eb         533
6     b04e41133dd3d30a5631cc8589a1eadd48a8bd53         523
7     15eeb36ae1c62d60de9fdeea0d121eb7d08713be         522
8     a15075a926c1998d91940f118342ba8356efc7d4         502
9     ce5c912bb8044f23fc0fc31bd986b8d0a7303db5         489
10    6a9cf03dfb2fc82f5b3b043c9c3fdbab997fd54d         487


Top 10 Most Listened Artist:
                 artist_name  play_count
Rank
1                   Coldplay       70138
2              Kings Of Leon       68570
3     Florence + The Machine       60066
4              Dwight Yoakam       54136
5                      Björk       53814
6             The Black Keys       52220
7               Jack Johnson       44083
8              Justin Bieber       41645
9                OneRepublic       40981
10                     Train       39279


Top 10 most Played Songs:
                                                   title  \
Rank
1                                         You're The One
2                                                   Undo
3                                                Revelry
4     Horn Concerto No. 4 in E flat K495: II. Romanc…
5                                          Sehr kosmisch
6                           Dog Days Are Over (Radio Edit)
7                                                Secrets
8                                                 Canada
9                                                Invalid
```

```
10                                      Ain't Misbehavin

                                            artist_name  play_count
Rank
1                                         Dwight Yoakam       54136
2                                                 Björk       49253
3                                         Kings Of Leon       41418
4     Barry Tuckwell/Academy of St Martin-in-the-Fie…       31153
5                                              Harmonia       31036
6                                 Florence + The Machine       26663
7                                           OneRepublic       22100
8                                      Five Iron Frenzy       21019
9                                              Tub Ring       19645
10                                            Sam Cooke       18309


Top 10 Songs with most number of listeners:


                                                  title  \
Rank
1                                        Sehr kosmisch
2                                                  Undo
3                          Dog Days Are Over (Radio Edit)
4                                        You're The One
5                                               Revelry
6                                               Secrets
7     Horn Concerto No. 4 in E flat K495: II. Romanc…
8                                             Fireflies
9                                       Hey_ Soul Sister
10                                             Tive Sim

                                            artist_name  Number_of_Listeners  \
Rank
1                                              Harmonia                 8277
2                                                 Björk                 7032
3                                 Florence + The Machine                 6949
4                                         Dwight Yoakam                 6412
5                                         Kings Of Leon                 6145
6                                           OneRepublic                 5841
7     Barry Tuckwell/Academy of St Martin-in-the-Fie…                 5385
8                                    Charttraxx Karaoke                 4795
9                                                 Train                 4758
10                                              Cartola                 4548

                 song_id
Rank
1     SOFRQTD12A81C233C0
```

```
2     SOAUWYT12A81C206F1
3     SOAXGDH12A8C13F8A1
4     SOBONKR12A58A7A7E0
5     SOSXLTC12AF72A7F54
6     SONYKOW12AB01849C9
7     SOEGIYH12A6D4FC0E3
8     SOLFXKT12AB017E3E0
9     SODJWHY12A8C142CCE
10    SOFLJQZ12A6D4FADA6
```

```
                 song_id  count
2220  SOFRQTD12A81C233C0   8277
317   SOAUWYT12A81C206F1   7032
352   SOAXGDH12A8C13F8A1   6949
614   SOBONKR12A58A7A7E0   6412
7416  SOSXLTC12AF72A7F54   6145
…                    …     …
8747  SOWNLZF12A58A79811     51
4492  SOLIGVL12AB017DBAE     51
622   SOBPGWB12A6D4F7EF3     50
9638  SOYYBJJ12AB017E9FD     48
2666  SOGSPGJ12A8C134FAA     48

[10000 rows x 2 columns]
```

```
Estimating biases using als…
Computing the msd similarity matrix…
Done computing similarity matrix.
Estimating biases using als…
Computing the msd similarity matrix…
Done computing similarity matrix.
Estimating biases using als…
Computing the msd similarity matrix…
Done computing similarity matrix.
Computing the msd similarity matrix…
Done computing similarity matrix.
Computing the msd similarity matrix…
Done computing similarity matrix.
Computing the msd similarity matrix…
Done computing similarity matrix.
Computing the msd similarity matrix…
Done computing similarity matrix.
Computing the msd similarity matrix…
Done computing similarity matrix.
```

```
Computing the msd similarity matrix…
Done computing similarity matrix.
Computing the msd similarity matrix…
Done computing similarity matrix.
Computing the msd similarity matrix…
Done computing similarity matrix.
Computing the msd similarity matrix…
Done computing similarity matrix.
Estimating biases using als…
Estimating biases using als…
Estimating biases using als…
                 test_rmse    fit_time  test_time
Algorithm
BaselineOnly      5.384732    1.958337   3.614825
KNNBaseline       5.387468   15.956658  70.437691
KNNWithZScore     5.398714   12.596456  71.014284
CoClustering      5.417637   12.846680   3.501397
SlopeOne          5.420629    9.018290  23.741897
KNNWithMeans      5.428411   12.115448  65.781161
NMF               5.557056   42.861590   3.565619
KNNBasic          5.597561   11.745206  63.213809
NormalPredictor   6.641988    1.059326   3.476840
SVD               8.439450   37.268810   2.902564
SVDpp             8.442772  587.291683  27.076039
```

MODEL #1 - Collaborative Filtering - Using Cosine Similarity of User_ids ### dfus dataset(all users & songs data combo) ### It is possible to use altenate route(not implmented here) of ### just using top songs and users

[8]:
```
init_model_1_CF_matrices()
from Functions_Million_Songs import tmp_pivot_table, sparse_matrix, dense_matrix

# pick user_id and # of song recommendations desired
user_id_to_recommend_songs_to = '01845f57f5c8b3309233e5a4a7145a7d33ad3d52'
num_of_songs_to_recommend = 5

# Show songs user has already listened to
x = pd.DataFrame(dfus.query('user_id ==␣
↪@user_id_to_recommend_songs_to')[['title','artist_name']])
print(f'\n\nSongs {user_id_to_recommend_songs_to} has already listened to:\n␣
↪{x[["title","artist_name"]]}')

# Recommend songs:
# find ordinal position of user_id index in tmp_pivot_table(user_id,song_id,␣
↪play_count)
user_id_index = tmp_pivot_table.index.get_loc(
                tmp_pivot_table[tmp_pivot_table.index ==␣
↪user_id_to_recommend_songs_to]
```

```
                 .iloc[-1].name)
# get cosine similarity based recommendations
recommended_song_id_column_index = (recommendations_cosine_model_1
                                                       (user_id_index,
                                                        num_of_songs_to_recommend,
                                                        dense_matrix))

print(f'\n\nTop {num_of_songs_to_recommend} song recommendations for␣
 ↪"{user_id_to_recommend_songs_to}"": \n')

get_song_ids_from_index(recommended_song_id_column_index, tmp_pivot_table)
```

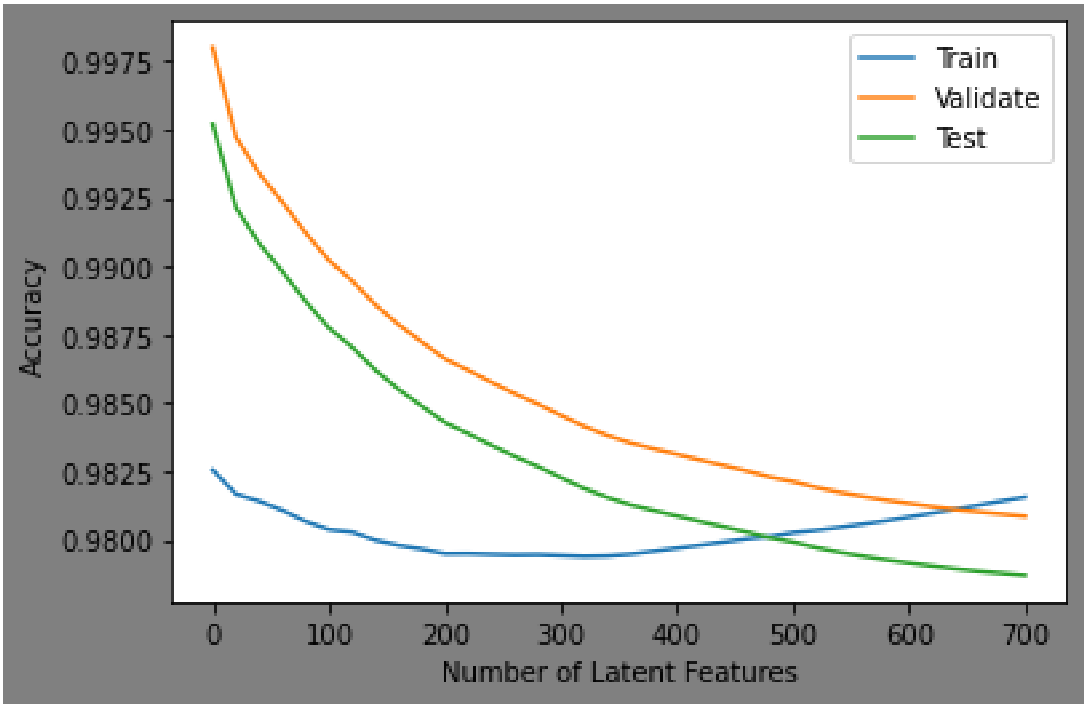
```
Songs 01845f57f5c8b3309233e5a4a7145a7d33ad3d52 has already listened to:
                                                         title  \
1702697                                             Orgelblut
1702698                                                  Welk
1702699                                        In These Arms
1702700  Die Kunst der Fuge_ BWV 1080 (2007 Digital Rem…
1702701                                  Don't Start Me Talkin'
1702702                                            Creil City
1702703              Together Again (Jimmy Jam Deep Remix)
1702704                                          Silent Shout
1702705                          Marshall Examines His Carcass
1702706                                       Streets On Lock
1702707                                     Le Jardin d'Hiver
1702708                            Schwarze Biene (Black Maja)
1702709                                            Représente
1702710                                    Love Is Not A Fight
1702711                                              The Gift
1702712                                                 Trash
1702713                                               Kill Me
1702714        What We Do (Explicit) (Feat. Memphis Bleek)

                           artist_name
1702697            Bohren & Der Club Of Gore
1702698            Bohren & Der Club Of Gore
1702699                             Bon Jovi
1702700                          Lionel Rogg
1702701                 Sonny Boy Williamson
1702702                      Alliance Ethnik
1702703                        Janet Jackson
1702704                            The Knife
1702705                      Octopus Project
1702706                          Young Jeezy
1702707                      Jacky Terrasson
```

```
1702708           Bohren & Der Club Of Gore
1702709                    Alliance Ethnik
1702710                    Warren Barfield
1702711                Angels and Airwaves
1702712                 The New York Dolls
1702713                Make the Girl Dance
1702714  Sauce Money Featuring Memphis Bleek
```

```
Top 5 song recommendations for "01845f57f5c8b3309233e5a4a7145a7d33ad3d52"":
```

[8]:
```
['Make Love To Your Mind',
 'En Algún Lugar Del Puerto (2001 Digital Remaster)',
 'Sinisten tähtien alla',
 'Forever & Always',
 'How Long']
```

[9]:
```
## MODEL #2 - SVD Matrix Factorization
##### Split data into train,validate and test sets on dfus dataset(all users &
 ↪songs data combo)
##### same user_id may have been randomly split across train,validate and test
 ↪sets
##### It is possible to use altenate route(not implmented here) of keeping
##### user_id  & song_id of a user either in train,validate and test but not
 ↪split across
##### these three sets
X_train, X_test = train_test_split(dfu_small,    test_size=0.2, random_state=42)
X_train, X_val  = train_test_split(X_train,      test_size=0.1, random_state=42)
u_train, s_train, vt_train, u_val, vt_val, u_test, vt_test =
 ↪perform_svd_test(X_train, X_val, X_test)
```



0.1 Accuracy Results:

For Train set the accurcy dips and then seems to increase as new features are added. The intersection of train &

The Graph is behaving oddly. For Validate & Test sets he accuracy is going down with adding more latent features

test sets suggests 450 might be optimal # of latent features to use.