PierroSarahPuriYuvrajToolTopicCode

December 6, 2024

1 Tools Topic

1.1 CSCI E-82

1.1.1 Sarah Pierro and Yuvraj Puri

December 4, 2024

2 set up

[1]:
```
pip install lightfm
```

```
Collecting lightfm
  Downloading lightfm-1.17.tar.gz (316 kB)
                             0.0/316.4
kB ? eta -:--:--
                        316.4/316.4 kB
9.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from lightfm) (1.26.4)
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.10/dist-
packages (from lightfm) (1.13.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from lightfm) (2.32.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (from lightfm) (1.5.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->lightfm) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->lightfm) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->lightfm) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->lightfm) (2024.8.30)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn->lightfm) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from scikit-learn->lightfm) (3.5.0)
Building wheels for collected packages: lightfm
  Building wheel for lightfm (setup.py) … done
  Created wheel for lightfm: filename=lightfm-1.17-cp310-cp310-linux_x86_64.whl
size=808328
sha256=2cf4cb34be9b92582b7284c56a06b4c044641ec6e5cf832649870eaf1fc2b88d
  Stored in directory: /root/.cache/pip/wheels/4f/9b/7e/0b256f2168511d8fa4dae4fa
e0200fdbd729eb424a912ad636
Successfully built lightfm
Installing collected packages: lightfm
Successfully installed lightfm-1.17
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from lightfm import LightFM
from lightfm.data import Dataset
from lightfm.evaluation import precision_at_k, auc_score, recall_at_k
# from lightfm.cross_validation import random_train_test_split
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from scipy.sparse import coo_matrix
from sklearn.feature_extraction.text import CountVectorizer
from scipy.sparse import hstack
```

3 load data

[3]:
```
music_info = pd.read_csv("Music Info.csv")
music_info.head()
```

[3]:

	track_id	name	artist	\
0	TRIOREW128F424EAF0	Mr. Brightside	The Killers	
1	TRRIVDJ128F429B0E8	Wonderwall	Oasis	
2	TROUVHL128F426C441	Come as You Are	Nirvana	
3	TRUEIND128F93038C4	Take Me Out	Franz Ferdinand	
4	TRLNZBD128F935E4D8	Creep	Radiohead	

	spotify_preview_url	spotify_id	\
0	https://p.scdn.co/mp3-preview/4d26180e6961fd46…	09ZQ5TmUG8TSL56n0knqrj	
1	https://p.scdn.co/mp3-preview/d012e536916c927b…	06UfBBDISthj1ZJAtX4xjj	
2	https://p.scdn.co/mp3-preview/a1c11bb1cb231031…	0keNu0t0tqsWtExGM3nT1D	
3	https://p.scdn.co/mp3-preview/399c401370438be4…	0ancVQ9wEcHVd0RrGICTE4	
4	https://p.scdn.co/mp3-preview/e7eb60e9466bc3a2…	01QoK9DA7VTeTSE3MNzp4I	

	tags	genre	year	duration_ms	\

```
0  rock, alternative, indie, alternative_rock, in…   NaN  2004       222200
1  rock, alternative, indie, pop, alternative_roc…   NaN  2006       258613
2   rock, alternative, alternative_rock, 90s, grunge   RnB  1991       218920
3  rock, alternative, indie, alternative_rock, in…   NaN  2004       237026
4  rock, alternative, indie, alternative_rock, in…   RnB  2008       238640

   danceability  …  key  loudness  mode  speechiness  acousticness  \
0         0.355  …    1    -4.360     1       0.0746      0.001190
1         0.409  …    2    -4.373     1       0.0336      0.000807
2         0.508  …    4    -5.783     0       0.0400      0.000175
3         0.279  …    9    -8.851     1       0.0371      0.000389
4         0.515  …    7    -9.935     1       0.0369      0.010200

   instrumentalness  liveness  valence    tempo  time_signature
0          0.000000    0.0971    0.240  148.114               4
1          0.000000    0.2070    0.651  174.426               4
2          0.000459    0.0878    0.543  120.012               4
3          0.000655    0.1330    0.490  104.560               4
4          0.000141    0.1290    0.104   91.841               4

[5 rows x 21 columns]
```

[4]:
```
user_listening_history = pd.read_csv("User Listening History.csv")
user_listening_history.head()
```

[4]:

	track_id	user_id	playcount
0	TRIRLYL128F42539D1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	1
1	TRFUPBA128F934F7E1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	1
2	TRLQPQJ128F42AA94F	b80344d063b5ccb3212f76538f3d9e43d87dca9e	1
3	TRTUCUY128F92E1D24	b80344d063b5ccb3212f76538f3d9e43d87dca9e	1
4	TRHDDQG12903CB53EE	b80344d063b5ccb3212f76538f3d9e43d87dca9e	1

4 explore data

We're using the Million Song + Spotify + LastFM dataset from Kaggle for our LightFM demo. In this section, we're exploring the data prior to modeling.

[7]:
```
print("Music Info:\n")
print(music_info.info())
print("\n\nUser Listening History:\n")
print(user_listening_history.info())
```

```
Music Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50683 entries, 0 to 50682
Data columns (total 21 columns):
 #   Column               Non-Null Count  Dtype
```

```
---  ------                --------------  -----
 0   track_id              50683 non-null  object
 1   name                  50683 non-null  object
 2   artist                50683 non-null  object
 3   spotify_preview_url   50683 non-null  object
 4   spotify_id            50683 non-null  object
 5   tags                  49556 non-null  object
 6   genre                 22348 non-null  object
 7   year                  50683 non-null  int64
 8   duration_ms           50683 non-null  int64
 9   danceability          50683 non-null  float64
 10  energy                50683 non-null  float64
 11  key                   50683 non-null  int64
 12  loudness              50683 non-null  float64
 13  mode                  50683 non-null  int64
 14  speechiness           50683 non-null  float64
 15  acousticness          50683 non-null  float64
 16  instrumentalness      50683 non-null  float64
 17  liveness              50683 non-null  float64
 18  valence               50683 non-null  float64
 19  tempo                 50683 non-null  float64
 20  time_signature        50683 non-null  int64
dtypes: float64(9), int64(5), object(7)
memory usage: 8.1+ MB
None


User Listening History:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9711301 entries, 0 to 9711300
Data columns (total 3 columns):
 #   Column     Dtype
---  ------     -----
 0   track_id   object
 1   user_id    object
 2   playcount  int64
dtypes: int64(1), object(2)
memory usage: 222.3+ MB
None
```

[8]:
```
print("Music Info:\n")
print(music_info.describe())
print("\n\nUser Listening History:\n")
print(user_listening_history.describe())
```

```
Music Info:
```

```
                year   duration_ms  danceability        energy           key  \
count  50683.000000  5.068300e+04  50683.000000  50683.000000  50683.000000
mean    2004.017323  2.511551e+05      0.493537      0.686486      5.312748
std        8.860172  1.075860e+05      0.178838      0.251808      3.568078
min     1900.000000  1.439000e+03      0.000000      0.000000      0.000000
25%     2001.000000  1.927330e+05      0.364000      0.514000      2.000000
50%     2006.000000  2.349330e+05      0.497000      0.744000      5.000000
75%     2009.000000  2.881930e+05      0.621000      0.905000      9.000000
max     2022.000000  3.816373e+06      0.986000      1.000000     11.000000

           loudness          mode   speechiness  acousticness  \
count  50683.000000  50683.000000  50683.000000  50683.000000
mean      -8.291204      0.631060      0.076023      0.213808
std        4.548365      0.482522      0.076007      0.302848
min      -60.000000      0.000000      0.000000      0.000000
25%      -10.375000      0.000000      0.035200      0.001400
50%       -7.200000      1.000000      0.048200      0.039900
75%       -5.089000      1.000000      0.083500      0.340000
max        3.642000      1.000000      0.954000      0.996000

       instrumentalness      liveness       valence         tempo  \
count      50683.000000  50683.000000  50683.000000  50683.000000
mean           0.225283      0.215425      0.433134    123.507682
std            0.337049      0.184697      0.258779     29.621125
min            0.000000      0.000000      0.000000      0.000000
25%            0.000018      0.098400      0.214000    100.683000
50%            0.005630      0.138000      0.405000    121.989000
75%            0.441000      0.289000      0.634000    141.639000
max            0.999000      0.999000      0.993000    238.895000

       time_signature
count    50683.000000
mean         3.898151
std          0.419670
min          0.000000
25%          4.000000
50%          4.000000
75%          4.000000
max          5.000000


User Listening History:

          playcount
count  9.711301e+06
mean   2.630946e+00
std    5.706324e+00
min    1.000000e+00
```

```
25%    1.000000e+00
50%    1.000000e+00
75%    2.000000e+00
max    2.948000e+03
```

[12]:
```
# Examine nonnumeric columns
print("Music Info:\n")
print(music_info.describe(include=['O']))
print("\n\nUser Listening History:\n")
print(user_listening_history.describe(include=['O']))
```

```
Music Info:

                  track_id            name              artist  \
count                50683           50683               50683
unique               50683           50683                8317
top     TRIOREW128F424EAF0  Mr. Brightside  The Rolling Stones
freq                     1               1                 132

                                    spotify_preview_url  \
count                                             50683
unique                                            50620
top     https://p.scdn.co/mp3-preview/82e186b1ddf2b2a5…
freq                                                  2

                    spotify_id     tags  genre
count                    50683    49556  22348
unique                   50674    20057     15
top     5vYA1mW9g2Coh1HUFUSmlb  country   Rock
freq                         2      506   9965


User Listening History:

                  track_id                                   user_id
count              9711301                                   9711301
unique               30459                                    962037
top     TRONYHY128F92C9D11  ec6dfcf19485cb011e0b22637075037aae34cf26
freq                 80656                                       784
```

The Music Info dataset contains ~50k songs from ~8k artists, over 15 genres.

The User Listening History dataset contains ~30k tracks (presumably a subset of the music info, but we will check) played by ~962k unique users.

[13]:
```
# check nulls
print("Music Info:\n")
print(music_info.isnull().sum())
print("\n\nUser Listening History:\n")
```

```
print(user_listening_history.isnull().sum())
```

```
Music Info:

track_id                   0
name                       0
artist                     0
spotify_preview_url        0
spotify_id                 0
tags                    1127
genre                  28335
year                       0
duration_ms                0
danceability               0
energy                     0
key                        0
loudness                   0
mode                       0
speechiness                0
acousticness               0
instrumentalness           0
liveness                   0
valence                    0
tempo                      0
time_signature             0
dtype: int64


User Listening History:

track_id     0
user_id      0
playcount    0
dtype: int64
```

[14]:
```
# Check Music Info tracks overlap with User Listening History tracks
user_listening_history['track_id'].isin(music_info['track_id']).value_counts()
```

[14]:
```
track_id
True    9711301
Name: count, dtype: int64
```

[15]:
```
music_info['track_id'] = music_info['track_id'].astype(str)
user_listening_history['track_id'] = user_listening_history['track_id'].
 ↪astype(str)

# Join music info data (track_id grain) to users' listening history (user_id␣
 ↪grain) on track_id
```

```
music_and_listeners = pd.merge(music_info, user_listening_history,␣
 ↪left_on='track_id', right_on='track_id', how='inner')

music_and_listeners.head()
```

[15]:

```
             track_id            name       artist  \
0  TRIOREW128F424EAF0  Mr. Brightside  The Killers
1  TRIOREW128F424EAF0  Mr. Brightside  The Killers
2  TRIOREW128F424EAF0  Mr. Brightside  The Killers
3  TRIOREW128F424EAF0  Mr. Brightside  The Killers
4  TRIOREW128F424EAF0  Mr. Brightside  The Killers

                                spotify_preview_url              spotify_id  \
0  https://p.scdn.co/mp3-preview/4d26180e6961fd46…  09ZQ5TmUG8TSL56n0knqrj
1  https://p.scdn.co/mp3-preview/4d26180e6961fd46…  09ZQ5TmUG8TSL56n0knqrj
2  https://p.scdn.co/mp3-preview/4d26180e6961fd46…  09ZQ5TmUG8TSL56n0knqrj
3  https://p.scdn.co/mp3-preview/4d26180e6961fd46…  09ZQ5TmUG8TSL56n0knqrj
4  https://p.scdn.co/mp3-preview/4d26180e6961fd46…  09ZQ5TmUG8TSL56n0knqrj

                                                tags genre  year  duration_ms  \
0  rock, alternative, indie, alternative_rock, in…   NaN  2004       222200
1  rock, alternative, indie, alternative_rock, in…   NaN  2004       222200
2  rock, alternative, indie, alternative_rock, in…   NaN  2004       222200
3  rock, alternative, indie, alternative_rock, in…   NaN  2004       222200
4  rock, alternative, indie, alternative_rock, in…   NaN  2004       222200

   danceability  …  mode  speechiness  acousticness  instrumentalness  \
0         0.355  …     1       0.0746       0.00119               0.0
1         0.355  …     1       0.0746       0.00119               0.0
2         0.355  …     1       0.0746       0.00119               0.0
3         0.355  …     1       0.0746       0.00119               0.0
4         0.355  …     1       0.0746       0.00119               0.0

   liveness  valence    tempo  time_signature  \
0    0.0971     0.24  148.114               4
1    0.0971     0.24  148.114               4
2    0.0971     0.24  148.114               4
3    0.0971     0.24  148.114               4
4    0.0971     0.24  148.114               4

                                  user_id  playcount
0  e95303796148a664af00f0e25059e38ab1823b30          1
1  d7a8dcae75b104184e9e243bd7d6f8c79567f286          1
2  c097ce6d176770f976b7fc98e8bcb0421b83c3b9          1
3  1c9fcbf32a26843605b2a0daf1beb42d52924903          1
4  37bb354237cf3a83cfe0d71d89b97a1d68e7d082          1
```

```
[5 rows x 23 columns]
```

[16]:
```
music_and_listeners.columns
```

[16]:
```
Index(['track_id', 'name', 'artist', 'spotify_preview_url', 'spotify_id',
       'tags', 'genre', 'year', 'duration_ms', 'danceability', 'energy', 'key',
       'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness',
       'liveness', 'valence', 'tempo', 'time_signature', 'user_id',
       'playcount'],
      dtype='object')
```

5 preprocess data

Since we have users' listening history, we can create some interests metadata. Specifically, let's list the genres they've listened to and their top 3 songs by play count.

If we want to use **genre** in our recommender system, we should figure out what to do with the nulls. Wondering if we can do without those rows entirely, or if we should do without the column.

[17]:
```
# count rows of dataframe where genre is not null
music_and_listeners['genre'].notnull().sum()
```

[17]:
```
6323150
```

[18]:
```
# drop rows without genre
music_and_listeners = music_and_listeners.dropna(subset=['genre'])
```

Starting out, we'll use a subset of the data before scaling up.

[19]:
```
# Take top users and tracks, and create a subset that pulls a bit from each of␣
 ↪these

top_users = music_and_listeners['user_id'].value_counts().head(10000).index
top_tracks = music_and_listeners['track_id'].value_counts().head(30000).index

subset_music_and_listeners = music_and_listeners[music_and_listeners['user_id'].
 ↪isin(top_users) & music_and_listeners['track_id'].isin(top_tracks)]

len(subset_music_and_listeners)
```

[19]:
```
695955
```

Let's create a user metadata feature containing genres they've listened to.

[20]:
```
subset_music_and_listeners['genre_list'] = subset_music_and_listeners.
 ↪groupby('user_id')['genre'].transform(lambda x: ','.join(x))

subset_music_and_listeners['genre_list'] =␣
 ↪subset_music_and_listeners['genre_list'].fillna('')
```

```
# Using the following code from ChatGPT to help me dedupe genres
# Create a new column for unique genres per user
def generate_user_genres(df):
    for user_id, group in df.groupby('user_id'):
        genres = sorted(set(genre.strip() for genres in group['genre_list'].
 ↪dropna() for genre in genres.split(',')))
        yield user_id, ', '.join(genres)

# Process in chunks
user_genres = pd.DataFrame(generate_user_genres(subset_music_and_listeners),␣
 ↪columns=['user_id', 'unique_genres_list'])
subset_music_and_listeners = subset_music_and_listeners.merge(user_genres,␣
 ↪on='user_id', how='left')
```

```
<ipython-input-20-cd5767c4344f>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  subset_music_and_listeners['genre_list'] =
subset_music_and_listeners.groupby('user_id')['genre'].transform(lambda x:
','.join(x))
<ipython-input-20-cd5767c4344f>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  subset_music_and_listeners['genre_list'] =
subset_music_and_listeners['genre_list'].fillna('')
```

We need to ensure unique user-item pairs. Play count tracks frequency, so we can safely drop this before engineering new user metadata features.

[21]:
```
# Drop duplicate user_id | track_id rows
subset_music_and_listeners = subset_music_and_listeners.
 ↪drop_duplicates(subset=['user_id', 'track_id'])
```

[22]:
```
len(subset_music_and_listeners)
```

[22]: 695955

[56]:
```
subset_ml = subset_music_and_listeners.copy()
```

Let's normalize the continuous features in the item metadata using minmax scalar

[24]:
```
# normalize select features
scaler = MinMaxScaler()
features_to_normalize = ['danceability', 'loudness', 'tempo', 'energy']
subset_music_and_listeners[features_to_normalize] = scaler.
 ↪fit_transform(subset_music_and_listeners[features_to_normalize])
```

[25]:
```
subset_music_and_listeners.head()
```

[25]:

	track_id	name	artist	spotify_preview_url	spotify_id	tags	genre	year	duration_ms	danceability	…	acousticness	instrumentalness	liveness	valence	tempo	time_signature	user_id	playcount	genre_list
0	TRIODZU128E078F3E2	Under the Bridge	Red Hot Chili Peppers	https://p.scdn.co/mp3-preview/90e41778392f27b6…	06zh28PcYIFvNOAz5Wq2Xb	rock, alternative, alternative_rock, 90s, funk	Pop	2003	265506	0.565306	…	0.0168	0.000534	0.136	0.513	0.382997	4	80c64182aea519818391137e81df0712126002b5	1	Pop,Rock,Rock,Folk,Rock,Pop,Electronic,Rock,Ro…
1	TRIODZU128E078F3E2	Under the Bridge	Red Hot Chili Peppers	https://p.scdn.co/mp3-preview/90e41778392f27b6…	06zh28PcYIFvNOAz5Wq2Xb	rock, alternative, alternative_rock, 90s, funk	Pop	2003	265506	0.565306	…	0.0168	0.000534	0.136	0.513	0.382997	4	3e6ef2a572d1f6f06df71bf28190eae9e1934a61	1	Pop,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Ro…
2	TRIODZU128E078F3E2	Under the Bridge	Red Hot Chili Peppers	https://p.scdn.co/mp3-preview/90e41778392f27b6…	06zh28PcYIFvNOAz5Wq2Xb	rock, alternative, alternative_rock, 90s, funk	Pop	2003	265506	0.565306	…	0.0168	0.000534	0.136	0.513	0.382997	4	638bbafa728e8865d627d7fdc48d3e1d4903f257	1	Pop,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Ro…
3	TRIODZU128E078F3E2	Under the Bridge	Red Hot Chili Peppers	https://p.scdn.co/mp3-preview/90e41778392f27b6…	06zh28PcYIFvNOAz5Wq2Xb	rock, alternative, alternative_rock, 90s, funk	Pop	2003	265506	0.565306	…	0.0168	0.000534	0.136	0.513	0.382997	4	63e7a5b340455557e4ab6fd10be9ea77db5098bb	1	
4	TRIODZU128E078F3E2	Under the Bridge	Red Hot Chili Peppers	https://p.scdn.co/mp3-preview/90e41778392f27b6…	06zh28PcYIFvNOAz5Wq2Xb	rock, alternative, alternative_rock, 90s, funk	Pop	2003	265506	0.565306	…	0.0168	0.000534	0.136	0.513	0.382997	4	e626376ea0d6e980f021160b3df07c38740c7e09	1	

```
3          2  Pop,Rock,RnB,Rock,Electronic,Rock,Electronic,E…
4          2  Pop,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Ro…

                                  unique_genres_list
0  Blues, Electronic, Folk, Jazz, Metal, New Age,…
1  Electronic, Jazz, Metal, Pop, Punk, Rap, RnB, …
2  Electronic, Folk, Latin, Metal, Pop, Reggae, R…
3  Electronic, Jazz, Metal, New Age, Pop, Rap, Re…
4        Country, Electronic, Pop, Rap, Reggae, Rock

[5 rows x 25 columns]
```

[ ]:
```
# item data

# Need to normalize as well
# duration_ms too? and year? minmax scale these?
# ['year', 'duration_ms', 'danceability', 'energy', 'loudness', 'speechiness',␣
 ↪'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo']

#subset_music_and_listeners[['track_id', 'artist', 'tags', 'genre', 'year'␣
 ↪,'danceability', 'energy', 'speechiness', 'acousticness',␣
 ↪'instrumentalness', 'liveness','valence','tempo']]
```

We're getting user-item interactions from the playcount column.

If using implicit feedback, we'll want to turn playcounts into a binary indicator of the interaction (0 is no, >=1 is yes) and use a loss like WARP or logistic.

If we want to use **explicit feedback**, we can use raw, normalized, or binned playcounts and use a loss like MSE.

[26]:
```
# Use this for implicit feedback (make all playcounts 1)
# replace playcount with a binary indicator
subset_music_and_listeners['playcount'] =␣
 ↪subset_music_and_listeners['playcount'].apply(lambda x: 1 if x >= 1 else 0)

subset_music_and_listeners['playcount'].value_counts()
```

[26]:
```
playcount
1    695955
Name: count, dtype: int64
```

We binarized the output to try implicit feedback. Any unseen user-item interactions in the above dataset should be interpreted by lightfm as a negative interaction.

First, we have to convert to a format compatible with LightFM (akin to converting to dmatrix for xgboost) and create the interaction matrix.

[27]:
```
# Map user_id and track_id to unique indices
user_id_map = {user_id: idx for idx, user_id in␣
 ↪enumerate(subset_music_and_listeners['user_id'].unique())}
track_id_map = {track_id: idx for idx, track_id in␣
 ↪enumerate(subset_music_and_listeners['track_id'].unique())}

# Add mapped indices to the dataframe
subset_music_and_listeners['user_idx'] = subset_music_and_listeners['user_id'].
 ↪map(user_id_map)
subset_music_and_listeners['track_idx'] =␣
 ↪subset_music_and_listeners['track_id'].map(track_id_map)

# Code modified from ChatGPT to make the coomatrix
# Create the interaction matrix
interactions = coo_matrix(
    (subset_music_and_listeners['playcount'],
     (subset_music_and_listeners['user_idx'],␣
 ↪subset_music_and_listeners['track_idx']))
)
```

Next, we'll prepare the user-base and item-based metadata features, which is what makes this a hybrid recommender system.

[28]:
```
# Vectorize the user features
# I had a conversation with ChatGPT on vectorizing the features, so this code␣
 ↪is modified from that.
# genres users have listened to is our user-based metadata
vectorizer = CountVectorizer(tokenizer=lambda x: x.split(', '))
user_features = vectorizer.fit_transform(subset_music_and_listeners.
 ↪drop_duplicates('user_id')['unique_genres_list'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:521:
UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is
not None'
  warnings.warn(
```

[29]:
```
# Item metadata
item_metadata = subset_music_and_listeners[['track_id', 'tags', 'genre',␣
 ↪'danceability', 'energy']].drop_duplicates('track_id')

# Vectorize tags and genres
tags_vectorized = vectorizer.fit_transform(item_metadata['tags'].fillna(''))
genres_vectorized = vectorizer.fit_transform(item_metadata['genre'].fillna(''))

# Combine all item features (numeric + vectorized)
item_features = hstack([tags_vectorized, genres_vectorized,␣
 ↪item_metadata[['danceability', 'energy']].values])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:521:
UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is
not None'
  warnings.warn(
```

6 split data

Split the data into train, validation, and test sets.

[30]:
```
# reserving 60% for train and 20% for both val and test sets
train_data, test_data = train_test_split(subset_music_and_listeners,␣
 ↪test_size=0.2, random_state=42)
train_data, val_data = train_test_split(train_data, test_size=0.25,␣
 ↪random_state=42)

# Create sparse matrices for train, val, test
train_interactions = coo_matrix(
    (train_data['playcount'], (train_data['user_idx'],␣
 ↪train_data['track_idx'])),
    shape=interactions.shape
)
val_interactions = coo_matrix(
    (val_data['playcount'], (val_data['user_idx'], val_data['track_idx'])),
    shape=interactions.shape
)
test_interactions = coo_matrix(
    (test_data['playcount'], (test_data['user_idx'], test_data['track_idx'])),
    shape=interactions.shape
)
```

[32]:
```
print(f"Train size: {len(train_data)}")
print(f"Validate size: {len(val_data)}")
print(f"Test size: {len(test_data)}")
```

```
Train size: 417573
Validate size: 139191
Test size: 139191
```

7 LightFM demo

LightFM requires interaction data and metadata features in a sparse matrix format, which we've already handled. Now, we can initialize the recommender system. We'll start by using WARP loss.

7.1 train

Build the model

[33]:
```
# Initialize the model
model = LightFM(loss='warp', no_components=100)

# Fit the model
model.fit(
    train_interactions,
    user_features=user_features,
    item_features=item_features,
    epochs=30,
    num_threads=4
)
```

[33]: <lightfm.lightfm.LightFM at 0x78187d7af400>

7.2 validate

[34]:
```
# Evaluate on val set for top 5 recommendations
val_auc = auc_score(
    model, val_interactions, user_features=user_features,␣
 ↪item_features=item_features
).mean()

print(f"Validation AUC: {val_auc:.4f}")
```

```
Validation AUC: 0.8007
```

7.3 Make recommendations for a given user

[35]:
```
# Find a user!
test_data.head()
```

[35]:

	track_id	name	artist	spotify_preview_url	spotify_id
695075	TRRNMCY128E078E5F7	Jails And Bombs	Amos Lee	https://p.scdn.co/mp3-preview/43b7243096b2ffd8…	0pOGsIEVyVyMs7Z2oOrWcU
443998	TRGBQBV128F9344C34	Celestial Crown	The Sword	https://p.scdn.co/mp3-preview/78e0f2f42563e281…	0I4hmroT87YMn2nJTjPG6a
422223	TRXOEPI128EF35353F	Just an Illusion	Imagination	https://p.scdn.co/mp3-preview/30895ac2d9196331…	
337125	TRFIIHS128F427EF9D	On the Bus Mall	The Decemberists	https://p.scdn.co/mp3-preview/fd1dc366d55dc949…	
79112	TRLIDNT128F930327C	Dull Life	Yeah Yeah Yeahs	https://p.scdn.co/mp3-preview/83f01f361cfa04b4…	

```
422223  01ZMt4u9VTEX0QEefb2k6N
337125  12R4KTIkIv3PkBTQNkcbO4
79112   1pvlznzPWn4XiPzQzrrdIU

                                                    tags  genre  year  \
695075                           jazz, folk, soul, chill   Folk  2005
443998         metal, heavy_metal, soundtrack, doom_metal  Metal  2006
422223                            dance, 80s, soul, funk   Rock  2012
337125  indie, folk, acoustic, indie_pop, beautiful, m…   Rock  2005
79112   rock, alternative, indie, female_vocalists, al…   Rock  2009

        duration_ms  danceability  …  liveness  valence     tempo  \
695075       251200      0.766327  …    0.3860    0.247  0.562668
443998       117640      0.134694  …    0.2790    0.202  0.456151
422223       388640      0.770408  …    0.0455    0.633  0.477761
337125       364413      0.276531  …    0.2940    0.367  0.432747
79112        248386      0.393878  …    0.0880    0.207  0.486146

        time_signature                                 user_id  playcount  \
695075               4  ff0a796eed4f91150a08d48aaa115308cc14ca8e          1
443998               4  bdfac34b2c85a695713cc728e7d859c2bf5b4962          1
422223               4  d6f01c0697732308a8a635c40a4899f80d121066          1
337125               4  7991284f490649f97528471e5803762d7d3d10db          1
79112                4  2c722b0d57e847b366b78f4573531cddeaa44dc3          1

                                                genre_list  \
695075  Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,R…
443998  Rock,Rock,Rock,Rock,Rock,Metal,Rock,Rock,Metal…
422223  Rock,Electronic,Rock,Rock,Electronic,Electroni…
337125  Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…
79112   Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,R…

                                    unique_genres_list  user_idx  track_idx
695075  Country, Electronic, Folk, Metal, Pop, Rap, Rock      9617      14210
443998                              Metal, Punk, Rock       738       5344
422223                      Electronic, Pop, Rap, Rock      6998       4576
337125                     Electronic, Folk, Pop, Rock      7768       3279
79112        Country, Electronic, Folk, Punk, Rap, Rock      4153        404

[5 rows x 27 columns]
```

[39]:
```
# subset_music_and_listeners.where(subset_music_and_listeners['user_id'] ==␣
↪'bdfac34b2c85a695713cc728e7d859c2bf5b4962').dropna().reset_index()
```

[38]:
```
# subset_music_and_listeners.where(subset_music_and_listeners['user_id'] ==␣
↪'ff0a796eed4f91150a08d48aaa115308cc14ca8e').dropna().reset_index()
```

Let's look at the user's listening history (rows in the original dataset)

[37]:
```
subset_music_and_listeners.where(subset_music_and_listeners['user_id'] ==␣
↪'7991284f490649f97528471e5803762d7d3d10db').dropna().reset_index()
```

[37]:

	index	track_id	name	artist	spotify_preview_url	spotify_id
0	23421	TRWJYOT128F935572B	In the Aeroplane Over the Sea	Neutral Milk Hotel	https://p.scdn.co/mp3-preview/23f5b39b7eb6b130…	3xEbrpJTTcsaWMX4gHKd4O
1	40068	TRRWJLU128F92F9912	My Boy Builds Coffins	Florence + the Machine	https://p.scdn.co/mp3-preview/c973d886c10a763c…	1KxcbEY0asYVMR043MjZWq
2	44337	TROUKLP12903CB6364	White Sky	Vampire Weekend	https://p.scdn.co/mp3-preview/3d58c03e6fc6ef30…	0dAp6Ptz8UlHxZ9uaydqMR
3	45781	TRBVNWT128F93173BA	Help I'm Alive	Metric	https://p.scdn.co/mp3-preview/757d19768368f201…	0VERkYcvGRi1PJwrroc1OB
4	47587	TRTNFRQ12903CB6360	Horchata	Vampire Weekend	https://p.scdn.co/mp3-preview/ca1b4c65328b1746…	
..	…	…	…	…	…	
131	685473	TROPNKC128F42A52DD	Entering White Cecilia	The New Pornographers	https://p.scdn.co/mp3-preview/ab66940b6a71d809…	
132	686596	TRBCJAL128F429EA70	Ballad of a Comeback Kid	The New Pornographers	https://p.scdn.co/mp3-preview/cb7f7094c5b93588…	
133	686927	TRHHMHF128F423F004	Have to Explode	The Mountain Goats	https://p.scdn.co/mp3-preview/df37c48db6b7e66b…	
134	693500	TRAVRKY128F429EA68	Loose Translation	The New Pornographers	https://p.scdn.co/mp3-preview/152b87cb5edf228d…	
135	693964	TRJNMNC128F427ED16	Your Secrets	Belle and Sebastian	https://p.scdn.co/mp3-preview/bd4b8406b11ab7a5…	

```
4    22d5vvCijMTue7PvUrGiz9
..                      …
131  3eNTBstSSJ9oD2q5XR5eNJ
132  4sFpslecNLE12QU9r4xtbC
133  2iEcCqHFQFh91nl0IDYDd1
134  5Qu79Xa89JOQyZgMKWYtRk
135  0FUpLggWrnfOzlYa5tLiLZ

                                                  tags genre    year  \
0    rock, alternative, indie, folk, indie_rock, 90…  Rock  1998.0
1    rock, alternative, indie, pop, female_vocalist…  Rock  2009.0
2    rock, electronic, alternative, indie, pop, ind…  Rock  2010.0
3    rock, alternative, indie, female_vocalists, in…  Rock  2010.0
4    rock, alternative, indie, pop, indie_rock, ame…  Rock  2010.0
..                                                 …   …     …
131                                indie, indie_rock  Rock  2007.0
132                      rock, indie, pop, indie_rock  Rock  2003.0
133                           indie, folk, indie_rock  Rock  2002.0
134                            indie, pop, indie_rock  Rock  2003.0
135                                             indie  Rock  2004.0

     duration_ms  …  liveness  valence     tempo  time_signature  \
0       202560.0  …    0.1140    0.222  0.425948             3.0
1       176733.0  …    0.0847    0.188  0.554633             4.0
2       178666.0  …    0.3260    0.944  0.463636             3.0
3       290194.0  …    0.4430    0.225  0.526429             4.0
4       206733.0  …    0.3560    0.837  0.545244             4.0
..           …  …       …       …        …                …
131     208760.0  …    0.2770    0.828  0.594157             4.0
132     231093.0  …    0.2360    0.743  0.636709             4.0
133     201506.0  …    0.1110    0.286  0.613227             4.0
134     179600.0  …    0.2310    0.813  0.572721             4.0
135     191706.0  …    0.1120    0.847  0.520489             4.0

                                      user_id  playcount  \
0    7991284f490649f97528471e5803762d7d3d10db        1.0
1    7991284f490649f97528471e5803762d7d3d10db        1.0
2    7991284f490649f97528471e5803762d7d3d10db        1.0
3    7991284f490649f97528471e5803762d7d3d10db        1.0
4    7991284f490649f97528471e5803762d7d3d10db        1.0
..                                        …        …
131  7991284f490649f97528471e5803762d7d3d10db        1.0
132  7991284f490649f97528471e5803762d7d3d10db        1.0
133  7991284f490649f97528471e5803762d7d3d10db        1.0
134  7991284f490649f97528471e5803762d7d3d10db        1.0
135  7991284f490649f97528471e5803762d7d3d10db        1.0
```

```
                                                  genre_list  \
0    Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…
1    Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…
2    Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…
3    Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…
4    Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…
..                                                 …
131  Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…
132  Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…
133  Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…
134  Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…
135  Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,E…

              unique_genres_list  user_idx  track_idx
0    Electronic, Folk, Pop, Rock    7768.0       91.0
1    Electronic, Folk, Pop, Rock    7768.0      192.0
2    Electronic, Folk, Pop, Rock    7768.0      212.0
3    Electronic, Folk, Pop, Rock    7768.0      215.0
4    Electronic, Folk, Pop, Rock    7768.0      218.0
..                           …       …         …
131  Electronic, Folk, Pop, Rock    7768.0    13782.0
132  Electronic, Folk, Pop, Rock    7768.0    13787.0
133  Electronic, Folk, Pop, Rock    7768.0    13790.0
134  Electronic, Folk, Pop, Rock    7768.0    14173.0
135  Electronic, Folk, Pop, Rock    7768.0    14176.0

[136 rows x 28 columns]
```

Make recommendations for this user

[40]:
```python
# Predict scores for all tracks
user_id = 7768
scores = model.predict(
    user_id,
    np.arange(interactions.shape[1]),
    user_features=user_features,
    item_features=item_features
)

# Get top 5 recommendations
top_items = np.argsort(-scores)[:5]

# Map indices back to track_id
track_mapping = {v: k for k, v in track_id_map.items()}
recommended_tracks = [track_mapping[idx] for idx in top_items]

# join the recommended tracks with the track metadata
```

```
recommended_tracks_meta = subset_music_and_listeners[['track_id', 'name',␣
 ↪'artist', 'genre', 'danceability', 'loudness', 'tempo', 'energy']].
 ↪where(subset_music_and_listeners['track_id'].isin(recommended_tracks)).
 ↪dropna().drop_duplicates('track_id').reset_index()

recommended_tracks_meta
```

[40]:

	index	track_id	name	artist	genre	danceability	loudness	tempo	energy
0	120818	TRZKOKE128E0786527	Earthquake Weather	Beck	Rock	0.805102	0.743970	0.665476	0.636993
1	128190	TRXMILX128F147DF70	Nostrand	Ratatat	Electronic	0.724490	0.740579	0.772256	0.374987
2	241980	TROJVOC128F92EC9F4	Bird Song	Florence + the Machine	Folk	0.457143	0.793854	0.550988	0.513990
3	289631	TRAXREG128F4281B39	Out of Egypt, Into the Great Laugh of Mankind,…	Sufjan Stevens	Folk	0.638776	0.618891	0.513700	0.555991
4	316449	TROIVZG128F9326122	San Francisco	Foxygen	Folk	0.635714	0.749671	0.530506	0.670993

7.3.1 Qualitative assessment

[41]:
```
# get the user's playcount for each genre
subset_music_and_listeners[subset_music_and_listeners['user_idx'] == user_id].
 ↪groupby('genre')['playcount'].sum()
```

[41]:
```
genre
Electronic    40
Folk           4
Pop            1
Rock          91
Name: playcount, dtype: int64
```

7.3.2 Quantitative assessment

[42]:
```
# Evaluate on test set
test_auc = auc_score(
    model, test_interactions, user_features=user_features,␣
 ↪item_features=item_features
```

```
).mean()

print(f"Test AUC: {test_auc:.4f}")
```

```
Test AUC: 0.8006
```

8 NEW MODEL - using almost all features

[57]:
```
subset_ml.head()
```

[57]:

	track_id	name	artist	spotify_preview_url	spotify_id	tags	genre	year	duration_ms	danceability	…	liveness	valence	tempo	time_signature	user_id	playcount	genre_list
0	TRIODZU128E078F3E2	Under the Bridge	Red Hot Chili Peppers	https://p.scdn.co/mp3-preview/90e41778392f27b6…	06zh28PcYIFvNOAz5Wq2Xb	rock, alternative, alternative_rock, 90s, funk	Pop	2003	265506	0.565306	…	0.136	0.513	0.382997	4	80c64182aea519818391137e81df0712126002b5	1	Pop,Rock,Rock,Folk,Rock,Pop,Electronic,Rock,Ro…
1	TRIODZU128E078F3E2	Under the Bridge	Red Hot Chili Peppers	https://p.scdn.co/mp3-preview/90e41778392f27b6…	06zh28PcYIFvNOAz5Wq2Xb	rock, alternative, alternative_rock, 90s, funk	Pop	2003	265506	0.565306	…	0.136	0.513	0.382997	4	3e6ef2a572d1f6f06df71bf28190eae9e1934a61	1	
2	TRIODZU128E078F3E2	Under the Bridge	Red Hot Chili Peppers	https://p.scdn.co/mp3-preview/90e41778392f27b6…	06zh28PcYIFvNOAz5Wq2Xb	rock, alternative, alternative_rock, 90s, funk	Pop	2003	265506	0.565306	…	0.136	0.513	0.382997	4	638bbafa728e8865d627d7fdc48d3e1d4903f257	1	
3	TRIODZU128E078F3E2	Under the Bridge	Red Hot Chili Peppers	https://p.scdn.co/mp3-preview/90e41778392f27b6…	06zh28PcYIFvNOAz5Wq2Xb	rock, alternative, alternative_rock, 90s, funk	Pop	2003	265506	0.565306	…	0.136	0.513	0.382997	4	63e7a5b340455557e4ab6fd10be9ea77db5098bb	1	
4	TRIODZU128E078F3E2	Under the Bridge	Red Hot Chili Peppers	https://p.scdn.co/mp3-preview/90e41778392f27b6…	06zh28PcYIFvNOAz5Wq2Xb	rock, alternative, alternative_rock, 90s, funk	Pop	2003	265506	0.565306	…	0.136	0.513	0.382997	4	e626376ea0d6e980f021160b3df07c38740c7e09	1	

```
1  Pop,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Ro…
2  Pop,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Ro…
3  Pop,Rock,RnB,Rock,Electronic,Rock,Electronic,E…
4  Pop,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Ro…

                                unique_genres_list  user_idx  track_idx
0  Blues, Electronic, Folk, Jazz, Metal, New Age,…         0          0
1  Electronic, Jazz, Metal, Pop, Punk, Rap, RnB, …         1          0
2  Electronic, Folk, Latin, Metal, Pop, Reggae, R…         2          0
3  Electronic, Jazz, Metal, New Age, Pop, Rap, Re…         3          0
4        Country, Electronic, Pop, Rap, Reggae, Rock         4          0

[5 rows x 27 columns]
```

[58]:
```
# normalize select features
scaler1 = MinMaxScaler()
ftn = ['year', 'duration_ms', 'danceability', 'energy', 'loudness',␣
 ↪'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence',␣
 ↪'tempo']

subset_ml[ftn] = scaler1.fit_transform(subset_ml[ftn])
```

[47]:
```
subset_ml.head()
```

[47]:
```
             track_id              name                 artist  \
0  TRIODZU128E078F3E2  Under the Bridge  Red Hot Chili Peppers
1  TRIODZU128E078F3E2  Under the Bridge  Red Hot Chili Peppers
2  TRIODZU128E078F3E2  Under the Bridge  Red Hot Chili Peppers
3  TRIODZU128E078F3E2  Under the Bridge  Red Hot Chili Peppers
4  TRIODZU128E078F3E2  Under the Bridge  Red Hot Chili Peppers

                                spotify_preview_url              spotify_id  \
0  https://p.scdn.co/mp3-preview/90e41778392f27b6…  06zh28PcYIFvNOAz5Wq2Xb
1  https://p.scdn.co/mp3-preview/90e41778392f27b6…  06zh28PcYIFvNOAz5Wq2Xb
2  https://p.scdn.co/mp3-preview/90e41778392f27b6…  06zh28PcYIFvNOAz5Wq2Xb
3  https://p.scdn.co/mp3-preview/90e41778392f27b6…  06zh28PcYIFvNOAz5Wq2Xb
4  https://p.scdn.co/mp3-preview/90e41778392f27b6…  06zh28PcYIFvNOAz5Wq2Xb

                                          tags genre      year  \
0  rock, alternative, alternative_rock, 90s, funk   Pop  0.858333
1  rock, alternative, alternative_rock, 90s, funk   Pop  0.858333
2  rock, alternative, alternative_rock, 90s, funk   Pop  0.858333
3  rock, alternative, alternative_rock, 90s, funk   Pop  0.858333
4  rock, alternative, alternative_rock, 90s, funk   Pop  0.858333

   duration_ms  danceability  …  acousticness  instrumentalness  liveness  \
0     0.134514      0.565306  …      0.016867          0.000535  0.127442
```

```
1     0.134514      0.565306  …      0.016867          0.000535  0.127442
2     0.134514      0.565306  …      0.016867          0.000535  0.127442
3     0.134514      0.565306  …      0.016867          0.000535  0.127442
4     0.134514      0.565306  …      0.016867          0.000535  0.127442

    valence     tempo  time_signature  \
0  0.519757  0.382997               4
1  0.519757  0.382997               4
2  0.519757  0.382997               4
3  0.519757  0.382997               4
4  0.519757  0.382997               4

                                    user_id  playcount  \
0  80c64182aea519818391137e81df0712126002b5          1
1  3e6ef2a572d1f6f06df71bf28190eae9e1934a61          1
2  638bbafa728e8865d627d7fdc48d3e1d4903f257          1
3  63e7a5b340455557e4ab6fd10be9ea77db5098bb          2
4  e626376ea0d6e980f021160b3df07c38740c7e09          2

                                          genre_list  \
0  Pop,Rock,Rock,Folk,Rock,Pop,Electronic,Rock,Ro…
1  Pop,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Ro…
2  Pop,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Ro…
3  Pop,Rock,RnB,Rock,Electronic,Rock,Electronic,E…
4  Pop,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Rock,Ro…

                                unique_genres_list
0  Blues, Electronic, Folk, Jazz, Metal, New Age,…
1  Electronic, Jazz, Metal, Pop, Punk, Rap, RnB, …
2  Electronic, Folk, Latin, Metal, Pop, Reggae, R…
3  Electronic, Jazz, Metal, New Age, Pop, Rap, Re…
4        Country, Electronic, Pop, Rap, Reggae, Rock

[5 rows x 25 columns]
```

[59]:
```python
# Map user_id and track_id to unique indices
user_id_map1 = {user_id: idx for idx, user_id in enumerate(subset_ml['user_id'].
 ↪unique())}
track_id_map1 = {track_id: idx for idx, track_id in␣
 ↪enumerate(subset_ml['track_id'].unique())}

# Add mapped indices to the dataframe
subset_ml['user_idx'] = subset_ml['user_id'].map(user_id_map1)
subset_ml['track_idx'] = subset_ml['track_id'].map(track_id_map1)

# Code modified from ChatGPT to make the coomatrix
# Create the interaction matrix
```

```
interactions = coo_matrix(
    (subset_ml['playcount'],
     (subset_ml['user_idx'], subset_ml['track_idx']))
)
```

[60]:
```
# Vectorize the user features
# I had a conversation with ChatGPT on vectorizing the features, so this code␣
 ↪is modified from that.
# genres users have listened to is our user-based metadata
vectorizer1 = CountVectorizer(tokenizer=lambda x: x.split(', '))
user_features1 = vectorizer1.fit_transform(subset_ml.
 ↪drop_duplicates('user_id')['unique_genres_list'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:521:
UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is
not None'
  warnings.warn(
```

[61]:
```
# Item metadata
item_metadata1 = subset_ml[['track_id', 'tags', 'genre', 'year', 'duration_ms',␣
 ↪'danceability', 'energy', 'loudness', 'speechiness', 'acousticness',␣
 ↪'instrumentalness', 'liveness', 'valence', 'tempo'
]].drop_duplicates('track_id')
'year', 'duration_ms', 'danceability', 'energy', 'loudness', 'speechiness',␣
 ↪'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo'
# Vectorize tags and genres
tags_vectorized1 = vectorizer1.fit_transform(item_metadata1['tags'].fillna(''))
genres_vectorized1 = vectorizer1.fit_transform(item_metadata1['genre'].
 ↪fillna(''))

# Combine all item features (numeric + vectorized)
item_features1 = hstack([tags_vectorized1, genres_vectorized1,␣
 ↪item_metadata1[['year', 'duration_ms', 'danceability', 'energy', 'loudness',␣
 ↪'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence',␣
 ↪'tempo'
]].values])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:521:
UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is
not None'
  warnings.warn(
```

[62]:
```
# reserving 60% for train and 20% for both val and test sets
train_data, test_data = train_test_split(subset_ml, test_size=0.2,␣
 ↪random_state=42)
train_data, val_data = train_test_split(train_data, test_size=0.25,␣
 ↪random_state=42)
```

```
# Create sparse matrices for train, val, test
train_interactions = coo_matrix(
    (train_data['playcount'], (train_data['user_idx'],␣
 ↪train_data['track_idx'])),
    shape=interactions.shape
)
val_interactions = coo_matrix(
    (val_data['playcount'], (val_data['user_idx'], val_data['track_idx'])),
    shape=interactions.shape
)
test_interactions = coo_matrix(
    (test_data['playcount'], (test_data['user_idx'], test_data['track_idx'])),
    shape=interactions.shape
)
```

[63]:
```
# Initialize the model
model = LightFM(loss='warp', no_components=100)

# Fit the model
model.fit(
    train_interactions,
    user_features=user_features1,
    item_features=item_features1,
    epochs=30,
    num_threads=4
)
```

[63]:
```
<lightfm.lightfm.LightFM at 0x78187ba63220>
```

[64]:
```
# Evaluate on val set
val_auc = auc_score(
    model, val_interactions, user_features=user_features1,␣
 ↪item_features=item_features1
).mean()

print(f"Validation AUC: {val_auc:.4f}")
```

```
Validation AUC: 0.8013
```

[65]:
```
# Evaluate on test set
test_auc = auc_score(
    model, test_interactions, user_features=user_features1,␣
 ↪item_features=item_features1
).mean()

print(f"Test AUC: {test_auc:.4f}")
```

```
Test AUC: 0.8013
```