Data and Web Mining

Project Report

Music Recommendation System

TEAM MEMBERS

D Ravi Ram Karthik (AP19110010436)
T Dhinesh Kumar (AP19110010479)
P Prudhvi (AP19110010419)
K Pranay (AP19110010349)
Anishka Chauhan (AP19110010518)

Abstract

A music recommendation system helps a user find songs suited to the user's music taste. In this day and age, a music connoisseur has to choose from the millions of songs available which is a problem so music streaming services like Apple Music, Spotify, and YouTube Music provide users with personalized songs using similar recommendation engines. In our project, we have used Apriori Algorithm on a huge song dataset to recommend songs to users based on their artist's preferences.

Introduction

Thanks to the rapid growth of mobile devices and the internet, innumerable music resources are available to us with a single click. The number of songs available in these humongous online music libraries is way ahead of anyone's ability to listen to them all. People find it tough to choose from millions of tunes at times. Furthermore, music service providers want an efficient method of managing songs and assisting their customers in discovering music through quality recommendations. As a result, the need for an effective recommendation system is critical.

A music recommendation system is one that learns from a user's previous listening experience and suggests tracks that they might enjoy listening to, in the future. We used a variety of algorithms to try to create a useful recommender system. We started with a popularity-based paradigm that was straightforward and intuitive. Collaborative filtering algorithms are also implemented, which forecast (filter) a user's liking by collecting preferences and tastes from many other users (collaborating). We have used Apriori Algorithm to narrow down to final recommendations in the music recommendation system that we have developed in this project.

Where to get the data from?

1. Spotify gives access to its dataset which has more than 100,000 songs and includes over 30 parameters for each song such as mood, context, segments, etc.
2. We can get the dataset from Kaggle which has a "Million Song Dataset" that offers over a million songs of data with various parameters.

Collaborative Filtering:

Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. The system generates recommendations only by using information about rating profiles for different users or items. By locating peer users/items with a rating history similar to the current user or item, it generates recommendations using this neighborhood. We have implemented an item-based collaborative filtering model. Listen count parameter is used as implicit feedback for training. To calculate the similarity between two items, we look into the set of items the target user has rated and compute how similar they are to the target item I, and then select K most similar item. The similarity between two items is calculated by taking the ratings of the users who have rated both the items and thereafter using the cosine similarity function as in.

Association Rules

Association rules are "if-then" statements that illustrate the likelihood of associations between data items in huge data sets in a variety of databases. Association rule mining is extensively used to uncover sales connections in transactional data or medical data sets, and it has a variety of uses.

Working:

At its most basic level, association rule mining entails applying machine learning models to analyze data in a database for patterns, or co-occurrences. It looks for common if-then relationships, which are the laws of association.

An association rule has two parts: an antecedent (if) and a consequence (if) (then). An antecedent is a piece of data that appears in the data set. A consequent is an object that is experienced in conjunction with the antecedent.

Scanning data for common if-then patterns and using the criteria support and confidence to locate them yields the most important correlations. Support indicates the frequency with which the items appear in the data. The number of times the if-then propositions are determined to be true is referred to as the number of times the if-then propositions are determined to be true.

Itemsets, which are made up of two or more things, are used to calculate association rules. If rules are created by examining all potential itemsets, there may be so many rules that they are meaningless. As a result, association rules are frequently derived from rules that are well-represented in data.

AIS, SETM, Apriori, and variants of the latter are examples of popular algorithms that use association rules.

Candidate itemsets are generated utilizing only the large itemsets from the previous pass with the Apriori algorithm. The preceding pass's huge itemset is linked with itself to generate all itemsets with a size greater than one. After that, each created item set with a small portion is removed. The candidates are the remaining itemsets. Any subset of a frequent itemset is considered a frequent itemset by the Apriori algorithm.

Apriori Algorithm

This useful algorithm is used to calculate the association rules between objects. It basically concludes how two or more objects are related to one another. In other words, we can say that the apriori algorithm is an association rule learning that analyzes that people who bought product A also bought product B.

The apriori algorithm's main goal is to construct an association rule between different things. The association rule outlines the relationship between two or more items. Frequent pattern mining is another name for the Apriori algorithm. In most cases, the Apriori algorithm is used on a database with a large number of transactions.

Proposed Model

We'll select a music and find how many users are listening to the same music. After finding out we'll be filtering out frequent item sets using apriori algorithm. The frequent item set having lift ,support and confidence more than or equal to the minimum number will be the output excluding the given song. We'll be using association also to find the frequent item set.

Process

- Data Preprocessing

Code:

```
In [ ]: print("The column names of the dataframe are: ",data.columns)
        data.describe()
```

Printing the columns and describing the characteristics like mean std etc.

Output:

Out[32]:

	user
count	1000.00000
mean	34.36900
std	18.53468
min	1.00000
25%	20.00000
50%	35.00000
75%	47.00000
max	69.00000

Code:

Checking the null values

```
In [ ]: data.isnull().sum() #prints the sum of null values
```

```
Out[33]: user        0
         artist      8
         sex         3
         country     7
         dtype: int64
```

Code:

```
In [ ]: encoder = ce.OrdinalEncoder(cols=['country']) # helps the data to tran
        data = encoder.fit_transform(data)
        print(data)
```

Importing ordinal and fitting the data to predict

Output:

```
     user                        artist sex  country
0       1         red hot chili peppers   f        1
1       1        the black dahlia murder   f        1
2       1                      goldfrapp   f        1
3       1               dropkick murphys   f        1
4       1                       le tigre   f        1
..    ...                            ...  ..      ...
995    67                 sufjan stevens   f        2
996    67                   beastie boys   f        2
997    67   creedence clearwater revival   f        2
998    69                      ghostface   m        2
999    69            explosions in the sky   m        2

[1000 rows x 4 columns]
```

Code:

```
In [ ]: data['country']=data['country'].fillna(data['country'].mode()) #filling most probable value for country
```

```
In [ ]: data = data.drop('sex',axis=1)
```

```
In [ ]: data= data.dropna(subset=['artist'])
```

```
In [ ]: data.head()
```

Filling null values in country with mode and dropping the sec column, lastly removing remaining null values.

Output:

Out[38]:

	user	artist	country
0	1	red hot chili peppers	1
1	1	the black dahlia murder	1
2	1	goldfrapp	1
3	1	dropkick murphys	1
4	1	le tigre	1

Code:

```
In [ ]: data = data.drop('country',axis=1)
```

```
In [ ]: columnsname = list(data["artist"].unique())
        username = list(data["user"].unique())
```

```
In [ ]: newData=pd.DataFrame(columns=columnsname,index=username)
        newData.reset_index(inplace=True)
        newData.head()
```

Dropping the country column getting uniques from artist and user and making the dataframe.

Output:

Out[42]:

	index	red hot chili peppers	the black dahlia murder	goldfrapp	dropkick murphys	le tigre	schandmaul	edguy	jack johnson	eluveitie	...	underworld	joni mitchell	frou frou	dido	spoon	the jimi hendrix experience	elliott smith
0	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 545 columns

Code:

```
In [ ]: transactions = []
        for i in data['user'].unique():
            transactions.append(list(data[data['user'] == i]['artist'].values))
```

```
In [ ]: x=len(columnsname)
        y=len(username)
        for i in range(y):
            for j in range(x):
                if columnsname[j] in transactions[i]:
                    newData.iloc[i,j]="True"
                else:
                    newData.iloc[i,j]="False"
```

```
In [ ]: print(newData)
```

Appending the songs to the list which were listened by the user

Output:

```
     index red hot chili peppers the black dahlia murder goldfrapp  \
0     True                 True                   True      True
1    False                False                   True     False
2    False                False                  False     False
3    False                False                  False     False
4    False                False                  False     False
5    False                False                  False     False
6    False                False                  False     False
7    False                False                  False     False
8    False                False                  False     False
9    False                False                  False     False
10   False                False                  False     False
11   False                False                  False     False
12   False                False                  False     False
13   False                False                  False     False
14   False                False                  False     False
15   False                False                  False     False
16   False                False                  False     False
17   False                False                  False      True
```

- **Data Mining**

Code:

```
frequent_items=apriori(newData,min_support=0.07,use_colnames=True)
print("Total number of frequent items with support more than 0.07 is {}".format(len(frequent_items)))
frequent_items
```

This filters out the frequent itemsets which crosses the minimum support 0.07

Output:

```
Total number of frequent items with support more than 0.07 is 67
```

	support	itemsets
0	0.104167	(index)
1	0.083333	(eluveitie)
2	0.187500	(guano apes)
3	0.083333	(the rolling stones)
4	0.104167	(aphex twin)
...	...	...
62	0.083333	(n*e*r*d, crystal castles)
63	0.083333	(paul mccartney, blur)
64	0.083333	(doves, blur)
65	0.083333	(paul mccartney, doves)
66	0.083333	(paul mccartney, doves, blur)

67 rows × 2 columns

Code:

```
rules=association_rules(frequent_items,metric="confidence",min_threshold=0.3)
print(rules)
```

This filters out the item set based on minimum confidence 0.3

Output:

```
               antecedents              consequents  antecedent support  \
0             (guano apes)          (fleetwood mac)            0.187500
1          (fleetwood mac)             (guano apes)            0.125000
2             (neil young)             (guano apes)            0.104167
3             (guano apes)             (neil young)            0.187500
4            (tenacious d)                    (air)            0.125000
5                    (air)            (tenacious d)            0.145833
6            (max richter)            (frank zappa)            0.250000
7            (frank zappa)            (max richter)            0.104167
8              (blink-182)            (max richter)            0.104167
9            (max richter)              (blink-182)            0.250000
10               (n*e*r*d)        (crystal castles)            0.083333
11       (crystal castles)                (n*e*r*d)            0.166667
12         (paul mccartney)                   (blur)            0.104167
13                  (blur)         (paul mccartney)            0.083333
14                 (doves)                   (blur)            0.083333
15                  (blur)                  (doves)            0.083333
16         (paul mccartney)                  (doves)            0.104167
17                 (doves)         (paul mccartney)            0.083333
18  (paul mccartney, doves)                   (blur)            0.083333
19   (paul mccartney, blur)                  (doves)            0.083333
20            (blur, doves)         (paul mccartney)            0.083333
21         (paul mccartney)            (blur, doves)            0.104167
22                 (doves)   (paul mccartney, blur)            0.083333
23                  (blur)  (paul mccartney, doves)            0.083333
```

```
    consequent support   support  confidence       lift  leverage  conviction
0             0.125000  0.083333    0.444444   3.555556  0.059896    1.575000
1             0.187500  0.083333    0.666667   3.555556  0.059896    2.437500
2             0.187500  0.104167    1.000000   5.333333  0.084635         inf
3             0.104167  0.104167    0.555556   5.333333  0.084635    2.015625
4             0.145833  0.083333    0.666667   4.571429  0.065104    2.562500
5             0.125000  0.083333    0.571429   4.571429  0.065104    2.041667
6             0.104167  0.083333    0.333333   3.200000  0.057292    1.343750
7             0.250000  0.083333    0.800000   3.200000  0.057292    3.750000
8             0.250000  0.104167    1.000000   4.000000  0.078125         inf
9             0.104167  0.104167    0.416667   4.000000  0.078125    1.535714
10            0.166667  0.083333    1.000000   6.000000  0.069444         inf
11            0.083333  0.083333    0.500000   6.000000  0.069444    1.833333
12            0.083333  0.083333    0.800000   9.600000  0.074653    4.583333
13            0.104167  0.083333    1.000000   9.600000  0.074653         inf
14            0.083333  0.083333    1.000000  12.000000  0.076389         inf
15            0.083333  0.083333    1.000000  12.000000  0.076389         inf
16            0.083333  0.083333    0.800000   9.600000  0.074653    4.583333
17            0.104167  0.083333    1.000000   9.600000  0.074653         inf
18            0.083333  0.083333    1.000000  12.000000  0.076389         inf
19            0.083333  0.083333    1.000000  12.000000  0.076389         inf
20            0.104167  0.083333    1.000000   9.600000  0.074653         inf
21            0.083333  0.083333    0.800000   9.600000  0.074653    4.583333
22            0.083333  0.083333    1.000000  12.000000  0.076389         inf
23            0.083333  0.083333    1.000000  12.000000  0.076389         inf
```

Code:

```
rules.sort_values(by='lift',inplace=True,ascending=False)
rules
```

It sorts the values in the dataset according to the lift values in descending order

Output:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
23	(blur)	(paul mccartney, doves)	0.083333	0.083333	0.083333	1.000000	12.000000	0.076389	inf
22	(doves)	(paul mccartney, blur)	0.083333	0.083333	0.083333	1.000000	12.000000	0.076389	inf
19	(paul mccartney, blur)	(doves)	0.083333	0.083333	0.083333	1.000000	12.000000	0.076389	inf
18	(paul mccartney, doves)	(blur)	0.083333	0.083333	0.083333	1.000000	12.000000	0.076389	inf
15	(blur)	(doves)	0.083333	0.083333	0.083333	1.000000	12.000000	0.076389	inf
14	(doves)	(blur)	0.083333	0.083333	0.083333	1.000000	12.000000	0.076389	inf
13	(blur)	(paul mccartney)	0.083333	0.104167	0.083333	1.000000	9.600000	0.074653	inf
21	(paul mccartney)	(blur, doves)	0.104167	0.083333	0.083333	0.800000	9.600000	0.074653	4.583333
20	(blur, doves)	(paul mccartney)	0.083333	0.104167	0.083333	1.000000	9.600000	0.074653	inf
17	(doves)	(paul mccartney)	0.083333	0.104167	0.083333	1.000000	9.600000	0.074653	inf
16	(paul mccartney)	(doves)	0.104167	0.083333	0.083333	0.800000	9.600000	0.074653	4.583333
12	(paul mccartney)	(blur)	0.104167	0.083333	0.083333	0.800000	9.600000	0.074653	4.583333
11	(crystal castles)	(n*e*r*d)	0.166667	0.083333	0.083333	0.500000	6.000000	0.069444	1.833333

Results

Code:

```
rules[ (rules['lift'] >= 6) & (rules['confidence'] >= 0.8) ].sort_values(['confidence','lift'],ascending=False)
```

Sorting out the elements which have are above are equal to a certain values of lift and confidence

Output:

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
(doves)	(blur)	0.083333	0.083333	0.083333	1.0	12.0	0.076389	inf
(blur)	(doves)	0.083333	0.083333	0.083333	1.0	12.0	0.076389	inf
(paul mccartney, doves)	(blur)	0.083333	0.083333	0.083333	1.0	12.0	0.076389	inf
(paul mccartney, blur)	(doves)	0.083333	0.083333	0.083333	1.0	12.0	0.076389	inf
(doves)	(paul mccartney, blur)	0.083333	0.083333	0.083333	1.0	12.0	0.076389	inf
(blur)	(paul mccartney, doves)	0.083333	0.083333	0.083333	1.0	12.0	0.076389	inf
(blur)	(paul mccartney)	0.083333	0.104167	0.083333	1.0	9.6	0.074653	inf
(doves)	(paul mccartney)	0.083333	0.104167	0.083333	1.0	9.6	0.074653	inf
(blur, doves)	(paul mccartney)	0.083333	0.104167	0.083333	1.0	9.6	0.074653	inf
(n*e*r*d)	(crystal castles)	0.083333	0.166667	0.083333	1.0	6.0	0.069444	inf

Conclusion

Here we used association and apriori to find out the corelation and frequent item sets to find music which could be recommended. The result consists of song(antecedents) and it's recommendations(consequents)

References

- https://towardsdatascience.com/create-music-recommendation-system-using-python-ce5401317159
- https://towardsdatascience.com/part-iii-building-a-song-recommendation-system-with-spotify-cf76b52705e7
- https://medium.com/@briansrebrenik/introduction-to-music-recommendation-and-machine-learning-310c4841b01d
- https://www.frontiersin.org/articles/10.3389/fams.2019.00044/full
- https://www.kaggle.com/datasets/ravichaubey1506/lastfm