Data Mining

Music Recommendation system

I. Introduction

With the appearance of online administrations, for example, Pandora and iTunes, the music business has moved away from the customary appropriation model of offering physical duplicates of music to a cloud-based model that gives music to clients to tune in to. In this model, esteem is determined when these administrations present tunes that the client is keen on; either the client buys a membership, or the client pays for the tune. In the two cases, these music administrations can determine monetary profit by improving their proposals to possible clients of the tunes they like. Hence, there is solid monetary motivating force to execute a decent tune suggestion framework. There are many fascinating issues to handle when building a melody proposal framework. One issue that makes suggesting melodies troublesome is that there is no direct comparability measure between tunes; two melodies that sound comparable might be altogether different tunes that have totally various crowds. In this way, we expected to figure out how to increase helpful data that could be utilized to suggest different melodies. Another issue is in distinguishing valuable highlights of the melodies; figuring out which highlights would be best for a learning calculation is significant, since acceptable component determination could improve suggestions.

Recommendation Engines

Proposal motors are a method of displaying and revising data accessible about client inclinations and afterward utilizing this data to give educated suggestions on the premise regarding that data.

It is excessively significant for online organizations, outside the ability to grasp and section the client, which have basically no physical association with their clients. Proposal motors give awesome chances to these associations to comprehend their customers as well as to utilize that data to expand their incomes. Another significant bit of leeway, you can without much of a stretch incorporate an unrefined proposal motor in its collaboration with the clients and afterward, based on its presentation, settle on the choice to build up a more advanced adaptation.

The Million Song Dataset

The million-melody dataset is an exceptionally famous dataset and is accessible at https://labrosa.ee.columbia.edu/millionsong/. The first dataset contained measured sound highlights of around a million tunes extending over numerous years.

Despite the fact that we won't utilize this dataset legitimately, we will utilize a few pieces of it. A few different datasets were brought forth from the first million melody dataset. One of those datasets was known as The Echonest Taste Profile Subset. This specific dataset was made by The Echonest with some undisclosed accomplices. The dataset contains play tallies by unknown clients for tunes contained in the million tunes dataset. Getting ready condition and transferring information, You can download this python journal and information from GitHub archive.

The two datasets can be downloaded on Kaggle, song_data.csv and tally play of 10,000 melodies.

The libraries used are listed as below: -

Os- Python permits the designer to utilize a few OS-subordinate functionalities with the Python module os. This bundle abstracts the functionalities of the stage and gives the python capacities to explore, make, erase and adjust documents and organizers.

Warnings- Alerts are given to caution the designer of circumstances that aren't really special cases. Typically, an admonition happens when there is some out of date of certain programming components, for example, catchphrase, capacity or class, and so on. Alternately, an admonition isn't basic. It gives some message; however, the program runs. The admonition module is really a subclass of Exception which is a class in Python.

Panda- Pandas is a library made to assist designers with working with "named" and "social" information naturally. It depends on two fundamental information structures: "Arrangement" (one-dimensional, similar to a rundown of things) and "Information Frames" (two-dimensional, similar to a table with various sections). Pandas permits changing over information structures to Data Frame objects, dealing with missing information, and including/erasing sections from Data Frame, attributing missing records, and plotting information with histogram or plot box.

NumPy-NumPy (Numerical Python) is an ideal apparatus for logical processing and performing essential and propelled cluster tasks. The library offers numerous convenient highlights performing procedure on n-exhibits and networks in Python. It assists with preparing exhibits that store estimations of similar information type and makes performing math procedure on clusters (and their vectorization) simpler

Time- python has a module named time to handle time-related tasks. The Python time module gives numerous methods of speaking to time in code, for example, items, numbers, and strings. It likewise gives usefulness other than speaking to time, such as holding up during code execution and estimating the productivity of your code.

Matplotlib- This is a standard information science library that assists with creating information perceptions, for example, two-dimensional outlines and charts (histograms, scatterplots, non-Cartesian directions diagrams). Matplotlib is one of those plotting libraries that are extremely valuable in information science ventures — it gives an article arranged API to installing plots into applications.

Scikit Learn- This is an industry-standard for information science ventures situated in Python. Scikits is a gathering of bundles in the SciPy Stack that were made for explicit functionalities – for instance, picture preparing. Scikit-learn utilizes the math tasks of SciPy to uncover a brief interface to the most widely recognized AI calculations.

Instead of using any readily available packages for building our recommendation system, We have used the class approach that permits us to understand what goes on behind a recommendation engine.

```
class popularity_recommender_py():
    def __init__(self):
        self.train_data = None
        self.user_id = None
        self.item_id = None
        self.popularity_recommendations = None

    #Create the popularity based recommender system model
    def create(self, train_data, user_id, item_id):
        self.train_data = train_data
        self.user_id = user_id
        self.item_id = item_id

        #Get a count of user_ids for each unique song as recommendation score
        train_data_grouped = train_data.groupby([self.item_id]).agg({self.user_id: 'coun
t'}).reset_index()
        train_data_grouped.rename(columns = {'user_id': 'score'},inplace=True)

        #Sort the songs based upon recommendation score
        train_data_sort = train_data_grouped.sort_values(['score', self.item_id], ascendin
g = [0,1])

        #Generate a recommendation rank based upon score
        train_data_sort['Rank'] = train_data_sort['score'].rank(ascending=0, method='firs
t')
```

```
        #Get the top 10 recommendations
        self.popularity_recommendations = train_data_sort.head(10)

    #Use the popularity based recommender system model to
    #make recommendations
    def recommend(self, user_id):
        user_recommendations = self.popularity_recommendations

        #Add user_id column for which the recommendations are being generated
        user_recommendations['user_id'] = user_id

        #Bring user_id column to the front
        cols = user_recommendations.columns.tolist()
        cols = cols[-1:] + cols[:-1]
        user_recommendations = user_recommendations[cols]

        return user_recommendations
```

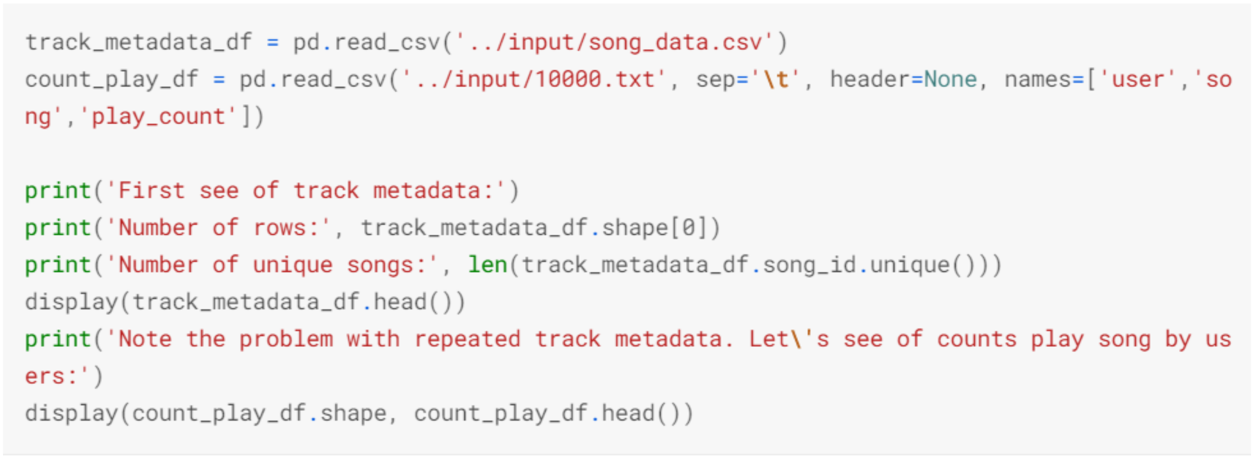
Here we have created the class for our recommendation system that is Popularity based recommendation system model. We have worked on the attribute of user listening the song like user_id and use the popularity based recommender system model to make recommendations.

Preparing Dataset

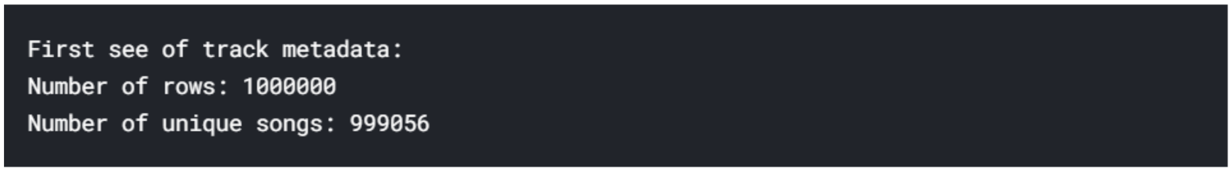
The greater part of suggestion motor utilizes a scanty grid. Along these lines, we need load all information and combined them so we can work with melodies names rather than ID's. In a genuine creation work, you'd stay with ID's and stress over the names at the showcase layer to make things more effective. Be that as it may, this lets us comprehend what's happening better and you can utilize your comprehend to make suggestions for tunes, yet for band/craftsman and collection/discharges.

Load Dataset

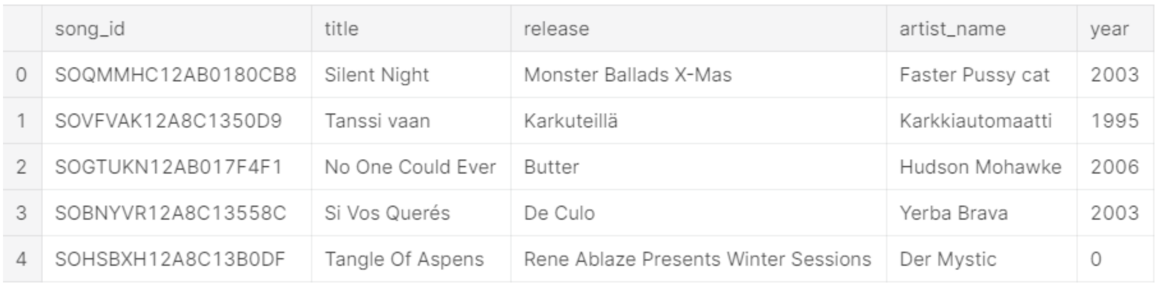
Since we have two datasets that can be joined by the song Id, first we need load then and make a check of consistent. We expect that track metadata has only one record per song, but we know that it is a common problem.



```python
track_metadata_df = pd.read_csv('../input/song_data.csv')
count_play_df = pd.read_csv('../input/10000.txt', sep='\t', header=None, names=['user','so
ng','play_count'])

print('First see of track metadata:')
print('Number of rows:', track_metadata_df.shape[0])
print('Number of unique songs:', len(track_metadata_df.song_id.unique()))
display(track_metadata_df.head())
print('Note the problem with repeated track metadata. Let\'s see of counts play song by us
ers:')
display(count_play_df.shape, count_play_df.head())
```



```
First see of track metadata:
Number of rows: 1000000
Number of unique songs: 999056
```



	song_id	title	release	artist_name	year
0	SOQMMHC12AB0180CB8	Silent Night	Monster Ballads X-Mas	Faster Pussy cat	2003
1	SOVFVAK12A8C1350D9	Tanssi vaan	Karkuteillä	Karkkiautomaatti	1995
2	SOGTUKN12AB017F4F1	No One Could Ever	Butter	Hudson Mohawke	2006
3	SOBNYVR12A8C13558C	Si Vos Querés	De Culo	Yerba Brava	2003
4	SOHSBXH12A8C13B0DF	Tangle Of Aspens	Rene Ablaze Presents Winter Sessions	Der Mystic	0

Thus, as we expect, before go ahead first we need deal with the rehashed track metadata. Since the majority of the case are nulls and a few cases are disparate qualities, we will substitute all irregularities by their separate max esteem by every melody Id.

```python
unique_track_metadata_df = track_metadata_df.groupby('song_id').max().reset_index()

print('Number of rows after unique song Id treatment:', unique_track_metadata_df.shape[0])
print('Number of unique songs:', len(unique_track_metadata_df.song_id.unique()))
display(unique_track_metadata_df.head())
```

```
Number of rows after unique song Id treatment: 999056
Number of unique songs: 999056
```

	song_id	title	release	artist_name	year
0	SOAAABI12A8C13615F	Afro Jazziac	To Birdland And Hurry	Herbie Mann	2000
1	SOAAABT12AC46860F0	Herre Gud Ditt Dyre Namn Og Ære	Som Den Gyldne Sol Frembryter	Bergen Big Band	0
2	SOAAABX12A8C13FEB2	N.Y.C. Remix	Paris Can´t Wait	Guardner	0
3	SOAAACR12A58A79456	Irresistible	Wowie Zowie	Superchumbo	2002
4	SOAAACY12A58A79663	Untitled 1	Pine Cone Temples	Thuja	0

Then we need merge the two datasets based on the songs ids.

```python
user_song_list_count = pd.merge(count_play_df,
                                unique_track_metadata_df, how='left',
                                left_on='song',
                                right_on='song_id')
user_song_list_count.rename(columns={'play_count':'listen_count'},inplace=True)
del(user_song_list_count['song_id'])
```

Exploratory Data Analysis (EDA)

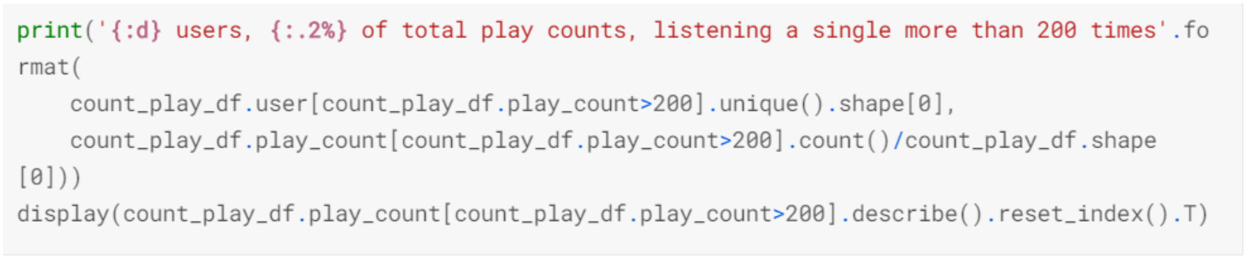
```
display(user_song_list_count.head())
user_song_list_count.listen_count.describe().reset_index().T
```

	user	song	listen_count	title	release
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1	The Cove	Thicker Than Water
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2	Entre Dos Aguas	Flamenco Para Niños
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1	Stronger	Graduation
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1	Constellations	In Between Dreams
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODACBL12A8C13C273	1	Learn To Fly	There Is Nothing Left To Lose

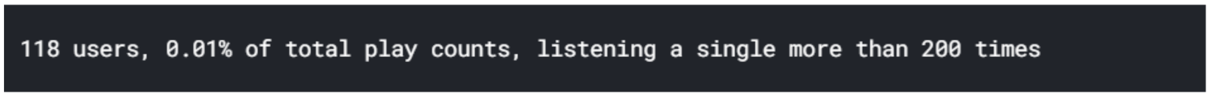
:

	0	1	2	3	4	5	6	7
index	count	mean	std	min	25%	50%	75%	max
listen_count	2e+06	3.04548	6.57972	1	1	1	3	2213

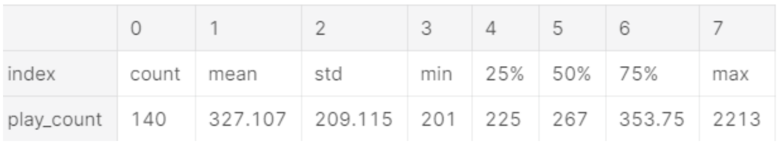
It looks like we have some users obsessed with a single song. A user having heard 2,213 times a single song is something very extreme considering that we have only 3 times in the third quartile. So we have check how many users listen to a single song more than 3 times and more than 200 times.



```
print('{:d} users, {:.2%} of total play counts, listening a single more than 200 times'.fo
rmat(
    count_play_df.user[count_play_df.play_count>200].unique().shape[0],
    count_play_df.play_count[count_play_df.play_count>200].count()/count_play_df.shape
[0]))
display(count_play_df.play_count[count_play_df.play_count>200].describe().reset_index().T)
```



```
118 users, 0.01% of total play counts, listening a single more than 200 times
```



	0	1	2	3	4	5	6	7
index	count	mean	std	min	25%	50%	75%	max
play_count	140	327.107	209.115	201	225	267	353.75	2213

So, as we can see only 118 users listen to a single song more than 200 times

Recommendation Engine
The basis of the recommendation engine is always the recorded interaction between the users and products.

Types of Recommendation Engines

The major area of distinction in different recommendation engines comes from the entity that they assume is the most important in the process of generating recommendations.

Popularity Based Recommendation Engine

The most least complex suggestion motor is the Popularity-Based, that essentially standing, if some thing is loved by a greater part of our client base, at that point it is a smart thought to prescribe that thing to clients who have not associated with that thing. The code to build up this sort of proposal is very simple and is successfully only an outline strategy that figure out which things of the substance have the most clients and afterward that will end up being our standard suggestion set for every client.

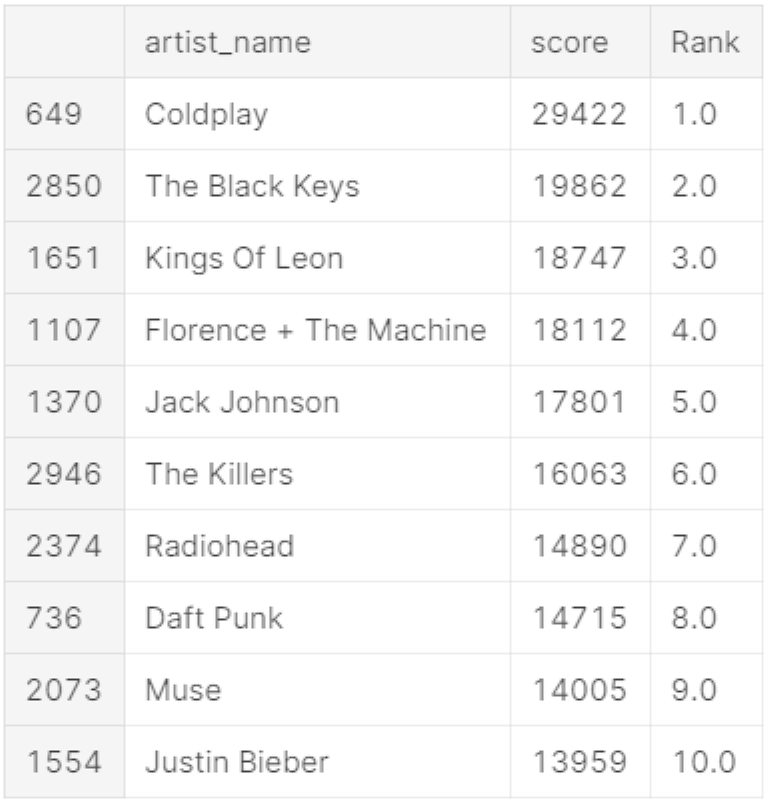
```
def create_popularity_recommendation(train_data, user_id, item_id, n=10):
    #Get a count of user_ids for each unique song as recommendation score
    train_data_grouped = train_data.groupby([item_id]).agg({user_id: 'count'}).reset_index
()
    train_data_grouped.rename(columns = {user_id: 'score'},inplace=True)

    #Sort the songs based upon recommendation score
    train_data_sort = train_data_grouped.sort_values(['score', item_id], ascending = [0,
1])

    #Generate a recommendation rank based upon score
    train_data_sort['Rank'] = train_data_sort.score.rank(ascending=0, method='first')

    #Get the top n recommendations
    popularity_recommendations = train_data_sort.head(n)
    return popularity_recommendations
```

```
recommendations = create_popularity_recommendation(user_song_list_count,'user','title', 1
5)
display(recommendations)
```

	title	score	Rank
6837	Sehr kosmisch	8277	1.0
8726	Undo	7032	2.0
1965	Dog Days Are Over (Radio Edit)	6949	3.0
9497	You're The One	6729	4.0
6499	Revelry	6145	5.0
6826	Secrets	5841	6.0
3438	Horn Concerto No. 4 in E flat K495: II. Romanc...	5385	7.0
2596	Fireflies	4795	8.0
3323	Hey_ Soul Sister	4758	9.0
8495	Tive Sim	4548	10.0
8781	Use Somebody	3976	11.0
5721	OMG	3947	12.0
2120	Drop The World	3879	13.0
5000	Marry Me	3578	14.0
1265	Canada	3526	15.0

We have use our popularity recommendation function to find the 10 artists recommendations too.

```
display(create_popularity_recommendation(user_song_list_count,'user','artist_name', 10))
```



	artist_name	score	Rank
649	Coldplay	29422	1.0
2850	The Black Keys	19862	2.0
1651	Kings Of Leon	18747	3.0
1107	Florence + The Machine	18112	4.0
1370	Jack Johnson	17801	5.0
2946	The Killers	16063	6.0
2374	Radiohead	14890	7.0
736	Daft Punk	14715	8.0
2073	Muse	14005	9.0
1554	Justin Bieber	13959	10.0

As I said before that the class Popularity_recommender_py() gives the recommendation to the user which is passed as a parameter. So, it returns the list of the popular songs for the user but since it is popularity-based recommendation system the recommendation for the users will not be affected.

So, even if we change the user the result that we get from the system is the same since it is a popularity-based recommendation system.

Conclusion:

As you see, the assignment to building a suggestion motor isn't to hard, and we didn't utilize any library to can have better comprehension of what goes on behind a proposal motor. When we have characterized the center element, our most prominent exertion will be to set up the information as indicated by this element and to meet the suggested proposal calculation.

The decision of the calculation additionally goes through the data we have, yet as we have seen, even with scarcely any qualities, we can create frameworks of proposals of differing intricacy and with expressive outcomes for the business.