INTERNSHIP REPORT

Recommending Songs for Instagram reels to Influencers:

Medicento (Yoursbiz ad-network)

Name : Vijaykumar Arun Lokhande

Roll No: 18135119

Branch : Mechanical

Email: vijaykumral.mec18@itbhu.ac.in

Acknowledgement

I would like to take this opportunity to thank Medicento (Yours biz Ad-Network) and the entire recruitment team, for seeing the potential in me and offering me the position of Machine Learning Intern with the firm and giving me such an enriching experience even in virtual mode.

I am highly indebted to my reporting manager Mr. Rahul Vidyarthi and my mentor Ms. Rekha, for constantly helping, guiding and supporting me in my responsibilities and project for the past 8 weeks of my summer internship at the company. I hope that the insights and knowledge I gained about Machine Learning would be very helpful for my future endeavors.

In the end, I express my overall gratitude to my institute and the Training and Placement Cell for providing me with such an opportunity.

About Medicento(Yourbiz)

Medicento(Yoursbiz) is an Ad Network in affiliation Marketing space redefining Network and digital Marketing products using a data driven approach.

We intend to bring Advertisers, Influencers and Retailers onto 1 single platform to create this Unified Medicine cum Ad-Network Channel.

We intend to change the Affiliation Marketing ecosystem by introducing demand (pull based) flow of advertisement and healthcare rather than push based, thus optimising Margins and engagement for both advertisers and influencers

Our smart AI based recommendation system would suggest future activities based on the demand of the market, to influencers and advertisers.

Contents:

1. Introduction

Social Media is not just social media, it has become a platform for advertisement. Social Media like Instagram, Facebook gained Popularity. Recently Reels came out to be very popular hence widely used in affiliate marketing to promote the advertisement. It has become a source of income for many.

Rapid development of mobile devices and the internet has made it possible for us to access different music resources freely. The number of songs available exceeds the listening capacity of a single individual. People sometimes feel it is difficult to choose from millions of songs. Moreover, music service providers need an efficient way to manage songs and help their customers to discover music by giving quality recommendations. Thus, there is a strong need for a good recommendation system.

Currently, there are many music streaming services, like Pandora, Spotify, etc. which are working on building high-precision commercial music recommendation systems. These companies generate revenue by helping their customers discover relevant music and charging them for the quality of their recommendation service. Thus, there is a strong thriving market for good music recommendation systems.

Music recommender system is a system which learns from the users past listening history and recommends songs which they would probably like to hear in future. We have implemented various algorithms to try to build an effective recommender system. We firstly implemented a popularity based model which was quite simple and intuitive. Collaborative filtering algorithms which predict (filtering) taste of a user by collecting preferences and tastes from many other users (collaborating) is also implemented. We have also done experiments on content based models, based on latent factors and metadata.

2. Dataset

Name: Vijaykumar Arun Lokhande

Project name: Insta-Influencer

Song Recommendation: Data and Sources

Code File: Data Cleaning and EDA.ipynb

Dataset: data1.xlsx, data2.xlsx

Objective:

To explain various features in the dataset, their meaning. To mention sources from which they data has been collected.

Libraries Used: -

Numpy, Pandas, Selenium, BeautifulSoup Excel.

** Sources **

1. User Profile Data

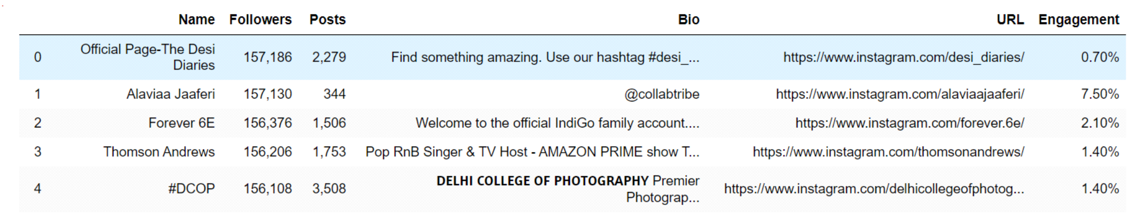
Source: https://starngage.com/app/global/influencer/ranking/india

Technologies Used: Selenium.

Language : Python (Jupyter Notebooks)

Code : Scraping Influencer from StarEngage Website.ipynb

Preview:



	Name	Followers	Posts	Bio	URL	Engagement
0	Official Page-The Desi Diaries	157,186	2,279	Find something amazing. Use our hashtag #desi_...	https://www.instagram.com/desi_diaries/	0.70%
1	Alaviaa Jaaferi	157,130	344	@collabtribe	https://www.instagram.com/alaviaajaaferi/	7.50%
2	Forever 6E	156,376	1,506	Welcome to the official IndiGo family account....	https://www.instagram.com/forever.6e/	2.10%
3	Thomson Andrews	156,206	1,753	Pop RnB Singer & TV Host - AMAZON PRIME show T...	https://www.instagram.com/thomsonandrews/	1.40%
4	#DCOP	156,108	3,508	DELHI COLLEGE OF PHOTOGRAPHY Premier Photograp...	https://www.instagram.com/delhicollegeofphotog...	1.40%

problems:

Although Many attempts were made to scrape the data directly from Instagram, we could not collect a mass data from instagram. There is no official API. There were some issues of account restrictions.

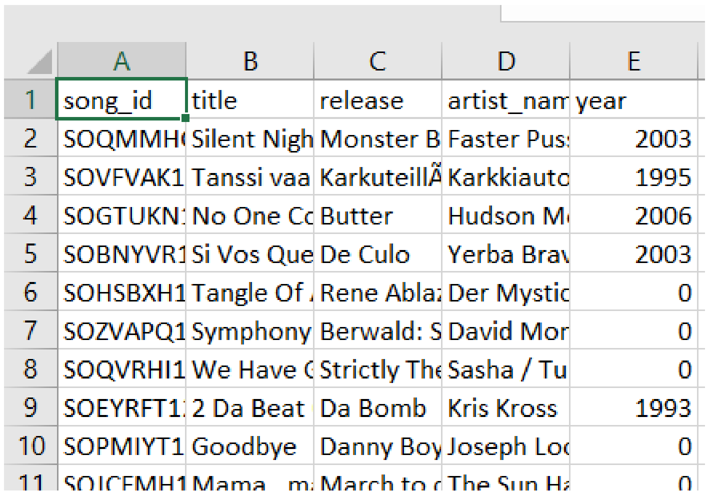
2. Songs History and Metadata

Same above mentioned difficulty in getting data was there. So I collected Data of songs from million songs dataset.

Source: https://www.kaggle.com/c/msdchallenge

File : Song_data.csv

Preview:



	A	B	C	D	E
1	song_id	title	release	artist_nam	year
2	SOQMMH(	Silent Nigh	Monster B	Faster Pus:	2003
3	SOVFVAK1	Tanssi vaa	KarkuteillÃ	Karkkiauto	1995
4	SOGTUKN:	No One Cc	Butter	Hudson M(	2006
5	SOBNYVR1	Si Vos Que	De Culo	Yerba Brav	2003
6	SOHSBXH1	Tangle Of ,	Rene Abla:	Der Mystic	0
7	SOZVAPQ1	Symphony	Berwald: S	David Mor	0
8	SOQVRHI1	We Have (	Strictly The	Sasha / Tu	0
9	SOEYRFT1:	2 Da Beat	Da Bomb	Kris Kross	1993
10	SOPMIYT1	Goodbye	Danny Boy	Joseph Loc	0
11	SOICFMH1	Mama m:	March to c	The Sun Ha	0

3. Combining Metadata and User Profile.

We'll store the results in two seperate xlsx files because it can't store more than 60k urls.

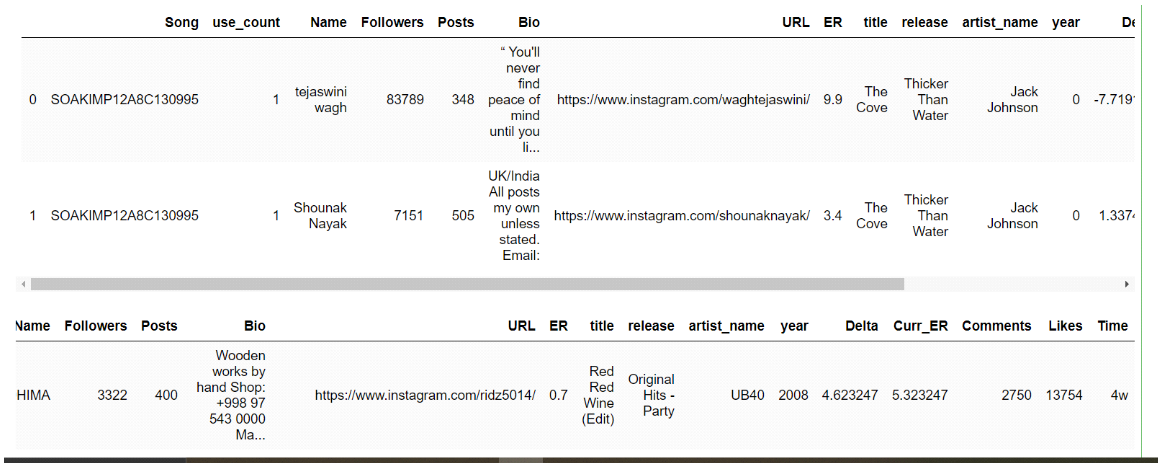
1. Criteria for combination:

i) Select most of nano/micro influencer + add some clinch of celebrities (10k nano/micro + 2k celebrities.)

ii) Select random songs and ensure that songs may repeat most of the times.

File : data1.xlsx, data2.xlsx

Preview:



	Song	use_count	Name	Followers	Posts	Bio	URL	ER	title	release	artist_name	year	De
0	SOAKIMP12A8C130995	1	tejaswini wagh	83789	348	“ You'll never find peace of mind until you li...	https://www.instagram.com/waghtejaswini/	9.9	The Cove	Thicker Than Water	Jack Johnson	0	-7.719
1	SOAKIMP12A8C130995	1	Shounak Nayak	7151	505	UK/India All posts my own unless stated. Email:	https://www.instagram.com/shounaknayak/	3.4	The Cove	Thicker Than Water	Jack Johnson	0	1.337

Name	Followers	Posts	Bio	URL	ER	title	release	artist_name	year	Delta	Curr_ER	Comments	Likes	Time
HIMA	3322	400	Wooden works by hand Shop: +998 97 543 0000 Ma...	https://www.instagram.com/ridz5014/	0.7	Red Red Wine (Edit)	Original Hits - Party	UB40	2008	4.623247	5.323247		2750	13754	4w

** Features Meaning **

'**Song',** => song used for reels

'**use_count'** => No. of Times a particular user/Influencer used this song for reels,

'**Name'** => Name of the user

'**Followers'** => Follower Count of the influencer

'**Posts'** => Total No. of posts poste from the influencer account

'**Bio**',=> Biography

'**URL**'=> Profile URL,

'ER',=> Engagement Rate

{Song Metadata} 'title', 'release', 'artist_name', 'year',

'**Comments**',=> Comments for this reels

'**Likes**', => Likes for this reels

Curr_ER', => ER for this reel.

Ex. If comments for this reel = 100, If Likes for this reel = 1000, If user have followers = 5000. Then curr_ER is

Curr_ER = (Likes + Comments) /Followers = (100+1000)/5000

'Delta' => AVG ER - Curr_ER

Avg ER is calculated by the past consistent history of users over a number of posts.

Delta = Avg ER - Curr_ER

Ex. AVG ER = 2.4 and Curr_ER for a reel is 4.0 then Delta = 4.0-2.4 = 1.6

Note Delta can be negetive.

'**Time**' => days before the reel has been posted

Results: Overviewed Data and understand its feature briefly.

3. Cleaning

Name: Vijaykumar Arun Lokhande

Project name: Insta-Influencer

Song Recommendation: Cleaning+EDA

Code File: Data Cleaning and EDA.ipynb

Dataset: data1.xlsx, data2.xlsx

Objective:

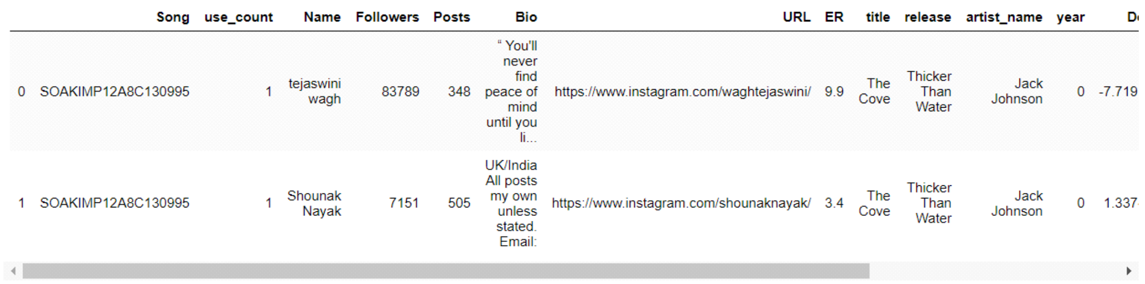
We are cleaning the dataset with the help of EDA.

Libraries Used: -

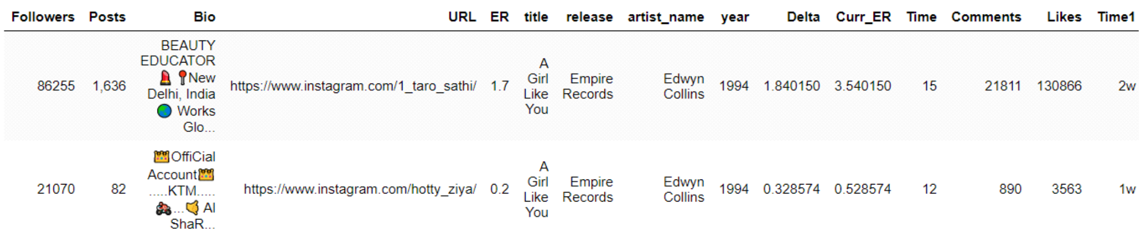
Numpy, Pandas, Seaborn, Matplotlib

Stepwise Explanation: -

This is the dataframe.



	Song	use_count	Name	Followers	Posts	Bio	URL	ER	title	release	artist_name	year	D
0	SOAKIMP12A8C130995	1	tejaswini wagh	83789	348	" You'll never find peace of mind until you li...	https://www.instagram.com/waghtejaswini/	9.9	The Cove	Thicker Than Water	Jack Johnson	0	-7.719
1	SOAKIMP12A8C130995	1	Shounak Nayak	7151	505	UK/India All posts my own unless stated. Email:	https://www.instagram.com/shounaknayak/	3.4	The Cove	Thicker Than Water	Jack Johnson	0	1.337



Followers	Posts	Bio	URL	ER	title	release	artist_name	year	Delta	Curr_ER	Time	Comments	Likes	Time1
86255	1,636	BEAUTY EDUCATOR 👠📍New Delhi, India 🌏 Works Glo...	https://www.instagram.com/1_taro_sathi/	1.7	A Girl Like You	Empire Records	Edwyn Collins	1994	1.840150	3.540150	15	21811	130866	2w
21070	82	👑OffiCial Account👑 .....KTM..... 🏍...📢Al ShaR...	https://www.instagram.com/hotty_ziya/	0.2	A Girl Like You	Empire Records	Edwyn Collins	1994	0.328574	0.528574	12	890	3563	1w

- ## 1. Remove Repeating Entries:

Influencer is uniquely identified by 'URL' and Song for reel is uniquely identified by 'Song'. So we will remove the rows with duplicate ['URL' + 'Song']. (ie same user have used the same song)

- ## 2. Convert Str format to Int:

Some of the numerical data is present in string format. ex. Likes is string format, then convert it to int format.

- ## 3. Remove entries 'Use Count'<'Posts':

'Use Count': The no. of times a particular influencer with 'URL' and used a particular Song named 'Song' is used for reel.

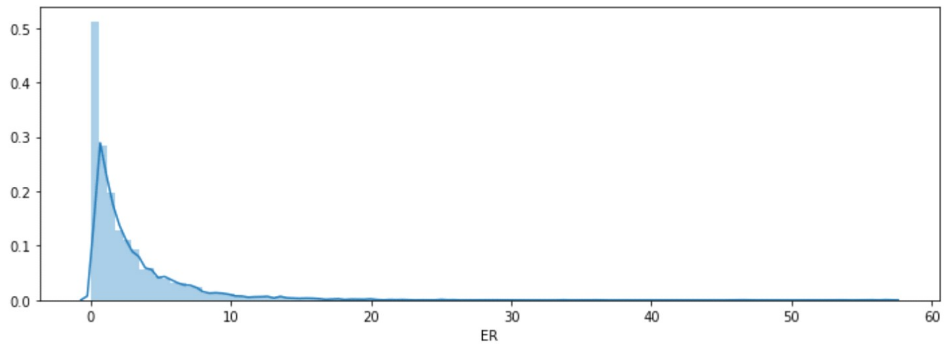
So the posts made by the influencer must be greater than use_count of the song.

- ## 4: Remove entries with ER>12:

ER means Engagement Rate; ER = (likes + comment)/Followers

Usually ER is pretty low for every influencer < 5%. because most of the followers are passive followers. To support our fact, here is distplot for our dataset.

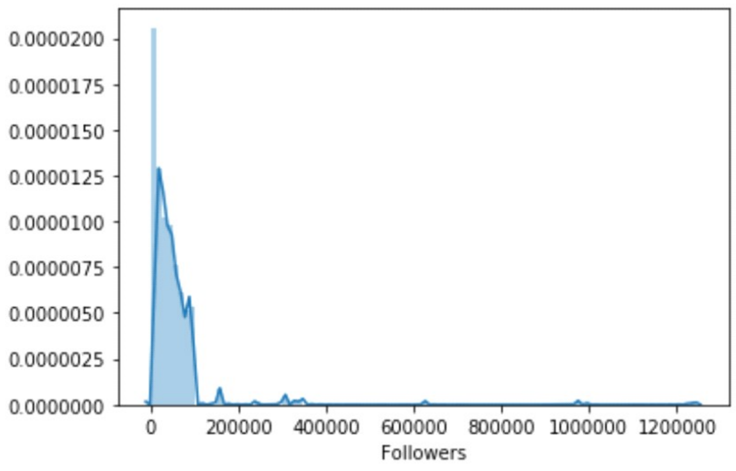
```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x2a2a9a20f48>
```



So we filtered out ER<12%.

- ## 5: Visualise User distribution based on Follower Count

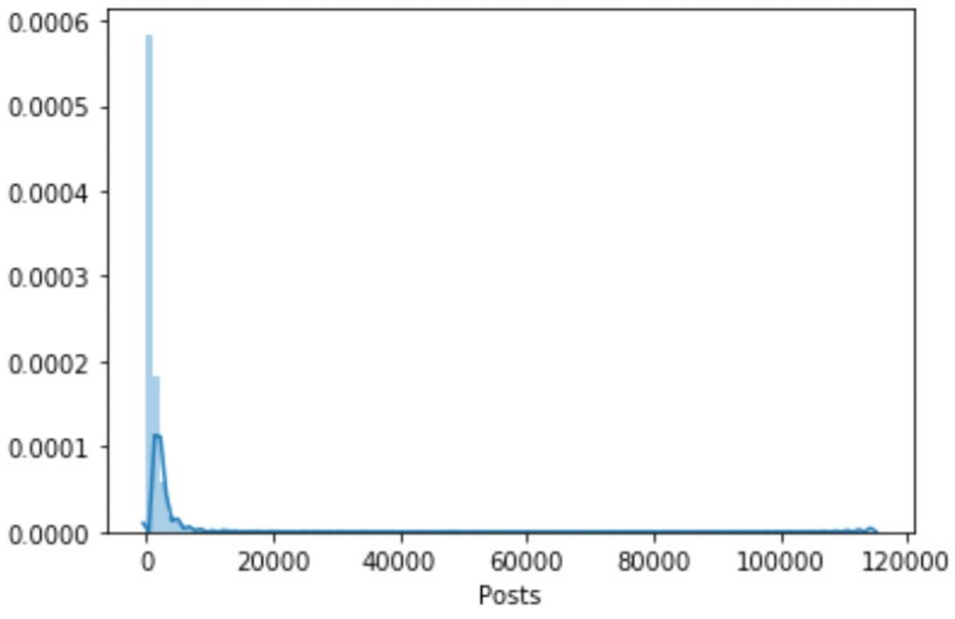
```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x2a2a7143808>
```



Most of the influencer are nano/micro <60k followers. Some of the influncer have huge followers following; These influencer with more than 80k followers are celebrities. Also not everybody is celebrity, only a few people are celebrity. Make Sense!

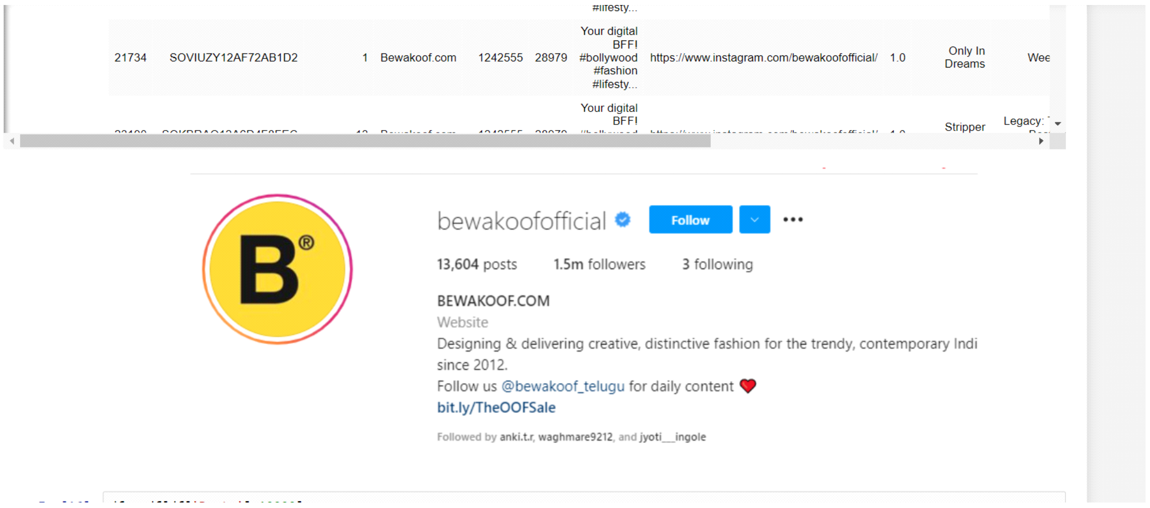
6: Remove entries Post>15k

```
[11]: <matplotlib.axes._subplots.AxesSubplot at 0x2a2a897ef88>
```



Most of the influencer have posted < 15k posts in their lifetime. Let's observe some of the outliers. In the dataset 'Bewakoof' is the profile with over more than 1milion post. For this I manually checked the profile of 'Bewakoof' and the account hasn't posted more than 13k posts.

Even the mighty Priyanka Chopra hasn't.



21734	SOVIUZY12AF72AB1D2	1	Bewakoof.com	1242555	28979	Your digital BFF! #bollywood #fashion #lifesty...	https://www.instagram.com/bewakoofofficial/	1.0	Only In Dreams	Wee
						Your digital BFF!			Stripper	Legacy:

bewakoofofficial Follow

13,604 posts 1.5m followers 3 following

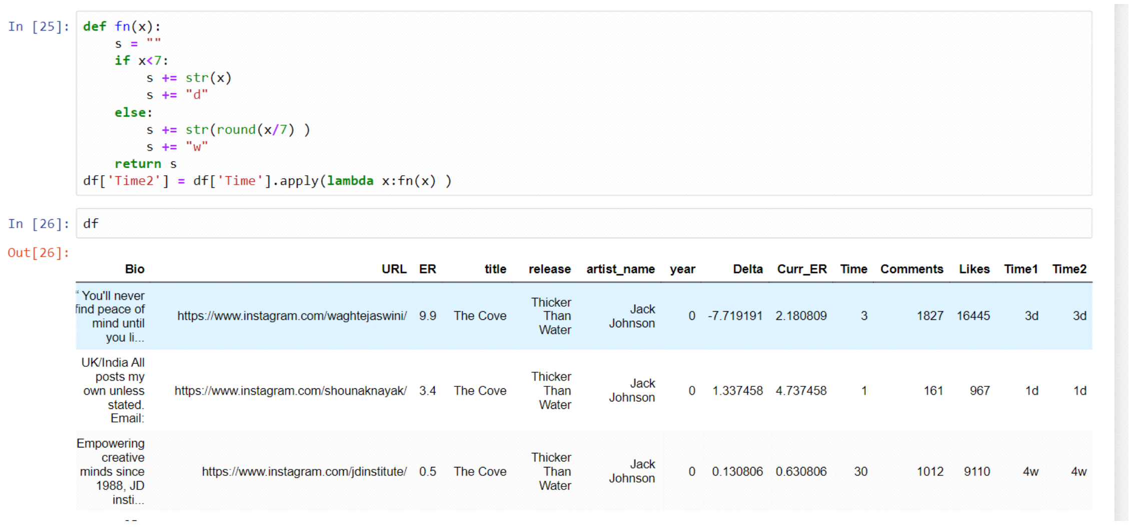
BEWAKOOF.COM
Website
Designing & delivering creative, distinctive fashion for the trendy, contemporary Indi since 2012.
Follow us @bewakoof_telugu for daily content ❤
bit.ly/TheOOFSale

Followed by anki.t.r, waghmare9212, and jyoti__ingole

So these are non accurate data points. We'll remove it.

- ## 7. Convert Time to it's format:

observe the time is in numeric. But when we actually collect the data, data will have time in format 5d, 7w. Time means the days/weeks before the reel has been posted.



```
In [25]: def fn(x):
             s = ""
             if x<7:
                 s += str(x)
                 s += "d"
             else:
                 s += str(round(x/7) )
                 s += "w"
             return s
         df['Time2'] = df['Time'].apply(lambda x:fn(x) )
```

```
In [26]: df
```

Out[26]:

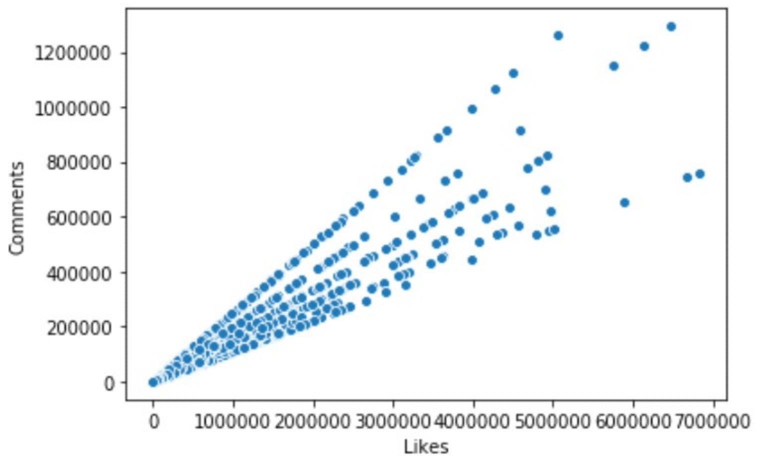
Bio	URL	ER	title	release	artist_name	year	Delta	Curr_ER	Time	Comments	Likes	Time1	Time2
" You'll never find peace of mind until you li...	https://www.instagram.com/waghtejaswini/	9.9	The Cove	Thicker Than Water	Jack Johnson	0	-7.719191	2.180809	3	1827	16445	3d	3d
UK/India All posts my own unless stated. Email:	https://www.instagram.com/shounaknayak/	3.4	The Cove	Thicker Than Water	Jack Johnson	0	1.337458	4.737458	1	161	967	1d	1d
Empowering creative minds since 1988, JD insti...	https://www.instagram.com/jdinstitute/	0.5	The Cove	Thicker Than Water	Jack Johnson	0	0.130806	0.630806	30	1012	9110	4w	4w

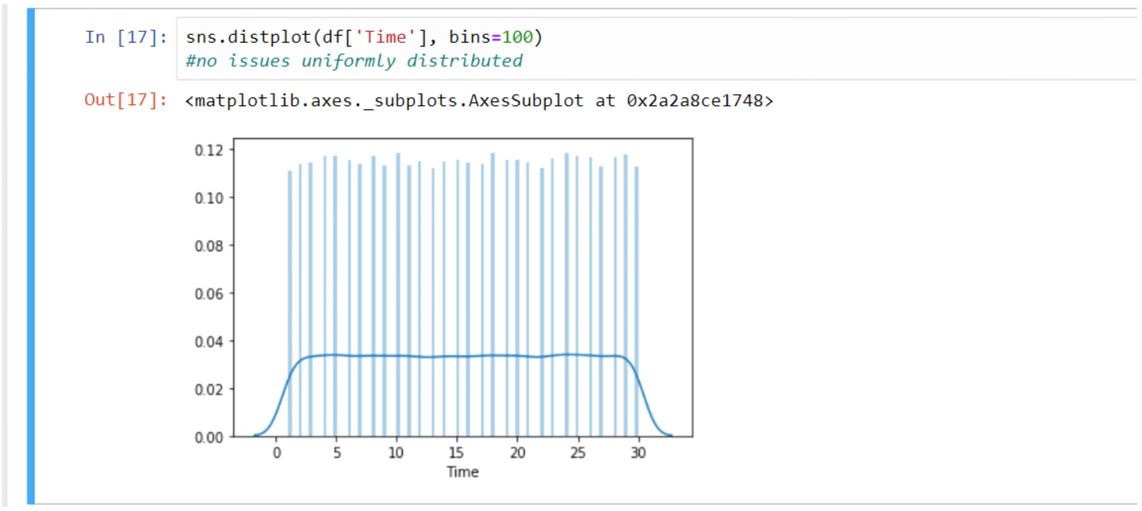
- ## 8. Some more EDA:

Below are the graphs. Graphs are perfectly linear because the data is dummy and has been created manually using logic.

```
[19]: sns.scatterplot(x='Likes', y='Comments', data=df)
      #have relationship, perfect lines since our data is dummy
```

```
[19]: <matplotlib.axes._subplots.AxesSubplot at 0x2a2ad8289c8>
```





```
In [17]: sns.distplot(df['Time'], bins=100)
         #no issues uniformly distributed

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x2a2a8ce1748>
```

- 9. Remove Unnecessary Features:

remove all the unncessary column/features. Only keep those attributes which are scrapable.

<u>4.1 Popularity Based</u>

Name: Vijaykumar Arun Lokhande

Project name: Insta-Influencer

<u>Song Recommendation: Popularity Based</u>

Code File: Celebrity Usage.ipynb, Count Based Popularity (MAP).ipynb, More Engagement + Count

Dataset: Cleaned_data1.xlsx, Cleaned_data2.xlsx

Objective:

To recommend songs for reels to Influencer tracked by Yoursbiz Ad-Network.

Libraries Used: -

Numpy, Pandas, Sklearn.

Model: -Popularity(Sorting)

<u>1st Variation:</u> Celebrity

- **Explanation**:

Note:

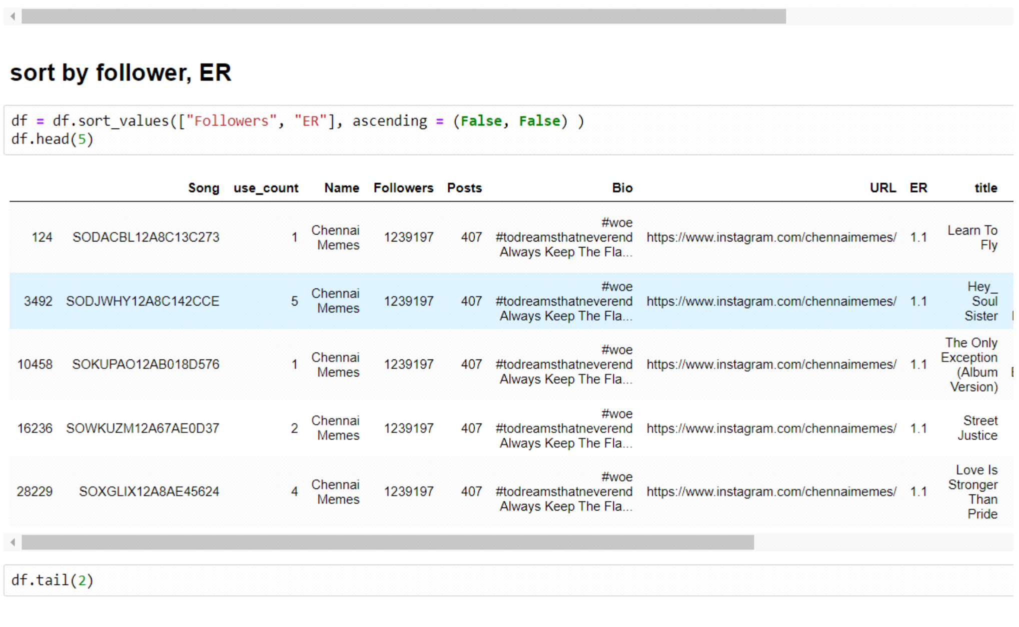
'Delta' = (Curr_ER - AVG ER) if you don't know it, refer documentation of Dataset.

We'll Treate as if they are rating as in movie rating predictions.

- 0. Filter out user with <80k followers:

Celebrity meaning: Here celebrity means the influencer with more followers. (ie we assume >80k followers). We did so because our focus is on nano and micro influencers.

- 1. Sort by Followers Count and then By Engagment Rate.



sort by follower, ER

```
In [5]: df = df.sort_values(["Followers", "ER"], ascending = (False, False) )
        df.head(5)
Out[5]:
```

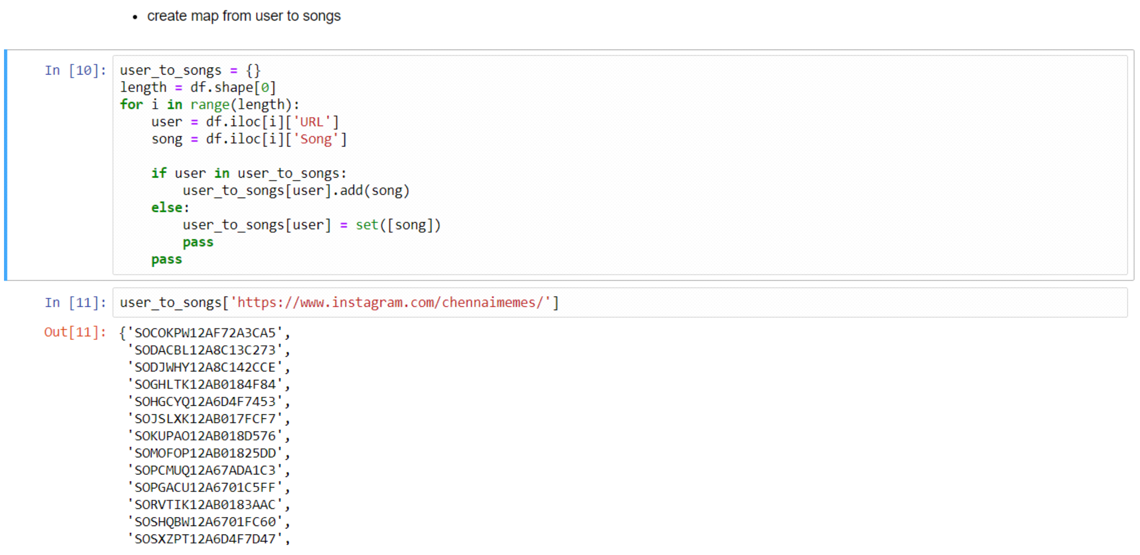
	Song	use_count	Name	Followers	Posts	Bio	URL	ER	title
124	SODACBL12A8C13C273	1	Chennai Memes	1239197	407	#woe #todreamsthatneverend Always Keep The Fla...	https://www.instagram.com/chennaimemes/	1.1	Learn To Fly
3492	SODJWHY12A8C142CCE	5	Chennai Memes	1239197	407	#woe #todreamsthatneverend Always Keep The Fla...	https://www.instagram.com/chennaimemes/	1.1	Hey_ Soul Sister
10458	SOKUPAO12AB018D576	1	Chennai Memes	1239197	407	#woe #todreamsthatneverend Always Keep The Fla...	https://www.instagram.com/chennaimemes/	1.1	The Only Exception (Album Version)
16236	SOWKUZM12A67AE0D37	2	Chennai Memes	1239197	407	#woe #todreamsthatneverend Always Keep The Fla...	https://www.instagram.com/chennaimemes/	1.1	Street Justice
28229	SOXGLIX12A8AE45624	4	Chennai Memes	1239197	407	#woe #todreamsthatneverend Always Keep The Fla...	https://www.instagram.com/chennaimemes/	1.1	Love Is Stronger Than Pride

```
In [15]: df.tail(2)
Out[15]:
```

So we sorted users based on their follower counts and then by their performance(ie. ER).

- ## 2. Create User to Song Mapping:

We are going to create a user to songs dictionary. This dictionary will help us to 'NOT TO RECOMMEND THE SONG WHICH HAS ALREADY BEEN USED BY THE INFLUENCER/USER'



- create map from user to songs

```
In [10]: user_to_songs = {}
         length = df.shape[0]
         for i in range(length):
             user = df.iloc[i]['URL']
             song = df.iloc[i]['Song']

             if user in user_to_songs:
                 user_to_songs[user].add(song)
             else:
                 user_to_songs[user] = set([song])
                 pass
             pass

In [11]: user_to_songs['https://www.instagram.com/chennaimemes/']

Out[11]: {'SOCOKPW12AF72A3CA5',
          'SODACBL12A8C13C273',
          'SODJWHY12A8C142CCE',
          'SOGHLTK12AB0184F84',
          'SOHGCYQ12A6D4F7453',
          'SOJSLXK12AB017FCF7',
          'SOKUPAO12AB018D576',
          'SOMOFOP12AB01825DD',
          'SOPCMUQ12A67ADA1C3',
          'SOPGACU12A6701C5FF',
          'SORVTIK12AB0183AAC',
          'SOSHQBW12A6701FC60',
          'SOSXZPT12A6D4F7D47',
```

- ## 3. Take i/p User:

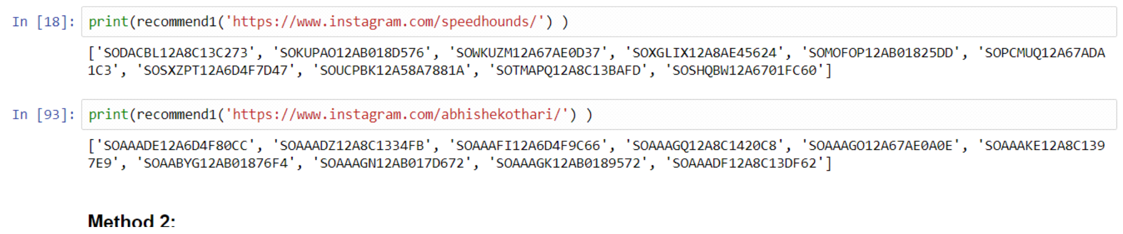
Take input user

- ## 4. Recommend Song:

Recommend the song from dataframe which sorted in decreasing manner, and the song should not has been used the user already(ie not found in mapping)

- **Results:**

list of top 10 songs that to be recommended to the influencers.



```
In [18]: print(recommend1('https://www.instagram.com/speedhounds/') )

         ['SODACBL12A8C13C273', 'SOKUPAO12AB018D576', 'SOWKUZM12A67AE0D37', 'SOXGLIX12A8AE45624', 'SOMOFOP12AB01825DD', 'SOPCMUQ12A67ADA
         1C3', 'SOSXZPT12A6D4F7D47', 'SOUCPBK12A58A7881A', 'SOTMAPQ12A8C13BAFD', 'SOSHQBW12A6701FC60']

In [93]: print(recommend1('https://www.instagram.com/abhishekothari/') )

         ['SOAAADE12A6D4F80CC', 'SOAAADZ12A8C1334FB', 'SOAAAFI12A6D4F9C66', 'SOAAAGQ12A8C1420C8', 'SOAAAGO12A67AE0A0E', 'SOAAAKE12A8C139
         7E9', 'SOAABYG12AB01876F4', 'SOAAAGN12AB017D672', 'SOAAAGK12AB0189572', 'SOAAADF12A8C13DF62']
```

Method 2:

All the step remains same, just 'Step 1' Changes in other the variations:

2nd Variation: *Celebrity*

- 1. Sort by Followers Count and then groupby songs.

* based on no. of celebrities that have used the song as reel

* Assumption:

1. define celebrity: influencer with follower count > 80k
2. Celebrities tend to follow the trends
3. more celebrities using the song, more likely it to be trending
4. more trending is a song, more useful for influencer.

We will be having how many celebrities(influencer>80k) have used the song for making reels

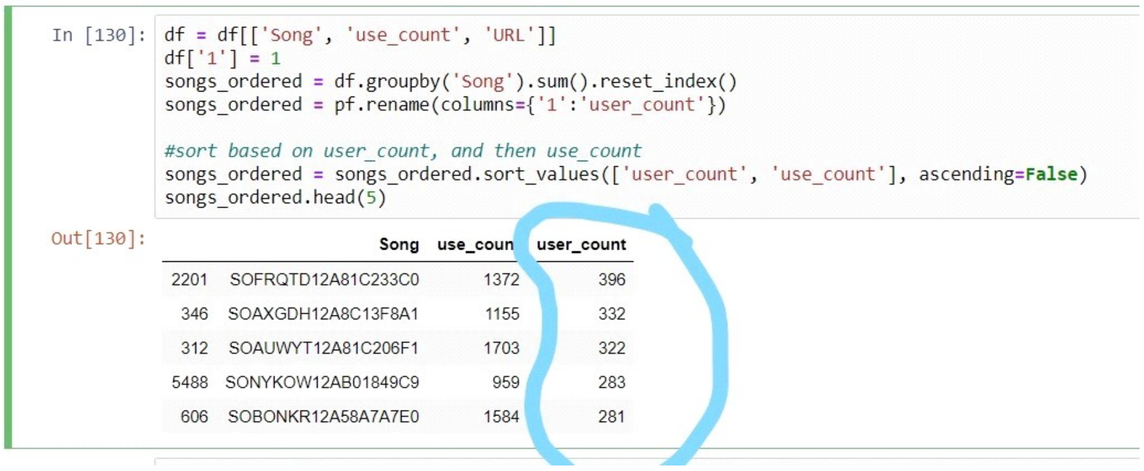
3rd Variation: *Count*

* If 80-90% of the influencers tracked by Yoursbiz have used a particular song, implies that the song is popular song

* Assumption : using popular song is likely to boost the engagement rate and attract more fans/followers to the influencer.

- 1.1 User Count

Create a variable 'User Count' which tells us the no. of unique user that have used the song once in life as reel.



```
In [130]: df = df[['Song', 'use_count', 'URL']]
          df['1'] = 1
          songs_ordered = df.groupby('Song').sum().reset_index()
          songs_ordered = pf.rename(columns={'1':'user_count'})

          #sort based on user_count, and then use_count
          songs_ordered = songs_ordered.sort_values(['user_count', 'use_count'], ascending=False)
          songs_ordered.head(5)
```

Out[130]:

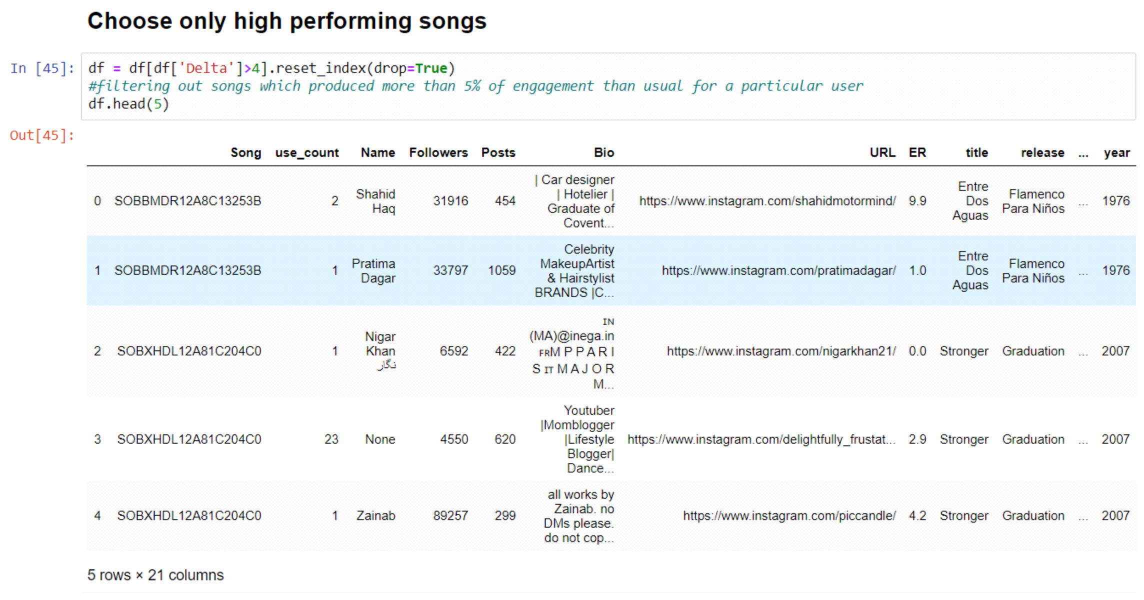
	Song	use_coun	user_count
2201	SOFRQTD12A81C233C0	1372	396
346	SOAXGDH12A8C13F8A1	1155	332
312	SOAUWYT12A81C206F1	1703	322
5488	SONYKOW12AB01849C9	959	283
606	SOBONKR12A58A7A7E0	1584	281

- 1.2 Sort on User Count:

sort decreasingly on user count. dataframe looks like above.

<u>4th Variation:</u> More Performing

- 1.1 Only keep song which have performed. ie(Delta>4)



Choose only high performing songs

```
In [45]: df = df[df['Delta']>4].reset_index(drop=True)
         #filtering out songs which produced more than 5% of engagement than usual for a particular user
         df.head(5)
```

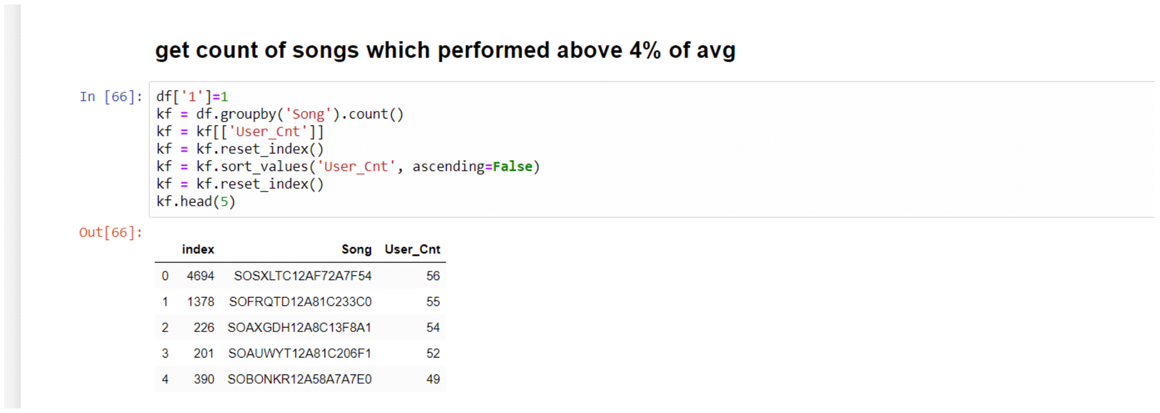
Out[45]:

	Song	use_count	Name	Followers	Posts	Bio	URL	ER	title	release	...	year
0	SOBBMDR12A8C13253B	2	Shahid Haq	31916	454	| Car designer | Hotelier | Graduate of Covent...	https://www.instagram.com/shahidmotormind/	9.9	Entre Dos Aguas	Flamenco Para Niños	...	1976
1	SOBBMDR12A8C13253B	1	Pratima Dagar	33797	1059	Celebrity MakeupArtist & Hairstylist BRANDS |C...	https://www.instagram.com/pratimadagar/	1.0	Entre Dos Aguas	Flamenco Para Niños	...	1976
2	SOBXHDL12A81C204C0	1	Nigar Khan نگار	6592	422	IN (MA)@inega.in FRM P P A R I S IT M A J O R M...	https://www.instagram.com/nigarkhan21/	0.0	Stronger	Graduation	...	2007
3	SOBXHDL12A81C204C0	23	None	4550	620	Youtuber |Momblogger |Lifestyle Blogger| Dance...	https://www.instagram.com/delightfully_frustat...	2.9	Stronger	Graduation	...	2007
4	SOBXHDL12A81C204C0	1	Zainab	89257	299	all works by Zainab. no DMs please. do not cop...	https://www.instagram.com/piccandle/	4.2	Stronger	Graduation	...	2007

5 rows × 21 columns

1.2 Groupby the songs

make 'User Count'. The number of different unique user for which the song has performed and brought more change engagement rate > +4% in atleast once in the lifetime.



get count of songs which performed above 4% of avg

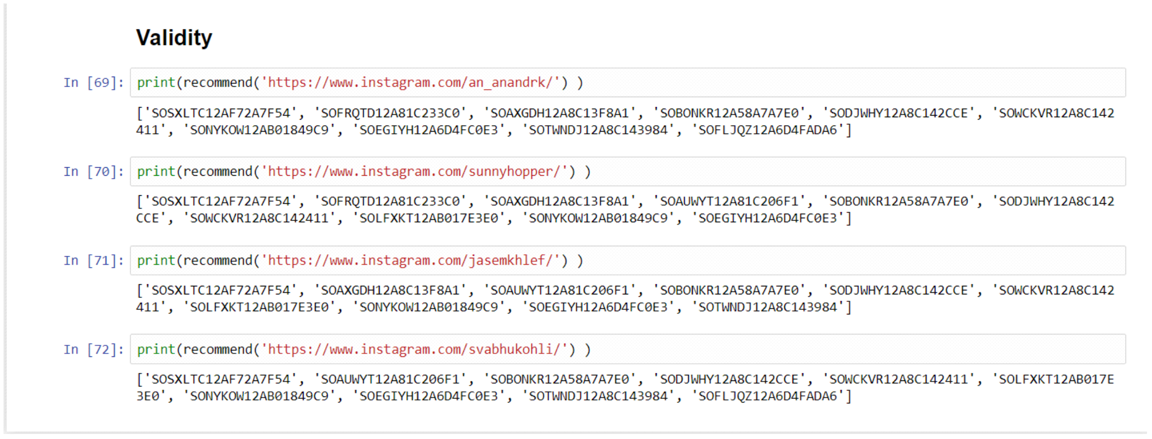
```
In [66]: df['1']=1
         kf = df.groupby('Song').count()
         kf = kf[['User_Cnt']]
         kf = kf.reset_index()
         kf = kf.sort_values('User_Cnt', ascending=False)
         kf = kf.reset_index()
         kf.head(5)
```

Out[66]:

	index	Song	User_Cnt
0	4694	SOSXLTC12AF72A7F54	56
1	1378	SOFRQTD12A81C233C0	55
2	226	SOAXGDH12A8C13F8A1	54
3	201	SOAUWYT12A81C206F1	52
4	390	SOBONKR12A58A7A7E0	49

1.3 Sort in Decreasing order on User_Cnt.

Result:



Validity

```
In [69]: print(recommend('https://www.instagram.com/an_anandrk/') )

['SOSXLTC12AF72A7F54', 'SOFRQTD12A81C233C0', 'SOAXGDH12A8C13F8A1', 'SOBONKR12A58A7A7E0', 'SODJWHY12A8C142CCE', 'SOWCKVR12A8C142
411', 'SONYKOW12AB01849C9', 'SOEGIYH12A6D4FC0E3', 'SOTWNDJ12A8C143984', 'SOFLJQZ12A6D4FADA6']

In [70]: print(recommend('https://www.instagram.com/sunnyhopper/') )

['SOSXLTC12AF72A7F54', 'SOFRQTD12A81C233C0', 'SOAXGDH12A8C13F8A1', 'SOAUWYT12A81C206F1', 'SOBONKR12A58A7A7E0', 'SODJWHY12A8C142
CCE', 'SOWCKVR12A8C142411', 'SOLFXKT12AB017E3E0', 'SONYKOW12AB01849C9', 'SOEGIYH12A6D4FC0E3']

In [71]: print(recommend('https://www.instagram.com/jasemkhlef/') )

['SOSXLTC12AF72A7F54', 'SOAXGDH12A8C13F8A1', 'SOAUWYT12A81C206F1', 'SOBONKR12A58A7A7E0', 'SODJWHY12A8C142CCE', 'SOWCKVR12A8C142
411', 'SOLFXKT12AB017E3E0', 'SONYKOW12AB01849C9', 'SOEGIYH12A6D4FC0E3', 'SOTWNDJ12A8C143984']

In [72]: print(recommend('https://www.instagram.com/svabhukohli/') )

['SOSXLTC12AF72A7F54', 'SOAUWYT12A81C206F1', 'SOBONKR12A58A7A7E0', 'SODJWHY12A8C142CCE', 'SOWCKVR12A8C142411', 'SOLFXKT12AB017E
3E0', 'SONYKOW12AB01849C9', 'SOEGIYH12A6D4FC0E3', 'SOTWNDJ12A8C143984', 'SOFLJQZ12A6D4FADA6']
```

We'll Get top 10 recommend songs, when given i/p the User.

This Technique does not give Personalised

Recommendations.

Name: Vijaykumar Arun Lokhande

Project name: Insta-Influencer

4.2 Collaborative Filtering (Model Based)

Song Recommendation: Collaborative Filtering(Memory Based)

Code File: SVD Based.ipynb

Dataset: Cleaned_data1.xlsx, Cleaned_data2.xlsx

Objective:

To recommend songs for reels to Influencer tracked by Yoursbiz Ad-Network.

Libraries Used: -

Numpy, Pandas, Sklearn, Math.

Model: -Collaborative Filtering(Using pearson Similarity)

i) Item Based

ii) User Based

Resource:

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html To know more about root mean square evaluation.

http://research.microsoft.com/pubs/115396/EvaluationMetrics.TR.pdf To know more about how recommender systems are evaluated in different sitiations.

- ## **Explanation**:

we'll evaluate both(item based, user based similaraties)togather.

Note:

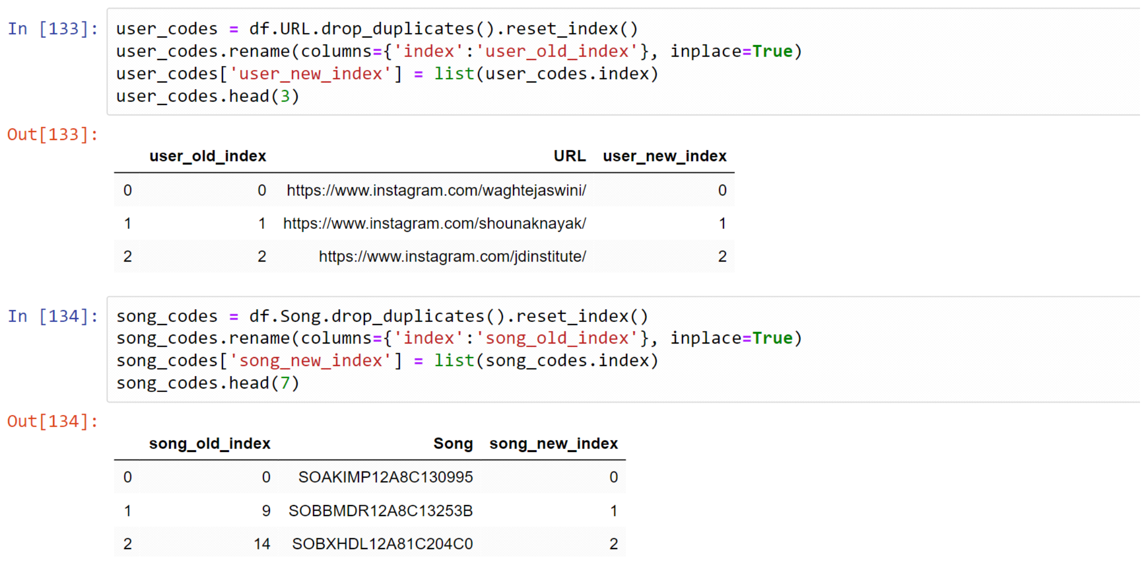
'Delta' = (Curr_ER - AVG ER) if you don't know it, refer documentation of Dataset.

We'll Treate as if they are rating as in movie rating predictions.

- ## 1. Split the Data into Training and Testing:

split data into training and testing (75:25). Converting to testing is assuming we don't have rating for these entries. which will be used to calculate accuaracy of the predictions.

- ## 2. Encode Songs and Influencers:



In [133]:
```
user_codes = df.URL.drop_duplicates().reset_index()
user_codes.rename(columns={'index':'user_old_index'}, inplace=True)
user_codes['user_new_index'] = list(user_codes.index)
user_codes.head(3)
```

Out[133]:

	user_old_index	URL	user_new_index
0	0	https://www.instagram.com/waghtejaswini/	0
1	1	https://www.instagram.com/shounaknayak/	1
2	2	https://www.instagram.com/jdinstitute/	2

In [134]:
```
song_codes = df.Song.drop_duplicates().reset_index()
song_codes.rename(columns={'index':'song_old_index'}, inplace=True)
song_codes['song_new_index'] = list(song_codes.index)
song_codes.head(7)
```

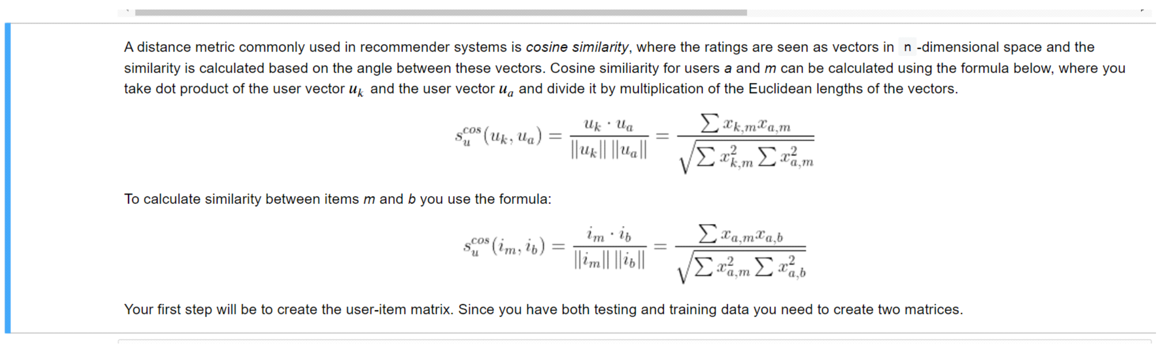
Out[134]:

	song_old_index	Song	song_new_index
0	0	SOAKIMP12A8C130995	0
1	9	SOBBMDR12A8C13253B	1
2	14	SOBXHDL12A81C204C0	2

So that influencer and songs are now encoded/mapped to numbers(integers) which can be then used to make pivot table.

- ## 3. Make the pivot Table:
- ## 4. Create a Distance Matrix:

Brief Theory:



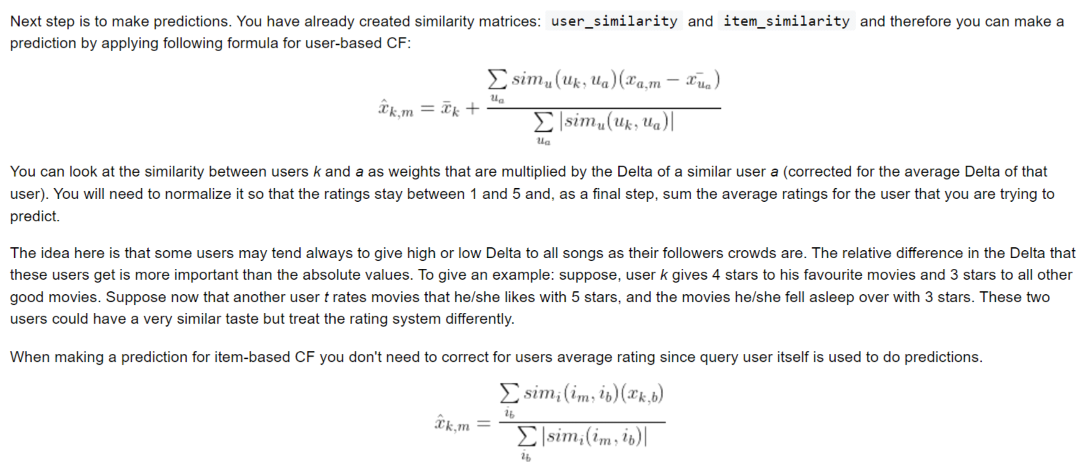
A distance metric commonly used in recommender systems is *cosine similarity*, where the ratings are seen as vectors in `n` -dimensional space and the similarity is calculated based on the angle between these vectors. Cosine similiarity for users *a* and *m* can be calculated using the formula below, where you take dot product of the user vector u_k and the user vector u_a and divide it by multiplication of the Euclidean lengths of the vectors.

$$s_u^{cos}(u_k, u_a) = \frac{u_k \cdot u_a}{||u_k|| \, ||u_a||} = \frac{\sum x_{k,m} x_{a,m}}{\sqrt{\sum x_{k,m}^2 \sum x_{a,m}^2}}$$

To calculate similarity between items *m* and *b* you use the formula:

$$s_u^{cos}(i_m, i_b) = \frac{i_m \cdot i_b}{||i_m|| \, ||i_b||} = \frac{\sum x_{a,m} x_{a,b}}{\sqrt{\sum x_{a,m}^2 \sum x_{a,b}^2}}$$

Your first step will be to create the user-item matrix. Since you have both testing and training data you need to create two matrices.


Next step is to make predictions. You have already created similarity matrices: `user_similarity` and `item_similarity` and therefore you can make a prediction by applying following formula for user-based CF:

$$\hat{x}_{k,m} = \bar{x}_k + \frac{\sum\limits_{u_a} sim_u(u_k, u_a)(x_{a,m} - \bar{x_{u_a}})}{\sum\limits_{u_a} |sim_u(u_k, u_a)|}$$

You can look at the similarity between users *k* and *a* as weights that are multiplied by the Delta of a similar user *a* (corrected for the average Delta of that user). You will need to normalize it so that the ratings stay between 1 and 5 and, as a final step, sum the average ratings for the user that you are trying to predict.

The idea here is that some users may tend always to give high or low Delta to all songs as their followers crowds are. The relative difference in the Delta that these users get is more important than the absolute values. To give an example: suppose, user *k* gives 4 stars to his favourite movies and 3 stars to all other good movies. Suppose now that another user *t* rates movies that he/she likes with 5 stars, and the movies he/she fell asleep over with 3 stars. These two users could have a very similar taste but treat the rating system differently.

When making a prediction for item-based CF you don't need to correct for users average rating since query user itself is used to do predictions.

$$\hat{x}_{k,m} = \frac{\sum\limits_{i_b} sim_i(i_m, i_b)(x_{k,b})}{\sum\limits_{i_b} |sim_i(i_m, i_b)|}$$

here in our case we have delta, which varies from -7 to +5 acts as rating. Each influencer would have different crowd which reacts differently, so as the movie ratings, songs may not be rated/delta in same manner. Therefore we use corrected mean.

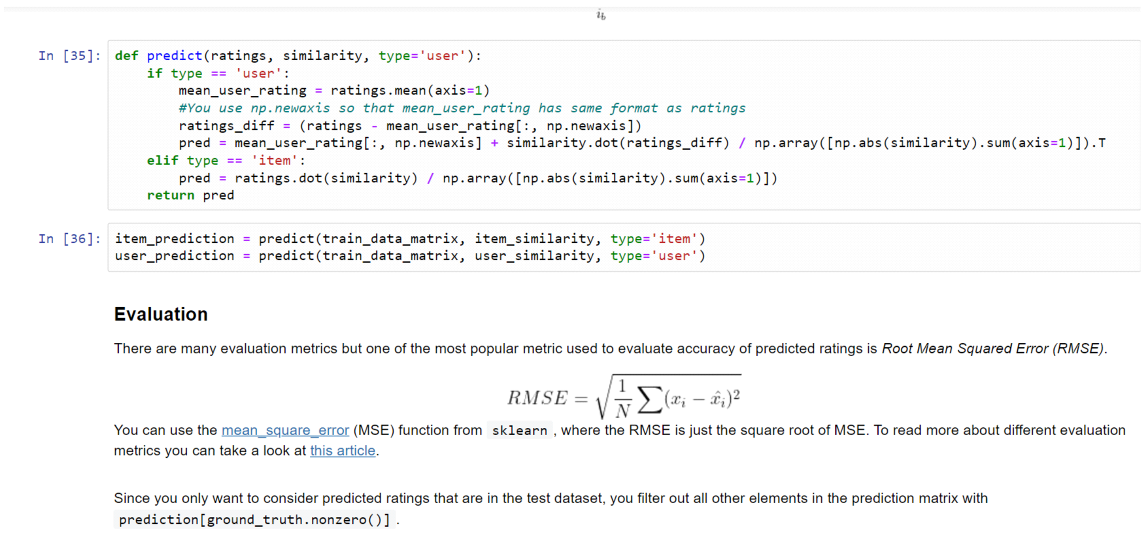
Also there may be many influencers in the data, so calculating distance matrix for user based similarity will be calculation intensive, and heavy. since in that case we need to compute for each user against all remaining users. So sometimes item based similarities are used. As they are computationally cheaper.

see below image for code.

- ## 4. Predict and Evaluate:

evaluate against test data. Using Root Mean square error.

ALSO PLEASE FIND THE BRIEF THEORY IN NOTEBOOKS.


i_b

```
In [35]: def predict(ratings, similarity, type='user'):
             if type == 'user':
                 mean_user_rating = ratings.mean(axis=1)
                 #You use np.newaxis so that mean_user_rating has same format as ratings
                 ratings_diff = (ratings - mean_user_rating[:, np.newaxis])
                 pred = mean_user_rating[:, np.newaxis] + similarity.dot(ratings_diff) / np.array([np.abs(similarity).sum(axis=1)]).T
             elif type == 'item':
                 pred = ratings.dot(similarity) / np.array([np.abs(similarity).sum(axis=1)])
             return pred
```

```
In [36]: item_prediction = predict(train_data_matrix, item_similarity, type='item')
         user_prediction = predict(train_data_matrix, user_similarity, type='user')
```

Evaluation

There are many evaluation metrics but one of the most popular metric used to evaluate accuracy of predicted ratings is *Root Mean Squared Error (RMSE)*.

$$RMSE = \sqrt{\frac{1}{N}\sum(x_i - \hat{x}_i)^2}$$

You can use the mean_square_error (MSE) function from `sklearn`, where the RMSE is just the square root of MSE. To read more about different evaluation metrics you can take a look at this article.

Since you only want to consider predicted ratings that are in the test dataset, you filter out all other elements in the prediction matrix with `prediction[ground_truth.nonzero()]`.

Result:

User-based CF RMSE: 2.8828404060427064

Item-based CF RMSE: 2.8853598092386368. Which means the results are quite good.

By arranging the predicted ratings in decreasing order and skipping songs that are already used by influencer, we can recommend songs to the influencer

4.3 Collaborative Filtering(Memory Based)

Name: Vijaykumar Arun Lokhande

Project name: Insta-Influencer

Song Recommendation: Collaborative Filtering(Memory Based)

Code File: CF(KNN Model).ipynb

Dataset: Cleaned_data1.xlsx, Cleaned_data2.xlsx

Objective:

To recommend songs for reels to Influencer tracked by Yoursbiz Ad-Network.

Libraries Used: -

Numpy, Pandas, Sklearn, Scipy.

Model: -Collaborative Filtering(KNN)

Resource:

https://www.youtube.com/watch?v=_hf_y-_sj5Y&list=PLZoTAELRMXVN7QGpcuN-Vg35Hgjp3htvi Recommendation system playlist.

- ## **Explanation**:

we'll evaluate both(item based, user based similaraties)togather.

Note:

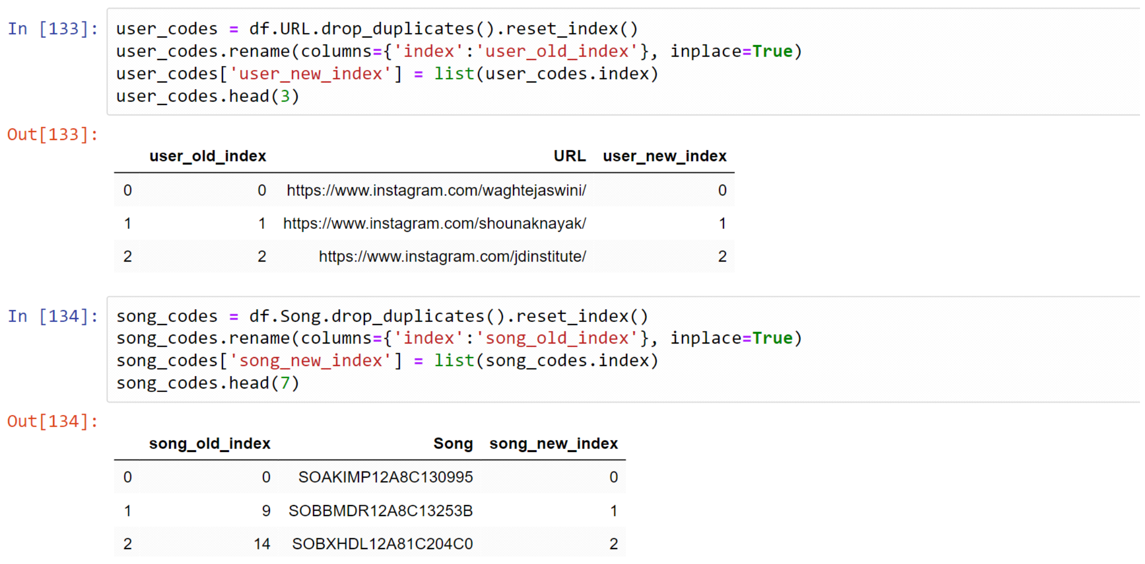
'Delta' = (Curr_ER - AVG ER) if you don't know it, refer documentation of Dataset.

We'll Treate as if they are rating as in movie rating predictions.

- ## 1. Split the Data into Training and Testing:

split data into training and testing (75:25). Converting to testing is assuming we don't have rating for these entries. which will be used to calculate accuaracy of the predictions.

- ## 2. Encode Songs and Influencers:



```
In [133]: user_codes = df.URL.drop_duplicates().reset_index()
          user_codes.rename(columns={'index':'user_old_index'}, inplace=True)
          user_codes['user_new_index'] = list(user_codes.index)
          user_codes.head(3)
```

Out[133]:

	user_old_index	URL	user_new_index
0	0	https://www.instagram.com/waghtejaswini/	0
1	1	https://www.instagram.com/shounaknayak/	1
2	2	https://www.instagram.com/jdinstitute/	2

```
In [134]: song_codes = df.Song.drop_duplicates().reset_index()
          song_codes.rename(columns={'index':'song_old_index'}, inplace=True)
          song_codes['song_new_index'] = list(song_codes.index)
          song_codes.head(7)
```

Out[134]:

	song_old_index	Song	song_new_index
0	0	SOAKIMP12A8C130995	0
1	9	SOBBMDR12A8C13253B	1
2	14	SOBXHDL12A81C204C0	2

So that influencer and songs are now encoded/mapped to numbers(integers) which can be then used to make pivot table.

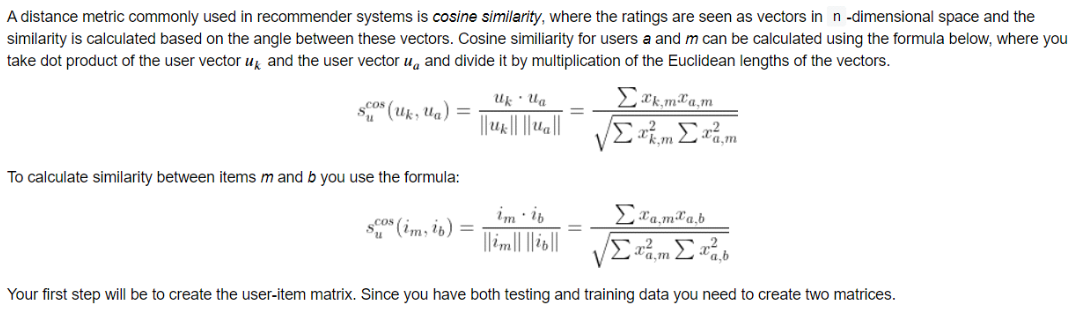
3. Make the pivot Table:

4. Choose a song as i/p:

5. Model Fit

K-nearest neighbor finds the k most similar items to a particular instance based on a given distance metric like euclidean, jaccard similarity , minkowsky or custom distance measures. In this my model, I used to cosine as metric.

Cosine Similarity:



A distance metric commonly used in recommender systems is *cosine similarity*, where the ratings are seen as vectors in `n` -dimensional space and the similarity is calculated based on the angle between these vectors. Cosine similiarity for users *a* and *m* can be calculated using the formula below, where you take dot product of the user vector u_k and the user vector u_a and divide it by multiplication of the Euclidean lengths of the vectors.

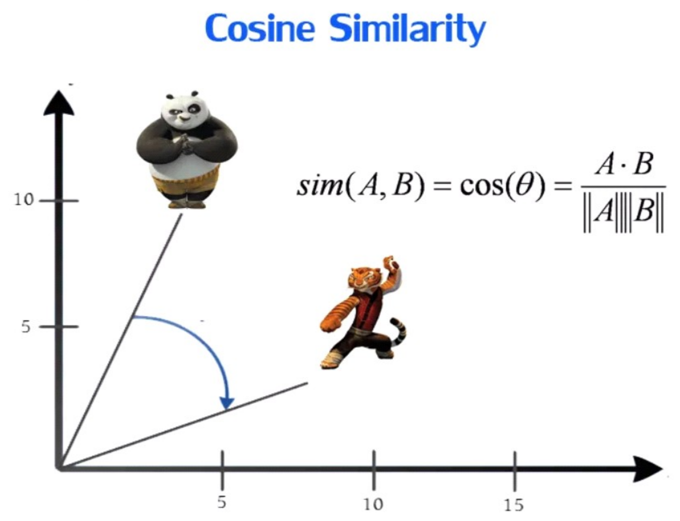
$$s_u^{cos}(u_k, u_a) = \frac{u_k \cdot u_a}{||u_k|| \, ||u_a||} = \frac{\sum x_{k,m}x_{a,m}}{\sqrt{\sum x_{k,m}^2 \sum x_{a,m}^2}}$$

To calculate similarity between items *m* and *b* you use the formula:

$$s_u^{cos}(i_m, i_b) = \frac{i_m \cdot i_b}{||i_m|| \, ||i_b||} = \frac{\sum x_{a,m}x_{a,b}}{\sqrt{\sum x_{a,m}^2 \sum x_{a,b}^2}}$$

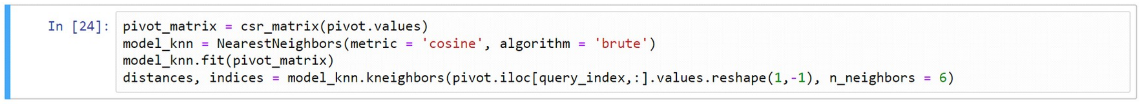
Your first step will be to create the user-item matrix. Since you have both testing and training data you need to create two matrices.

Cosine Similarity



$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

==

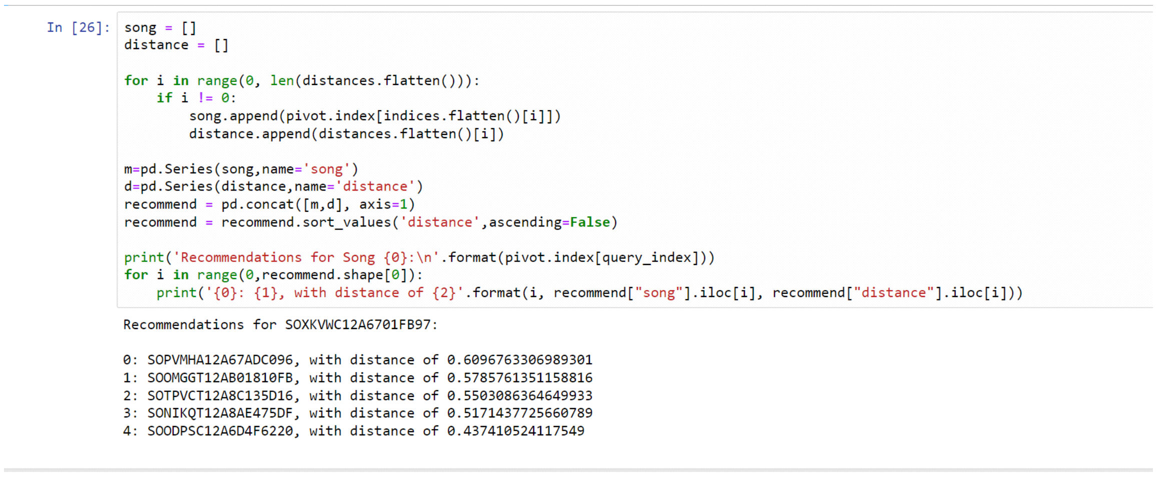


```
In [24]: pivot_matrix = csr_matrix(pivot.values)
         model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
         model_knn.fit(pivot_matrix)
         distances, indices = model_knn.kneighbors(pivot.iloc[query_index,:].values.reshape(1,-1), n_neighbors = 6)
```

It is calculated similarity items in user vs item matrix. For example, we let think that there are two movies: Lord of the Rings and Hobbit. Three people watched lord of the rings and hobbit. If fourth person watched lord of the rings. He/she could like Hobbit. So that the system recommends Hobbit to fourth people.

In general recommendation systems use to item based collaborative filtering. Item based CF improved to solve the problem of user based CF. As people minds and habits can change and items doesn't change. It is prefered.

- ## 6. Predict and Evaluate:



```
In [26]: song = []
         distance = []

         for i in range(0, len(distances.flatten())):
             if i != 0:
                 song.append(pivot.index[indices.flatten()[i]])
                 distance.append(distances.flatten()[i])

         m=pd.Series(song,name='song')
         d=pd.Series(distance,name='distance')
         recommend = pd.concat([m,d], axis=1)
         recommend = recommend.sort_values('distance',ascending=False)

         print('Recommendations for Song {0}:\n'.format(pivot.index[query_index]))
         for i in range(0,recommend.shape[0]):
             print('{0}: {1}, with distance of {2}'.format(i, recommend["song"].iloc[i], recommend["distance"].iloc[i]))
```

```
Recommendations for SOXKVWC12A6701FB97:

0: SOPVMHA12A67ADC096, with distance of 0.6096763306989301
1: SOOMGGT12AB01810FB, with distance of 0.5785761351158816
2: SOTPVCT12A8C135D16, with distance of 0.5503086364649933
3: SONIKQT12A8AE475DF, with distance of 0.5171437725660789
4: SOODPSC12A6D4F6220, with distance of 0.437410524117549
```

- take distances of the all songs with the i/p song.

- sort the distances in decreasing order

- here the distance is value between 0-1. since cos function have o/p upto 1. cos(0)=1

implies more closer the songs the value be will close to 1. Less similar the song, distance will be near to zero.

Result:

The Songs which are more closer/symmetrical in behavioural user will be recommended.

ex. i/p => SOXKVWC12A6701FB97:

o/p => 0: SOPVMHA12A67ADC096, with distance of 0.6096763306989301

1: SOOMGGT12AB01810FB, with distance of 0.5785761351158816

2: SOTPVCT12A8C135D16, with distance of 0.5503086364649933

3: SONIKQT12A8AE475DF, with distance of 0.5171437725660789

4: SOODPSC12A6D4F6220, with distance of 0.437410524117549

Name: Vijaykumar Arun Lokhande

Project name: Insta-Influencer

4.4 Matrix Factorization (SVD)

Song Recommendation: Matrix Factorization (SVD)

Code File: SVD Based.ipynb

Dataset: Cleaned_data1.xlsx, Cleaned_data2.xlsx

Objective:

To recommend songs for reels to Influencer tracked by Yoursbiz Ad-Network.

Libraries Used: -

Numpy, Pandas, Sklearn, Scipy.

Model: -Matrix Factorization using Singular Value Decomposition.

Resource:

https://www.youtube.com/watch?v=ZspR5PZemcs&t=6s Intuition for Matrix Factorization

https://www.youtube.com/watch?v=nbBvuuNVfco for mathematics behind singular value decomposition

Also refer some article if you still want to dig deeper.

Explanation: -

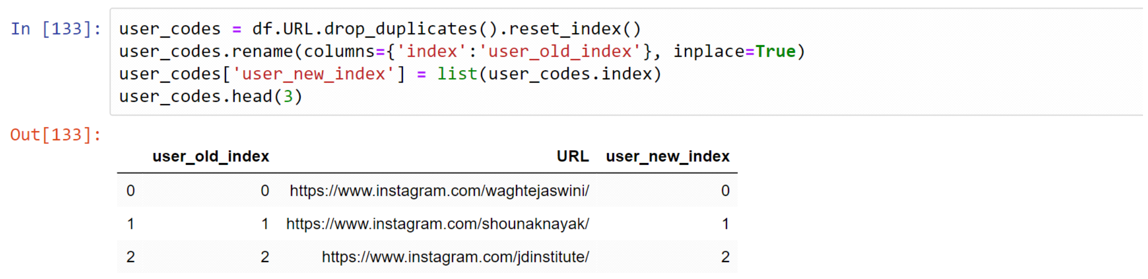
'Delta' = (Curr_ER - AVG ER) if you don't know it, refer documentation of Dataset.

We'll Treate as if they are rating as in movie rating predictions.

- ### 1. Split the Data into Training and Testing:

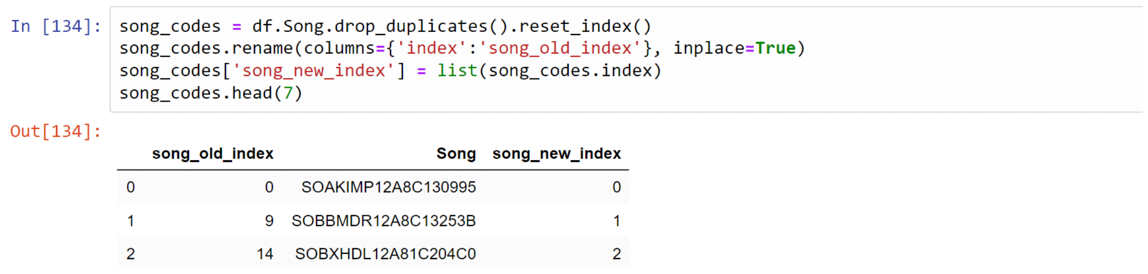
split data into training and testing (75:25). Converting to testing is assuming we don't have rating for these entries. which will be used to calculate accuaracy of the predictions.

- ### 2. Encode Songs and Influencers:



```
In [133]: user_codes = df.URL.drop_duplicates().reset_index()
          user_codes.rename(columns={'index':'user_old_index'}, inplace=True)
          user_codes['user_new_index'] = list(user_codes.index)
          user_codes.head(3)
Out[133]:
```

	user_old_index	URL	user_new_index
0	0	https://www.instagram.com/waghtejaswini/	0
1	1	https://www.instagram.com/shounaknayak/	1
2	2	https://www.instagram.com/jdinstitute/	2



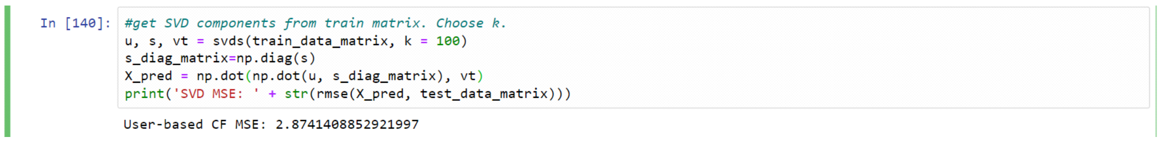
```
In [134]: song_codes = df.Song.drop_duplicates().reset_index()
          song_codes.rename(columns={'index':'song_old_index'}, inplace=True)
          song_codes['song_new_index'] = list(song_codes.index)
          song_codes.head(7)
Out[134]:
```

	song_old_index	Song	song_new_index
0	0	SOAKIMP12A8C130995	0
1	9	SOBBMDR12A8C13253B	1
2	14	SOBXHDL12A81C204C0	2

So that influencer and songs are now encoded/mapped to numbers(integers) which can be then used to make pivot table

- ## 3. Make the pivot Table:
- ## 4. Use SVD to factorise:

Even though mathematics is much complex to understand. It is just few lines of code to get it all done!



```
In [140]: #get SVD components from train matrix. Choose k.
          u, s, vt = svds(train_data_matrix, k = 100)
          s_diag_matrix=np.diag(s)
          X_pred = np.dot(np.dot(u, s_diag_matrix), vt)
          print('SVD MSE: ' + str(rmse(X_pred, test_data_matrix)))

          User-based CF MSE: 2.8741408852921997
```

- ## 5. Get Prediction and Evaluate Accuracy:

ALSO PLEASE FIND THE BRIEF THEORY IN NOTEBOOKS.

Result:

SVD Based RMSE(Root Mean Square Error): 2.8741408852921997. Which means the results are quite good.

By arranging the predicted ratings in decreasing order and skipping songs that are already used by influencer, we can recommend songs to the influencer.

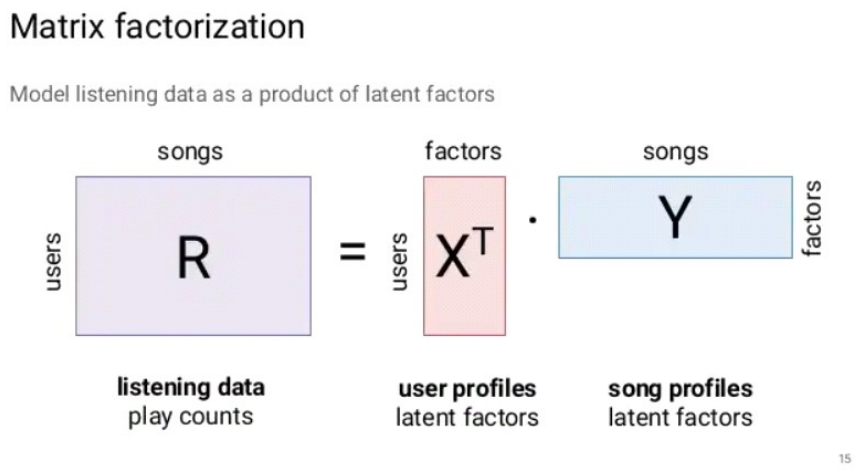
Brief Theory of this Method:

The starting point of any matrix factorization-based method is the utility matrix, a matrix of user Vs item dimension. Not, this is a sparse matrix, since not all item is used by the user. The process of matrix factorization means finding out a low rank approximation of the utility matrix. So we want to break down the utility matrix U into two low rank matrices so that we can recreate the matrix U by multiplying those two matrices:

Assuming the process helps us identify latent factors/features, meaning as K, our aim is to find two matrices X and Y such that their product (matrix multiplication) approximates R.

X = |U| x K matrix (A matrix with dimensions of num_users * factors)

Y = |P| x K matrix (A matrix with dimensions of factors * num_songs)



To make a recommendation to the user, we can multiply the corresponding user's row from the first matrix by the item matrix and determine the items from the row with maximum ratings. That will become our recommendations for the user. The first matrix represents the association between the users and the latent features, while the second matrix takes care of the associations between items (songs in our case) and the latent features.

There are multiple algorithms available for determining factorization of any matrix. We use one of the simplest algorithms, which is the singular value decomposition or SVD. You can follow these steps to determine the factorization of a matrix using the output of SVD function.

- Factorize the matrix to obtain U, S, and V matrices.

- Reduce the matrix S to first k components. (The function we are using will only provide k dimensions, so we can skip this step.)

- Compute the square root of reduced matrix Sk to obtain the matrix $Sk^{1/2}$</sup>.

- Compute the two resultant matrix $U*Sk^{1/2}$and $Sk^{1/2}*V$ as these will serve as our two factorized matrices

We can then generate the prediction of user i for product j by taking the dot product of the i th row of the first matrix with the j th column of the second matrix.

5. Results:

We got best results for SVD based latent factor model and memory based collaborative filtering algorithm. Our gives better results than popularity based model. It lags behind collaborative filtering algorithm because the ma_trix was too sparse which prevented objective functions to converge to global optimum. Our K-NN model did not work well and performs worse than even the popularity model. The reason behind this is that we have the features of only 10,000 songs, which is less than 3 % of the whole dataset so only some of these 10,000 songs could be recommended. The huge lack of information leads to the bad performance of this method.

6. Summary:

1. Popularity Based:

i) Count: Filter out songs which has user_used_count<100. Sort All the song in Descending Order. Recommend Top 20 Songs which is never used by user/influencer in the past. Evaluate the result with Mean Average Precision.

ii) More Engaging + Count: Filter out entries of the dataframe such that Change in Engagement for that entry is > +(%4). Now Sort the entries based on user_counts (user_counts -> The number of users for which the song proved to have more than 4% change in engagement when used in reels), recommend the top 10 entries

iii) Celebrity Usage: Celebrities are influencers with more followers. Filter out all entries corresponding to Celebrities. The More celebrities use the song with better ER will be recommended to the users.

Collaborative Filtering:

1. Model Based CF: We'll evaluate change in ER (Delta) that the corresponding song drives/produce for the user. It will act as 'Rating in the movie' for our model. We'll Split the data into training and testing. We'll Create a pivot table. Predict the similarity based on KNN using brute algo based on Cosine Similarity. We'll Associate a product with more similar products based on user preferences. Final recommendation will be songs closest to a given song.

2. Memory Based CF: We'll evaluate change in ER that the corresponding song drives for the user. It will act as 'Rating in the movie' for our model. We'll Split the data into training and testing. We'll Create a pivot table, and predict using Pearson's Similarity. Then we'll verify the results with Test data using RMSE(Root Mean Square Error).

i) Item-based CF RMSE: 2.8853598092386368

ii) User-based CF RMSE: 2.8828404060427064

Matrix Factorization Based on Singular Value Decomposition: We'll Factorize pivot matrix mentioned above in two/three smaller matrix. There are several latent features that are hidden(latent), which can be approximated. Maths is a little complex. So we can predict ratings for all possible combinations. We'll evaluate predictions using RMSE(Root Mean Square Error).

i) SVD Based RMSE: 2.8741408852921997