SONGS RECOMMENDATION USING SVD

Shreya Kumari, Neeraj

Ch. Brahm Prakash Government Engineering College, New Delhi, India

ABSTRACT

Even if a wide variety of songs are listened to these days, algorithms continue to need help in many areas. How does the system determine whether people enjoy a new song or artist when there are fewer historical figures? How do you choose which music to recommend to brand-new users? From this angle, the suggested research project attempts to evaluate the probability that a user would keep listening to the song after the initial apparent listening experience starts within the given time range.

The user's objective is 1, and 0 is determined if, within a month of the initial apparent listening event, there is a recurrent auditory occurrence.

There is also usage of methods like content-based filtering, factorization machines (FM), singular value decomposition (SVD), and collaborative filtering. Finally, SVD and FM are used to hybridise the suggested system.

INTRODUCTION

Since the majority of music is now purchased and listened to digitally, automated song recommendation is becoming more and more crucial. The music industry has shifted more and more in recent years towards digital distribution with the establishment of online music stores and customer services like Google Play, Grooveshark, Spotify, and iTunes [1, 2].

Hybrid recommenders integrate recommendation processes to decrease specific policy issues and increase assessment accuracy overall. To address this, mixed recommendations integrate content-based and social suggestions into various configurations across various applications [3, 4].

METHODOLOGY

A. Survey of Datasets

Seven million three hundred seventy-seven thousand four hundred eighteen observations total from the original dataset were included in it; each word relates to the first apparent auditory occurrence for a specific user's song pair [5].

The data is selected for our training models by choosing users with 500 events and at least 1000 songs. Of the filtered dataset, 75% is used for training, and the remaining 25% is tested. As a result, there are 289,928 observations in the testing set and 869,783 observations in the training set. The songs' metadata is also looked at. The characteristics will come in handy for FM and content-based models.

B. Collaborative Filtering

By examining how the other person has rated the musical compositions, interaction methods suggest them to the user. Consider, for example, that there is a picky user who adores component A. However, a few others love A and B; therefore, the consumer will recommend B. This approach has been practical and is frequently utilised in e-commerce systems (such as Amazon.com and the iTunes music store) [5].

Two groups of collaborative systems are distinguished:

1. User-Based Collaborative Filtering (UBCF):

As illustrated in Figure 1, UBCF applies that logic to make recommendations for items by identifying similar consumers to the dynamic consumer (to whom the recommended music is directed) [6]. One of its specific functions is the Nearest Neighbour algorithm, which is based on user input.

2. Item-Based Collaborative Filtering (IBCF):

This filtering technique, also known as item-item collaborative filtering or item-based collaborative filtering, is used for recommendation systems. Its basis is the similarity between the items, which is determined by using user ratings of those items [7], as illustrated in Figure 1.

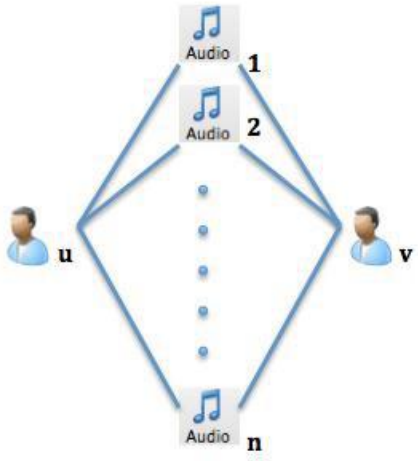


Fig. 1 User based collaborative filtering

Use the Recommender lab package as a basis for future models; these are essentially based on item- and user-based collaborative filter models.

"User-Based Collaborative Filter Accuracy was: 0.6743" and "Item-Based Collaborative Filter Accuracy was: 0.7073" are two of the model's key parameters.

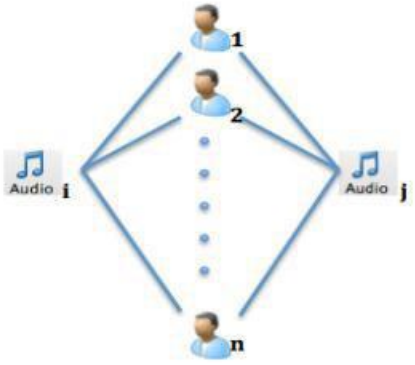


Fig. 2 Example of Item based collaborative filtering

C. Machines for Factorization

The factorization machine is examined first. In essence, a factorization machine weights higher-order interactions by the inner product of latent vectors, allowing us to model datasets rich in features [8, 9].

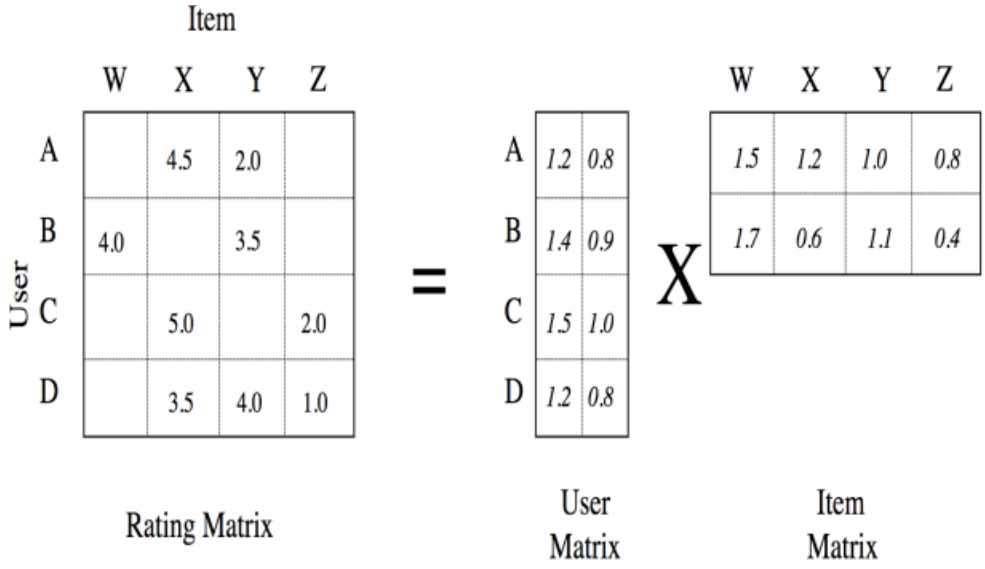


Fig. 3 Example of factorization machine

Thus, even in highly scant data, the key traits are calculated! The parameters that make up the factorization machine are as follows, which need to be learned (where n is the number of features in the dataset and k is the dimensionality of latent features):

According to the parameter adjustment for the previous section, latent dimensions provide the best accuracy when k = 25 is used. Furthermore, the fastest error rate convergence is achieved with an initialized standard deviation of 0.1.

Therefore, for the final model, these values must be set for the hyperparameters. In this instance, the final factorization machine model is trained, and it is subsequently applied to produce predictions for dataset testing.

Lastly, the accuracy, precision, recall, and error metrics assess the suggested model. To begin, a single generic evaluation function has been written. This will be used to compare the performance of the various models. The newly created role is called across the FM model's output probabilities.

• It is then recognised that the approximate accuracy of 0.7482 is significantly greater than the item-based (i.e., 0.71) and user-based (0.67) filtering. Additionally, it shows that our model's coverage is accurate because it predicts a value of 1 for both the most well-known songs and the most addicted or active users. Nonetheless, the level of popularity of the selected music may have an impact on our prediction accuracy.

The content-based techniques recommend short musical segments based on the attributes of the user's preferred songs. This caters to a broad spectrum of vocalists; that is, several works are recommended even after they have yet to be ranked [11, 12].

D. Content Based Recommender

Content-based techniques are designed to use scenarios in which the objects can be explained using factual feature sets and the functionality of the content-based recommender system, as seen in Figure 4 below.

However, these methods have significant issues with the accuracy of recommendations since the potential content similarity between them is simply one of many factors that characterise consumer preferences.

Furthermore, adding user-approved songs with meaningful music can be challenging when using an actual database where most users provide lower ratings for the tracks.

Content-based systems are primarily used when a sufficient amount of characteristic knowledge is genuinely available, as opposed to when it is not. In this instance, structured data is obtained from song duration, lyricist, composer, performer genres and the corresponding language.

1. Pre-processing and Feature Extraction: During the feature mining step, the various objects' descriptions are acquired. Nonetheless, since this technique is prevalent, it probably uses any interpretation, including a multidimensional information interpretation.

For the objects represented in feature space, we will create a single desirable encoded vector.

2. KNN classification model: One of the simplest classification methods is the closest neighbour classifier, which has a very straightforward application. Based on each user's ratings for the items, we created a model for them. Every user may view content-based recommender systems as a categorization challenge [13].

For every neighbour that is a part of the content-based model, the overall accuracy is 0.6880569. We may conclude from this that the KNN - content based recommendation system will get more accurate as the number of KNN neighbours increases.

The content-based model beat the user-based collaborative filtering strategy when we compared it to other collaborative filtering models. However, the KNN model's neighbour count was actually limited to 32.

E. Singular Value Decomposition (SVD)

The SVD algorithm is a matrix factorization process that lowers the amount of extraneous parameters in the dataset; Figure 5 illustrates how this model operates. By breaking down the rating matrix and reducing its space dimensions from r to k, the SVD model, for a large M×N user-item rating matrix A of rank r, effectively maps the users and items to a joint latent factor space with k singular values. The latent factor space automatically uses the user's responses to explain the rankings by characterizing the things and the users on the topics.

1. Proposed Model

Initially, we will transform the given data into an M×N rating matrix containing all songs and users, where each element 'r' indicates whether or not the user plans to listen to music 'I' again. If a user-song pair is absent, NA will be used in its place.

To formulate our prediction, we will substitute "NA" for the detected values in the original test data. In an upcoming section on model evaluation, we will be contrasting model predictions with actual values.

Next, we use a train proportion 0.8 to split the matrix into training and testing groups.

Next, we used the training matrix to begin training our models.

We utilised five different k values to select the ideal parameter afterwards.

2. Adjustment Factors:

To select the most advantageous latent component, we tested our SVD model with k=5, 8, 10, 20, 40) and compared their training RMSE. More latent variables will, of course, produce more accurate results, but we also considered the fact that the training and prediction times differ significantly with different k values.

It is evident from RSME and our prediction accuracy that a higher value of k will not significantly enhance the model. Consequently, we determine that 10 is the ideal value of k.

With k = 10, the SVD approximation's final accuracy is 0.71.

F. Hybridisation

Conventional recommendation system techniques, such as knowledge-based, content-based, and collaborative filtering, all have advantages and disadvantages of their own.

For example, collaborative filtering faces issues with sparsity and cold starts, whereas content-based approaches face issues with narrowness and a high description need.

However, a hybrid approach can use a way to provide projections in areas where the other system is unable to, resulting in a more robust recommendation scheme.

* We found that using a novel approach to combine the outputs from SVD and Factorization Machines to create a deep neural network-based recommendation system improves prediction accuracy in the work that is being recommended.

• After noting the results of the SVD and Factorization Machines, we handle this as a classification problem to produce hybrid song recommendations.

* Using the testing data portion of the original data, we construct the hybrid recommender. To create our model, we further divided the original testing data into separate training and testing sets.

1. Adjusting Parameters

To obtain the optimal feature set, we train our deep learning model on a range of net sizes and decay rates.

2. Accuracy

We compared the obtained accuracy to the output of the factorization machine and the SVD output. We found a modest improvement in the accuracy of the neural net hybrid recommendation system. The following part will go over our findings in more detail.

RESULT

In this work, we developed several cutting-edge recommendation systems, including a Content-Based Recommender System, SVD, User-Based Collaborative Filtering (UBCF), Item-Based Collaborative Filtering (IBCF), and Factorization Machines.

Before implementing the models, we have to import the important libraries and data set in the jupyter notebook.

```
%matplotlib inline

import pandas
from sklearn.cross_validation import train_test_split
import numpy as np
import time
from sklearn.externals import joblib
import Recommenders as Recommenders
import Evaluation as Evaluation
```

(figure 4)

Importing libraries

```
#Read userid-songid-listen_count triplets
#This step might take time to download data from external sources
triplets_file = 'https://static.turi.com/datasets/millionsong/10000.txt'
songs_metadata_file = 'https://static.turi.com/datasets/millionsong/song_data.csv'

song_df_1 = pandas.read_table(triplets_file,header=None)
song_df_1.columns = ['user_id', 'song_id', 'listen_count']

#Read song  metadata
song_df_2 =  pandas.read_csv(songs_metadata_file)

#Merge the two dataframes above to create input dataframe for recommender systems
song_df = pandas.merge(song_df_1, song_df_2.drop_duplicates(['song_id']), on="song_id", how="left")
```

(figure 5)

Loading data

The loaded data is very huge so we extract the subset of the data.

```
song_df = song_df.head(10000)

#Merge song title and artist_name columns to make a merged column
song_df['song'] = song_df['title'].map(str) + " - " + song_df['artist_name']
```

(figure 6)

Subset of data

1. Popularity Based Model:

```
song_grouped = song_df.groupby(['song']).agg({'listen_count': 'count'}).reset_index()
grouped_sum = song_grouped['listen_count'].sum()
song_grouped['percentage']  = song_grouped['listen_count'].div(grouped_sum)*100
song_grouped.sort_values(['listen_count', 'song'], ascending = [0,1])
```

(figure 7)

Dividing the data set into train set and test set.

```
train_data, test_data = train_test_split(song_df, test_size = 0.20, random_state=0)
print(train_data.head(5))
```

(figure 8)

```
pm = Recommenders.popularity_recommender_py()
pm.create(train_data, 'user_id', 'song')
```

Use the popularity model to make some predictions

```
user_id = users[5]
pm.recommend(user_id)
```

(figure 9)

Instance of Recommenders class Output of popularity-based model:

	user_id	song	score	Rank
3194	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Sehr kosmisch - Harmonia	37	1
4083	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Undo - Björk	27	2
931	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Dog Days Are Over (Radio Edit) - Florence + Th...	24	3
4443	4bd88bfb25263a75bbdd467e74018f4ae570e5df	You're The One - Dwight Yoakam	24	4
3034	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Revelry - Kings Of Leon	21	5
3189	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Secrets - OneRepublic	21	6
4112	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Use Somebody - Kings Of Leon	21	7
1207	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Fireflies - Charttraxx Karaoke	20	8
1577	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Hey_ Soul Sister - Train	19	9
1626	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Horn Concerto No. 4 in E flat K495: II. Romanc...	19	10

(figure 10)

2. Item based collaborative filtering:

Creating an instance of Item_similarity_recommender class.

```
is_model = Recommenders.item_similarity_recommender_py()
is_model.create(train_data, 'user_id', 'song')
```

(figure 11)

Getting recommendation for song.

```
#Print the songs for the user in training data
user_id = users[5]
user_items = is_model.get_user_items(user_id)
#
print("----------------------------------------------------------------------------------------")
print("Training data songs for the user userid: %s:" % user_id)
print("----------------------------------------------------------------------------------------")

for user_item in user_items:
    print(user_item)

print("----------------------------------------------------------------------")
print("Recommendation process going on:")
print("----------------------------------------------------------------------")

#Recommend songs for the user using personalized model
is_model.recommend(user_id)
```

(figure 12)

Output for Item based collaborative filtering:

```
----------------------------------------------------------------------------------------
Training data songs for the user userid: 4bd88bfb25263a75bbdd467e74018f4ae570e5df:
----------------------------------------------------------------------------------------
Just Lose It - Eminem
Without Me - Eminem
16 Candles - The Crests
Speechless - Lady GaGa
Push It - Salt-N-Pepa
Ghosts 'n' Stuff (Original Instrumental Mix) - Deadmau5
Say My Name - Destiny's Child
My Dad's Gone Crazy - Eminem / Hailie Jade
The Real Slim Shady - Eminem
Somebody To Love - Justin Bieber
Forgive Me - Leona Lewis
Missing You - John Waite
Ya Nada Queda - Kudai
----------------------------------------------------------------------
Recommendation process going on:
----------------------------------------------------------------------
No. of unique songs for the user: 13
no. of unique songs in the training set: 4483
```

(figure 13)

	user_id	song	score	rank
0	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Superman - Eminem / Dina Rae	0.088692	1
1	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Mockingbird - Eminem	0.067663	2
2	4bd88bfb25263a75bbdd467e74018f4ae570e5df	I'm Back - Eminem	0.065385	3
3	4bd88bfb25263a75bbdd467e74018f4ae570e5df	U Smile - Justin Bieber	0.064525	4
4	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Here Without You - 3 Doors Down	0.062293	5
5	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Hellbound - J-Black & Masta Ace	0.055769	6
6	4bd88bfb25263a75bbdd467e74018f4ae570e5df	The Seed (2.0) - The Roots / Cody Chestnutt	0.052564	7
7	4bd88bfb25263a75bbdd467e74018f4ae570e5df	I'm The One Who Understands (Edit Version) - War	0.052564	8
8	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Falling - Iration	0.052564	9
9	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Armed And Ready (2009 Digital Remaster) - The ...	0.052564	10

(figure 14)

3. SVD matrix factorization based collaborative filtering:

Importing the libraries.

```
import math as mt
import csv
from sparsesvd import sparsesvd #used for matrix factorization
import numpy as np
from scipy.sparse import csc_matrix #used for sparse matrix
from scipy.sparse.linalg import * #used for matrix multiplication
```

(figure 15)

Implementation.

```
def computeSVD(urm, K):
    U, s, Vt = sparsesvd(urm, K)

    dim = (len(s), len(s))
    S = np.zeros(dim, dtype=np.float32)
    for i in range(0, len(s)):
        S[i,i] = mt.sqrt(s[i])

    U = csc_matrix(np.transpose(U), dtype=np.float32)
    S = csc_matrix(S, dtype=np.float32)
    Vt = csc_matrix(Vt, dtype=np.float32)

    return U, S, Vt

#Compute estimated rating for the test user
def computeEstimatedRatings(urm, U, S, Vt, uTest, K, test):
    rightTerm = S*Vt

    estimatedRatings = np.zeros(shape=(MAX_UID, MAX_PID), dtype=np.float16)
    for userTest in uTest:
        prod = U[userTest, :]*rightTerm
        #we convert the vector to dense format in order to get the indices
        #of the movies with the best estimated ratings
        estimatedRatings[userTest, :] = prod.todense()
        recom = (-estimatedRatings[userTest, :]).argsort()[:250]
    return recom
```

(figure 16)

Quantitative comparison between the models

We now formally compare the popularity and the personalized models using precision-recall curves.

precision recall calculator class to calculate the evaluation measures

```
start = time.time()

#Define what percentage of users to use for precision recall calculation
user_sample = 0.05

#Instantiate the precision_recall_calculator class
pr = Evaluation.precision_recall_calculator(test_data, train_data, pm, is_model)

#Call method to calculate precision and recall values
(pm_avg_precision_list, pm_avg_recall_list, ism_avg_precision_list, ism_avg_recall_list) = pr.calculate_measures(user_sample)

end = time.time()
print(end - start)
```

(figure 17)

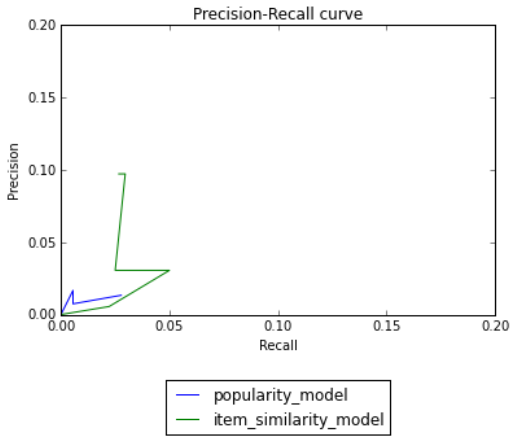
Code to plot precision recall curve¶

```
import pylab as pl

#Method to generate precision and recall curve
def plot_precision_recall(m1_precision_list, m1_recall_list, m1_label, m2_precision_list, m2_recall_list, m2_label):
    pl.clf()
    pl.plot(m1_recall_list, m1_precision_list, label=m1_label)
    pl.plot(m2_recall_list, m2_precision_list, label=m2_label)
    pl.xlabel('Recall')
    pl.ylabel('Precision')
    pl.ylim([0.0, 0.20])
    pl.xlim([0.0, 0.20])
    pl.title('Precision-Recall curve')
    #pl.legend(loc="upper right")
    pl.legend(loc=9, bbox_to_anchor=(0.5, -0.2))
    pl.show()
```

```
print("Plotting precision recall curves.")

plot_precision_recall(pm_avg_precision_list, pm_avg_recall_list, "popularity_model",
                      ism_avg_precision_list, ism_avg_recall_list, "item_similarity_model")
```

(figure 18)

Plotting precision recall curves.



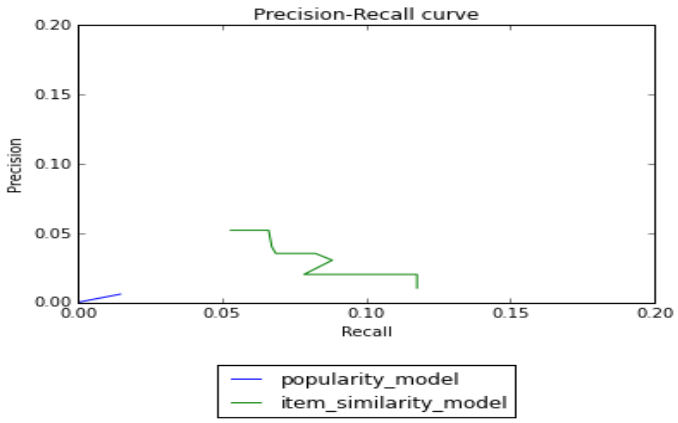
(figure 19)

Generate Precision Recall curve using pickled results on a larger data subset.

```
print("Plotting precision recall curves for a larger subset of data (100,000 rows) (user sample = 0.005).")

#Read the persisted files
pm_avg_precision_list = joblib.load('pm_avg_precision_list_3.pkl')
pm_avg_recall_list = joblib.load('pm_avg_recall_list_3.pkl')
ism_avg_precision_list = joblib.load('ism_avg_precision_list_3.pkl')
ism_avg_recall_list = joblib.load('ism_avg_recall_list_3.pkl')

print("Plotting precision recall curves.")
plot_precision_recall(pm_avg_precision_list, pm_avg_recall_list, "popularity_model",
                      ism_avg_precision_list, ism_avg_recall_list, "item_similarity_model")
```

(figure 20)

Output:



(figure 21)

CONCLUSION

Additionally, we have used deep neural networks to hybridise Factorization Machines and SVD models to improve the recommendation system's overall accuracy.

We observed that in the content-based recommendation system, growing the KNN's near neighbours also improved the model's accuracy—a little bit better than SVD performance. Nevertheless, the overall expense of creating a content-based recommender is significantly more significant because each user has a unique model based on the characteristics of the music they like.

Although factorization machines are less precise than other models, they have a relatively high recall value. Additionally, we noticed that the developed model outperforms songs that appear frequently in the data; therefore, we may want to improve the accuracy of less well-known songs in the future.

We saw that, with hybridization, recall decreased somewhat, but accuracy marginally increased. Compared to the FM Model, the hybrid model has a lower recall value but better precision.

REFERENCES

[1] McFee, B., Bert in Mahieux,T., Ellis, D. P., Lanckriet, G. R., “The million song dataset challenge”, In Proceedings of the 21st international conference companion on World Wide Web, pp. 909916, April 2012.

[2] Aiolli, F., “A preliminary study on a recommender system for the million songs dataset challenge”, Preference Learning: Problems And Applications In AI, 2012.

[3] Koren, Yehuda. "Recommender system ut ilizing collaborat ive filtering combining explicit and implicit feedback with both neighborhood and latent factor models." U.S. Patent No. 8,037,080. 11 Oct. 2011.

[4] Cremonesi, Paolo, Yehuda Koren, and Roberto Turrin, "Performance of recommender algorithms on top-n recommendat ion tasks." Proceedings of the fourth ACM conference on Recommender systems. ACM, 2010.

[5] Szu-Yu Chou, Li-Chia Yang, Yi-Hsuan Yang, and Jyh-Shing Roger Jang, "Condit ional preference nets for user and item cold start problems in music recommendat ion." (ICME 2017) 2017 IEEE Internat ional Conference on Mult imedia and Expo, pp. 1147–1152, 2017.

[6] Charu C. Aggarwal, «Recommender Systems. The Textbook», pp. 518, 2016.

[7] Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich, Recommender Systems. An Introduction, 2011.

[8] Mohapatra, H., Panda, S., Rath, A., Edalatpanah, S., & Kumar, R., "A tutorial on powershell pipeline and its loopholes", International journal of emerging t rends in engineering research, 8(4), pp. 975- 982, 2018.

[9] Kumar, R., Edalatpanah, S. A., Jha, S., & Singh, R. , "A Pythagorean fuzzy approach to the transportat ion problem", Complex & intelligent systems, 5(2), 255-263, 2018.

[10] Kumar, R., Edalatpanah, S. A., Jha, S., & Singh, R., "A Pythagorean fuzzy approach to the transportat ion problem", Complex & intelligent systems, 5(2), pp. 255-263, 2018.

[11] Svetlin Bostandjiev, John Donovan, Tobias Höllerer, "TasteWeights:A Visual Interact ive Hybrid Recommender System", ACM/IEEE Computer Science Curricula, 2012.

[12] Royi Ronen, Noam Koenigstein, Elad Ziklik and Nir Nice," Select ing Content-Based Features for Collaborat ive Filtering Recommenders", ACM/IEEE Computer Science Curricula, 2013.

[13] Mohammad Yahya H. Al-Shamri, Nagi H. Al-Ashwal, "Fuzzy- Weighted Similarity Measures for Memory-Based Collaborative Recommender Systems", Journal of Intelligent Learning Systems and Applicat ions, 2014.

[14] Manoharan, Samuel,"Pat ient Diet Recommendat ion System Using K Clique and Deeplearning Classifiers." Journal of Artificial Intelligence 2, no. 02, pp.121-130, 2017.

[15] Smys, S., and C. Vijesh Joe, "Big data business analyt ics as a strategic asset for healthcare indust ry." Journal of ISMAC 1, no. 02,pp.92-100,2010.