MUSIC RECOMMENDATION SYSTEM

Kahaan Patel
AU1841110
Ahmedabad University
kahaan.p@ahduni.edu.in

Vidit Vaywala
AU1841128
Ahmedabad University
vidit.v@ahduni.edu.in

Hemil Shah
AU1841135
Ahmedabad University
hemil.s@ahduni.edu.in

Meet kadiya
AU1841099
Ahmedabad University
meet.k@ahduni.edu.in

I. ABSTRACT

Our goal is to build a music recommender system that can be used by a user to get recommendations based on his/her preferences. We are using the collaborative filtering method in order to create our recommender system. We first tried to create our recommender system model using the K-nearest neighbors algorithm of classification. In order to improve our model and make it more personalized, in the second half of the project, we refined our model and applied the Singular Value Decomposition (SVD) algorithm on it. Finally we used the Root Mean Square Error method to evaluate the accuracy of our model. The results of both the models - K-nearest neighbours and Singular Value Decomposition can be found in the results section of the report. The rest of the report is divided as follows - firstly we have introduced the topic, then we have carried out the literature survey of the related topic where we have explored the K-nearest Neighbours and Singular Value Decomposition algorithms in detail. Then we have written about the implementation of our model and how we have built it step by step - for both KNN and SVD algorithm. After the implementation, we have shown the results obtained from our recommendation system. Finally we have concluded the report.

II. KEY WORDS

K-nearest neighbours (KNN), Fuzzy string matching , Classification Algorithms, Music Recommendation , CSR Matrix, Singular value decomposition(SVD), Root mean square error(RMSE), GridSearch CV, Accuracy, Cross validate, Surprise library.

III. INTRODUCTION

Music Recommend system is a system that seeks to predict or filter preference according to the user's different choices.It learns from the user's past listening history and recommends them various songs which they would probably like to hear in the future.Collaborative filtering algorithms which predict (filtering) taste of a user by collecting preferences and tastes from many other users (collaborating) are also implemented. In this first phase of the project, we have build a recommender system using k-nearest neighbors algorithm.Our system also uses RMSE which stands for Root Mean Squared Error. The standard deviation of the errors that exist when making a forecast on a dataset is known as the RMSE. The Root Mean Square Error (RMSE) is a metrix for determining how well a regression line matches the data points.Along with RMSE, we have also applied The singular value decomposition (SVD).SVD is another method for factorising a matrix into singular vectors and singular values. The SVD is commonly used in machine learning as a data reduction tool as well as in the estimation of other matrix operations such as matrix inverse. Our dataset consists of 2 million records containing user ids, song ids and the number of times a user has listened to a particular song. [1]

IV. LITERATURE SURVEY

A. *k NEAREST NEIGHBOURS -KNN algorithm*

The algorithm k-nearest neighbours is used to predict a user's rating of unrated items. When KNN makes a song prediction, it calculates the "distance" between the target song and every other song in its database. It then ranks the songs based on their distances and returns the top k as the most similar song recommendations. The k-nearest neighbours (KNN) algorithm relies on item feature similarity rather than making any assumptions about the underlying data distribution. A data point is classified in KNN by a majority vote of its neighbours, with the data point assigned to the most common class among its k-nearest neighbours, as measured by a distance function. inclass if k = 1, then simply the data point is assigned to the class of its nearest neighbor—i.e., itself. In general; to reduce the overall noise ,a large k value is more favourable. [2]

B. *THEORY OF SINGULAR VALUE DECOMPOSITION (SVD)*

The aim of singular value decomposition is to minimise a dataset with a large number of values to a dataset with considerably fewer values while also retaining a substantial portion of the original data's heterogeneity. How will we find systems from all of the raw data we've gathered? When dealing with high-dimensional raw data, this becomes much more difficult. It's like looking for a needle in a haystack. We can use SVD to extract and untangle data. SVD does everything from diagonalizing a matrix to creating special matrices that are simple to manipulate and evaluate. It lays the groundwork for detangling data into separate components. Whereas Less important elements are skipped by Principal component analysis. SVD is a collaborative filtering technique

which is used in the recommender system. [4] SVD uses a matrix structure where each row and column represents as a user and item. it also reduces the number of features of a dataset by reducing the space dimension because of this it is also called as a matrix factorisation technique. singular value decomposition decomposes a matrix into three other matrices [1] orthogonal left singular matrix:- which displays the interaction between the user and unidentified factors [2] diagonal matrix :- which specifies the latent factor's intensity [3] diagonal right singular matrix :- Which shows that items and latent factors are equivalent.

IMPLEMENTATION

1) **Collecting Data**- We have used the million song dataset that is available online for developing our model. It consists of users id, songs id ,title ,author and how many times a user has listened to a song.

2) **Filtering and Visualizing data**- Using the Pandas and the seaborn library we filtered out the dataset and then explored its characteristics after visualizing it using boxplots, histograms, etc.

A. Recommendation using KNN algorithm

3) **Using Scipy Sparse CSR Matrix**- A lot of values in input matrix are 0 and hence we are dealing with an extremely sparse matrix. We will use the CSR matrix to represent our sparse matrix.

4) **Using Fuzzy String Matching**- When a user enters a song's name partially correctly then our fuzzy matcher matches the user's input to all the songs in our database. And then it selects the movie with the highest ratio. For example, if the user enters "Mary mee" instead of "Marry me(this is the actual song name)" then the fuzzy matcher will understand that the user wanted to select the song "Marry me".

5) **Using KNN algorithm to make predictions**- Using the scikit-learn library we used the k-nearest neighbor algorithm to find the best recommendations to a particular song that user inputted

6) **Getting the Recommendation**- Our model provides recommendations to the user based on evaluation of the k-nearest neighbors of our input song.

B. Recommendation system using SVD algorithm

7) **Normalizing our data**- we have lot of 0 values in our cleaned data, So for finding SVD of our data we have to do normalization of that data. So first we convert our cleaned data to array using tonumpy() and after that we found mean of every user's listen count and after that we subtract mean value from data to normalize our data.

8) **Using scipy.sparse.linalg svds**- we have use scipy.sparse.linalg library because we have very sparse data. So we are using svds method of scipy.sparse.linalg for computing the SVD values of our matrix. So basically we are giving our normalized data and the value of k(total K number of singular values) as input for finding SVD. And it give us 3 matrix as a output matrix(Here we have data of m x n matrix).
   a) U matrix - The relationship between users and latent factors is represented by U, which is m x r orthogonal left singular matrix.
   b) sigma Matrix - S is a r x r diagonal matrix, which describes the strength of each latent factor.basically it has all singular values as diagnoal matrix.
   c) V transpose Matrix - The relationship between songs and latent factors is represented by U, which is m x r orthogonal right singular matrix.

9) **predicting ratings**- After we have three matrix we are first computing dot product of u and sigma matrix and after than with V transpose matrix to get every user's rating value. but sometime this will overfit our data. So we add mean of user's listen count data which we found earlier to regularize out ratings. This term is called as regularization term.

10) **Function for Recommendation**- So here we create our function to recommend song to user. our task is to recommend songs to the user that the user hasn't listened to yet but he will find it interesting , We will only take the songs that the user hasn't listened to yet by taking help of operator and .isin() operator of Python , Then we will merge the songs dataframe with our user predicted ratings dataframe and sort it in descending order. As we are sorting it in descending order, we are getting the best songs that we would want to recommend to our user , finally using the iloc function of python, we will show the top n (number of songs the user wanted) songs recommended by our system to the user.

11) **Getting the Recommendation**- We are taking user id And N(how many songs user want as recommendation) as input and then passing this value to our recommendation function and we are getting N songs as the recommendation system's output.

12) **Calculating the RMSE value using SURPRISE library**- Surprise is a library in Python used for analyzing recommender systems. [5] We will be using it to calculate the RMSE of our model. So once specifying hyperparameters and an array of potential values in the param grid dictionary, GridSearchCV() calculates

a score for each combination of hyperparameters on a k-fold cross validated dataset and returns the set of parameters that minimises the mean score across folds. Both the number of folds and the score can be selected by the user — we use 3 folds and RMSE accuracy score. After that we will perform the cross validation using the cross validate function. Then we will fit the algorithm to our training set and finally we will test the test dataset and get the RMSE score.

RESULTS

1) **Fuzzy Matching -** When a user enters a song's name partially correctly then our fuzzy matcher matches the user's input to all the songs in our database. And then it selects the movie with the highest ratio. For example, if the user enters "Mary mee" instead of "Marry me (this is the actual song name)" then the fuzzy matcher will understand that the user wanted to select the song "Marry Me" .Hence through simple language processing we can exempt the user from entering the correct name everytime. Little mistakes will be taken care of by the fuzzy matcher. We have implemented this using the fuzzywuzzy library of python.

```
Congrats we found matches in our database:
0             Marry Me
1            Mary Jane
2            Mardy Bum
3           Army of Me
4          Nearly Home
             ...
71        Sacred Flame
72        Already Gone
73    Have Mercy On Me
74           Arroyito
75           Cover Me
Name: 0, Length: 76, dtype: object
5083
```

Fig. 1. Fuzzy Matching

2) **Recommendation using KNN -** Using the scikit-learn library we used the k-nearest neighbor algorithm to find the best recommendations to a particular song that user inputted. Our version of KNN algorithm Calculates the Cosine distance (Cosine distance metric is used to find similarities between different documents) from the new song inputted by users to all the other songs. The songs with the higher value (nearer to 1) are considered good recommendations and our model outputs those songs in the form of recommendations to the user

```
Our recommendation system is starting....

The Recommendations for your song Marry Me are as follows:
1: Each Coming Night, with distance of 0.8464234687989661
2: Dog Days Are Over (Radio Edit), with distance of 0.8379670379111354
3: Maybe, with distance of 0.835318025177499
4: OMG, with distance of 0.817101016304944
5: Sehr kosmisch, with distance of 0.8153417297405798
6: Secrets, with distance of 0.8107638302168317
7: Fireflies, with distance of 0.8103488650069546
8: Hey_ Soul Sister, with distance of 0.795120973864746
```

Fig. 2. Recommendation using KNN

.

3) **Recommendation using SVD -** The below output shows the songs recommended to the user with the user id = 30. Our main task here is to recommend those songs to user that he has not listened to before but he will find it interesting. Using the algorithm described in the implementation section above, we calculated the best possible songs that user would find the most interesting. Finally We are taking user id and N(how many songs user want as recommendation) as input and then passed this value to our recommendation function and we are getting N songs as the recommendation system's output.

The Recommendation for user id 30 are :

	song_id	title
155260	SOAFTRR12AF72A8D4D	Harder Better Faster Stronger
929922	SODACBL12A8C13C273	Learn To Fly
410996	SOWCKVR12A8C142411	Use Somebody
136053	SOLLNTU12A6701CFDC	Kryptonite
871448	SOPSOHT12A67AE0235	Almaz
329598	SOCKSGZ12A58A7CA4B	Bleed It Out [Live At Milton Keynes]
867043	SOPQLBY12A6310E992	Creep (Explicit)
420069	SOPXKYD12A6D4FA876	Yellow
319597	SOKLRPJ12A8C13C3FE	The Scientist
210381	SOUJVIT12A8C1451C1	Savior

Fig. 3. Recommendation using SVD

.

4) **Root Mean Square Error(RMSE) -** Below we have attached two results of RMSE. The first result is showing the 5 fold cross validation of our algorithm. Here, we use surprise library for calculating values for RMSE. First we use GridSearchCV method to find best parameter for our algorithm , we fit our data into that model and doing 12 parallel jobs for finding best parameter and at the end

of these jobs we get all best fit parameters and then we cross validate this with our data . So here we have 5 - fold cross validation, so in results we can see that at every fold we get the RMSE value, train time and test time. And we also have the mean and standard deviation of all 5 fold and in the second results we have the final value of RMSE , which is 2.1508

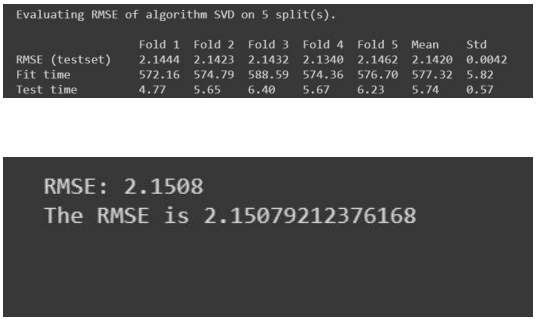

Fig. 4. RMSE

.

CONCLUSION

Our aim throughout the project was to build a music recommendation system that uses collaborative filtering and machine learning algorithms like K-nearest neighbours and Singular Value Decomposition (SVD) to recommend songs to the user based on their preferences. In the first phase of the project, we started exploring the K-nearest neighbours algorithm and found out its usefulness in building our recommendation system. We finalized the dataset, filtered and visualized it to have a better understanding, and finally built the model using the K-nearest neighbours algorithm. In the second phase of our project, we wanted to make this recommendation system more personalized to a single user and hence we used the Singular Value Decomposition (SVD) algorithm to improve it further. SVD is a collaborative filtering technique that is used in the recommender system. The aim of singular value decomposition is to minimise a dataset with a large number of values to a dataset with considerably fewer values while also retaining a substantial portion of the original data's heterogeneity. We were able to get good results from the recommendation system using both the KNN and SVD algorithms. Finally, we checked the Root Mean Square Error (RMSE) of our model using the Surprise library of Python. The future work in this project is to try and predict a user's emotion and then recommend appropriate songs to him/her.

REFERENCES

[1] B. Srebrenik, "Introduction to Music Recommendation and Machine Learning," Medium, 04-Dec-2018. [Online]. Available: https://medium.com/@briansrebrenik/introduction-to-music-recommendation-and-machine-learning-310c4841b01d: :text=These

[2] "Using KNN algorithm for classification of textual documents," IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8079924/?$casa_token$=35bn2HhwmIMAA

[3] C.-S. Mike Wu and D. Garg, "Movie Recommendation System Using Collaborative FilteringChing-Seh Mike Wu," IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org/document/8663822. [Accessed: 17-Mar-2021].

[4] A. Nguyen, "Singular Value Decomposition in Recommender Systems," Home, 19-May-2016. [Online]. Available: https://repository.tcu.edu/handle/116099117/11320. [Accessed: 11-Apr-2021].

[5] N. Hug, "Surprise: A Python library for recommender systems," 05-Aug-2020. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.02174.pdf. [Accessed: 22-Mar-2021].