Music Recommendation System (MRS)

By

YASHI KESARWANI (Reg. no: 0000197)

Supervisor: Dr. Anirban Lakshman

A Thesis submitted to

Indian Institute of Information Technology Kalyani

for the partial fulfillment of the degree of

Bachelor of Technology

in

Computer Science and Engineering

June, 2020

Certificate

This is to certify that the thesis entitled “Music Recommendation System (MRS)” being submitted by Yashi Kesarwani, an undergraduate student, Reg No: 0000197, Roll No- 39/CSE/16042, in the Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India, for the award of Bachelors of Technology in Computer Science and Engineering is an original research work carried by her under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of Indian Institute of Information Technology Kalyani and in my opinion, has reached the standards needed for submission. The work, techniques and the results presented have not been submitted to any other University or Institute for the award of any other degree or diploma.

(Dr. Anirban Lakshman)
Assistant Professor
Indian Institute of Information Technology, Kalyani

Declaration

I hereby declare that the work being presented in this thesis entitled, "Music Recommendation System (MRS)", submitted to Indian Institute of Information Technology, Kalyani in partial fulfillment for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering during the period from January 2020 to May 2020 under the supervision of Dr. Anirban Lakshman, Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal, 741235, India, does not contain any classified information.

Yashi Kesarwani
(Reg No: 0000197)
Department of Computer Science and Engineering
Indian Institute of Information Technology, Kalyani

Acknowledgment

First of all, I would like to take this opportunity to thank and express my gratitude to my supervisor Dr. Anirban Lakshman for his guidance, and instruction that he has given to me from time to time. I have been doing this thesis under his supervision through the completion of my undergraduate project. The skills I have been able to learn from him during my thesis work have benefited me immensely and will continue to do so throughout my future endeavors.

Yashi Kesarwani
(Reg No-0000197)
Department of Computer Science Engineering
Indian Institute of Information Technology, Kalyani

Place: Kalyani
Date: 20/06/2020

Abstract

The concept of filtering out songs based on the interest of a user is the core principle of today's Music Streaming (MS) service. Recommendation Systems (RS) are a key component of the Music Streaming companies. Different companies use different types of Recommendation Systems. Since the web is now an important medium for almost every kind of business and electronic transaction, it serves up as the driving force for the development of Recommendation System technology. There is significant dependency that exists between user and item-based activity which is the basic principle of recommendation. With millions of songs to choose from, people sometimes feel overwhelmed. Most common Recommendation System are designed using the concept of filtering techniques and deal with the count and similarities between the likenesses of the users. This project is basically aimed upon building a Music Recommendation System that gives the user recommendations on music based on his music taste by analyzing his previously heard music and playlist. This project is done by building a basic Content based and Collaborative Filter Recommender System. K-Nearest Neighbors algorithm & Matrix Factorization method is used for the building of Collaborative filtering Music Recommendation System.

Contents

Chapters

Chapter-1
Introduction

1.1 Overview

Music is one of the most popular entertainment media in the digital era. Music is considered as the work of human creativity to express ideas and emotions in the form of sounds that consist of melody, harmony and rhythm. Music can be categorized into several genres, such as pop, rock, jazz, blues, folk etc. Listening to music in the digital age is easier because of the features on the smartphone that can play music offline and online. Nowadays, the availability of digital music is very abundant compared to the previous era, so to sort out all this digital music is very time consuming and causes information fatigue. Therefore, it is very useful to develop a music recommender system that can search music libraries automatically and suggest songs that are suitable for users.

1.2 Recommendation System

A R**ecommender System**, or a R**ecommendation System** is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. They are primarily used in commercial applications.

Recommender systems aim to predict users' interests and recommend product items that quite likely are interesting for them. They are among the most powerful machine learning systems that online retailers implement in order to drive sales.

The latest generation of music listeners currently rely on music suggestions provided by music streaming services. Therefore, the music streaming services need a capable recommendation system which can suggest music based on the listener's interests. Most of the streaming services rely on CF algorithms to suggest music. CF is just a single parameter in determining the taste of the listener. To overcome this, top streaming services use a combination of algorithms to form what is called as a hybrid recommender system.

1.3 Why do I need recommender systems?

Companies using recommender systems focus on increasing sales as a result of very personalized offers and an enhanced customer experience. Recommendations typically speed up searches and make it easier for users to access content they're interested in, and surprise them with offers they would have never searched for.

What is more, companies are able to gain and retain customers by sending out emails with links to new offers that meet the recipients' interests, or suggestions of films and TV shows that suit their profiles.

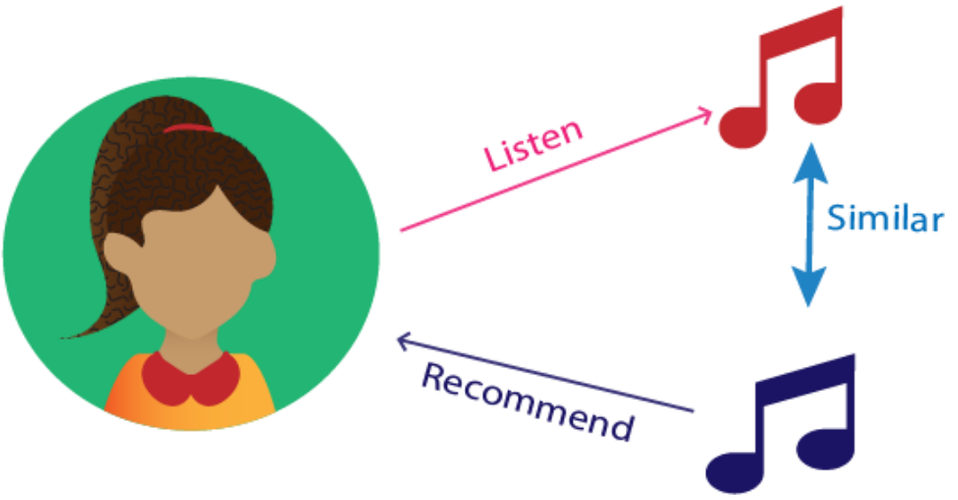
The user starts to feel known and understood and is more likely to buy additional products or consume more content. By knowing what a user wants, the company gains competitive advantage and the threat of losing a customer to a competitor decrease.
Providing that added value to users by including recommendations in systems and products is appealing. Furthermore, it allows companies to position ahead of their competitors and eventually increase their earnings.

1.4 Music Recommender System

Recommender System is a software tool and algorithm that gives recommendations for items that is most interesting to a user. Recommendations is related to many kinds of real applications, such as what commodities are purchased, what songs is listened, or what latest news is read. On the other hand, there is a transformation of the recorded commodity music, specifically when Apple bought Beats Music in 2014. Recently, the business model in music industry is changing from being dependent on commodity sales to a model based on subscriptions and streaming. With the new business model in music industry, the availability of digital music currently is abundant compared to previous era. Therefore, the role music recommender system for the music providers is essential. It can predict and then offer the appropriate songs to their users, consequently the music providers can increase user satisfaction and sell more diverse music.
Generally, music recommender system can be divided into three main parts that is: (i) users, (ii) items and (iii) user-item matching algorithms. Firstly, to differentiate user's music tastes, I can develop user modelling based on user profiling such as geographic region, age, gender, lifestyles, and interests. User modelling will model the difference in user profile to determine their choices of music. Second, item profiling comprises of three kinds of metadata, that is: editorial, cultural and acoustic. They can be used in a variety of recommender system. Finally, the matching algorithm should be able to automatically recommend personalized music to listeners. There are two main approaches of matching algorithm, that is content-based filtering and collaborative filtering.

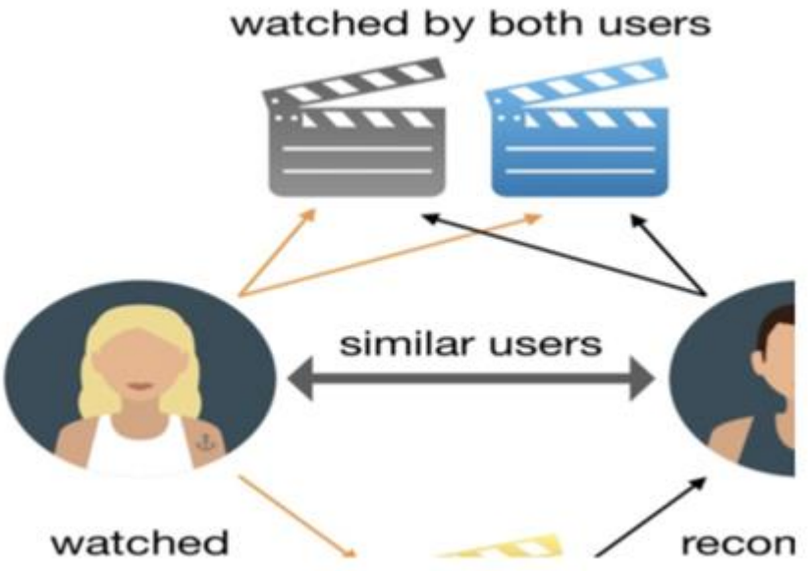
1.5 Types of Recommendation System

A **content-based recommender** recommends based on history of similar items purchased or interacted with in the past. One can attempt to recommend music that is perceptually similar to what the user has previously listened to, by measuring the similarity between audio signals. This approach requires the definition of a suitable similarity metric. Such metrics are often defined ad hoc, based on prior knowledge about music audio.



Content-based recommendation

A **collaborative recommender** recommends based on usage trends of similar users. For instance, if Alice and Bob like movies X, Y and Z, and you like movies X and Y, you may be recommended movie Z. Or, if people who listen to the Beatles and the Rolling Stones also typically play Bob Dylan, then when you input the Beatles into a recommender system, Bob Dylan may be a valid recommendation.



Collaborative based recommendation

A **hybrid recommender** utilizes features of both content-based and collaborative recommenders with an aim to improve quality of results. Amazon may recommend a stereo speaker to you based on my viewing history of sound systems (content-based filtering) as well as on our website usage behavior similar to other users of the Amazon platform (collaborative filtering).

A **popularity-based recommender** does not take usage history into consideration other than sheer popularity of the content. If Britney Spears and Justin Bieber are on top of the Billboard charts (due to sheer popularity), either (or both) artist could be recommended to each new user of Spotify in lieu of basing the recommendation on their usage history.

A recommender system works by mapping the distance (i.e. the "similarity") between points (i.e. artists, users, songs) in my dataset. Three popular metrics include:

- The **Jaccard similarity** is a good choice for implicit item feedback (i.e. binary feedback such as like/dislike or played/not played)?
- The **Cosine similarity** is good for comparing the ratings of items, but does not consider the differences in mean and variance of the items
- The **Pearson Correlation similarity** also compares the ratings of items and effects of mean and variance have been removed

Chapter -2

Objective & Literature Review

2.1.1 Problem Statement

The main goal is suggesting best set of options to the user. For a specific user, I had their song history frequency list songs. From all this information, I have to predict what songs user might like then the question comes: how can I use all this information to achieve my goal. As it is not a straight forward task to find the relevance between various songs. It might be possible that one song which looks similar to other may be completely different and users may dislike that song or may be that song is not of user's taste. There are lots of user around the world and lots of songs so making a relevance between songs and users is a tedious task.

2.1.2 Objective

The objective of this project is as follows:

- **Examine the data I am working with by performing initial Exploratory Data Analysis (EDA)**
- **Build a basic content recommender system using the concept of Term Frequency-Inverse Document Frequency (TF-IDF) for pattern matching.**
- **Build a couple versions of a basic collaborative recommender system using K Nearest Neighbors in sci-kit module and Matrix Factorization algorithm**

2.1.1 Learning Objective

Learning objective of doing this project is first to learn about machine learning, its key concepts and various data mining techniques and algorithms. Other goal is to learn a lot of machine learning algorithm and how to use them. Just learning algorithms doesn't makes you an engineer the real task is to learn which is the right algorithm to apply for a specific project.

2.1.2 Outcome objective

The main objective in terms of outcome is to create a framework for users which can help them in suggesting the right songs for them. This project aims to find the correlation and similarity between different music lovers, their tastes and various songs so that if a user's taste is similar to the other one, I can recommend the songs of one to another on the basis of similar taste. If a song is similar to the other, I can suggest that song to the user that listen the first one. One of the objectives of this project is to reduce the time that user generally wastes on looking for the right song

2.2 Literature Review

A. A recommender System Based on Genetic Algorithm for Music Data

The recommender system by Hyun-Tae Kim *et al* [1] is a hybrid approach of Collaborative Filtering (CF) and Genetic Algorithm (GA). The proposed system aims to effectively adapt and respond to immediate changes in users' preferences. The experiments conducted in an objective manner exhibit that my system is able to recommend items suitable with the subjective favorite of each individual user. The content-based filtering technique is applied to generate the initial population of GA. The recommender system is divided into three phases:

- Feature extraction
- Feature evaluation
- Interactive phase

Advantages:

- The experimental results exhibited that the average scores, which are objectively collected by means of user evaluations, increases by degrees as the generation grows.

Limitations:

- It's really hard for people to come up with a good heuristic which actually reflects what I want the algorithm to do. It might not find the most optimal solution to the defined problem in all cases.

B. Building Effective Recommender System Using Machine Learning Based Framework

The proposed paper shows the adaptation of collaborative filtering in Apache Mahout Platforms. Apache Mahout is an apache software foundation project to produce open source implementation of machine learning (ML) techniques like clustering, collaborative filtering and frequent item set mining. It is a scalable ML library.

Advantages:

- The problem of scalability is solved by the mahout to a certain extent.

Limitations:

- The methodologies are sometimes rejected because of the lack of features availability.

Content-Based Music Recommendation using underlying Music Preference Structure - The proposed paper by Soleymani *et al* [2] discusses about a content-based music recommender system which is based on a set of attributes derived from psychological studies of music preference.

Advantages:

- Content based music recommender systems can deal with unrated songs.

Limitations:

- The content-based recommender system compromises on quality. Genre and tags can be incorrectly identified.

C. A Personalized Music Recommendation System Using Convolutional Neural Networks Approach

In the paper by Shun-Hao Chang *et al* [3] “A Personalized Music Recommendation System Using Convolutional Neural Networks Approach” [3], they have used CNN to classify the music and generate a log file and used Collaborative Filtering to provide recommendations. There paper suggested that using traditional classifiers such as SVN or K-Nearest Neighbmy can reduce efficiency when applied on complex data.

D. Affective Music Recommendation System Reflecting the Mood of Input Image

“Affective Music Recommendation System Reflecting the Mood of Input Image”, byShotoSasaki *et al* [4] recommended music based on the present mood identified in the image. The Recommender System recommended based on the genre the user listens to and determines its mood (happy, sad, lonely etc.) and predicts music based on it.

Chapter -3

Methodology

3.1 Dataset Description

For Content-based recommendation system

I used dataset **songdata.csv** which contains name, artist, and lyrics for 57650 songs in English. The data has been acquired from LyricsFreak through scraping.
I build the content-based recommendation system using the TF-IDF technique.

For Collaborative filter recommendation system

I am going to use the **Million Song Dataset**, a freely-available collection of audio features and metadata for a million contemporary popular music tracks.
There are two files that will be interesting for us. The first of them will give us information about the songs. Particularly, it contains the user ID, song ID and the listen count. On the other hand, the second file will contain song ID, title of that song, release, artist name and year. I need to merge these two Dataframes. For that aim, I'll use the song_ID

In the Dataset, we have-

- 2000000 observations
- 9567 unique songs
- 3375 unique artists
- 76353 unique users

3.2 Methodology

For Content based Recommendation System

A content-based recommendation algorithm has to perform the following two steps-

- First, extract features out of the content of the song descriptions to create an object representation.
- Second, define a similarity function among these object representations which mimics what human understands as an item-item similarity.

In my case, because I am working with text and words, **Term Frequency-Inverse Document Frequency (TF-IDF)** is used for the matching process.

TF-IDF is a technique used for information retrieval. It lights a term's frequency (TF) and its inverse document frequency (IDF); two concepts that I am going to explain. The algorithm finds the score for TF and IDF for each term in the document. After that, the product of TF and IDF of each word is obtained. This is called the TF-IDF Light of that term. When I use this technique, I am just counting the occurrence of each keyword in a document, and finding its importance by calculating the TF-IDF score for that document. The higher the TF*IDF score, the stranger is that word in that context and thus, the more important the term.

The **Term frequency** of a word in the current document is solely the number of times that it appears to the total number of words in a document. For example, for the word music in the document- "I love music because it makes me feel like I can fly".

TF(music) = Number of times music appears/Total number of words = 1/12

Inverse document frequency of a term is the measure of how significant that term is in the whole corpus. It is defined as the total number of documents in a corpus to the frequency occurrence of documents that do contain the term following the formula:

IDF = log(Total number of documents/Number of documents containing the term)

If a word is very rare, it means that it has fiery occurrences, and as a consequence, the IDF increases. Because the TF-IDF score is used to evaluate how important a word is to a document in a corpus, the importance of that word increases when the number of occurrences increases but it is offset by the frequency of the word in the corpus.

If I calculate the TF-IDF score for each word, I have a vector that is commonly called the **TF-IDF Vector.** After that, I use TF-IDF vectorizer that calculates the TF-IDF score for each song lyric, word-by-word.

Here, I pay particular attention to the arguments I can specify:

- **analyzer:** Whether the feature should be made of word or character n-grams.
- **stop_word:** Remember that stop words are simply words that add no significant value to my system, so they should be ignored by the system. I pass English so that it is identified as the language of the lyrics.

I create a **lyric_matrix** variable where I store the matrix containing each word and its TF-IDF score with regard to each song lyric.

How do I use this matrix for a recommendation?

The answer is a word: **similarity**. I now need to calculate the similarity of one lyric to another. For this aim, I can use different metrics such as cosine similarity, or Euclidean distance, among others. For my song recommendation system, I am using **cosine similarity** and particularly, its implementation from Scikit-learn. I want to calculate the cosine similarity of each item with every other item in the dataset. So, I just pass the lyrics_matrix as argument. Once I get the similarities, I store it in a dictionary called similarities, the names of the 50 most similar songs for each song in my dataset.

Limitations of Content Based Recommender System-

- However, one of the biggest limitations of content-based recommendation systems is that the model only learns to recommend items of the same type that the user is already using or, in our case, listening to.
- Even though this could be helpful, the value of that recommendation is significantly less because it lacks the surprise component of discovering something completely new
- Last, this type of recommendation system is not able to capture any contextual or cultural information. Only information that can be derived from the content of a song can be obtained and used to generate recommendations.

For Collaborative filtering Recommendation System

Collaborative methods make a prediction on possible preferences using a matrix with ratings on different songs.

A question that arises is how can we know the preferences of a user?

The answer is something that we use in our everyday life: **Rating**.

There are two ways to collect user ratings.

- The first one is by using **Explicit Rating**, this means we explicitly ask the user to give a rating. This represents the most direct feedback from users to show how much they like a song.
- The second uses **Implicit Rating;** so, for example, we examine whether or not a user listened to a song, for how long or how many times, which may suggest that he/she liked that particular song.

These ratings can be translated to binary - user listening or not to a song, discrete - 1 to 5 stars, or the numbers of times someone listened to a song, or a continuous number - the total minutes a user spent listening to an artist. After we collect ratings, we can generate interaction matrices. **Interaction matrices** are based on the many entries that include a user-song pair as well as a value that represents the user's rating for that song.

In our case, we are going to use the **Million Song Dataset**, a freely-available collection of audio features and metadata for a million contemporary popular music tracks.

There are two files that will be interesting for us.

- The first file gives us information about user ID, song ID and the listen count. We can read it from static.turi.com/datasets/millionsong/10000.txt.
- The second file will contain song ID, the title of that song, and artist name. We can read it fromstatic.turi.com/datasets/millionsong/song_data.csv

After merging both files, we end up with a dataset consisting of 2 million observations with 9567 unique songs and 76353 unique users. We are going to use listen_count, the number of times a user listened to a song as an implicit rating. Doing some exploratory analysis, we can discover that a user listens to a mean number of 26 songs and a median of songs of 16. We can quickly see that not all users listen to all songs. So, a lot of values in the song x users' matrix are going to be zero. Thus, we'll be dealing with extremely sparse data. Dealing with such a sparse matrix, we'll take a lot of memory and resources. To make our life easier, let's just select all those users that have listened to at least 16 songs.

We need now to work with a **scipy-sparse matrix** to avoid overflow and wasted memory. For that purpose, we'll use the **csr_matrix** function from **scipy.sparse.** First, we'll reshape the data based on unique values from song_id as index and user_id as columns to form axes of the resulting Dataframe. Then, we'll use the function **pivot** to produce a **pivot table**. Then, we'll convert this table to a sparse matrix.

How do collaborative filters recommend songs to a particular user?
There are two main approaches to do this-

- It finds users with similar interests and behaviour, and considering what those similar users listened to, it makes a recommendation. This technique is known as **user-based approach** (or user-item).
- It can take into account what songs the user has considered in the past and recommend new similar songs that the user can enjoy; a technique that is called **item-based approach** (or item-item).

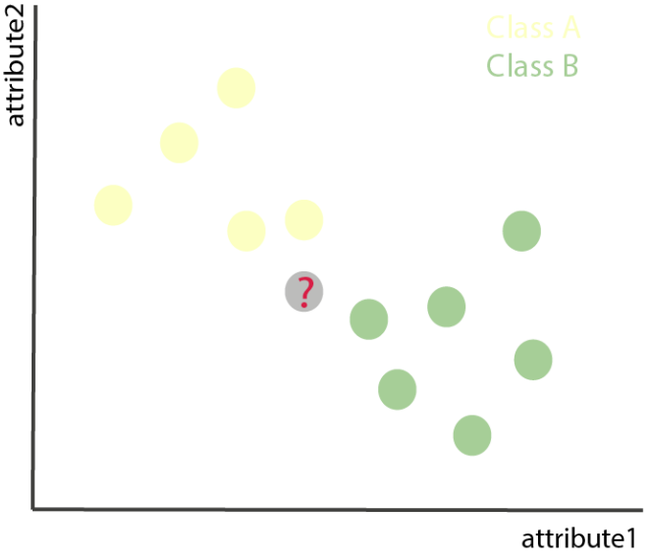
What algorithms do collaborative filters use to recommend new songs?
There are several machine learning algorithms that can be used in the case of collaborative filtering. Among them, I can mention nearest-neighbor, clustering, and matrix factorization.

K - Nearest Neighbors (KNN) is considered as one of the standard methods when it comes to both user-based and item-based collaborative filtering approaches. **So, I will be using K-Nearest Neighbors algorithm to build my collaborative filtering recommendation system.**

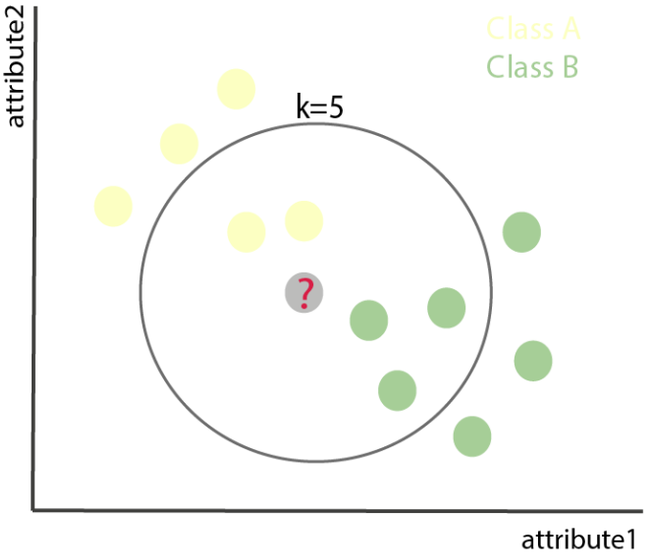
K Nearest Neighbors Algorithm

The KNN algorithm is a **supervised**, **non-parametric**, **lazy-learning** method used for both classification and regression. Supervised because it needs labeled data points to work. Non-parametric because it doesn't make any assumption on the data distribution. Lazy because does not use the training data points to do a generalization; it is delayed until the testing phase.

This algorithm is based on feature similarity- I can start to see what it can be used for recommendation systems. It basically assumes that similar things are located near to each other. And here **distances** strike again to measure how similar or 'close' two points are. Imagine that I have to classify a data point into one of two categories.
I have two attributes that describe the different points. If I represent the points and its attributes in an x-y graph, it will look something like this:



Now, how does the a KNN algorithm know if this new point belongs to the yellow or green category? First, it will look at its **k** nearest neighbors by taking a circle with center point taken as point to be classified. Then, it will determine which class that point should have by voting the most popular class among these k-points. Let's see this in action. If I put k = 5 in my example, I need to take the 5-nearest neighbors as follows:



So, I have 3 points that belong to the green class and 2 to the yellow. So, my new data point is classified as green. Pay attention here. The election of k is very important, as in my case if I were to set k=3, the new data point will be classified as yellow.
When I use KNN to make a prediction about a song, the algorithm will calculate the distance between the target song and every other song in my dataset. Then, it will make a ranking according to their distances. Finally, it will return the top k nearest neighbor songs as song recommendations.

Item Similarity with K-Nearest Neighbors in Sci-kit Learn-
As my first iteration of a basic collaborative recommender, I will build a sparse matrix comparing artist plays (rows) by user (columns). This matrix captures all relationships between artists and users with number of plays in each respective cell. This data will then be passed through a latent mapping algorithm, K-nearest neighbors, to determine cosine similarity amongst the user/artist relationships. This will help us determine which artists are most similar as in shortest distance apart within this latent mapping. For instance, as I would see with a relatively small cosine distance between them, when a user plays the Beatles, they also have a high probability of playing the Rolling Stones rather than a more distant artist such as Snoop Dogg.
fuzzy matching-
If I want to make a recommendation, I won't have an id. I just will know the title and sometimes this name is written in different ways. How do I solve this problem? A way to overcome this is to approximately match the string of the new song with the strings of all the songs in the dataset and determine how similar they are.
If I found one that is similar, I'll take that song. I am going to use a common technique to do this is called **fuzzy-matching**. I'll use the library **fuzzywuzzy**, this method uses **Levenshtein Distance**, which measures the minimum number of edits that you need to do to change a one-word sequence into the other. These edits can be insertions, deletions or substitutions.
I first need to create a dictionary to decode the song titles based on their ids. This will be passed to the recommender system. Then, we'll make a recommendation on the song.

There are some limitations to this method. It doesn't handle sparsity very well, and it's not computational efficient as I have more users or songs.
To overcome these limitations, I can use other methods such as matrix factorization.

Matrix Factorization Algorithm

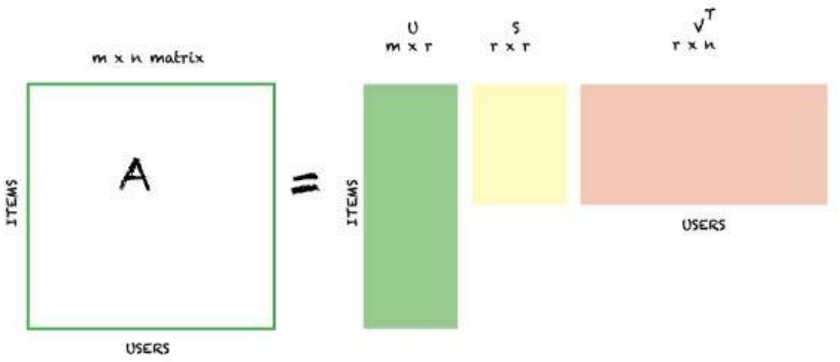
Matrix Factorization is a powerful way to implement a recommendation system. The idea behind it is to represent users and items in a lower-dimensional latent space. So, in other words, Matrix factorization methods decompose the original sparse user-item matrix into lower dimensionality less sparse rectangular matrices with latent features. This does not only solve the sparsity issue but also makes the method scalable. It doesn't matter how big the matrix is, you can always find lower dimensionality matrices that are a true representation of the original one.

But what are latent features, anyway? And, why does it make sense for recommendation systems?
Imagine a user gave a good rating to the following songs: Highway to Hell (AC DC), The number of the Beast (Iron Maiden) and Smoke on the Water (Deep Purple). I could go on and considered them as three separate ratings. However, if my aim is to find a simpler way of representing their preference,

I can just say that they like Hard Rock. Hard Rock is a **latent feature**. They are expressed by higher-level attributes instead of specific songs.

Matrix Factorization will tell us how much a user is related to a set of features as well as how much a song belongs to a particular set of latent features. This confers an advantage over other collaborative filtering: although two users haven't rated any same song, I can still find the similarity between them if they share the same latent features.

Among the different matrix factorization techniques, I used the popular **singular value decomposition (SVD).** This can be an abstract concept as, I deepened into the mathematical foundations. But I'll try to keep it as simple as possible. Imagine I have a matrix A that contains the data for n users x m songs. This matrix can be decomposed uniquely into three matrices; let's called them **U**, **S**, and **V**.
In terms of my song recommender:
- **U** is an *n* users x *r* user-latent feature matrix
- **V** is an *m* songs x *r* song-latent feature matrix
- **S** is an *r x r non-negative* diagonal matrix containing the singular values of the original matrix.
A singular value represents the importance of a specific feature in predicting a user preference.
A Diagonal matrix is a matrix in which the entries outside the main diagonal are all zero.



Single value decomposition: Any matrix A of n x m dimensions can be decomposed into three matrix U, S, and V.

These three matrices are very easy to handle. Moreover, multiplying them will give us back the original matrix **A**. I explored the data before in which I got to know that a user listens to an average of 26 songs. After a little more exploration, I discovered that a song is listened to by an average of 200 users, with minimum 48 and maximum 8277 users. So, I already know that it is a very sparse matrix. To make it easier, let's just filter further the data. I am going to select only those songs which have been listened to by at least 200 users.

Instead of working with the implicit rating as it is, I'll apply the binning technique. I'll define 10 categories. The original data values which fall into the interval from 0 to 1, will be replaced by the representative rating of 1; if they fall into the interval 1 to 2, they will be replaced by 2 and so on. The last category will be assigned to original values ranging from 9 to 2213.

Let's now try to take the matrix factorization approach to recommend songs using Python 3.7 and a fun library called surprise. **Surprise** is an easy-to-use Python library specially designed for recommender systems I'll use the built-in function for **SVD**. It's worth to notice that this implementation (popularized by Simon Funk during Netflix prize) factorizes a matrix into two other ones, and then it uses gradient descent to find optimal values of features and Lights. This is not strictly the mathematical SVD explained above. Instead, this algorithm factorizes the original matrix as the product of two lower-dimensional matrices- The first matrix has a row containing latent features associated with each user & The second one has a column containing latent features associated with a song.

Now, I need to find which are the best parameters for my model for the data I have. The **GridSearchCV** class will compute accuracy metrics for the SVD algorithm on the combinations of parameters selected, over a cross-validation procedure. This is useful for finding the best set of parameters for a prediction algorithm. Cross validate will run a cross-validation procedure for my best estimator found during the grid search, and it will report accuracy measures and computation times. This method uses K-Fold as the cross-validation technique. After finding the best parameters for the model, I create my final model. I use the method fit to train the algorithm on the trainset, and then, the method test to return the predictions obtained from the test set.

Chapter – 4

Evaluation & Results

4.1 Evaluation & Results

A **recommender** or **recommendation system** or **engine**) is a filtering system whose aim is to predict a rating or preference a user would give to an item, e.g. a film, a product, a song, etc.
There are two main types of recommender systems:

- Content-based filters
- Collaborative filters

We are building a Music Recommendation to recommend song from some past information of users and their preferences.

Content-based recommendation algorithm has to first, extract features out of the content of the song descriptions & Second, define a similarity function among these. In my case, because I am working with text and words, I used **Term Frequency-Inverse Document Frequency (TF-IDF)** for matching process.
Content-based methods are computationally fast and interpretable. Moreover, they can be efficiently adapted to new items or users. However, one of the biggest limitations of content-based recommendation systems is that the model only learns to recommend items of the same type that the user is already using or, in my case, listening to. Even though this could be helpful, the value of that recommendation is significantly less because it lacks the surprise component of discovering something completely new.

Collaborative-based methods work with an interaction matrix, also called rating matrix. The aim of this algorithm is to learn a function that can predict if a user will benefit from an item - meaning the user will likely buy, listen to, watch this item. We used two different algorithms for building collaborative filtering recommendation system.
K Nearest Neighbors (KNN) is considered the standard method when it comes to both user-based and item-based collaborative filtering approaches.
There are some limitations to this method. It doesn't handle sparsity very well, and it's not computational efficient as I have more users or songs. To overcome these limitations, I used another algorithm matrix factorization.
Matrix Factorization is a powerful way to implement a recommendation system. The idea behind it is to represent users and items in a lower-dimensional latent space. Matrix Factorization will tell us how much a user is related to a set of features as well as how much a song belongs to a particular set of latent features. For its implementation, we will use built-in SVD function and GridSearchCV to compute accuracy metrics for the SVD algorithm on the combinations of parameters selected, over a cross-validation procedure, to know the best set of parameters. After finding the best parameters for the model, we create our final model, train it and find the error for the test set.
The RMSE (Root Mean Square Error) for the final model is 2.18.

Chapter – 5

Conclusion & Future Work

6.1 Conclusion

I had a great learning experience while doing this project. I have learned about data mining and data cleaning while completing Exploratory Data Analysis (EDA). This is the very first task of a machine learning model to remove all the problem creating objects from the dataset. Data cleaning and Data exploration were very useful to make dataset algorithm ready. I have learnt to create machine learning model, train the model and then doing testing on it. Best suited algorithm for this project is K-Nearest Neighbors Algorithm.

This is a project of my Artificial Intelligence course. I find it is very good as I got a chance to practice theories that I have learnt in the course, to do some implementation and try to get a better understanding of a real artificial intelligence problem: Music Recommender System. There are many different approaches to this problem and I get to know some algorithms in detail like KNN and Matrix Factorization. By manipulating the dataset, changing the learning set and testing set, changing some parameters of the problem and analyzing the result, I learned a lot of practicing skills. I faced a lot of problem in dealing with this huge dataset, how to explore it in a better way and also had difficulties in some programming details. However, with lot of efforts, I have overcome all of these and got to know about several techniques of Data Analysis.

6.2 Future Work

I made content based and collaborative based recommendation systems. Due to lack of time I couldn't make a model for hybrid recommendation system. Popularity based models are also good at recommendations so I'll try to implement this also to predict top-N songs to the users which are most popular at a given time.

There are various types of the recommender system which are vast and cover various parameters. They are developing and emerging in modern day generation of e- services and commerce. But simultaneously, there is a need to develop and optimize the working and output of recommender system. Several service providers facilitate the users with a list of items. But this is not sufficient because customers have different preferences and choices which may mainly depend upon various factors and constraints. Also, in many cases it may not be possible to recommend specific items to particular users. Therefore, there is a scope for incorporating the concept of multiple dimensions in music recommender system particularly.

Most of the products and services provided by the various e-commerce sites are expensive and therefore less used by customers. This leads to the inability to rate an item or set of items correctly and specifically. Thus, traditional recommender system techniques are not optimum. This leads a way towards further work and development in building an optimized recommender system which also takes into considerations the feedback and all other constraints related to recommendation.

References-

[1]. Kim, H. T., Kim, E., Lee, J. H., & Ahn, C. W. (2010, April). A recommender system based on genetic algorithm for music data. In 2010 2nd International Conference on Computer Engineering and Technology (Vol. 6, pp. V6-414). IEEE.

[2]. Soleymani, M., Aljanaki, A., Wiering, F., & Veltkamp, R. C. (2015, June). Content-based music recommendation using underlying music preference structure. In Multimedia and Expo (ICME), 2015 IEEE International Conference on (pp. 1-6). IEEE.

[3]. S. Chang, A. Abdul, J. Chen and H. Liao, "A personalized music recommendation system using convolutional neural networks approach," 2018 IEEE International Conference on Applied System Invention (ICASI), Chiba, 2018, pp. 47-49.

[4]. S. Sasaki, T. Hirai, H. Ohya and S. Morishima, "Affective Music Recommendation System Reflecting the Mood of Input Image," 2013 International Conference on Culture and Computing, Kyoto, 2013, pp. 153-154.