**A**

**PROJECT ON**

**MUSIC RECOMMENDATION SYSTEMS**

TABLE OF CONTENTS

1.0 [Introduction](#INTRODUCTION)………………………………………………………………………………2

2.0 [Methodology](#METHODOLOGY)……………………………………………………………………………...2

2.1. Development Environment ……………………………………………………….3

2.2. Data Gathering…………………………………………………………………….3

2.3 Data Pre-Processing……………………………………………………………….3

2.4. Data Evaluation……………………………………………………………………4

2.5. Various Methods Of Building A Recommender System……………………….…5

2.5.1. Popularity Based Recommendation System: …………………………......5

2.5.2. Machine Learning…………………………………………………………5

2.5.3. Content Based Filtering (Classification Model) ………………………….6

2.5.4. Collaborative Based Filtering Recommender System……………..……...6

2.5.4.1. User-Based Filtering……………………………….……...7

2.5.4.2. Item-Based Filtering……………………………….………8

2.5.4.3. Matrix Factorization Method………………………………8

2.5.4.5. Kmeans Clustering……………………………….………10

2.5.5. Deep Learning……………………………….…………………………………...11

3.0. [Experiments and Discussions](#EXPERIMENTSANDDISCUSIONS)……………………………….……………………………12

3.1. Machine Learning Experiments……………………………….…………………...12

3.2. Deep Learning Experiments……………………………….……………………...13

4.0. [Conclusions](#CONCLUSIONS)……………………………….……………………………………….……..14

**1.0** **INTRODUCTION**

According to Forbes, in 2018, the Global Music revenue was worth $19.8 billion and digital music accounted for over half of it with a turn in of $11.2 billion. The digital music industry has given smallest artist without record label and little capital to release their song and reach out to a wider audience. Now music contents are generated and uploaded daily, leaving the internet with a vast library of music contents and listener with uncountable choices.

With the wide availability of smart devices and access to the internet, there has been an increase and an unlimited supply of digital music contents which can be accessed through various music streaming services such as iTunes, Amazon, Google Play, Spotify etc. A user might find it as a tedious and time-consuming exercise to go through all this content just to find similar music choices, thus, diminishing user experience.

To create an engaging environment for users, it is recommended that a music recommendation system is developed, the system will automatically suggest suitable songs by searching through the music library based on the previous interaction, choice pattern and historical data of other users with similar music taste. According to Oord Avd et al “Music recommendations are done by looking for similarities from one music to another or by giving preference from one user to another” (Oord Avd et al, 2013).

A critical success factor for the system is the ability to effectively recommend appropriate song choices for users, thus, increasing user engagement and maintaining a high standard of user experience.

To solve this problem, we are proposing to design and develop a machine learning system (Music Playlist Generator/Recommendation System). We intend to use Python as the primary programming language, python library tool and various machine learning and deep learning algorithm for the design, analysis and development of the system.

**2.0** **METHODOLOGY**

The aim of the project is to design and develop a Music Recommendation system using machine learning algorithm approach. The system should be able to generate a playlist of recommended songs for a listener based on his/her previous listened song, the number of times it was listen to and similar interaction of other users with the system to make prediction (recommendation).

In other to build our machine learning model, the following prerequisite was needed:

1. A development environment and tool for our machine learning model
2. An appropriate machine learning and deep learning algorithm for training and building our model
3. A rich library of songs with various attributes (dataset) for training and testing the dataset

The following methods was adopted in building our recommendation system model

**2.1. DEVELOPMENT ENVIRONMENT**

Jupyter Notebook was adopted as the development environment for writing and running our codes, python was also used as the programming language because it provides a wide variety of machine learning algorithm. These resources allowed for us to perform Data Pre-processing on our dataset, splitting our dataset for training and testing, selecting and building a machine learning model, making prediction with our built model and testing for accuracy.

**2.2. DATA GATHERING**

For the Machine Learning Model, we selected the **Million Song Dataset**, as it has a large library of songs with key attribute necessary for building our model. It is also a popular dataset used around for the world for implementing music recommendation system. It is free and available online. The total size of the dataset is about 300 GB, which is very large but for the purpose of our task, We used the **Million Song Subset** which contains 10,000 instances of data with a size of about 1.8 GB and the **Song Metadata** which contains detailed information about the songs. We didn’t have to download any of the files as we used a resource that provide access to the same file in its repository on the internet.

For Deep Learning, we used a different dataset that provided us with lots of details of a song and the genre that the song belongs to. We wanted to be able to predict the genre a song belong to correctly thus, we can create a model that recommends songs to users based on genre that a user has previously listen to.

**2.3 DATA PRE-PROCESSING**

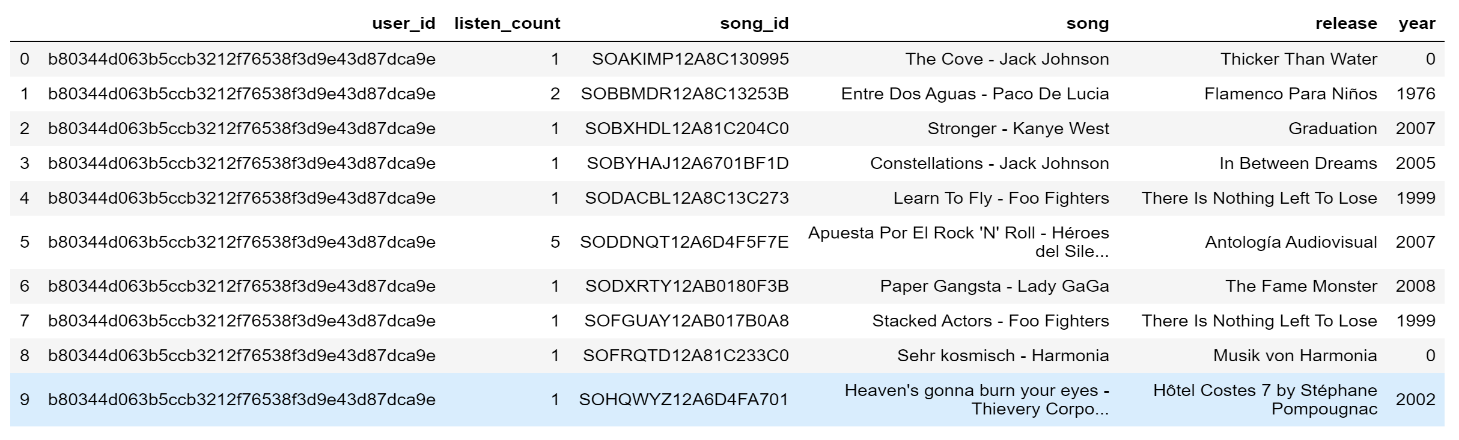
The two datasets selected holds valuable information that are essential to building our machine learning algorithm. The Million Song Subset dataset is a text file document which contain 3 attributes, the user identification number, the song identification number and the number of times a user listen to a song, with 10,000 instances of these attributes. The column name to describe the attributes was not specified in the file but a description of the file is available in the Million Song Dataset website.

The Song\_data dataset is in a csv file format, it contains information about the songs such as song\_id, title, release, artist\_name and year which are attributes of a song. It has 1,000,000 instances of songs.

To use machine learning algorithm in studying the features of the data and build a classification model, it was necessary that we better understand the data. It became necessary to merge both file which is give us a singular view for performing analysis.

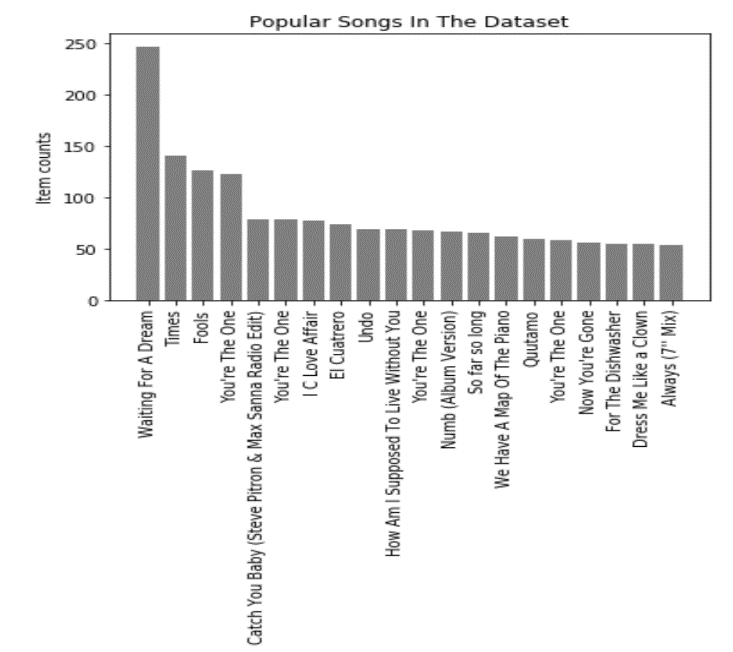
Both files are in different file format as mentioned above, but the python library provide useful tool such as pandas; an open source data analysis and manipulation tool to prepare our data.

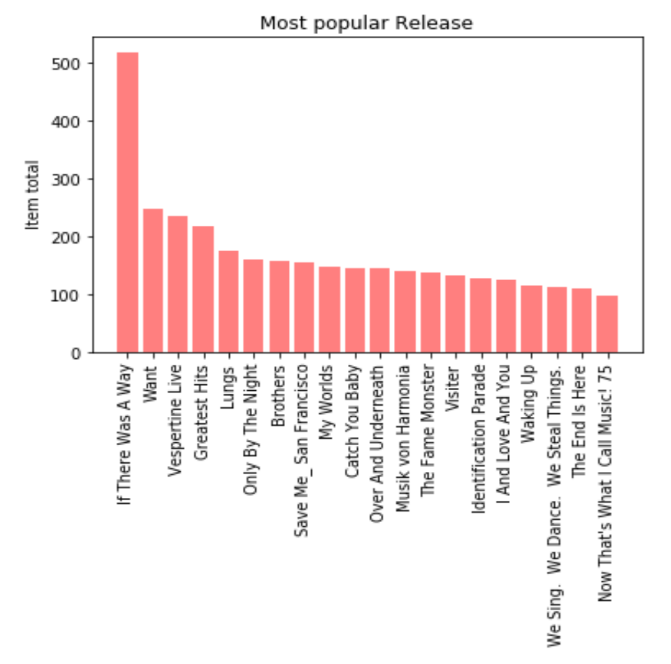
Using Pandas, we loaded both files, label all column in the Million Song Subset text file as described on the website, combined both files as a single data frame, drop duplicate columns, combine similar column in a single column for better readability i.e artist\_name + title = song. The resulting new dataset was used in performing our experiment.

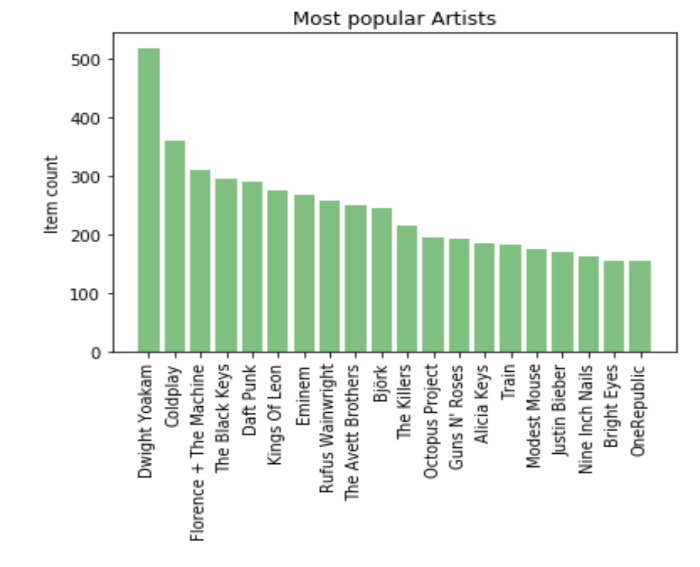


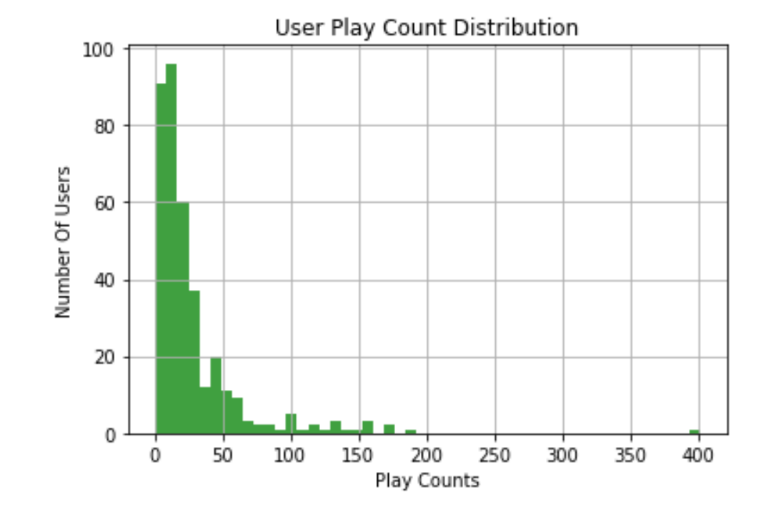
**2.4. DATA EVALUATION**

Evaluation was then carried out on the data, to better understand the dataset such as Popular Songs in the Dataset, Most popular Release, Most popular Artists and User Play Count Distribution.







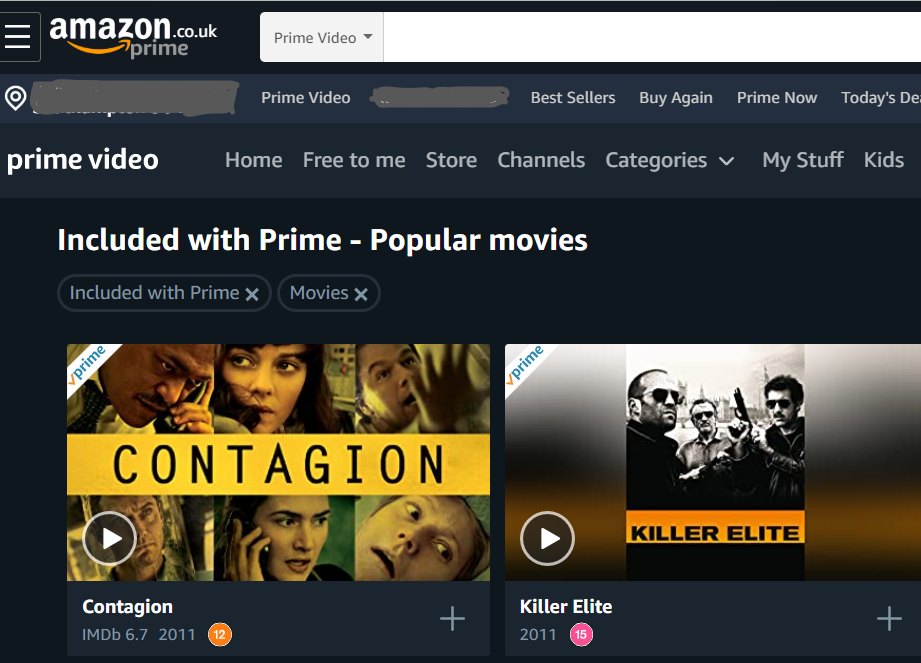


**2.5. VARIOUS METHODS OF BUILDING A RECOMMENDER SYSTEM**

There are various methods of building a recommender system,

**2.5.1. POPULARITY BASED RECOMMENDATION SYSTEM:**

The is a type of system that recommends items to users, based on how popular it is. The popularity ranking can be based on how many times an item was viewed or the number of times an item was purchased. The system keeps a list of all items ranked from top to bottom, starting from the item with the highest total number of counts to the item with the least count. It then makes recommendation based on this list, always recommending the items with the top counts to other users. The shortfall of this method is that it is not personalized to individual users, all users get the same recommendation.



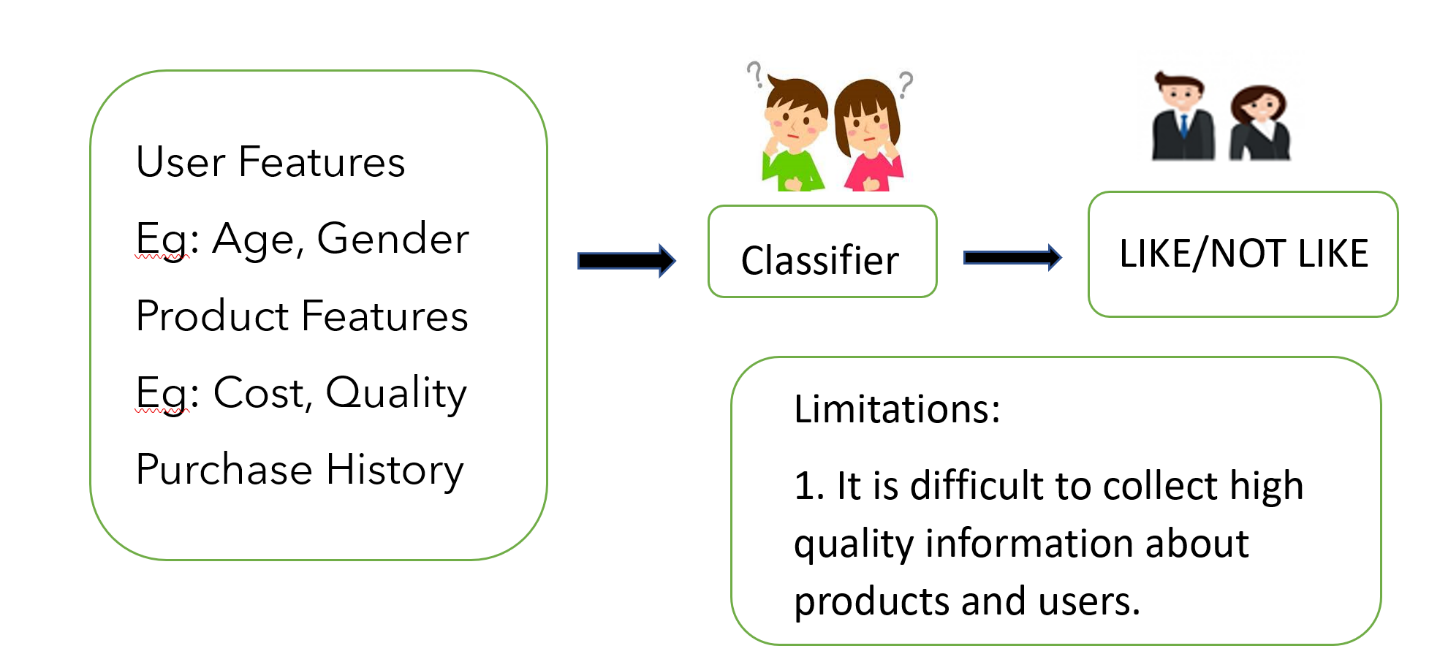
**2.5.2. MACHINE LEARNING**

We employed the method of a recommender that is based on filtering method to build our system. There are two (2) common filtering based method; the Content and Collaborative Based.

**2.5.3. CONTENT BASED FILTERING (CLASSIFICATION MODEL)**

This is a type of system that performs classification that is based on the attributes of an item and the user’s preferences profile. Recommendation is done by looking at user specific classification problem and learning a classifier to understand the user likes and dislikes, centred on item’s attributes.

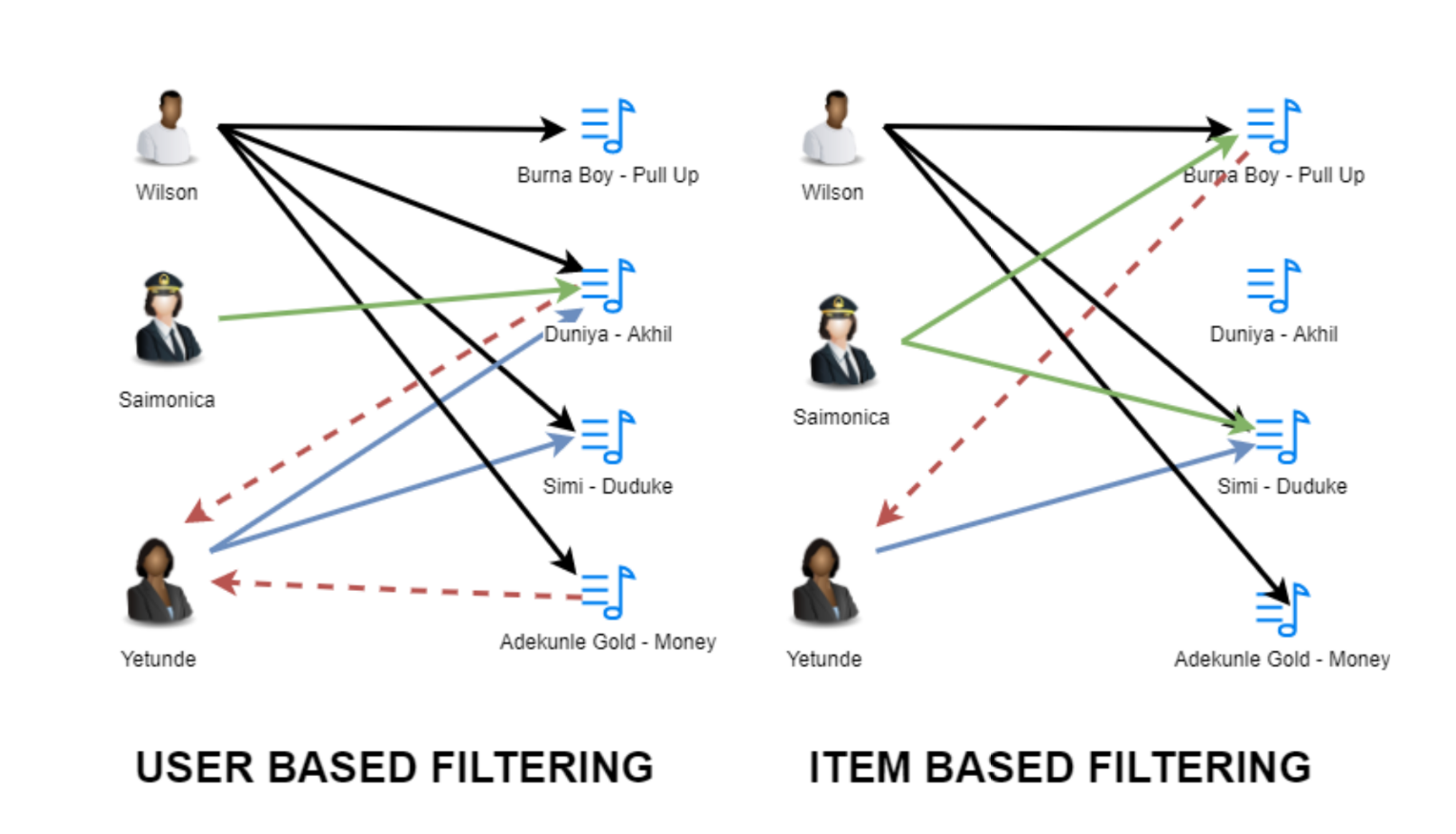
The classified model uses features the items and a profile of the user to make prediction of types of item the user will like, it recommends items that are similar to those the user have liked in the past. It evaluates the attributes of the items such as cost, quality etc., classifies users into various groups and make prediction whether a user will like an item or not, based on the purchase history. The drawback to this method is the difficulty in collecting high quality information about products and users.



**2.5.4. COLLABORATIVE BASED FILTERING RECOMMENDER SYSTEM**

The Collaborative Based Filtering method is implemented with the assumption that people who liked similar items in the past will like similar items in the future. It makes recommendation using user rating and item rating profile, through location of peer users and items with a similar rating history to the current user or item. Collaborative based Filtering can be classified into two (2) types of model; Memory-Based and Model-Based.

Collaborative Filtering is divided into two (2); User-Based and Item Based as depicted in the diagram below:

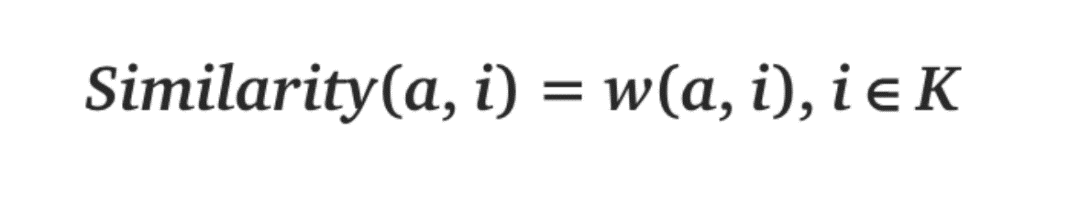


**2.5.4. USER-BASED FILTERING**

The diagram above shows a scenario where a user:Wilson likes to all Four (4) songs in this list, user:Saimonica likes to just One (1) song and user:Yetunde likes to Two (2) songs in the list. To make a prediction of songs that Yetunde might be interested in, it looks at the user who has similar songs taste, which is Wilson and Monica. Although Saimonica has only likes a song on the list, recommendation was still done based on other songs liked by Wilson. The analogy has helped us create a synthetic rating that is based on similar user’s taste in songs.

An application of this method id the user-based Nearest Neighbor Algorithm, which performs 2 key tasks for implementation:

1. It finds the k-nearest neighbors (kNN) to a user **a**, using the function **w** to measure the distance between each pair of users:



1. It then makes prediction on the rating that the user **a** will give to items that the k-neighbours have consumed but user **a** has not.

**2.5.5. ITEM-BASED FILTERING**

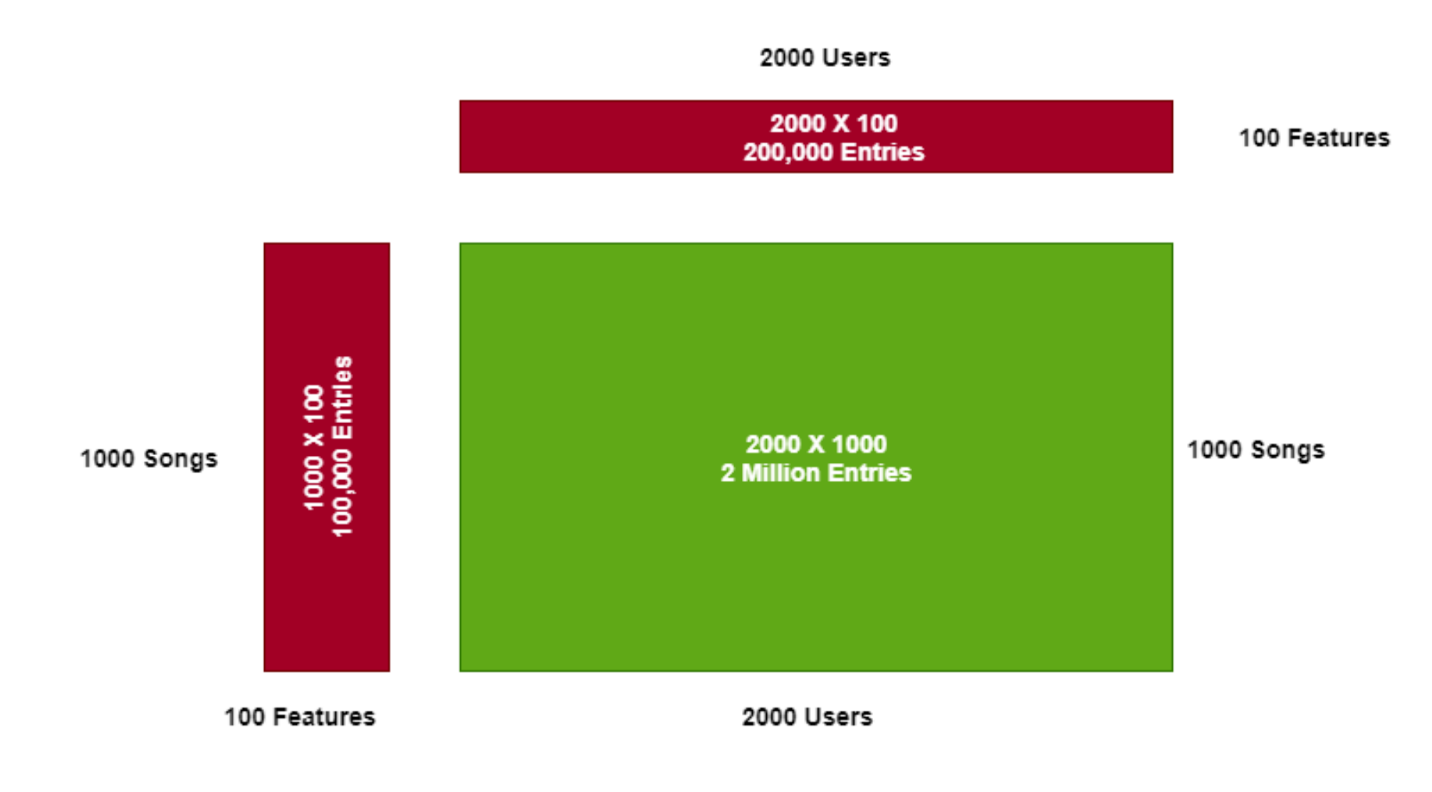
For Item-Based Filtering, as implied in the name, we focus on the item, in the diagram above, user:Wilson, user:Saimonica and user:Yetunde have all listen to the song by Simi:Duduke. Wilson and Saimonica have also listen to the song by Burna Boy – Pull, therefore the song Simi-Duduke is similar to the song Burna Boy-Pull, so the system make prediction that Yetunde will also like Burna-Boy-Pull and recommends it to her. The song Duniya-Akhil is not recommended as it is likely not similar to other songs.

To perform Item based filtering, the following is done:

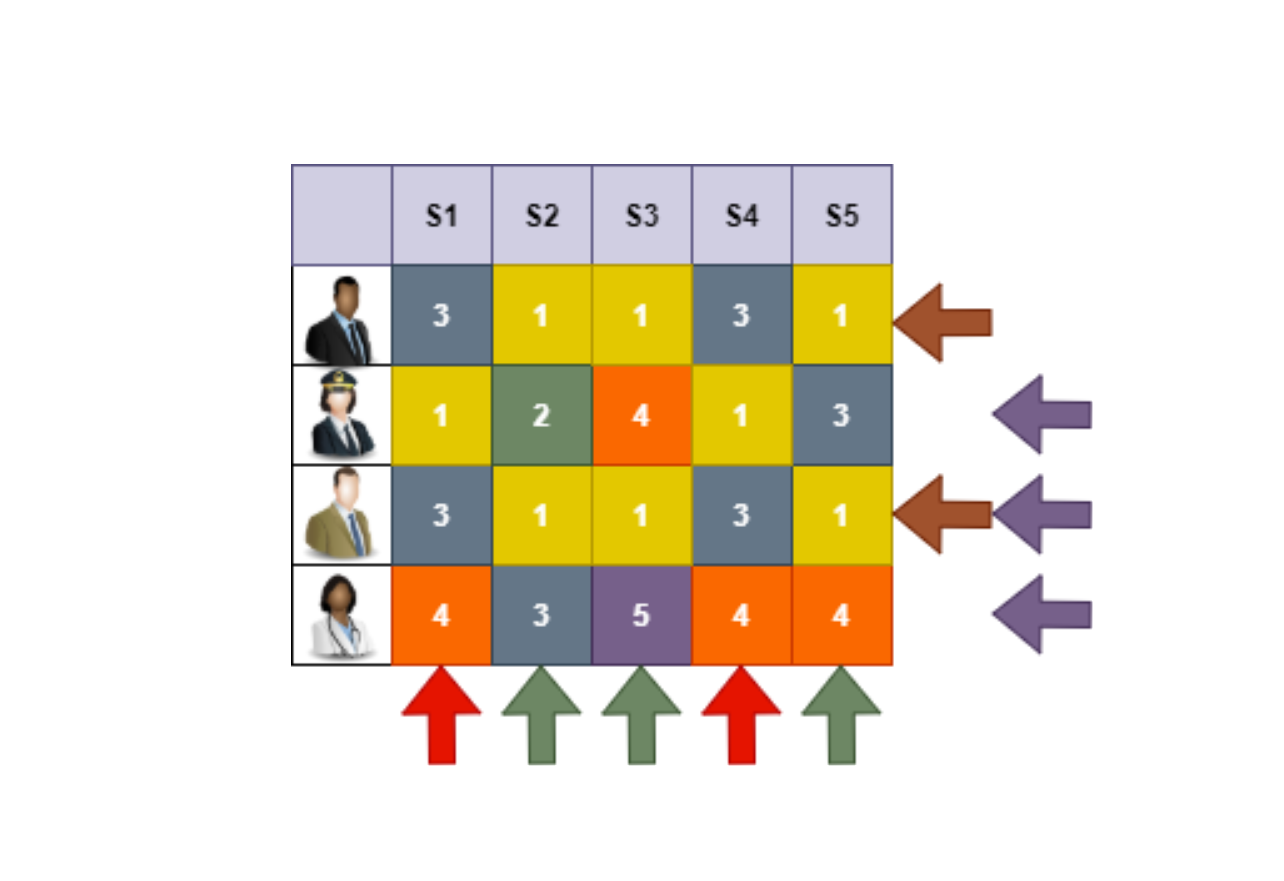
1. Calculation of Similarity amongst items using Cosine Based Similarity or Correlation Based Similarity of Adjusted Cosine Similarity or 1-Jaccard distance
2. Make prediction by calculating the Weighed Sum or Regression

**2.5.6. MATRIX FACTORIZATION METHOD**

We implemented the Model-Based Collaborative Filtering Method using the Matrix Factorization Algorithm. Matrix factorization algorithms work by disintegrating the user to item interaction [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)), into an output of two factorized matrices. This method allow us to reduce our large dataset into a two smaller factors of the dataset without the lose any information.



This can be achieved by looking for dependency in rows and column of the dataset, for example, if we have 5 songs and 4 users with the matrix rating as illustrated in the diagram below



Through observation, we can begin to see various dependency in the rows and columns as shown with the arrows;

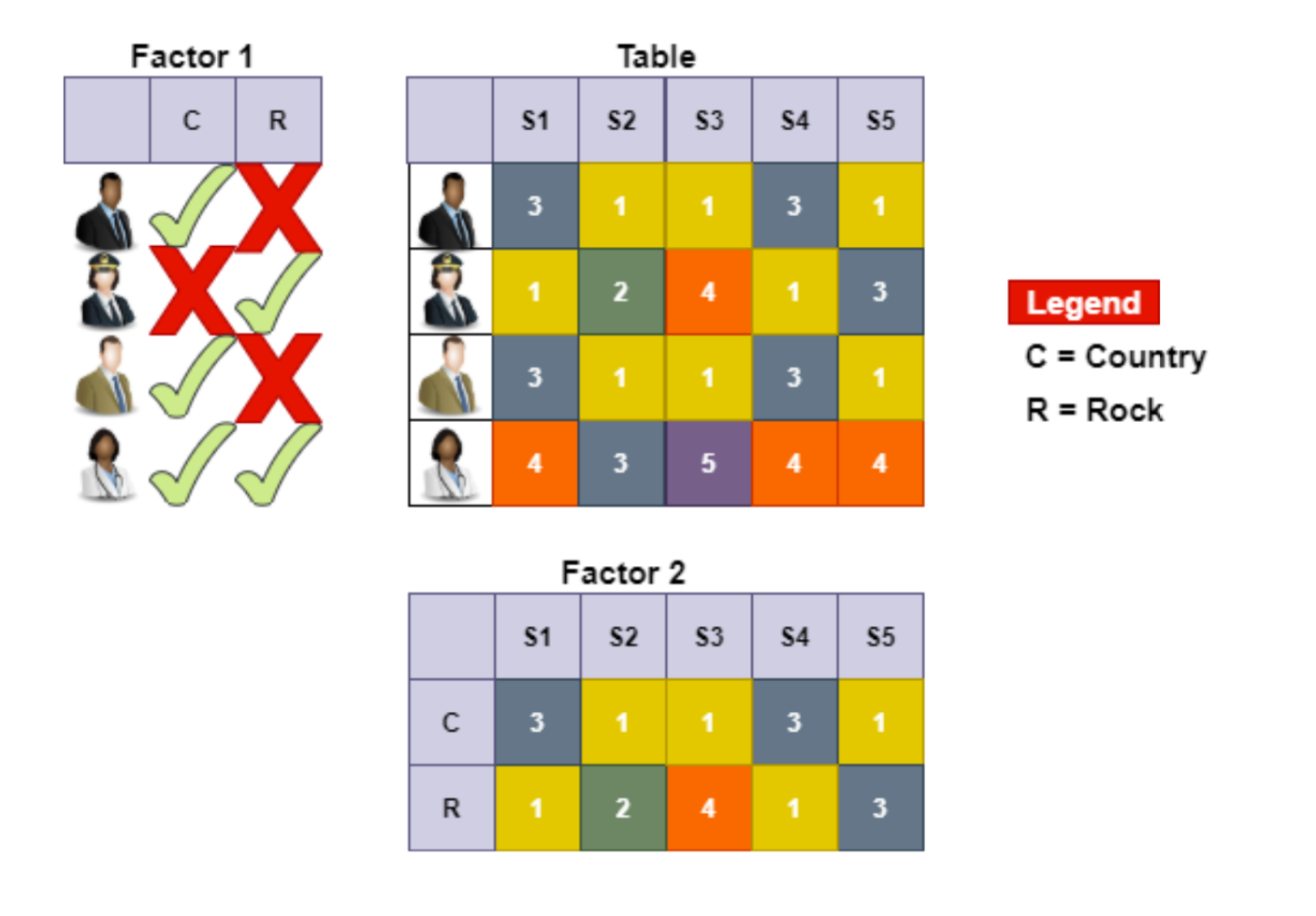
**Brown Arrow** – Both User 1 and User 3 have the same taste as they both rated all songs the same

**Red Arrow** – Song 1 and Song 4 must be very similar, likely the same genre as they are both rated the same by all users.

**Purple Arrow** – The rating of User 2 and User 3 is the sum total of User 4. It means that User2 can like country music and dislike rock while User3 likes rock music but dislike country music but User 4 like country and rock music.

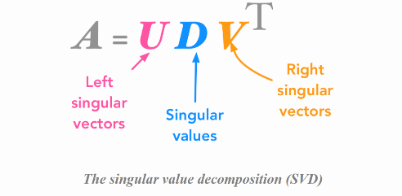
**Green Arrow** – Song 5 is the average of Song 2 and Song 3. These mean that Song 5 (Country/Rock) can be a blend of Song 2 (Country) and Song 3 (Rock).

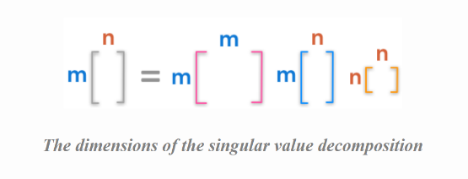
Using Matrix Factorization Method, we can reduce the above table by expressing it as a product of two smaller matrix table by looking factors such as various features and user ratings:



Based on the principle above, we applied the Singular Value Decomposition (SVD) Algorithm to split the Rating Matrix into constituent User Matrix and Item Matrix with minimum Sum is squared error (SSE) on the Train Dataset. Our goal is to predict unknown ratings for the remaining set of songs in the Test Dataset using the learned User Matrix and Item Matrix.

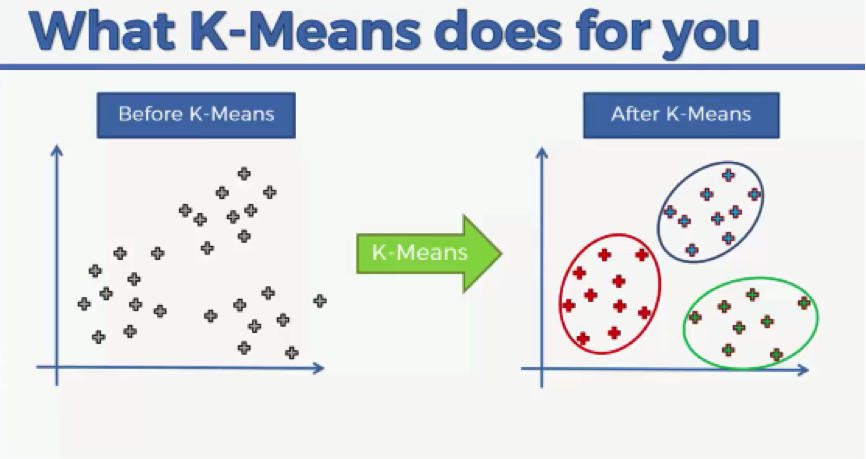
**Singular Value Decomposition** (SVD) is a Scipy lineal algebra function and it was used for factorizing our dataset into decomposed dataset.





**2.5.7. KMEANS CLUSTERING**

[KMeans](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans) Clustering was implemented by creating a cluster of songs in the dataset. The clustering was created on the listen counts of users and user index. Based on the cluster, recommendation will be done. If a user listens to a song that falls in a cluster, he will be recommended other songs within the same cluster. One of the advantage of this method is that it scales well with very large dataset.

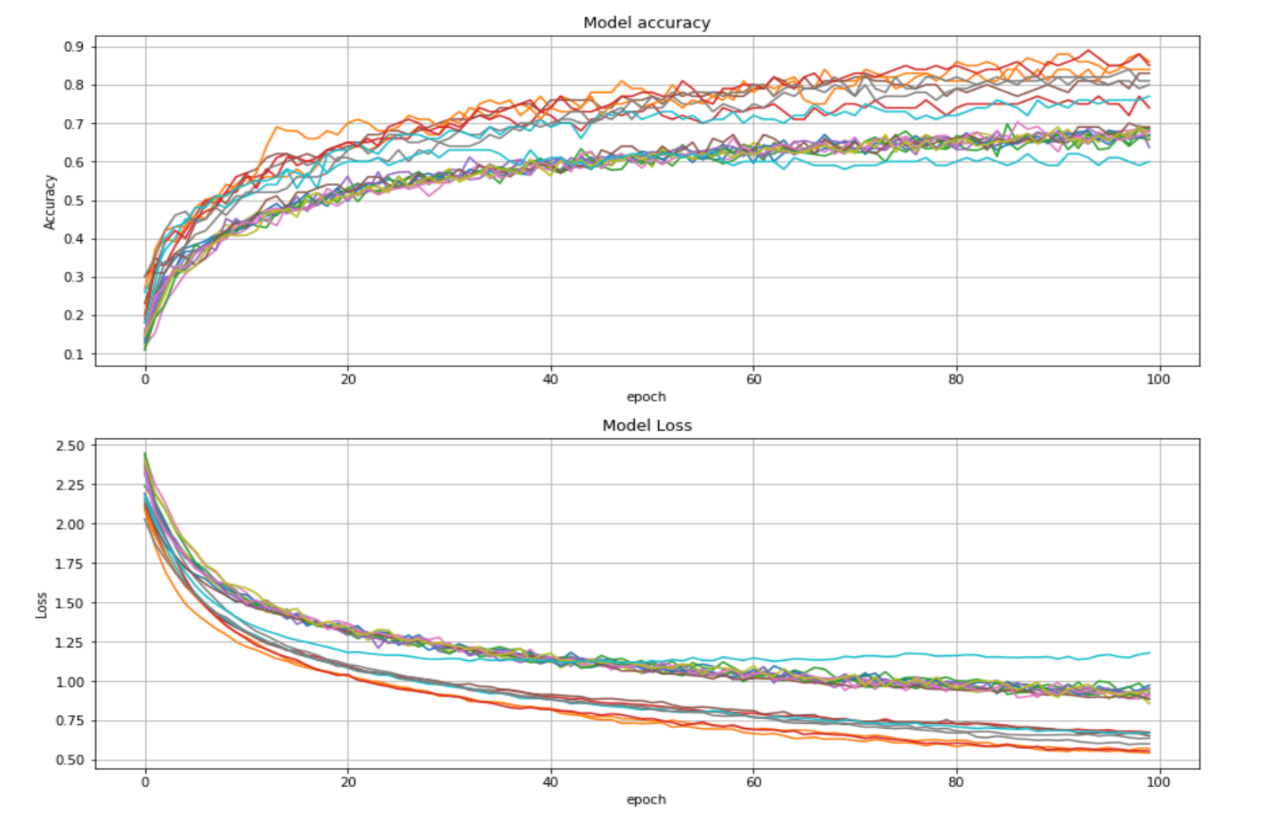


**2.5.8. DEEP LEARNING**

Deep learning provides network capability of learning unsupervised form data that are unstructured or unlabelled.

We adopted Keras, which was ran on the TensorFlow framework to build our neural network, it was adopted because it provides numerous implementations of commonly used neural-network building blocks such as layers, activation functions, optimizers etc.

For this experiment Keras layers was mostly used. Data Standardization was used for the process of rescaling attributes of the dataset, and after building the neural network, tensor board was used to check the accuracy, while Epochs(time) was also used in checking the Loss and Accuracy. The graph below shows the Model Accuracy and Loss.



**3.0.** **EXPERIMENTS AND DISCUSIONS**

Recommendation systems were implemented using various algorithms as discussed in the Methodology. For this project we compared the popularity based recommender system and the Collaborative Filtering (item similarity) model using the Precision and Recall Curve. The Collaborative Filtering Model based was also explored using Matrix Factorization principles.

Kmeans algorithm was implemented to create clusters of songs based on the listen count of all users.

Deep Learning was also explored using keras and tensorflow to train a dataset to classify different songs in various genre and accuracy was examined.

**3.1. MACHINE LEARNING EXPERIMENTS**

The first model we implemented is the **Popularity Recommender Model**, this model recommends music to users based on popular items in the dataset, this was achieved by making a listings of all song from the highest listen count to the lowest listen count, and then making recommendation of the top 10 songs in the list. This shortfall of this approach is that the model makes the same recommendation to all users and personalization for user cannot be attained. This was tested on 2 users; User 8 and User 9. The diagram below shows the result:

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

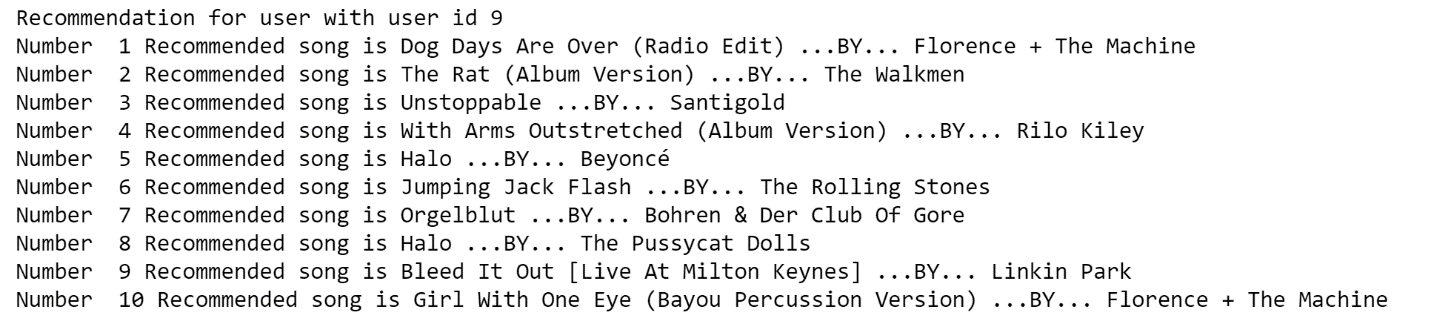
In order to personalize our model, we decided to implement the **Collaborative Filtering (Item Similarity) Model** using cooccurrence matrix, this model uses the item based filtering system to make recommendation to users. It’s performance was compared to the personalized model using the Precision and Recall Curve and it showed that the item similarity model performed better. Precision (Positive Predictive Value) gives insight into how relevant the list of recommended items are, while Recall(Sensitivity) gives insight into how well the recommender is able to recall all the items the user has rated positively in the dataset. From the Precision recall curve, we can see that the item similarity model has higher precision than the popularity model.

A screenshot of a cell phone

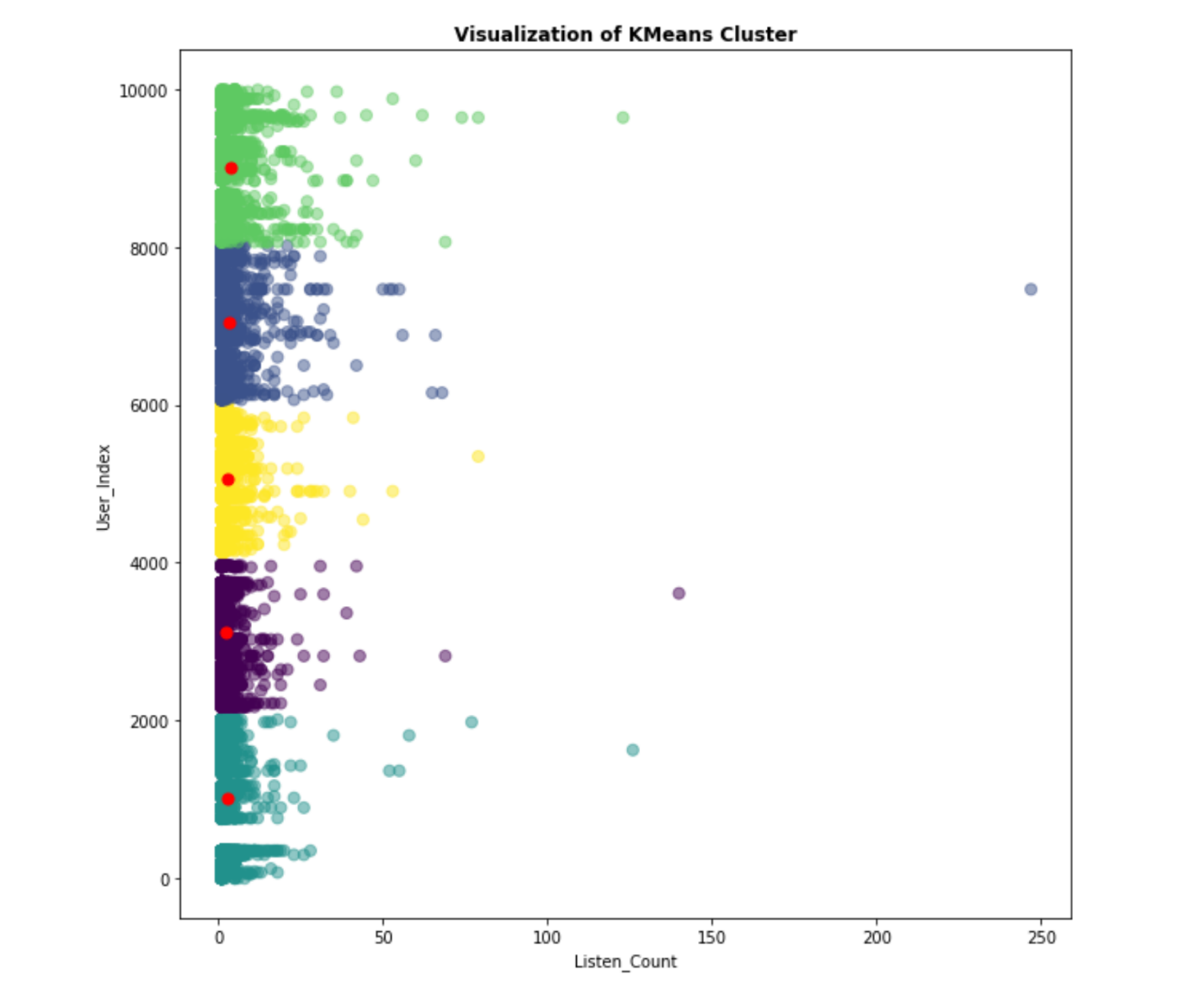
Description automatically generated

To improve our model further, Collaborative Filtering (Matrix Factorization) model was implemented by computing the Singular Value Decomposition (SVD) of our dataset. This improved our model accuracy and gave better recommendations. The model was ran for the two users, User 8 and User 9 and the recommendations are shown in the image below:





Kmean algorithm was implemented on the dataset using a k cluster value of 5. The input value are the user\_index and listen\_count. This help us cluster songs into 5 clusters. Based on this, songs will be recommended to users if a song within the same cluster will be recommended. The graph below shows the resulting clusters.



**3.2. DEEP LEARNING EXPERIMENTS**

We also experimented using Deep Learning on a different dataset to classify songs into different genre, songs will then be recommended based on genre that a listener has previously listen to. The model was built using, scikit-learn, keras and TensorFlow. We trained our network with the k-folds model and over 100 epochs. We regularize our model by using dropouts and weight decay in order to avoid overfitting and to improve accuracy. The loss function and accuracy for each epoch was also computed. From the loss function graph, we discover that the test loss decreases after each epoch

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**4.0.** **CONCLUSIONS**

From our results, it was concluded that the popularity-based model will work in cases where a business show their popular items but it is not suited for the type of recommender system we propose as it does not perform personalization.

Collaborative Filtering model using Matrix Factorization gave a good result, it was able to make personalized recommendations, which is better suited for our model.

The K- clustering model can also be explored as an alternative to building a recommender system, as we were able to create a model that cluster songs based on listen-count and recommendation can be made.

The Deep Learning model also performed well with a high accuracy, when used to predict genre, this is also an alternative solution to the recommender system.

**5.0 FUTURE IMPROVEMENTS**

While our Recommendation system performed really good, it might not perform well a new songs as it will have zero (0) listen count, we would like to improve on the collaborative filtering model to recommend unlisten songs too.

**REFERENCES**

Million Song Dataset, (2011). *Million Song Dataset.* [online] Available at: <<http://millionsongdataset.com/> > [Accessed 18/05/2020].

Pinela, C., 2017. *Recommender Systems — User-Based And Item-Based Collaborative Filtering*. [online] Medium. Available at: <https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f> [Accessed 19 May 2020].

Grover, P., 2017. *Various Implementations Of Collaborative Filtering*. [online] Medium. Available at: <https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0> [Accessed 19 May 2020].

J, H., n.d. *Deep Learning Book Series · 2.8 Singular Value Decomposition*. [online] Code · Data Science. Available at: <https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.8-Singular-Value-Decomposition/> [Accessed 18 May 2020].

En.wikipedia.org. Updated: 12 May 2020. *Recommender System*. [online] Available at: <https://en.wikipedia.org/wiki/Recommender\_system> [Accessed 23 May 2020].

Mwiti, D., 2018. *How To Build A Simple Recommender System In Python*. [online] Medium. Available at: <https://towardsdatascience.com/how-to-build-a-simple-recommender-system-in-python-375093c3fb7d> [Accessed 23 May 2020].

En.wikipedia.org. Updated:19 April 2020. *Matrix Factorization (Recommender Systems)*. [online] Available at: <https://en.wikipedia.org/wiki/Matrix\_factorization\_(recommender\_systems)> [Accessed 24 May 2020].