UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESIGN AND DEVELOPMENT OF A SONG RECOMMEDER SYSTEM BASED ON USER EXPERIENCE AND EMOTIONS

GONZALO JOSÉ OSENDE PÉREZ

JULIO 2019

TRABAJO DE FIN DE GRADO

Título:	DISEÑO Y DESARROLLO DE UN SISTEMA RECOMENDADOR DE CANCIONES BASADO EN EXPERIENCIA DE USUARIO Y EMOCIONES
Título (inglés):	DESIGN AND DEVELOPMENT OF A SONG RECOMMEDER SYSTEM BASED ON USER EXPERIENCE AND EMOTIONS
Autor:	Gonzalo José Osende Pérez
Tutor:	Carlos Ángel Iglesias Fernández
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DESIGN AND DEVELOPMENT OF A SONG RECOMMEDER SYSTEM BASED ON USER EXPERIENCE AND EMOTIONS

Gonzalo José Osende Pérez

Julio 2019

Resumen

Realización de una herramienta de escucha de música con recomendación basada en experiencia de usuario y con un factor condicional en la emoción o estado de ánimo producido por las canciones.

La herramienta contará con una interfaz con la que reproducir y ver las listas de canciones que el sistema o el usuario hayan creado, teniendo también la opción de filtrar por "emoción".

Las recomendaciones se realizan basadas en las escuchas que ha tenido un usuario, del modo que cuanto más haya escuchado una canción, podemos decir que dicha canción es de su agrado y por lo tanto le recomendaremos otra canción la cual suelan escuchar usuarios que también les gusta la primera canción.

Palabras clave: recomendación de canciones, machine learning, Música, Emociones

Abstract

This project consist on the making of a music listening program with songs recommendations based in user experience and with an extra condition in the emotion or mood produced by the song.

The program will have an interface that allows the user play his song lists or the ones generated by the recommender. In both cases, the user will have the option of filtering the songs by the "emotion" they produce.

The recommendation are made based in the listenings the user has done, the more the user listen to a song, the more he like it, and that is the way of getting the actual "taste" of a user. With this, the recommender can guess what song to recommend, comparing it with other user with similar "taste".

All the recommender system will be made by machine learning with the library scikit-learn for Python and other dependencies.

Keywords:song recommendations, machine learning, music, emotions

Agradecimientos

En primer lugar me gustaría dedicarle este trabajo y todo el esfuerzo realizado a mi madre. Mamá, sé que te hubiera gustado leer y corregir esta memoria tantas veces como fuera necesario antes de entregarla, creo que he conseguido de alguna forma hacerlo yo mismo. Lo que no he conseguido es llenar el vacío que se ha quedado tras tu marcha, te anhelo y te quiero.

Este año y durante toda mi vida mi familia ha estado ahí cuando se tenían que estar. Agradezco a mi padre y mi hermano el ser como son y lo que me quieren. También quiero agradecer a mis padrinos por el perfecto trato recibido por su parte en todos los años de mi vida y en especial los últimos meses tan duros.

Por supuesto a mi grupo de amigos de toda la vida (El Bunker), los cuales han sabido hacer que los malos momentos tengan un poco de alegría y me han demostrado lo bonita y satisfactoria que puede ser la amistad. A vosotros, Javi, Jesús, Nacho, Miguel, Carlos, Luis y por supuesto no podía faltar Pepo, va por ti también grande.

Desde que entré a la carrera ha estado conmigo, le tengo en estima como si de un hermano se tratara. Juan, desde aquí te agradezco todos los buenos momentos pasados y que pasarán y te envío todas las fuerzas y suerte para acabar tu enmienda en esta carrera de una vez por todas.

Inés, te doy las gracias por tu amistad en este último año. Por gente como tú merece la pena seguir adelante. Me has ayudado mucho y te mereces lo mejor ahora que emprendes el vuelo hacia terreno inexplorado en tu vida. ¡Mucha suerte y mucho amor!

Sofía, pese a que llevas poco tiempo en mi vida, creo que te debo un 30% de este trabajo y un 100% en agradecimiento. Eres la margarita en este campo de batalla que ha conseguido hacerme feliz pese a la adversidad. Te quiero.

Agradezco a mi tutor, Carlos Ángel, el apoyo dado para la realización de este trabajo.

Me gustaría incluir a todos los amigos y amigas y a toda la gente que de alguna forma me ha ayudado a seguir adelante. A todos vosotros os digo: ¡Agradecido!

Viva las ratas de Schrödinger y viva el Rock & Roll.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Context

At present, there are many ways to listen to music and one of the most used is the streaming platforms, such as Spotify, iTunes... Since each user has his own taste and opinion regarding all the forms of art, music is not an exception and that is the motivation for using recommendation systems in practically all this mentioned platforms.

The recommender systems help users to find the appropriate song or songs for the moment they are listening. Current systems use user information, such as recent listenings, last purchases or most listened genre. This "cold start" problem has made researchers focus on improving content-based recommender systems, which use audio features extracted automatically from the audio content as the basis for recommendations. In the [22] is stated that the construction of features and the definition of similarity in such systems are frequently ad-hoc and not explicitly optimized for the specific task.

Until now, there have not been self-emotion based systems, this is a system that recommends the user the most suitable song for his actual state of mind. This type of systems can be seen as subjective, but thanks to known markers in the melody and lyrics, we can consider these systems nearer to be the substitutes to actual.

The last implementations of machine learning systems have the capability of processing large

amounts of data in short time periods. This is aiming to achieve the real time processing of big data for the search of patterns that can lead to a machine learning.

1.2 Project goals

The principal objective of this project is to develop a platform for getting recommendations of songs to listen and the possibility of reproduce it. To make a dimensioning of the aiming of this project, the main objective can be divided in the next points:

- Make a dataset of songs that adds an emotion factor in order to take it into account for the recommendation.
- Create a model for the recommendation engine capable of recommend based on the musical taste of the user with the best parameters.
- Create an user interface with login and the implementation of the recommendation engine.
- Implement a song reproducing system in the dashboard.
- Design the dashboard to make it intuitive for users to use.

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document. The structure is as follows:

Chapter 1 treats the context wherein this project is located and reflects the objectives that the project aim to complete.

Chapter 2 describes all the technologies and software used in the development of this project.

Chapter 3 analyses the back end model of the whole system and the process of developing it.

Chapter 4 makes an overview of the structure of the data used for this application.

Chapter 5 shows the process followed in the implementation of the system, mainly the dashboard or frontend part.

Chapter 6 finishes the project giving a global conclusion view with the assessment of the final system and the problems faced. Also, shows the lines to follow once this project has finished.

CHAPTER 2

Enabling Technologies

2.1 Introduction

These are the diferent software dependencies, programs and technologies used in this project to take the best results in the possibilities. The first stadium of the project was the recommender engine and the making of the predcition system.

2.2 Machine Learning

The machine learning [4] is an analysis method that automatises the construction of analytical models and algorithms. It is a branch of artificial intelligence which gives the computing systems the ability of learning concepts using data, without being explicitly programmed for it.

In general terms the machine learning is caracterized in 3 main categories explained below:

1. ***Regression Method*:** used to predict the continous value of a variable. It finds the most profitable ecuation that satisfies a point group. Typically used when the precision is no critical and the number of variables is small.
2. ***Classification Method*:** this method is used to predict the result of a discrete value attribute (a,b,c,...) given some caracteristics (X_0,X_1,X_2,... X_n). The more common is the

binary.

3. ***Clustering Method*:** this last method is used when it is necessary to clasificate the data instances without knowing the categories. This type of grouping make possible the construction of consistent intances clusters considering the variables. This method is the most common when the amount of data is considerably large.

When it comes to the concept of learning, the machine learning is divided into 2 principal types of algortihms:

1. ***Supervised Learning*:** this is the one that given a data set in the input, there are known the correct results that should be at the output. This method consist of 2 main stages:

   (a) **Training:** it is used a set data (typically between the 60% and the 70% of the total data) to give the algorithm the capability of finding patterns and relations.

   (b) **Testing:** as the training process uses a percentage of the data, the rest is used to validate the performance and efficiency of the algorithm.

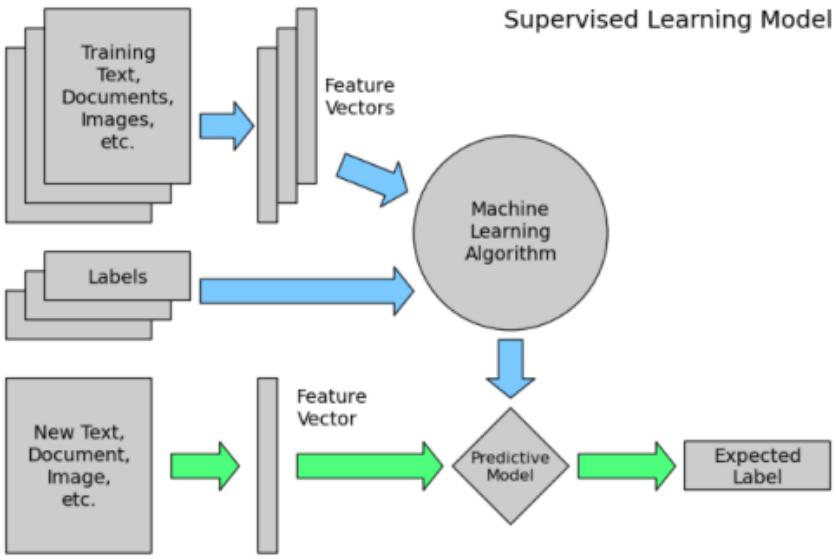

Figure 2.1: Supervised Learning model from the Course Notes for Learning Intelligent Systems by Carlos Ángel Iglesias

The methods used with this learning algorithm are:

- Regression
- Classification

2. ***Unsupervised Learning*:** this learning algorithm uses the data without knowing the results this may generate. The learning is made by searching structures or patterns in the data.

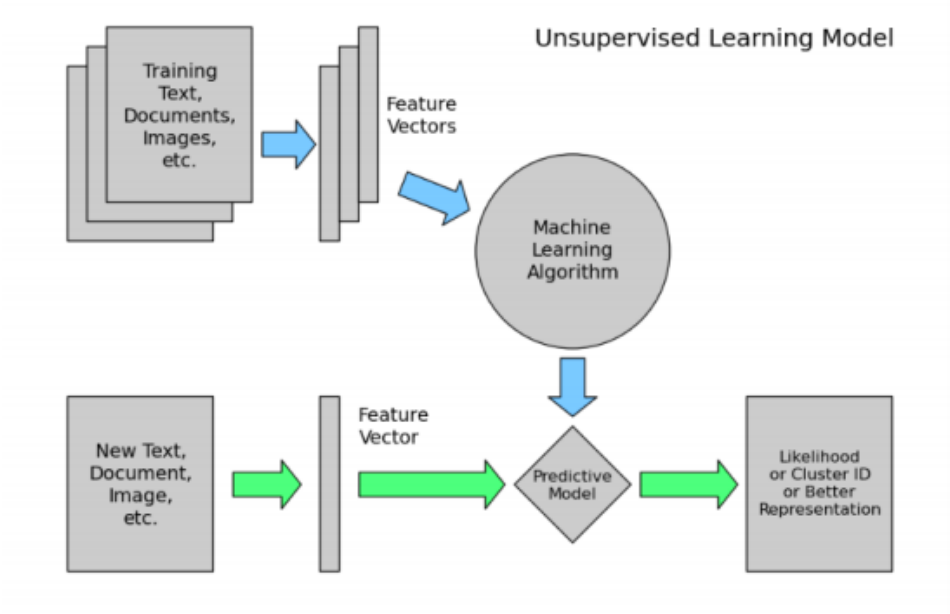

Figure 2.2: Unsupervised Learning model from the Course Notes for Learning Intelligent Systems by Carlos Ángel Iglesias

The methods used in this implementation of machine learning are:

- Clustering
- Dimensionality Reduction

2.3 Python Dependencies

The recommendation engine and logic for the project has been develop in Python. In order to archive the objectives and functionalities marked at the beginning of this project, there has been used the next dependencies and libraries.

2.3.1 Scikit-Learn

This is a Python library [17] witch provides diferent tools and algorithms for the making of this project. In this case, there was used a particular distribution of this library in order of getting the best affinity betwen the tools used and the requirements of the project.

2.3.2 Surprise

This library [15] conforms a distribution of Scikit-learn , aplying some of the functionalities of the main library to make recommender and prediction systems. In first instance, this library is built with a dataset of movies and user ratings. For this project, there has been used the following datasets:

- **Million Song Dataset (MSD) [2]:** contains information over 10.000 songs and the listenings of each song by 76.353 users. The informartion for each song is "Title", "Artist name", "Album" and "Release year".
- **MoodyLyrics [3]:** assigns a mood to each song from 4 categories/quadrants extracted from the Last.fm users assesments.

2.3.3 Pandas

This is a Data Analyst Library [23] gives to the project the tools for managing the data of songs and users in the first instance. Also it has the necessary tools for plotting the data in graphs that helped in the making of the early stages of this project.

2.3.4 Django

Django [5] is a Model View Controller (MVC)[29] based web framework for python applications which can be deployed with paralel services in the same server.

This software has been selected for its support in the implementation of end-user systems and the handling of the models. Thanks to these, the development of the dashboard has been fast and clean.

2.3.5 Pickle

Pickle [24] is a solution for serializing and de-serializing Python objects, in order to conserve its caracteristics and relations and getting a transferable stream of bytes.

In this project it has been used to serialize the trained algorithm for his posterior use in the final system.

2.3.6 Firefly

Firefly [25] is a function as a service framework which can be used to deploy functions as a web service. It makes the functions accesible via an API REST with an assigned IP or url.

This software has been used in the implementation of the prediction algorithm into the dashboard.

2.4 Postgre SQL

Postgre SQL [9] is an open source object-relational database that uses and extends the SQL language to access and handle the data combined with many features that help to the correct devepoment of the system. Also provides a series of security caracteristics optimum for an user-end system.

The election of this sofware was made due to the compatibility with the framework used and the capability of handling large amounts of data. Also it is well-known for his scalability.

2.5 Jupyter Notebook

This open-source Web Application [18] disposes the ability to create and share documents that contain live code, equations, visualizations and narrative text. Can be used with several code languages and supports Python.

The main use of this application is to make the code easily accessed and editable at the same time it can be explained or labelled to its demonstration or teaching.

In this project it has been used to make all the initial simulations and tests to select the algortihm to be used in the system. The notebook where this tasks are developed were used to demonstrate other students and professors the caracteristics of the machine learning used in this project.

2.5.1 Google Colaboratory

Google Colaboratory [8] is an enviroment based on Jupyter Notebook executed completely in Cloud systems provided by Google.

The use of this enviroment was made due to the limitations of the computer devices the author has access and to make the notebook more accesible. Since it provides the user with 13GB of memory RAM, 40GB of disk space and the computing power of the Google Cloud systems.

CHAPTER 3

The model

3.1 Introduction

The recommendation nowadays have become very important in the content industry, in special in the entertainment sectors such as Music, Books, Films, etc. The making of these systems have an important factor in the preprocessing of the data.

In this chapter a technical description of the developed model will be given.

Firstly, a description of the datasets used and the format of the data, next will be the preprocesing and munging of the data in order to obtain the optimum parametrers and format for the system. Lastly, the algorithm simulation and the results obtained.

3.1.1 Overview

This is the global view of the process followed in first instance to make the simulations to obtain the best algorithm for the system.

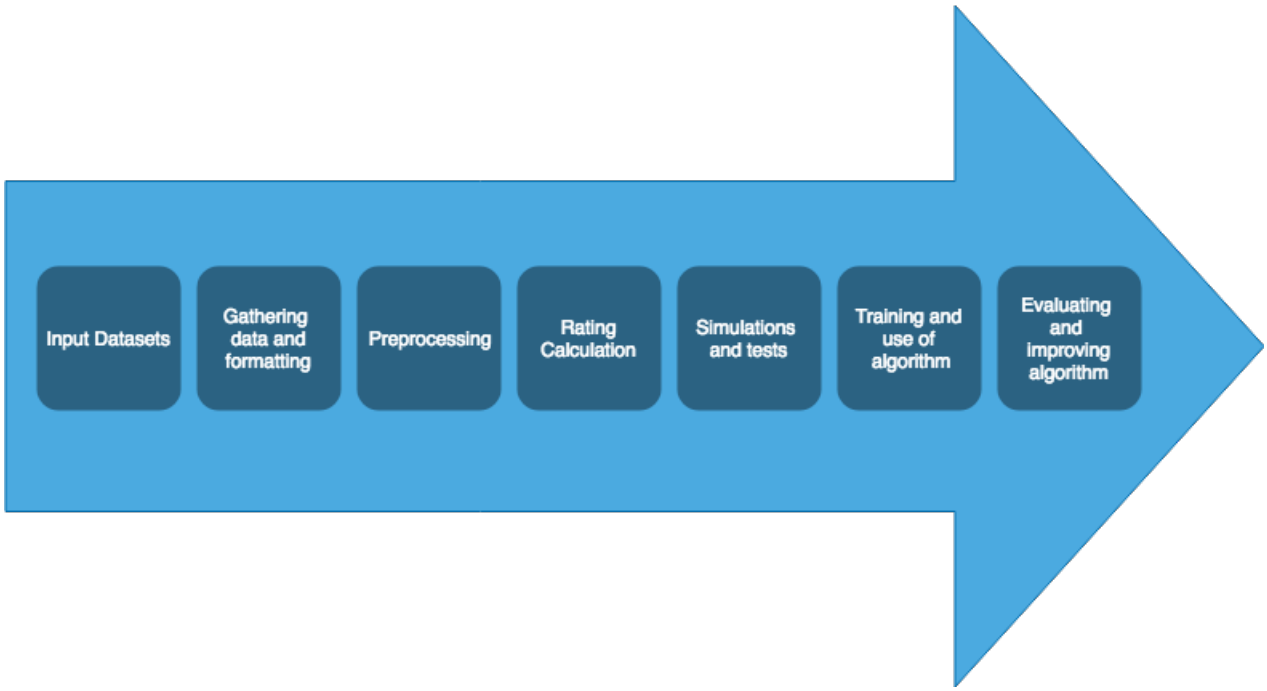

Figure 3.1: Process followed to make the model

The collection of data, as show in the previous chapter, is made from the Million Song Dataset [2] and several other smaller datasets that add relevant information to this. In this part, the main objective is the collection of important data to make the final program the more complete for the user.

The next parts of the process are described in the following sections.

3.2 Dataset

The Dataset used to gather all the data, One Million Song Dataset [2], has the basic information of 10,000 songs. This information covers the author, album, year and name of the song.

The second dataset added to the operation is the Echo Nest [1] which associates the songs to a list of 120.000 users and also provides the number of listenings for each user in each song.

The last dataser added is the Moody Lyrics [3] which has the component of giving a mood to each song. This information is considered as additional for the classification of the recommender once the recommedation is done.

With the three datasets mounted and reunited in one, the data is presented as the next table.

user_id	song_id	listen_count	title	release	artist_name	year	song	mood
b80344d0...	SOBYHAJ12A6701BF1D	1	Constellations	In Between Dreams	Jack Johnson	2005	Constellations - Jack Johnson	relaxed
b80344d0...	SOMSQJY12A8C138539	1	Breakout	There Is Nothing Left To Lose	Foo Fighters	1999	Breakout - Foo Fighters	happy
bd4c6e84...	SOJGMYY12AB01809BE	2	Who Knows Who Cares	Gorilla Manor	Local Natives	2009	Who Knows Who Cares - Local Natives	sad
e006b1a4...	SOBONKR12A58A7A7E0	2	You're The One	If There Was A Way	Dwight Yoakam	1990	You're The One - Dwight Yoakam	happy
b64cdd1a...	SOFWANS12AF72A12E6	1	Sad Songs And Waltzes	Fashion Nugget	Cake	1996	Sad Songs And Waltzes - Cake	sad

Table 3.1: Dataset used to calculate the ratings

3.3 Preprocessing

The preprocessing step is conceived to dispose of any information no needed for the correct functionality of the Machine Learning system. In this case, this stage has been applied to all songs in the dataset with the following steps:

3.3.1 Data depuration

The initial dataset contained a lot of information that exceeded the memory limits of the hardware. In order to provide a rational amount of data with the correct format, there has been removed all the entries that cumpliment these points.

- ***Cleaning Null fields*:** this include all the entries that contain some null data in the main fields, such as the song's name, the author name, the album or the year when it was released.
- ***No emotion recorded*:** in the merge of the song's dataset with the emotion/mood one, there were a significant cuantity of songs that did not have an actual emotion record. These songs were removed from the dataset in order to reduce the data given to the system and give it the as complete as possible.
- ***Random data reduction*:** as mentioned before, the hardware requeriments for the amount of data used in this system is too high for a normal system. Since there is no access to high level computing systems, we needed to reduce the data in some way. For making this reduction there has been used a random set of songs that changes every time the program is ran. The redcution made is a 85% of the original dataset.

3.4 Calculating the Ratings

For making a recommeder system, there is needed to take a rating field contained in a given scale of data so the Machine learning system can use this information to make the comparison and posterior prediction of this field.

In order to provide the best option to the final program, there has been used three different rating sets for the model simulation.

In the dataset used there is a "listen count" field that provides the listenings made by the user for a specific song. This information can give an aproximation of the users musical taste for the songs that has listened.

There are some points needed to know before taking part in further explanations:

$$N = (\text{Total number of listenings for all users})$$

$$user\_listen\_count = \begin{pmatrix}\text{Number of listenings of one con-}\\ \text{crete user for a concrete song}\end{pmatrix}$$

$$sum\_user\_listen\_count = \begin{pmatrix}\text{Sum of all the listenings made for}\\ \text{one concrete user}\end{pmatrix}$$

$$n\_users = (\text{Total number of users})$$

The rating measure is calculated assuming that the more one user listen a song, the more likes it. Therefor, with this calculation there must be a specific rating for each user associated with one particular song.

3.4.1 Rating 1

This rating is the most suitable with the comparison of users musical taste. The rating is calculated with the number of listenings for one song made by the user divided by the total number of listenings of all users.

$$rating\_1 = \frac{user\_listen\_count}{N} \qquad \begin{matrix}\text{min value} = 0\\ \text{max value} = 1\end{matrix}$$

The reason this rating is optimal for comparisons between users is because there is not ponderation of the value, is a direct relationship between the users listenings and the total listenings.

The demonstration of minimum and maximum values is trivial. In addition, the more user listenings, the more accurate will be the value of the rating.

3.4.2 Rating 2

This rating is an aproximation of the third one, saving processing capability of the system. Both ratings give an accurate measure of the musical taste of one particular user. The calculation is made by dividing the users listenings for the song by the mean of listenigns per user.

$$rating\_2 = \frac{user\_listen\_count}{N/n\_users} \qquad \begin{array}{l}\text{min value} = 0\\ \text{max value} = 1\end{array}$$

As previously said, the musical taste of each user is the most accurate with this rating. Nevertheless, this rating does not take into account the cuantity of listenings made by the user.

For example, if an user have 10.000 listenings and 1.000 were made for the song X and there is another user that has made 10 and 1 were for the song X. Both have the same rating for the song X, but obviously the first one has more probabilties of "liking" the song than the second one.

The demonstration of the maximum and minimum values is very simple:

$$min\{rating\_2\} = \frac{min\{user\_listen\_count\}}{min\{N/n\_users\}} = min\{user\_listen\_count\} = 0$$

$$max\{rating\_2\} = \frac{max\{user\_listen\_count\}}{max\{N/n\_users\}} = \frac{N}{N} = 1$$

3.4.3 Rating 3

This rating is similar to the previous one, with the caracteristic of taking into account the real number of listenings of the users. Is calculated with the divition by the users listenings for a particular song and the total listenings of this user.

$$rating\_3 = \frac{user\_listen\_count}{sum\_user\_listen\_count} \qquad \begin{matrix} \text{min value} = 0 \\ \text{max value} = 1 \end{matrix}$$

Like the previous one, this rating doesn't have recognise the amount of listenings made for the users. This only affects in extreme cases, like the previous example. Yet, this will affect mainly to the users with low listenings.

The demonstration of minimum and maximum values is trivial. Also, it is remarkable that the values of this and the previous rating will be significantly higher than the values taken in the first. This is caused by the N denominator, that can take very high values, especially if we are in a real music platform.

3.5 Simulations and tests

In order to find the most accurate combination between algorithm and the rating parameter. To determine this task, there has been made a study of the RMSE (Root Mean Square Error) returned by each algorithm trying diferent parameters and ratings.

3.5.1 Algorithms tested

All the simulations has been made with the tools given by Surprise [15]. Also, the algorithms used and their parameters were the offered by Surprise.

3.5.1.1 Normal Predictor

This algorithm [14] is custom built in the tools of "Surprise". It predicts a random rating based on the distribution of the training set, which assumed to be normal.

In this case, the prediction made for the rating $\hat{r}_{ui}$ is generated from a normal distribution $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$ where $\hat{\mu}$ and $\hat{\sigma}$ are estimated from the data using Maximum Likelihood estimation [28].

$$\hat{\mu} = \frac{1}{\|R_{train}\|} \sum_{r_{ui} \in R_{train}} r_{ui}$$

$$\hat{\sigma} = \sqrt{\sum_{r_{ui} \in R_{train}} \frac{(r_{ui} - \hat{\mu})}{\|R_{train}\|}}$$

3.5.1.2 Baseline Estimation

This algorithm [19] as previous one, is built for the "Surprise" tool. The prediction is made estimates the baseline for given user and song. This is:

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$

The parameters b_u and b_i indicate the observed deviations of user u and item i respectively, from the average.

For further information about the process and mathematical model of this algorithm, see the section 2.1 of [19].

3.5.1.3 Basic k-Nearest Neighbours

There are four algorithms derived from the Nearest Neighbours approach [16]. In this case, there is used a basic implementation [27] of this approach for the ML.

In consideration of the mathematical model, the prediction made has two implementations, one for the user comparison (3.1), and other for the item comparison (3.2):

$$\hat{r}_{ui} = \frac{\sum_{\nu \in N_i^k(u)} sim(u, \nu) \cdot r_{\nu i}}{\sum_{\nu \in N_i^k(u)} sim(u, \nu)} \tag{3.1}$$

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} sim(i, j) \cdot r_{ji}}{\sum_{j \in N_u^k(j)} sim(i, j)} \tag{3.2}$$

In this equations, the k parameter is adjustable. In addition, the similiraties for the prediction can be chosen to be made with different measures. The available measures [10] in Surprise are:

- Cosine similaritie between all pairs of users or items.
- Mean Squared Difference similarity between all pairs of users or items.
- Pearson correlation coefficient between all pairs of users or items.
- Pearson correlation coefficient between all pairs of users or items using baselines for centering instead of means.

3.5.1.4 k-Nearest Neighbours with Means

This is a built of the previous algorithm with the consideration of the mean ratings of each user. This approach is more favorable for predicting the rating of the users with low quantities of listenings.

The algorithm can be used to make similarities between users (3.7) or between items (3.8).

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{\nu \in N_i^k(u)} sim(u,\nu) \cdot (r_{\nu i} - \mu_\nu)}{\sum_{\nu \in N_i^k(u)} sim(u,\nu)} \tag{3.3}$$

$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} sim(i,j)} \tag{3.4}$$

The k parameter is adjustable and provides to the algorithm the number of neighbours to compare. It is important to choose an optimal value, since it can affect directly the bias and variance of the process.

3.5.1.5 k-Nearest Neighbours with Z score

In this case, the algorithm parts from the previous and adds the standard score normalization [32] for each user. This approach [13] has the factor of give a prediction weighted with the medium deviation of the nearest neighbours.

As the previous collaborative filtering algorithms, it support the configuration of making comparisons between users (3.5) or items (3.6).

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum_{\nu \in N_i^k(u)} sim(u,\nu) \cdot (r_{\nu i} - \mu_\nu)/\sigma_\nu}{\sum_{\nu \in N_i^k(u)} sim(u,\nu)} \tag{3.5}$$

$$\hat{r}_{ui} = \mu_i + \sigma_i \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)/\sigma_j}{\sum_{j \in N_u^k(i)} sim(i,j)} \tag{3.6}$$

If the σ of a group of neighbours is 0, it is used the overall deviation of all the users.

The k parameter is adjustable and provides to the algorithm the number of neighbours to compare.

3.5.1.6 k-Nearest Neighbours and baselines

This algorithm [11] takes the base of the nearest neighbours approach with baseline rating estimation to make the predictions.

As shown in the section 3.5.1.2, the b_{ui} parameter is an estimation made with the deviations of users and items.

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{\nu \in N_i^k(u)} sim(u,\nu) \cdot (r_{\nu i} - b_{\nu i})}{\sum_{\nu \in N_i^k(u)} sim(u,\nu)} \tag{3.7}$$

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - b_{uj})}{\sum_{j \in N_u^k(i)} sim(i,j)} \tag{3.8}$$

Similar to the previous collaborative filtering algorithms, the k parameter is adjustable to select the number of neighbours for each prediction.

For further information of the mathematical model of this algorithm, you can see the section 2.2 of [19].

3.5.1.7 Singular Value Descomposition

The Singular Value Descomposition [31] consist in the factorization of a matrix in order to get the reduced version of the matrix and then operate with it. This method applied to ML was popularized by Simon Funk in the Netflix Prize [6].

In the "Surprise" approach of this algorithm, the prediction $\hat{r}_{ui}$ is made from equation 3.9.

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{3.9}$$

If user u is unknown, then the bias b_u and the factor p_u are assumed to be zero. The same applies for item i with b_i and q_i.

For getting the unknown values of the previous equation, the "Surprise" built makes a minimization of the regulated squared error.

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2) \tag{3.10}$$

Where the minimization is made by a straightforward stocastic gradient descent.

$$\begin{aligned} b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \end{aligned} \tag{3.11}$$

In the equation 3.11 the $e_{ui} = r_{ui} - \hat{r}_{ui}$. γ is the learning rate and λ is the regularization term of the algorithm. Both are customized in the invocation of the algorithm.

The entire process of minimization is made a number of times that can be selected. Baselines of the ecuations are initialized in 0. The user and item factors are randomly initialized according to a normal distribution with adjustable parameters.

Singular Value Descomposition Plus

In addition to the default SVD algorithm previosuly presented, "Surprise" comes with another implementation of this algorithm which takes into account the implicit ratings.

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right) \tag{3.12}$$

In 3.12 the y_j terms are a new set of item factors that capture implicit ratings. Here, an implicit rating describes the fact that an user u rated an item j, regardless of the rating value.

If user u is unknown, then the bias b_u and the factor p_u are assumed to be zero. The same applies for item i with b_i, q_i and y_i.

3.5.1.8 Non-negative Matrix Factorization

This is a collaborative filtering algorithm [30] very similar to the SVD [3.5.1.7] with the exception of being unbiased. This makes the prediction more simple.

$$\hat{r}_{ui} = q_i^T p_u \tag{3.13}$$

In the 3.14 equation, both the user and items factors are maintained positive. This implementation is based [21] in the dense matrix [20].

In addition, as the previous one, the algorithm can be optimized with a stochastic gradient descent with the caracteristic of cualifying the process with adjustable size steps that ensures non-negative factors. It is remarkable to point that the initial values are also positive.

in each step of this SGD process, the f factors of users and items are updated as shown in the equations 3.14 and 3.15 where λ_u and λ_i are regularization parameters.

$$p_{uf} \leftarrow p_{uf} \cdot \frac{\sum_{i \in I_u} q_{if} \cdot r_{ui}}{\sum_{i \in I_u} q_{if} \cdot \hat{r}_{ui} + \lambda_u |U_i| p_{uf}} \tag{3.14}$$

$$q_{if} \leftarrow q_{if} \cdot \frac{\sum_{u \in U_i} p_{uf} \cdot r_{ui}}{\sum_{u \in U_i} p_{uf} \cdot \hat{r}_{ui} + \lambda_u |I_u| q_{if}} \tag{3.15}$$

In this algorithm, it is possible to choice the initial values for the process.

For further information of the mathematical model follow to [21].

3.5.1.9 Co-Clustering

This algorithm uses co-clustering [26] to get the similarities between users and items compare to the mean. The users are assigned to clusters C_u, items to C_i and both to co-clusters C_{ui}.

The process adapted in "Surprise" is an adaptation of [7].

$$\hat{r}_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i})$$

When user or item are unknown, the prediction is set to $\hat{r}_{ui} = \mu_i$ or $\hat{r}_{ui} = \mu_u$ respectively. Also, when both are unknown, the prediciton is calculated with $\hat{r}_{ui} = \mu$.

The clusters are assigned using a straightforward optimization method, as shown in the kNN based algorithms in 3.5.1.3.

3.5.2 Study and test

To ensure the system the best recomendation engine and adaptation to different environments of users and songs, there has been made all the possible combinations between algorithms, their parameters and the ratings calculated in 3.4.

In the simulations made there has been tested a range of parameters values for each algorithm in order to obtain the best results from it. In addition, since all the ratings are valid to measure the music taste of users in the system, the objective of this simulations is to get the one that provides more information to the recommender.

It is remarkable the factor that in the results shown in this section, there are only showing the best results for each algorithm, yet there has been made multiple simulations with different results for each one and with each rating.

3.5.2.1 Rating 1

As showed in 3.4.1, this rating gives the advantage of giving a global punctuation, making easier the comparisons between users.

As said previously, the rating has been tested with all the available algorithms [3.5.1] with all the possible combinations of parameters. The results of the simulations can be seen in the table 3.2.

Algorithm	RMSE	Fit time	Test time
BaselineOnly	0.000253	0.021981	0.049817
KNNBaseline	0.000258	0.582990	0.186703
SVD	0.000259	0.530536	0.026288
KNNBasic	0.000271	0.012747	0.041799
CoClustering	0.000271	0.462106	0.021970
NMF	0.000272	0.804233	0.024033
SlopeOne	0.000273	0.042067	0.059267
KNNWithMeans	0.000277	1.023241	0.189423
KNNWithZScore	0.000289	1.018636	0.203835
NormalPredictor	0.000339	0.011839	0.023541
SVDpp	0.029697	1.010221	0.050211

Table 3.2: Results of simulations for Rating 1

As shown in the results, the algorithm with the best RMSE is the Baseline estimation algorithm 3.5.1.2. It has not the best times for fitting the data and making the test, yet there are times under the mean of the rest algorithms.

3.5.2.2 Rating 2

This rating 3.4.2 has more accuracy in measuring the "musical taste" of users than the previous one. This can be useful to measure, yet not to make similarities between users with differing number of listenings.

The results have several orders of magnitude higher than the first rating, this is because the rating has been normalized with the mean user listenings, that has also several orders of magnitude higher than the first rating.

As detailed in the previous section, the rating has been tested with all the available algorithms [3.5.1] with all the possible combinations of parameters. The results of the simulations can be seen in the table 3.3..

Algorithm	RMSE	Fit time	Test time
BaselineOnly	1.206344	0.020444	0.022347
KNNBaseline	1.223250	0.587282	0.181548
SVD	1.246681	0.544684	0.027046
NMF	1.252903	0.788804	0.049367
SVDpp	1.260652	1.015368	0.052888
KNNBasic	1.305092	0.012552	0.067038
CoClustering	1.327864	0.460164	0.022309
KNNWithMeans	1.339165	0.926313	0.195695
SlopeOne	1.366803	0.041418	0.030758
KNNWithZScore	1.369889	1.020109	0.198244
NormalPredictor	1.617798	0.011622	0.023326.

Table 3.3: Results of simulations for Rating 2

As shown in the results, the algorithm with the best RMSE is the Baseline estimation algorithm 3.5.1.2. It has not the best times for fitting the data and making the test, yet there are times under the mean of the rest algorithms.

3.5.2.3 Rating 3

This is the last rating calculated 3.4.3 and as his predecessor, it has more accuracy in the "music taste" of the users, yet it does not take into account the users number of listens neither.

As well as the previous rating, the results have several orders of magnitude higher than the first rating, this is because the rating has been normalized with the specific user listenings and this gives a result higher than the normalization with all the users listenings

(First rating).

As seen in the rest of the ratings, this one has been tested with all the available algorithms [3.5.1] with all the possible combinations of parameters. The results of the simulations can be seen in the table 3.4.

Algorithm	RMSE	Fit time	Test time
BaselineOnly	1.206344	0.020444	0.022347
KNNBaseline	1.223250	0.587282	0.181548
SVD	1.246681	0.544684	0.027046
NMF	1.252903	0.788804	0.049367
SVDpp	1.260652	1.015368	0.052888
KNNBasic	1.305092	0.012552	0.067038
CoClustering	1.327864	0.460164	0.022309
KNNWithMeans	1.339165	0.926313	0.195695
SlopeOne	1.366803	0.041418	0.030758
KNNWithZScore	1.369889	1.020109	0.198244
NormalPredictor	1.617798	0.011622	0.023326

Table 3.4: Results of simulations for Rating 3

As shown in the results, the algorithm with the best RMSE is the Baseline estimation algorithm 3.5.1.2. It has not the best times for fitting the data and making the test, yet there are times under the mean of the rest algorithms.

CHAPTER 4

Architecture

4.1 Introduction

In this chapter, we cover the design phase of this project, as well as implementation details involving its architecture. Firstly, we present an overview of the project, divided into several modules. This is followed by the explanation of each module and its functionalities.

4.2 Overview

To make more comprehensible the architecture of the global system, there has been divided into 3 modules. Each module has an objective in the process.

- ***Attribution and persistence system*:** In this module is made the asignation of ratings to the songs and is created the database model of users, songs and ratings for later usage in the visualization and prediction systems.
- ***Prediction system*:** This is the module responsible to make the recommendations with the trained algorithm. This module is in continuous contact with the attribution module in order to make the most precise recommendations.

- ***Visualization system*:** This module is the last step in the string, showing the data and recommendations made to the user. In the development of this project, the dashboard has been made with the objective of showing the functionalities of the global system, instead of an user aimed system.

In the next figure it is showed the structure of the global system in order to make a more general view of the whole system.

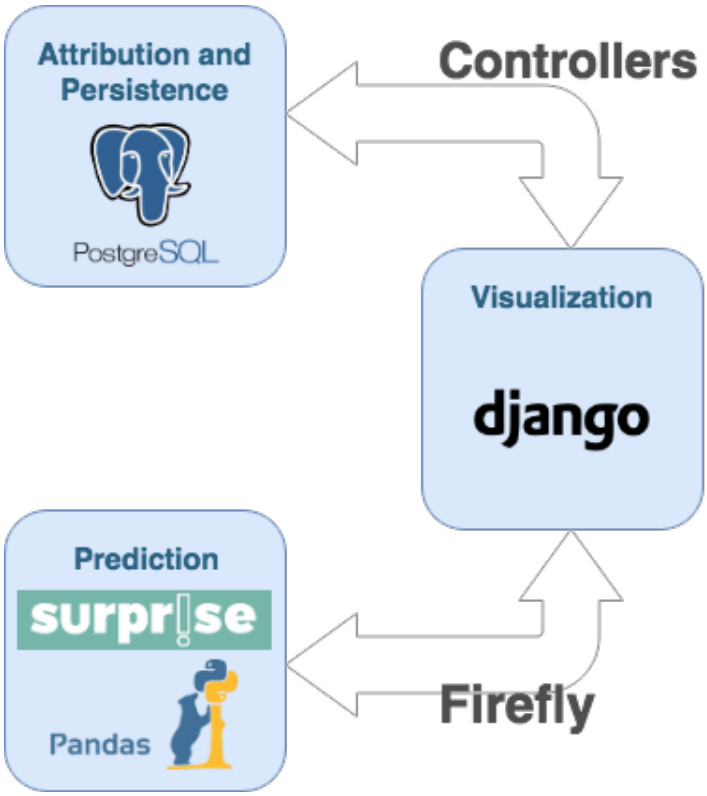

Figure 4.1: Architecture of the system

4.3 Attribution and persistence system

The principal labour of this system is to storage the data from users and songs in order to provide the prediction system the necessary information to make the recommendations. For making this posible there has been used PostgreSQL and the model clasification of Django.

In the system there are 2 principal tables of users and songs data correlated via the ratings table. The next diagram represents the actual model used in this project.

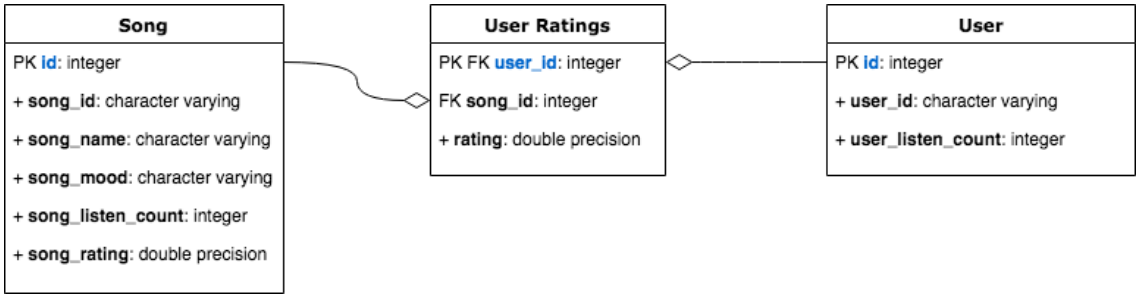

Figure 4.2: Database Model diagram

In the figure 4.2 is shown the relations between the models of this system. Both User and Song models have a relation one-to-many with the User Ratings Model, so each User can have many ratings and also each Song can have many ratings.

Since the ratings are caracterised with the song and user ids there is going to be one row per each song an user has listen at least one time. This is because the systems calculates the rating with the number of listenings of each song, as seen previously.

4.3.1 User table

This first table includes the relevant information of users in order to make the recommendation tool operate normally. In this version there has been no added personal information about the users so it is not necessary to the system operation.

Hovewer, this table is made to storage all the aditional information that an user can take, such as the name, the e-mail address, birth date, etc.

Field	**Explanation**
id	The index of each user, it identifies the position of each user in the table.
user_id	This is the user identification, it is an alphanumerical string of 40 characters. It has an unique value for each user in order to make posible the total identification of the user. This is the Primary Key of the table.
user_listen_count	Gives the information of how many listenings has made each user. This value is meant to be increment each time an user reproduces a song and it is used in the calculation of the rating as seen in previous chapter.

Table 4.1: Fields of User Model

As can be seen in the table, there are only 3 fields that define the model os User in this

system; The index, the user's identity and the total listen count of this user.

The user table has a size of 4842 unique users. These users have been randomly selected from one set of data used to train the algorithm, as can be seen in 5.3.

Considering the format of the user_id field, this system can hold the data of a maximum of $1.7868991 \cdot 10^{62}$ which is a correct value for the magnitude of this global system which objective is to serve as a reference in the music reproduction industry.

4.3.2 Song table

This table contains the relevant information of a song and a reduced identifying data. As the User model, this model has the the necessary data for the correct functionality of the global system.

If the implementation requires it, this is the table to include additional information of the song, such as the year of release, the album in which was published and other further information.

Field	**Explanation**
id	The index of each user, it identifies the position of each song in the table.
song_id	This is an alphanumerical string of 17 characters that identifies each song and it's unique value set. This is the Primary Key of the table.
song_name	Includes the name of the song followed by the name of the artist. There has been made in this format to make more accesible the data, since this is simply informative data for the dashboard.
song_mood	This field shows the mood induced by each song. As seen previously, the posible values are four: angry, sad, relaxed and happy.
song_listen_count	The number of listenings made by all the users of each song. This value has only statistical purposes.
song_rating	This field contains the mean rating of each song. This is made with all the ratings made for this song and it has statistical purposes.

Table 4.2: Fields of Song Model

This table has a lot of information that can be handled by the system in order to make global recommendations. For example a series of songs most liked by the users or the most listened.

The Mood field is included in order to provide the user a capability of decision based on the emotion desired. This is an additional functionality to the recommendation system that emphasizes the level of personalization reachable.

The initial size of the table is of 351 songs, significantly augmented in the development of this global system. The objective is to fill this table with new songs the users insert in the system in order to nourish the application and give more information to the algorithm.

With the format of song_id, the maximum number of unique song that can be obtained with this implementation is $2.865118 \cdot 10^{26}$ which is a correct value considering that in reproduction applications like Spotify or Deezer the total amount of songs rounds the 40 million songs.

4.3.3 User Ratings table

This last table contains the users ratings for each song. As said several times before and explained in 3.4, these ratings are calculated with a ponderation of the user listenings to each song and the total listenings of all the users.

The ratings have been set to a floating-point number type in order to cover all the range that this field can adopt. The minimum values of this field can get an order of magnitude 5 times under zero and the maximum can get values near one. As the number of users increase, the values of the ratings may get lower considering the method of calculation.

Field	**Explanation**
user_id	This is a foreign key of the id in the user table. This identifies the user associated with the rating. This is the Primary key of the table.
song_id	This is a foreign key of the id in the song table. This identifies the song associated with the rating.
rating	The "assessment" made by the user specified of the song specified. As told previously, this rating is calculated based in the user listenings of each song and can take values with several orders of magnitude under zero. That is the reason the type of this field is double precision.

Table 4.3: Fields of User Rating Model

In this table is included all the ratings made by the users in the user table for the songs in the song table. Considering that, there will be added ratings to this table every time an user listen to a new song, modifying all the registers the user has in this table.

The set of data in this table has been used to train the algorithm as seen in [5.3]. Since the recommendations are made with the algorithm designed in live time, this table would have no functionality in the final system other than to make statistics and represent the data to the user.

This data could be used to make a list of the preferred songs of one specific user or make playlists for each user based on the mood.

4.4 Prediction System

Once the simulations were made and the algorithm was selected [5.3] and trained, there was done a Pickle [24] serialization to keep the algorithm until the service is started.

As mentioned in the next chapter 5, the serialization is undone for mounting a series of functions as a service with Firefly. The functions made in this implementation were:

- ***Predict*:** { parameters: *user_id*, *song_id* } Given an user and a song, the function predicts using the k-NN Pearson Baseline algorithm the rating this user will most probably assign to the specific song.

- ***Recommend*:** { parameters: *user_id* } this function returns the song with the greater rating value predicted for the user given. It must be stressed the fact that the songs recommended are always songs the user have not listened to.

- ***Recommend List*:** { parameters: *user_id*, *mood* } returns a list of 5 songs ordered by the rating assigned to the user and with the optional filter of mood. The default mood is "none", so it selects all the possible options.

This three functions are available as a service with the Firefly framework and be accessed by the Visualization system with petitions HTTP GET to the IP direction where are published.

4.5 Visualization System

As mentioned previously, this is the final module until the end-user. This includes the dashboard and the synchronization of all the services used in the system.

There has been used a MVC pattern to be able of managing the data and functions more easily. With this pattern the whole system can be reduced to a module with access to the rest. In this case the visualization module has the control of the models and the services from the persistence and the predictions systems respectively.

The Visualization system was developed and deployed in a local environment of a personal computer, this means that the final system can be runned in a computer without a high-end performance, therefore in a conventional server will be satisfied the requirements for this application.

CHAPTER 5

Case study

5.1 Introduction

In this chapter will be explained the process of creation of the Visualization System and the results obtained once it was finished. This module is the last layer before the user-end and is in charge of gathering all the data in one dashboard adapted to the situations described previously.

5.2 Implementation and test of the dashboard

As mention previously, the dashboard had the initial objective of being an user interface with the functionalities this has inherited. Since the simulations and development of the prediction system take long time, there has been adopted the solution of making a dashboard for the representation of the system.

The dashboard made does not include the implementation to give service to end-users, yet it has the characteristics for doing it. There has been only implemented single access to the platform in order to see the recommendation made to every user in the platform.

For developing the interface there has been used Django [5], a python web framework based in the Model View Controller pattern [29]. The selection of this software was with the aim of including an user login system. Once the project change the focus to make a demonstration platform, this idea was discarded.

Once the algorithm were trained, there was mounted the models of database seen in the previous chapter 4 with the Django functionality. For making the database, there was used PostgreSQL software. The Django framework translate the objects defined to the SQL language that this software handles.

At this moment the main structure of the interface was made, including the database. It was then when the algorithm was trained and the data mounted in the model system of Django.

5.3 Algorithm selection

As seen in the chapter 3, there have been a whole process of simulation to obtain the best algorithm available in the resources used. After this simulation and test, there has been selected the algorithm k-NN [3.5.1.6] using a similarity of the pearson baseline.

In the results showed in the table 3.2 the algorithm with best results is the Baseline Only, yet there has been observed that with a sufficient amount of data for training the algorithm, the k-NN with pearson Baseline obtained better RMSE and similar execution times.

The both algorithms has a similar implementation, yet the k-NN have the bonus of making the similarities between users with a pearson baseline explained in the nest lines.

Considering the basic baseline algorithm [baseline-estimation], the Pearson baseline similarity is made by computing the Pearson correlation coefficient between all pairs of users using baselines for centering instead of means.

The Pearson-Baseline correlation coefficient used is defined as:

$$PBCC(u,v) = \hat{\rho}_{uv} = \frac{\sum_{i \in I_{uv}} (r_{ui} - b_{ui}) \cdot (r_{vi} - bvi)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - b_{ui})^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{vi} - bvi)^2}}$$

The computation of the coefficient is then defined as:

$$PBCC(u,v) = \frac{|I_{uv}| - 1}{|I_{uv}| - 1 + shrinkage} \cdot \hat{\rho}_{uv}$$

The shrinkage parameter helps to avoid overfitting when only few ratings are available. The value used in the development is 100.

The algorithm has been trained with a total of 40,000 users and 400 songs, with all their ratings calculated. Due to the limitations of the devices used during the development there has been divided the data into sets of 5,000 users to make the training using clustering.

To test the algorithm has been used the 25% of data from a set of 5,000 users selected randomly between the existing 8 sets. Subsequently, the 5,000 users selected were included in the mentioned model system of Django.

5.4 Dashboard functionalities

The dashboard initial page shows the list of 5,000 users each have their correspondent page of information and several functions for proving the recommendation functionality of the system.

The user page shows an interface that can be the initial page seen by one user when is logged in the system. In this interface is showed the information of the user listenings and the lists of songs with the users ratings for each one.

At the bottom of the page there are 2 buttons, one that give the prediction of one song the user will like and the other make a list of the songs it is more probable the user will be accommodated.

In both cases there can be selected the mood by the user. By default, there are selected all the possible moods.

Otherwise, there is an additional page for showing the list of songs and each song information.

CHAPTER 6

Conclusions and future work

In this chapter there will be described the conclusions extracted from this project, followed by the achieved objectives of the project and the problems encountered in the development. Finally, there will be shown the lines to follow in the future of this project.

6.1 Conclusions

Against to the expected in the initial stages, the project has adopted a more theoretical position in favour of optimize the recommendation engine and get the best markers for the algorithm. This has achieved an optimal characteristics that can be improved with the addition of new data to the system.

Therefore, due to the importance of the algorithm developed, the dashboard implemented has the focus in showing the capabilities of the recommendations made by the system and the possibilities of implementation in a reproduction system with users.

Once there was selected the k-NN Baseline algorithm and made the proper tests, the results obtained of the trained algorithm were:

Metric	Value
Mean Absolute Error (MAE)	$9.017 \cdot 10^{-5}$
Root Mean Square Error (RMSE)	$1.97 \cdot 10^{-4}$
Fraction of Concordant Pairs (FCP)	0.4879

Table 6.1: Metrics of the k-NN with Pearson Baseline algorithm

6.2 Achieved goals

The objectives achieved in the final moment of development of this project were:

- **Dataset with emotion factor:** there has been made an aggregation of several datasets in order to obtain a dataset that combines the principal information of songs and the emotion provoked by their listening.
- **Model creation:** there has been created a MVC dashboard with the characteristic of implement a machine learning algorithm that is capable of recommending songs and make predictions based in the information of some users.
- **Development of a dashboard:** this goal has been achieved with the Django framework. It is needed to mention that there has not been implemented an user login due to the limitation in time of the project.
- **Intuitive design of the dashboard:** using CSS and characteristics of the template aspects of Django, the web application developed has the correct simplicity to use it without knowing its functionalities.

6.3 Problems encountered

During the developing of this project the main problems faced were:

1. **Limited computing power:** at the beginning of the project, the development of the simulations were made in a computer with its consequent limitation. When it came to action the test of algorithms such as the k-NN, the system were unable to complete the tasks due to the large amount of RAM memory needed.

As solution to this problem, the development was deployed newly in the Google Colaboratory environment.

2. **Difficulties mounting the dashboard:** initially there was selected the software of Heroku considering it compatibility with Firefly. Nevertheless, when making the firsts test of the Heroku application, the majority of the dependencies used in the project had incompatibilities with the developing environment.

3. **Limited time for achieving objectives:** the previous problems and difficulties made the project to delay in its correct developing an in this way functionalities pointed in the objectives were impossible to make.

   The functionalities not achieved are:

   - *User login system with possibility of making the platform interactive.*
   - *The addition of a music reproduction system*

6.4 Future work

Due there are some functionalities that have not been implemented in the system and it has potential to be a reference in the music listening systems, next are shown and explained the future lines of this project to complete:

- ***Include an user login system*:** since it was an objective of the project and it has not been done due to the mentioned difficulties, the creation of a profile for each user with its credentials is a task with weight.
- ***Add music reproduction*:** one of the applications of this project is to be integrated with a music reproduction system to make the song recommendations. The implementation of a reproductor system with streaming reproduction can take this program to an advantage position in the world of machine learning systems.
- ***Increase the number of songs*:** the objective is to make this system the more relevant in the sector, it is necessary to include a huge range of songs to being able to engage more users to use the platform.
- ***Make recommendations based in the emotion of the user*:** currently the system takes de emotion as a filter to the recommendations made. It will be a great advantage to make the system recommendations taking into acount the feelings of the user.

APPENDIX A

Impact of this project

The automated recommendations are increasingly more present every day. In this project were intended to cover the aspects of high accuracy recommendation systems combined with the emotion factor.

This appendix reflects the impacts that meant to be made by this project in the different areas of the actual society

A.1 Social impact

Considering the relevance of the music reproduction systems in the actual society, the making of a music recommendation system with the added value of taking into account emotions could have multiple applications in the current music streaming platforms.

The music and emotions are two concepts linked by definition and with this system the principal objective was to make possible the recognition of the consequent emotions derived from listening to music and automate its recommendation by machine learning.

With this project, it is closer the possibility of reach a systems with implications in the real emotions of an user. The dataset created and the tool made are the pillars to make

this possible.

A.2 Economic impact

The economic impact of this project can be characterized in two different approaches, considering who is the purcharser of the project, an end-user or a company.

In the case it is an end-user, this system has the tools and functionalities to be a platform for consulting music. If it is considered the future lines of this project, the system can be adopted as a music reproduction platform with recommendation functionality.

Otherwise, considering the companies as the client, this project can be described as an optimized algorithm for music recommendation that can be implemented in a music reproduction platform.

A.3 Environmental impact

The computational systems and the telecommunication infrastructures used to supply the world with communications and computing power consume great amounts of resources of the planet, which has a negative aspect for the conservation and preservation of the environment.

In this project all the computing machines used contribute negatively to the environment. There were not used other resources than these.

A.4 Ethical impact

The principal ethical point treated in this project is the automation of process that can be made by human. With the implementation of high-end accurate machine learning systems in the society is being provoked a job loss in all the sectors of the society.

In the other hand is the more demanded content consumption specially in the audiovisual sector. The developed system comes to satisfying this demand and optimize the existing offers.

APPENDIX B

Economic budget

As a project with defined objectives and seeking its application into the world, it is necessary to dimension the costs and the resources used in the whole process. Having defined all the characteristics and consequences of the project, the economic assessment is necessary to terminate the definition of the project.

Firstly will be treated the physical resources used in the development, such as computing systems and connection systems. Thereupon, there will be justified the human resources and finally the taxes attached to the whole project.

B.1 Physical resources

This project required a personal computer for his development. The capabilities of the computer used are:

- **Hard Disk:** 250 GB SSD
- **RAM Memory:** 8GB DDR3
- **CPU:** Intel Core i5 1,8Ghz

- **Operating System:** Unix Based (Ubuntu, Linux, CentOs, MacOS,...)

The cost of a computer with this capabilities at least is around the 700€ in 2019.

As mentioned previously in the document, there has been used the computing systems of Google Colaboratory with no cost. Regardless, for taking into account all the factors, the capabilities of the machine provided were:

- **Hard Disk:** 33GB
- **RAM Memory:** 13GB
- **CPU:** 2vCPU @ 2.2GHz

As the use of this system does not take any cost, it will not be conceived into the global costs of the project. Therefore, a system with this capabilities costs around 900€ in 2019.

For the launch of this system is necessary to perform the deployment in a server system with suffice computing power. Since the project has been developed to show the power of the algorithm selected, the supposed cost of this deployment will not be considered neither in the total cost of the project.

B.2 Human resources

The human resources used for the realization of this system can be calculated with the total hours dedicated to the development of the entire project. Considering the project has had a duration of 7 months and a half with approximately 22 working days per month and a dedication of 5 hours per day, the total hours dedicated to the project are approximately 825 hours.

To make the project there is no need of having any degree, so considering it is realized by a undergraduate engineering student which gross salary per hour rounds the 10€, the total costs for human resources are 8.250€.

B.3 Taxes

The realization of this project should be considered as a job in Spain, in this case the state taxes are applied to the salary of the workers. Also, the devices bought for the project involve a VAT for the transaction.

Currently in Spain the taxes imposed to the salary are for paying the social security of the worker. From the point of view of the recruiter, the percentage to pay is an 23,6% of the salary. Considering the 8.250€ of salary for the employed, the taxes associated are a total of 1.947€.

Furthermore, the VAT applied over the computer systems in Spain is the 21% of the price. Therefore, the VAT to be payed for this project is 147€.

B.4 Conclusions

Taking into account all the aspects contemplated in this appendix, the total cost of the project can be calculated from the sum of the computer system and the salary of the worker.

Therefore, the total cost of the project ascend to **8.950€**.

Bibliography

[1] The echo nest taste profile subset, the official user data collection for the million song dataset.

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. in proceedings of the 12th international society. Music Information Retrieval Conference, February 2011.

[3] Erion ¸c ano. *MoodyLyrics: A Sentiment Annotated Lyrics Dataset*. Tft, Polytechnic University of Turin, Department of Control and Computer Engineering, Fourth International Conference on Artificial Intelligence and Applications, AIAP 2017, Vienna Austria., May 2017.

[4] Fabián A. Contreras. Introducción al machine learning. `https://docplayer.es/74142729-Whitepaper-introduccion-a-machine-learning-por-fabian-a-contreras-arqui.html`, 2016.

[5] Django Software Foundation. Django. `https://www.djangoproject.com/start/overview/`.

[6] Simon Funk. Singular Value Descomposition for netflix. `https://sifter.org/~simon/journal/20061211.html`, December 2006.

[7] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.6458&rep=rep1&type=pdf`, 2005.

[8] Google. Google colaboratory. `https://colab.research.google.com/notebooks/welcome.ipynb`, 2019.

[9] PostgreSQL Global Development Group. Postgresql. `https://www.postgresql.org/`, 2019.

[10] Nicolas Hug. Similarities module from surprise. `https://surprise.readthedocs.io/en/stable/similarities.html#module-surprise.similarities`, 2007.

[11] Nicolas Hug. k-nearest neighbors baseline algorithm. `https://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNBaseline`, 2017.

[12] Nicolas Hug. k-nearest neighbors with means algorithm. `https://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNWithMeans`, 2017.

[13] Nicolas Hug. k-nearest neighbors with z score algorithm. https://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNWithZScore, 2017.

[14] Nicolas Hug. Normal prediction algorithm. https://surprise.readthedocs.io/en/stable/basic_algorithms.html#surprise.prediction_algorithms.random_pred.NormalPredictor, 2017.

[15] Nicolas Hug. Surprise, a Python library for recommender systems. http://surpriselib.com, 2017.

[16] B. Mirkin I. Wasito. Nearest neighbour approach in the least-squares data imputation algorithms. http://www.dcs.bbk.ac.uk/~mirkin/papers/is05.pdf, February 2004.

[17] INRIA. Scikit-learn, 2017.

[18] Jupyter. Jupyter notebook. https://jupyter.org/, 2019.

[19] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. http://courses.ischool.berkeley.edu/i290-dm/s11/SECURE/a1-koren.pdf, January 2010.

[20] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf, 2001.

[21] Xin Luo, Mengchu Zhou, Yunni Xia, and Qinsheng Zhu. An efficient non-negative matrix factorization-based approach to collaborative filtering for recommender systems. https://ieeexplore.ieee.org/abstract/document/6748996, February 2014.

[22] Brian McFee, Luke Barrington, and Gert Lanckriet. Learning content similarity for music recommendation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(8):2207–2218, October 2012.

[23] NUMFOCUS. Pandas: Python data analyst library, 2018.

[24] Python. Pickle for python. https://docs.python.org/3/library/pickle.html.

[25] Rorodata. Firefly for python. https://firefly-python.readthedocs.io/en/latest/, December 2018.

[26] Wikipedia. Co clustering algorithm. https://en.wikipedia.org/wiki/Biclustering.

[27] Wikipedia. k-nearest neighbors algorithm. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

[28] Wikipedia. Maximum likehood estimation. https://en.wikipedia.org/wiki/Maximum_likelihood_estimation.

[29] Wikipedia. Model view controller pattern. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller.

[30] Wikipedia. Non-negative Matrix Factorization. https://en.wikipedia.org/wiki/Non-negative_matrix_factorization.

[31] Wikipedia. Singular Value Descomposition. https://en.wikipedia.org/wiki/Singular_value_decomposition.

[32] Wikipedia. Standard score normalization. https://en.wikipedia.org/wiki/Standard_score.