Especialización en Ciencia de Datos

TRABAJO FINAL INTEGRADOR

Análisis de Características de Escucha de Canciones para un Usuario en una Plataforma de Streaming de Música Mediante Métodos de Aprendizaje Automático No Supervisado

Alumno: MBA Francisco Seguí

Tutor: Dra. Leticia Gómez

Ciudad de Buenos Aires, Noviembre 2021

Índice

1. Introducción

La música fue desde la antigüedad a los tiempos presentes una de las formas predilectas de expresión artística de la humanidad. A través de distintas culturas, geografías y circunstancias históricas; la música siempre sirvió como forma de expresión, esparcimiento y transmisión de tradiciones entre generaciones.

Con el advenimiento de servicios de streaming, se comenzó a tener disponibles datos en grandes volúmenes y detalle sobre el consumo musical de las personas. Esto permite afrontar diversas problemáticas y responder preguntas para las que antes no se contaban con los datos ni las herramientas necesarios. Por ejemplo, existen estadísticas y análisis sobre las canciones más escuchadas tanto por un usuario en particular como a nivel de países. También hay diversos sistemas de recomendación de música, pero que no proveen las características de las canciones que están siendo propuestas.

El presente trabajo tiene por objetivo proveerle a un usuario una clasificación de las canciones que escucha, de acuerdo a sus características musicales, aplicando métodos de aprendizaje automático no supervisado a plataformas de streaming de música.

2. Contexto

2.1. Definición de Música y sus Elementos

Con el transcurso de la historia la música fue evolucionando y se fue formalizando el estudio de la Teoría Musical. Es así que se identifican cuatro elementos para describirla: ritmo, melodía, armonía y timbre. Particularidades en estos cuatro elementos permiten distinguir entre diferentes canciones y estilos.

Se presenta a continuación una definición de dichos elementos en base al trabajo de Copland (1939):

- **Ritmo:** es el elemento que más fácilmente percibe quien escucha. Se asocia con el movimiento físico. Está compuesto por la métrica (distribución espacial de las notas en un intervalo de tiempo) y los acentos de las notas. Por ejemplo, un ritmo simple

de dos tiempos, podría estar acentuado en el primer pulso (UNO-dos) o segundo pulso (uno-DOS) resultando en distintos efectos sobre la pieza. Existen ritmos regulares y otros más irregulares. (pp. 47-60)

- **Melodía:** después del ritmo es el segundo elemento más importante y fácilmente detectable. Se asocia con la emoción intelectual que produce la música. Quien la escucha puede determinar si encuentra que una melodía es de su agrado o no sin la necesidad de ser músico, similarmente a lo que sucede en la apreciación de otras artes. La melodía se asocia a la sucesión de una idea musical, y se da dentro de los límites de una escala de notas musicales. Existen criterios de que hacen agradable a una melodía, como tener proporciones satisfactorias, ser fluida, tener puntos altos y bajos, no tener repeticiones innecesarias y tener momento culminante de cierre que de la idea de una idea consumada e inevitable. (pp. 60-70)

- **Armonía:** el ritmo y la melodía son elementos que naturalmente surgieron en el hombre desde tiempos primitivos, la armonía es el desarrollo de un concepto intelectual aplicado a la música. Sus orígenes se encuentran en el siglo IX. La armonía consiste en la construcción de acordes, que resultan de tocar simultáneamente una serie de sonidos. La construcción de acordes se hace en base a la distancia entre notas dentro de una escala musical. Al agregar armónicos se agregan voces, textura y riqueza a los sonidos que escuchamos en una pieza. Una melodía sola no es suficiente para crear una obra musical, en gran parte el efecto de la armonía es complementarla y darle el sentido. Existen múltiples técnicas de armonizado, y es posible hacerlo de distintas formas para una misma pieza dando lugar a diferentes arreglos de una misma obra con resultados bien distintos. (pp. 70-84)

- **Timbre:** el cuarto y último elemento es el timbre. Viene asociado con el “color de un sonido”. Su definición formal es la cualidad del sonido producido por un agente sonoro, y es fácilmente distinguible por quien escucha. Por ejemplo, una guitarra y un violín tienen distintos timbres, así como un tenor, un barítono o una soprano. Los distintos timbres se complementan y generan un efecto distinto. Dependiendo del efecto que se busca generar, utilizando distintos timbres sobre la misma pieza es posible expresar una idea diferente. (pp. 84-103)

Una mención adicional es la de la textura musical, Copland (1939) distingue 3 tipos:

- **Música monofónica:** es el resultado de tocar una línea melódica sin ningún tipo de acompañamiento (pp.104-105)
- **Música homofónica:** a una línea melódica principal, se le agrega un acompañamiento de acordes. Introduce la armonía en la pieza (pp. 105-107)
- **Música polifónica:** este tercer tipo de textura musical es el más difícil de distinguir. Consiste en varias voces melódicas independientes que juntas forman una armonía. Es de común acuerdo en la música que una persona generalmente puede distinguir al menos dos o tres voces en una pieza polifónica. (pp. 107-113)

En resumen, se ve que como el estudio de la música está formalizado y es posible expresar estilos y canciones con variables numéricas y relaciones entre estas.

2.2. Tipos de Aprendizaje

El aprendizaje estadístico busca poder aprender de los datos para una aplicación en distintos campos del saber humano (Hastie, Tisbshirani & Friemdan, 2009, pp. 1-6). Se considera que un problema de ***aprendizaje supervisado*** cuando se cuenta con una medida de salida a predecir, que puede ser tanto cuantitativa como cualitativa. Utilizamos una serie de variables predictoras para tal fin. A su vez, cuando la variable de salida a predecir es un número, estamos ante un problema de regresión. Si por otra parte buscamos determinar una clase a la cual el elemento pertenece (por ejemplo, en un modelo de churn) estamos antes un problema de clasificación (Hastie, Tisbshirani & Friemdan, 2009, pp. 9-11).

Por otra parte, es un problema de ***aprendizaje no supervisado*** cuando no hay una variable de salida definida y buscamos definir como los datos se organizan o agrupan. No existe un “supervisor” que determine si las respuestas del modelo son correctas o que indique el grado de error de cada variable (Hastie, Tisbshirani & Friemdan, 2009, pp. 485-487). Dado que se pretende analizar los datos y como se relacionan entre sí

sin querer predecir una variable objetivo, analizaremos el conjunto de datos en su totalidad.

Existen múltiples métodos de aprendizaje de ambos tipos. Algunos ejemplos de supervisados son los árboles de decisión, redes neuronales de clasificación o regresión, Support Vector Machines. Entre algunos métodos no supervisados encontramos el algoritmo K-means, o las redes neuronales de Kohonen.

El objetivo de los algoritmos de clustering es poder particionar un número de observaciones en grupos de forma tal que las disimilaridades entre los elementos asignados al mismo cluster sean lo menor posibles. (Hastie, Tisbshirani & Friemdan, 2009, pp. 501-503). Se describen a continuación algunos de los métodos principales de este tipo.

El algoritmo **K-means** es uno de los más populares en su tipo, y se utiliza en situaciones donde las variables son del tipo cuantiativo. La medida de dismiliaridad que se utiliza es el cuadrado de la distancia Euclídea respecto al centroide de los distintos clusters, la cual se va optimizando con las distintas iteraciones (Hastie, Tisbshirani & Friemdan, 2009, pp. 509-510):

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = ||x_i - x_{i'}||^2$$

Los **mapas autoorganizados** pueden considerarse una versión restringida de un K-means donde las entradas deben pertenecer a uno de los colectores del plano de salida. Son también una forma de transformar una entrada de alta dimensionalidad, en una salida de dimensionalidad reducida (Hastie, Tisbshirani & Friemdan, 2009, pp. 528-523). Fueron propuestos por Kohonen (1990) y son un tipo particular de redes neuronales. Kohonen clasifica las redes neuronales en 3 categorías. En primer lugar, las de tipo "feedforward" donde se transforma una serie de señales de entrada en otras de salidas. En las mismas, los parámetros son determinados y ajustados externamente de forma supervisada. En segundo lugar, las redes neuronales de tipo "feedback", en las cuales las señales de entrada definen el estado inicial del sistema de retroalimentación y con las sucesivas ejecuciones, este estado tiende a un valor asintótico que es la salida del sistema. En tercer lugar, están las redes "competitivas",

"no-supervisadas" o "auto-organizadas", donde las neuronas vecinas compiten entre sí por medio de la interacción lateral mutua (función de vecindad). Estas se desarrollan adaptativamente en detectores de patrones de señales, en base a lo que genera una respuesta en un sistema de coordenadas de salida de menos dimensiones.

El **análisis de componentes principales** (PCA) es un método de reducción de la dimensionalidad, donde se representa el data set mediante un nuevo grupo de variables, y se retienen las que mayor varianza tengan (es decir, información) (Hastie, Tisbshirani & Friemdan, 2009, pp. 534-536).

2.3. Spotify: Plataforma de Streaming de Música

Existen distintos servicios de streaming de música, siendo Spotify uno de los más conocidos y utilizados. Spotify (2021a) es una plataforma digital de música, podcasts y videos a nivel global. Según estadísticas reportadas por la compañía en marzo de 2021, la plataforma contaba con 356 millones de usuarios activos por mes, presencia en 178 mercados, y más de 70 millones de canciones (Spotify, 2021b). A su vez esta plataforma permite acceder a datos de la misma por medio de una API web (Spotify, 2021c).

Los usuarios de plataformas de streaming de música como Spotify, usan estos servicios tanto para escuchar canciones que ya conocen, como para poder descubrir nuevos artistas y canciones. En este último tipo de uso es que los algoritmos de recomendación juegan un papel importante en poder sugerir opciones que sean compatibles con los gustos y hábitos musicales del usuario. Sin embargo, los usuarios desconocen los motivos por los que se hacen dichas recomendaciones. Este trabajo busca poder explorar formas de darle a un usuario mayor información sobre cómo escucha música.

3. Definición del Problema

En la actualidad existen diversos sistemas de recomendación de música en base a lo que un usuario escucha regularmente, pero sin explicar los motivos por los cuales se recomiendan dichas canciones. A su vez se proveen estadísticas sobre qué artistas y canciones fueron más veces escuchadas tanto a nivel individual como a nivel países, pero nuevamente sin proveer detalles sobre las características de dicha música. El problema que se buscar resolver es la falta de información para un usuario con conocimiento musical acerca de las características y patrones de la música que escucha.

4. Justificación del Estudio

A lo largo de la historia y diferentes culturas la música sirvió tanto como forma de expresión, esparcimiento y transmisión de tradiciones. Con el paso del tiempo y la evolución de este arte se fue formalizado la Teoría Musical. Una persona con conocimiento en esta área, adicionalmente a disfrutar de escuchar una canción, puede tener interés en entender mejor las características de dicha música. Mediante un análisis de las canciones que escucha esta persona podría conocer mejor sus hábitos musicales y cuáles son las características que hacen que una canción le agrade más que otras. También podría conocer cómo se agrupan los tipos de música que más escucha, y en base a qué tipo de características musicales.

5. Alcances del Trabajo y Limitaciones

El alcance de este trabajo se concentra en lo datos de un solo usuario del rango etario de 30-35 de Argentina. A su vez se limita la cantidad de canciones a las últimas 50 escuchadas en base a los límites provistos por Spotify al consultar música recientemente escuchada. Para un mayor alcance, el análisis realizado también puede ser extensible tanto en tiempo como en cantidad de usuarios y canciones.

Una limitación del presente trabajo es que el análisis se realiza sobre datos de la plataforma Spotify, que no es la única en el mercado. En estudios más extensivos podría analizarse incorporar datos de otros servicios similares.

6. Objetivos

6.1. Objetivo General

Desarrollar una herramienta que permita a los usuarios de Spotify conocer los patrones de la música que escuchan.

6.2. Objetivos Específicos

- Obtener mediante un proceso de ETL las canciones que escucha el usuario en Spotify, junto con las características de dichas canciones
- Clusterizar las canciones recientemente escuchadas por un usuario
- Determinar los factores que caracterizan a los distintos clusters
- Analizar que métodos de aprendizaje automático dan mejores resultados para abordar el tema propuesto

7. Metodología Aplicada

7.1. Modelo Conceptual

A los fines de mostrar una prueba conceptual sobre lo expuesto, desarrollaremos una herramienta que permita a los usuarios de Spotify conocer los patrones de la música que escuchan.

Detallamos en la Tabla 1 las variables principales de estudio, que se obtendrán mediante invocaciones a Spotify Web API.

Variable	Tipo	Definición
Id	Cuantitativa	Identificador único en Base62 de cada canción
acousticness	Cuantitativa	Medida del 0 al 1 de que tan acústica es la canción
danceability	Cuantitativa	Medida del 0 al 1 de que tan bailable es la canción
energy	Cuantitativa	Medida del 0 al 1 de que tan energética es una canción
instrumentalness	Cuantitativa	Medida del 0 al 1 de que tan probable es que la canción no contenga contenido vocal
liveness	Cuantitativa	Medida del 0 al 1 de que tan probable es que la canción sea grabada en vivo
loudness	Cuantitativa	Medida entre -60 y 0 en db de que tan fuerte es la canción, relacionado a la amplitud de onda
mode	Cuantitiva (binaria)	Medida del tono en que está compuesta la canción. (mayor=1) (menor=0)
speechiness	Cuantitativa	Medida del 0 al 1 de que tan probable es que haya palabras habladas en la canción
tempo	Cuantitativa	Medida del ritmo de la canción en beats por minuto (bpm)
valence	Cuantitativa	Medida del 0 al 1 de que tan positivas son las emociones que genera la canción. valores cercanos a 1 son canciones alegres, valores cercanos a más tristes
key	Cuantitativa	Tono musical en que está grabada la canción

Tabla 1: Variables principales

7.2. Algoritmos y Herramientas a Utilizar

Para la etapa ETL se utilizan la Spotify Web API, junto con librerías de Python, como numpy, pandas, matlibplot y spotipy. Se utilizan Jupyter Notebooks para presentar los resultados del análisis. A su vez para garantizar la portabilidad del ambiente donde corren los scripts y notebooks, se utiliza Docker para configurar un container. También se utiliza Anaconda para instalar las dependencias en dicho container. De esta forma se busca maximizar la portabilidad del código. Docker se corre en WSL2 (Windows Subsystem for Linux) utilizando la distribución Ubuntu 20.04 LTS.

Respecto de los métodos aprendizaje automático no supervisados, en esta propuesta utilizamos K-Means, PCA, SOM (Self-Organizing Maps – Kohonen NN).

8. Desarrollo del Modelo

8.1. Fuentes de Información

Los datos utilizados se obtienen mediante consultas a la Spotify Web API (Spotify, 2021c). Dentro de las distintas formas de obtener datos de esta fuente, se hace foco en los dos métodos mencionados en las Secciones 8.1.1. y 8.1.2.

8.1.1. Método “Get Recently Played Tracks”

Con el siguiente método se obtienen las últimas canciones que escuchó un usuario de Spotify, con un máximo de 50 registros (Spotify, 2021d). Como entrada se proveen las credenciales de autorización (se detalla este aspecto al describir el proceso de ETL en la sección 8.2). La salida tiene formato JSON con diversas variables, que pueden consultarse en el Anexo 12.2.7. Para los propósitos de este trabajo nos quedamos con la variable [item].[track].[id] (en adelante la llamamos “ID”) que contiene el identificador único de la canción en el repositorio de Spotify, que luego utilizaremos en el método descripto en la Sección 8.1.2 para obtener las características musicales de las canciones.

En el Anexo 12.1 se puede consultar la estructura de carpetas del repositorio.

8.1.2. Método “Get Several Audio Features”

Con el método de la Sección 8.1.1 obtuvimos los ID de las últimas canciones que escuchó un usuario. Utilizando dicha variable y las credenciales de autorización como entrada, obtenemos nuevamente una salida en formato JSON. En este caso las variables que obtenemos son las que describen las características musicales de una canción en el repositorio de Spotify (Spotify, 2021e). Estas son las mismas que ya han sido descriptas en la Sección 7.1 del marco conceptual del presente trabajo.

8.2. Proceso de ETL Aplicado

Para integrar la información se programó un ETL, cuyas etapas se describen en las siguientes subsecciones.

8.2.1. Configuración de la Cuenta y Aplicación de Spotify

Para poder obtener datos de Spotify se siguen los siguientes pasos:

1- Crear una cuenta de Spotify en https://spotify.com
2- Iniciar sesión con dicha cuenta en https://developer.spotify.com/ y aceptar los términos y condiciones del servicio para desarrolladores
3- Una vez creada esta cuenta se accede a una interfaz web donde es posible registrar diferentes aplicaciones de Spotify. Estas serán las que interactuarán entre el servicio de Spotify y el usuario, como se muestra en la Figura 1 (Spotify, 2021f)

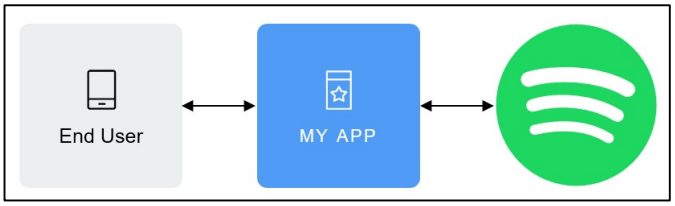


Figura 1: Diagrama de autorización de Spotify Web API

Fuente: Spotify (2021f). Recuperado de: https://developer.spotify.com/documentation/general/guides/authorization/

Se muestra a continuación, en la Figura 2, la interfaz con la aplicación que se utiliza en este trabajo, a la que llamamos “Spotify Music Analysis”

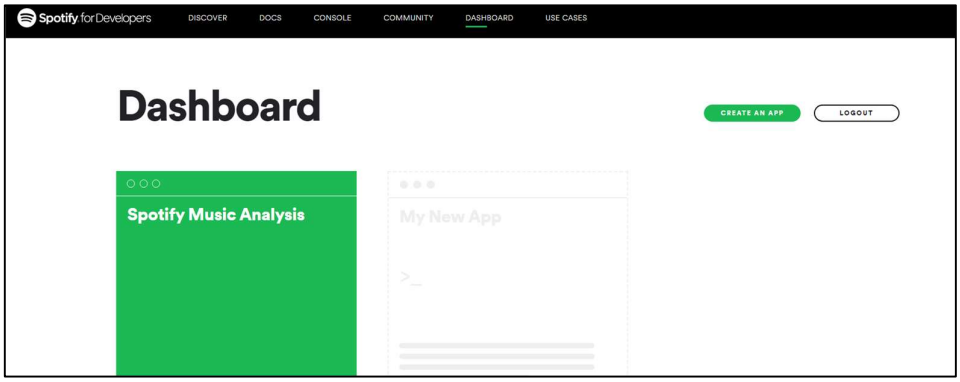


Figura 2: Interfaz de la aplicación “Spotify Music Analysis” en el portal de “Spotify for Developer”
Fuente: Elaboración propia dentro del portal Spotify for Developers

8.2.2. Configuración de las Credenciales de Spotify

Dentro de la configuración de la aplicación de Spotify es necesario configurar las credenciales para acceder a los datos. Para realizar este trabajo, se genera un archivo “credentials.py” que se configura y mantiene solamente en la máquina local y en ningún repositorio remoto para garantizar la seguridad de las credenciales (este archivo es parte del “gitignore” del repostorio).

Inicialmente se generan los tokens “Client ID” y “Client Secret”. Se obtienen y copian dichos valores a “credentials.py”. Ya que estos proveen acceso a información de la cuenta, deben ser guardados en un lugar seguro. En la Figura 3 se muestra la interfaz donde se generan los tokens de autorización.

Luego, dentro de las configuraciones, en “Redirect URIs” se inserta el valor “http://localhost:9000”, como se muestra en la Figura 4.

En el Anexo 12.2.5 se puede consultar el script es utilizado para guardar localmente los tokens de autorización.

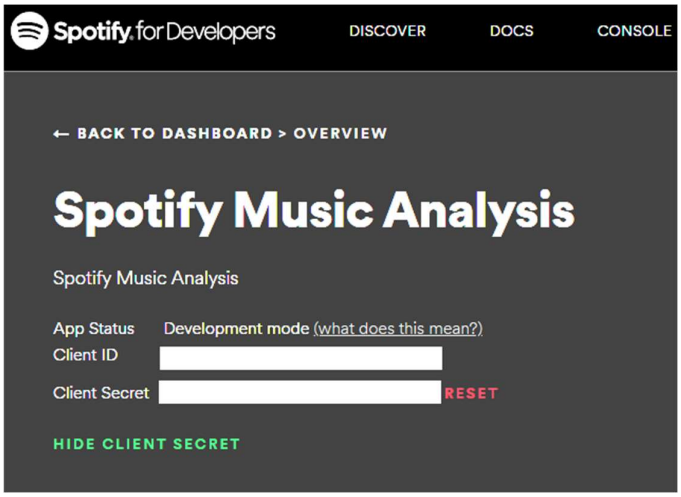


Figura 3: Configuración de tokens de autorización en la aplicación “Spotify Music Analysis”

Fuente: Elaboración propia dentro del portal Spotify for Developers

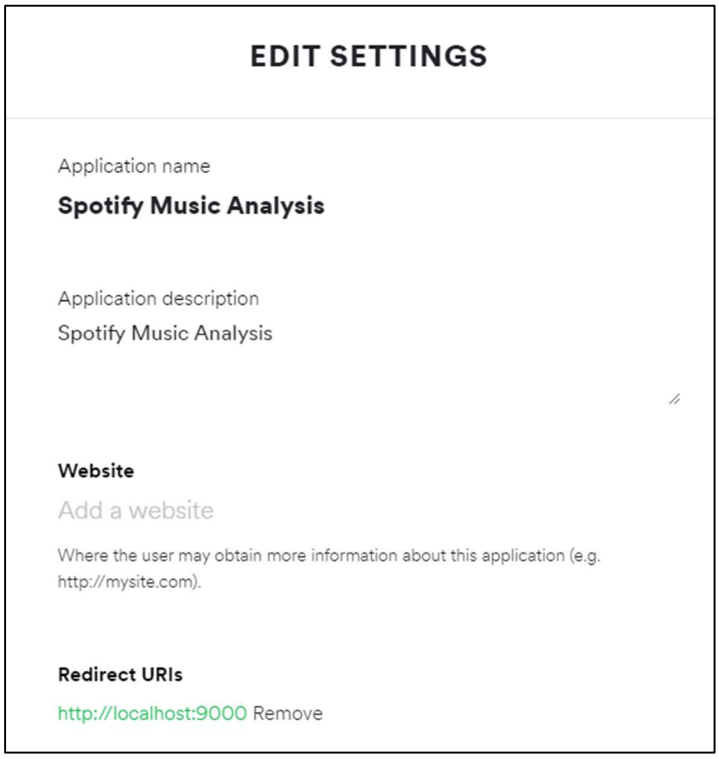


Figura 4: Configuración del parámetro “Redirect URIs” en la aplicación “Spotify Music Analysis”

Fuente: Elaboración propia dentro del portal Spotify for Developers

8.2.3. Configuración del Ambiente con Docker y Anaconda

El ambiente se corrió utilizando WSL2 (Windows Subsystem for Linux 2), y la distribución Ubuntu 20.04 LTS.

Para asegurar la portabilidad y correcto funcionamiento de la solución, los scripts y notebooks corren en un Docker container que está basado en la imagen "continuumio/miniconda3". Esto permite correr Anaconda en un container para manejar los paquetes.

La configuración de este container se encuentra en el "dockerfile". Después de seleccionar la imagen de Anaconda, el container crea y luego activa el ambiente personalizado llamado "spotify" basado en el archivo de configuración YAML "./config/Environment.yml". Esto instala los paquetes necesarios en el ambiente.

Para crear y correr el ambiente, las sentencias "docker build" y "docker run" se guardan en el archivo "./run.sh". El comando "docker run" publica el puerto 8888 en el host con el mismo puerto en el container, de forma tal que hace accesible el Jupyter notebook desde el host. También monta la carpeta del repo como un volumen del container. El comando "docker run" finalmente corre de forma interactiva el script bash "./config/startnotebook.sh". Este script bash corre el script Python "./scripts/getData.py" que utiliza la librería Spotipy para autenticarse usando tokens, y después hacer una llamada a la API para obtener los audio features de las últimas 50 canciones que el usuario escuchó. El output se guarda como un CSV en el archivo "./data/Data.csv". Este archivo es el final del ETL, y es lo que luego se utiliza en la Sección 8.3 al procesar los datos.

En los Anexos 12.2.1 a 12.2.6 se pueden consultar los scripts utilizados y los detalles del archivo YAML con la configuración del ambiente de Anaconda.

8.3. Procesamiento de los Datos

Una vez obtenidos los datos mediante el proceso de ETL, se realiza un análisis PCA. De forma exploratoria se muestra al usuario la varianza que explica dada componente principal. Luego se presenta un gráfico biplot con las dos primeras componentes principales, y vectores que muestran las cargas de las distintas variables. Los vectores de mayor módulo son aquellos que mayormente contribuyen a generar las componentes. De esta forma se provee una primera aproximación donde intuitivamente el usuario ve cuales son las características que mejor clasifican las canciones que escucha, generando foco en las variables más importantes. Finalmente, se le presentan al usuario los nombres de estas variables.

Una vez realizado este primer análisis de PCA, se evalúan los agrupamientos que se generan aplicando primero KMeans y luego SOM. Se hace un ajuste de hiperparámetros para buscar la configuración que mejores resultados provea. En el caso de KMeans se prueba con distintos números de clusters. En el de SOM con distintas dimensionalidades de matriz.

La métrica que se aplica para determinar la mejor configuración para cada algoritmo es el coeficiente silhouette. Este es una medida de la performance del algoritmo de clustering, basada en la diferencia entre la distancia media intra-cluster, y la distancia media al cluster más cercano (distinto del actual). Este coeficiente se usa también cuando comparamos la mejor corrida de cada algoritmo entre sí, para poder elegir el que mejor resultado provee.

8.4. Herramienta de Asistencia al Usuario

Se presenta al usuario un Jupyter Notebook interactivo donde se van generando los resultados del análisis descripto en la Sección 8.3, tanto de forma gráfica como analítica.

El propósito de esta interfaz es poder ayudar al usuario a visualizar y entender los patrones de las últimas canciones que escuchó, como se agrupan, y qué variables pesan más. Las celdas proveen comentarios e indicaciones, así como la posibilidad de

personalizar el análisis (por ejemplo, que el usuario pueda elegir las variables a graficar para visualizar los agrupamientos).

9. Resultados Experimentales

A modo de prueba conceptual, se presentan a continuación los resultados de aplicar el modelo descripto en la Sección 8. En primer lugar, se realiza el ETL descripto en el apartado 8.2 generando un dataset con las características musicales de las últimas 50 canciones que escuchó un usuario de Spotify de Argentina del rango etario 30-35.

Dichas características se capturan en las variables que fueron descriptas en la Sección 7. Como parte del proceso de ETL se crea y activa un ambiente para correr un Jupyter Notebook en un Docker Container con Anaconda. Se presentan a continuación los resultados obtenidos al procesar los datos según los pasos descriptos en la Sección 8.3.

Tras importar los datos, se le muestra al usuario los valores de los primeros 10 registros, a modo de familiarizarlo con el formato de los datos. En la Tabla 2 se muestran los resultados obtenidos para nuestro caso de uso.

En el Anexo 12.3 pueden consultase lo códigos de todo lo implementado en Jupyter Notebook.

9.1. Análisis de Componentes Principales (PCA)

Una vez importados los datos, se realiza el análisis PCA y se presentan la varianza que explica cada componente principal, tanto de forma numérica como gráfica. Puede observarse en la Tabla 3 que la primera componente principal explica más del 96% de la varianza. Si además sumamos la segunda componente se explica el 98,7% de la varianza, siendo el resto ya mejoras marginales. En la Figura 5 se presentan estos mismos datos de forma gráfica, siendo claro el quiebre y achatamiento de la curva después de la primera componente principal.

Id	dance-ability	energy	key	loud-ness	mode	speechi-ness	acoustic-ness	instrumen-talness	liven-ess	valence	tempo
3BiiSLo0wyHAY7lpCEO6ge	0.501	0.684	11	-6.706	1	0.0566	0.0049	0.000478	0.0524	0.442	131.95
2Ldb2ytuC9AxhpAkVEJBl6	0.822	0.3	0	-8.034	1	0.0673	0.882	9.15E-05	0.0665	0.734	125.95
0rt63HYAAlzUZo5O2D0uA6	0.471	0.396	8	-6.654	1	0.0336	0.77	0	0.0729	0.263	149.61
1i4F583d9v9RabcfOlKKVL	0.485	0.54	1	-5.4	1	0.0253	0.668	0	0.0849	0.426	62.631
0VuDtcT7H4CAljLZEH021x	0.686	0.765	8	-4.778	1	0.118	0.0122	0	0.127	0.743	99.998
2SUxn2O9NHL6GHGQFgwCY0	0.763	0.661	4	-5.592	0	0.056	0.169	0	0.117	0.632	95.002
5sv0WnUs74Orn6GoPmC5im	0.672	0.73	5	-4.77	1	0.0372	0.113	0	0.0612	0.559	123.97
0YbI9lWxUBsCJLb6EBydkr	0.578	0.558	2	-5.438	1	0.0872	0.236	0	0.143	0.54	75.087
3NuLxTXowTRcguTh9RsPmM	0.567	0.211	0	-9.975	1	0.0315	0.952	5.24E-05	0.275	0.289	134.02
3rmo8F54jFF8OgYsqTxm5d	0.807	0.893	11	-3.745	0	0.0347	0.0451	2.79E-05	0.366	0.537	126.01

Tabla 2: Extracto de las características musicales de las 10 primeras canciones del dataset estudiado

Componente Principal	Varianza Explicada
PC1	0.964574
PC2	0.022817
PC3	0.012267
PC4	0.000155
PC5	0.000079
PC6	0.000045
PC7	0.000035
PC8	0.000015
PC9	0.000006
PC10	0.000005
PC11	0.000002

Tabla 3: Varianza explicada por cada componente principal

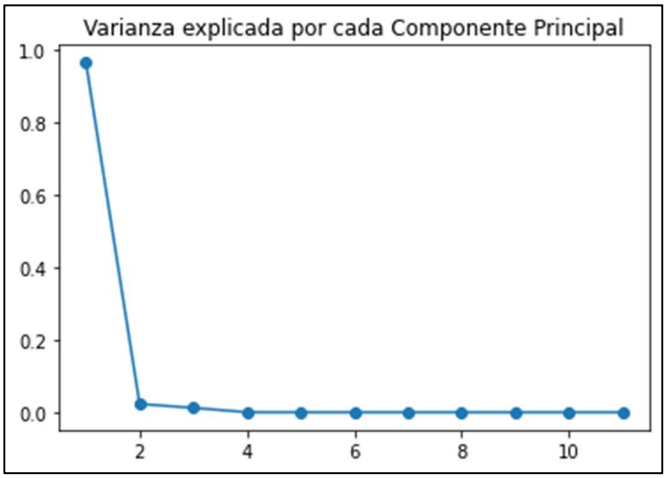


Figura 5: Varianza explicada por cada componente principal

Seguidamente, se le presenta al usuario un gráfico biplot para las dos primeras componentes principales. En la Figura 6 se puede ver como los puntos se distribuyen en un rango mayor en al eje horizontal que representa la primera componente principal, esto es consistente con que es la que mayor varianza explica.

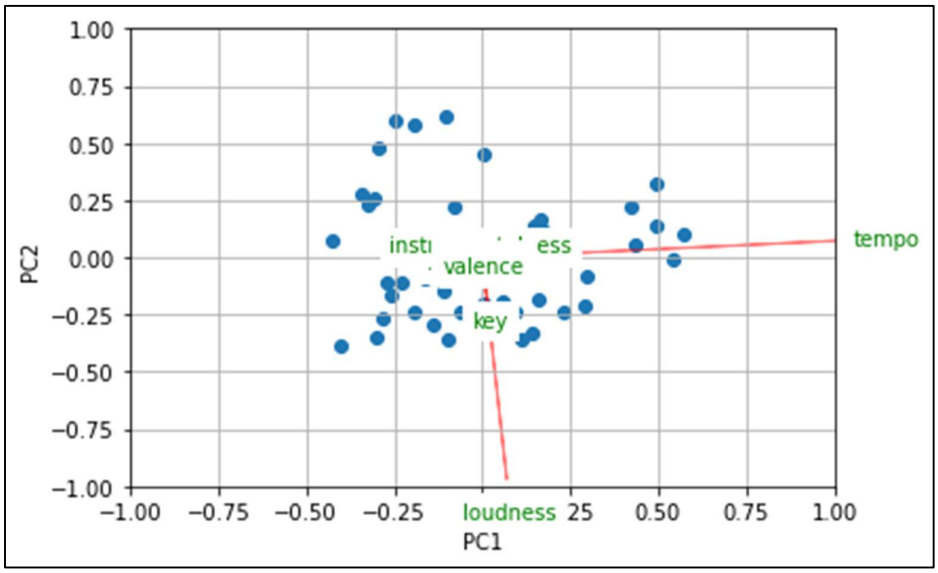


Figura 6: Gráfico biplot para las dos primeras componentes principales

Es interesante analizar los vectores de las cargas de las distintas componentes principales. Vemos como las variables “tempo” y “loudness” son las de mayor módulo, y son prácticamente ortogonales. La variable “tempo” es la mayor carga en la componente principal 1, y la variable “loudness” en la componente principal 2. Esta información se presenta en el Jupyter Notebook tanto de forma gráfica como analítica.

9.2. Agrupamiento por K-Means

Al realizar el agrupamiento por K-Means se verifican distintos valores de clusters, a fin de encontrar el que dé el mejor agrupamiento. Se prueba como números de clusters todos los enteros entre 2 y 12. El coeficiente de silhouette es evaluado para comparar los resultados. Con los datos utilizados la mejor configuración se da con 3 clusters, siendo el coeficiente de silhouette 0.58. En la Tabla 4 se muestran los centroides para este modelo.

Cluster	dance-ability	energy	key	loud-ness	mode	speechi-ness	acous-ticness	instrumen-talness	liven-ess	valence	tempo
1	0.613	0.583	5.947	-8.030	0.737	0.046	0.295	0.082	0.126	0.530	129.4
2	0.482	0.661	4.833	-8.558	0.667	0.079	0.147	0.181	0.176	0.467	174.0
3	0.493	0.408	3.880	-12.49	0.800	0.061	0.485	0.239	0.162	0.430	85.30

Tabla 4: Centroides para el modelo de agrupamiento K-Means con 3 clusters

Utilizando el análisis PCA de la Sección 9.1.1, se grafican los clusters utilizando las variables “tempo” y “loudness” que son las dos cargas con mayor peso en las dos primeras componentes principales. El resultado se puede ver en la Figura 7. Como puede apreciarse en el gráfico, los tres grupos se diferencian bien sin haber registros entremezclados. El algoritmo detecta 3 grupos bien distintos en función de estas variables.

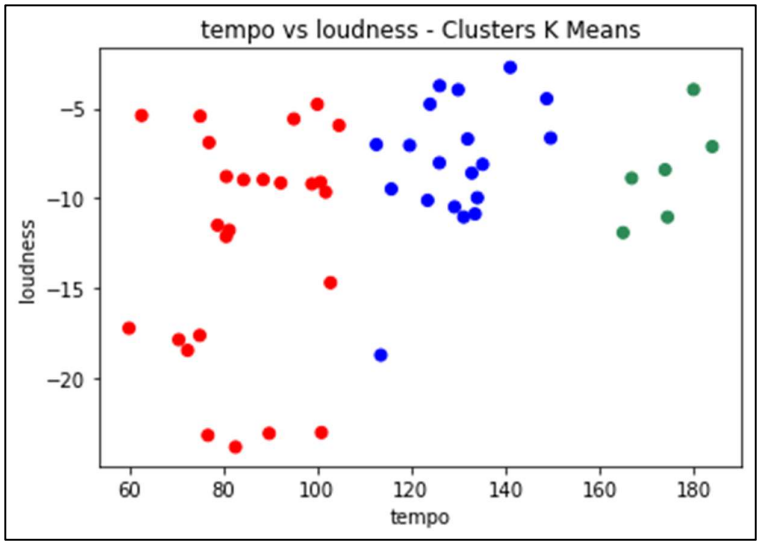


Figura 7: Gráfico del agrupamiento K-Means con 3 clusters, utilizando las variables “tempo” y “loudness”

Por otra parte, se provee al usuario la posibilidad de realizar este mismo gráfico para otras variables. En nutro caso de uso, el usuario eligió graficar las variables “acousticness” e “instrumentalness”. En la Figura 8 se muestra el resultado para las mismas. A diferencia del caso anterior, aquí puede verse como los puntos de los 3 clusters se entremezclan y no hay una clara clasificación según estas variables.

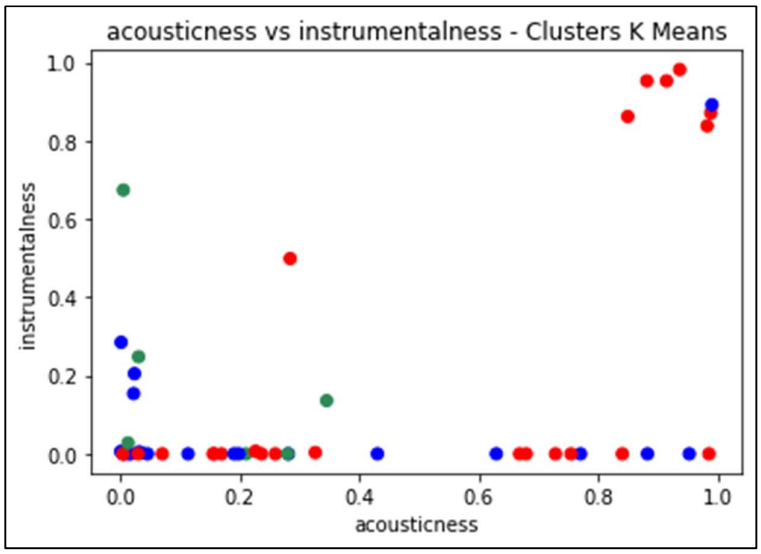


Figura 8: Gráfico del agrupamiento K-Means con 3 clusters, utilizando las variables “acousticness” e “instrumentalness”

9.3. Mapa Autoorganizado (SOM)

Para generar el modelo SOM de un mapa autoorganizado, se investigan distintos valores de dimensionalidad de la matriz (m x 1). En este caso utilizamos todos los enteros entre 2 y 12. Los resultados nuevamente se comparan con el coeficiente de silhouette. Para los datos utilizados la mejor configuración se da con una dimensionalidad 2, siendo el coeficiente de silhouette 0.57. Si bien el valor de este coeficiente es ligeramente mayor en K-Means, están en el mismo orden de magnitud. En la Tabla 5 se muestran los centroides (o pesos sinápticos de la red neuronal) para este modelo.

Cluster	dance-ability	energy	key	loud-ness	mode	speechi-ness	acous-ticness	instrumen-talness	liven-ess	valence	tempo
1	0.5779	0.5505	6.288	-8.769	0.6604	0.0517	0.3633	0.1123	0.1478	0.4910	122.6
2	0.5152	0.5309	5.413	-10.51	0.6405	0.0670	0.4238	0.2393	0.1909	0.4888	0.9925

Tabla 5: Centroides para el modelo de agrupamiento con SOM

Análogamente a lo realizado para K-Means, con el modelo SOM también utilizamos el resultado del análisis PCA de la Sección 9.1.1 para graficar los clusters utilizando las variables “tempo” y “loudness”, las mayores cargas en las dos primeras componentes principales. Se pyede ver el resultado en la Figura 9. En este caso hay solo dos clusters, pero vemos como la diferenciación entre los dos grupos es bien clara, sin zonas entremezcladas. Al compararlo con los grupos del K-Means vemos que uno de los grupos generados con SOM es igual a uno de los que es generado con K-Means. El restante grupo de SOM es la unión de los otros dos grupos que resultaron del análisis con K-Means.

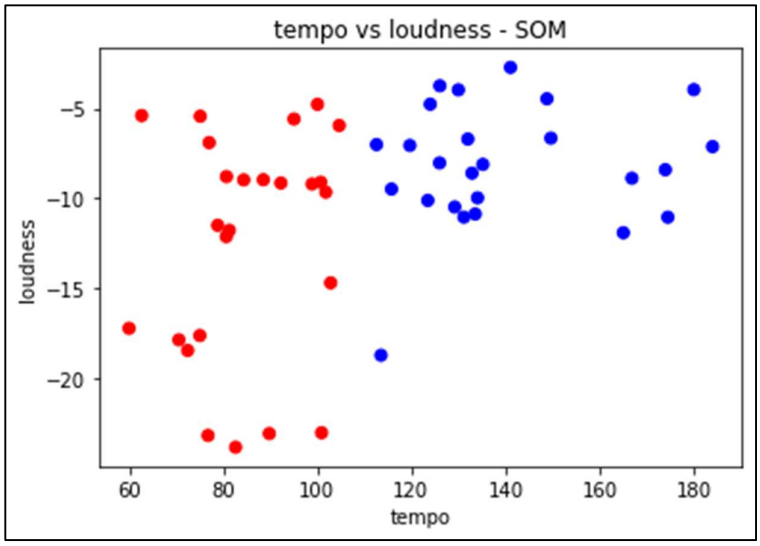


Figura 9: Gráfico del agrupamiento utilizando SOM, utilizando las variables “tempo” y “loudness”

Nuevamente se provee al usuario la posibilidad de realizar este mismo gráfico para otras variables. En nuestro caso, el usuario eligió graficar las variables “acousticness” e “instrumentalness”. En la Figura 10 se muestra el resultado para las mismas. Puede verse como los puntos de los 2 clusters se entremezclan y no hay una clara clasificación según estas variables, situación similar a la que observamos con K-Means, pero en este caso con 2 grupos.

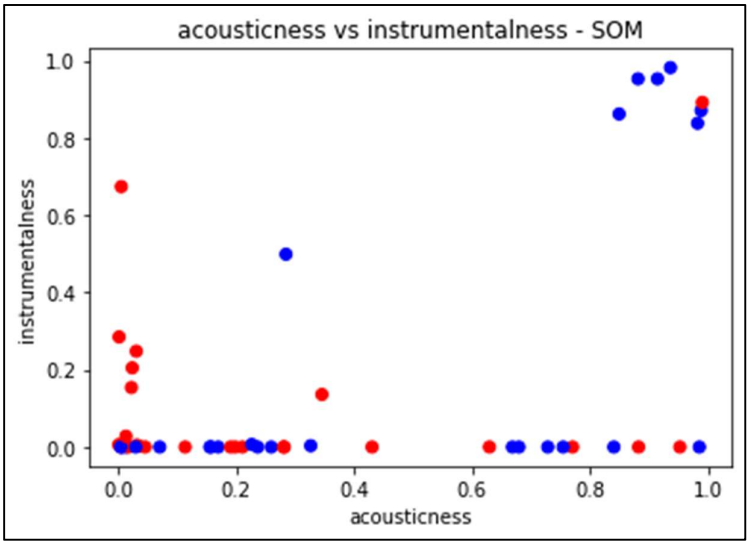


Figura 10: Gráfico del agrupamiento utilizando SOM,
utilizando las variables “acousticness” e “instrumentalness”

9.4. Comparación entre K-Means y Mapa Autoorganizado (SOM)

Vemos que la principal diferencia entre los agrupamientos que se generaron con K-Means y con SOM es la cantidad de clusters. En el primer caso se crean 3, y en el segundo 2. Los coeficientes silhouette son muy cercanos, siendo ligeramente superior el del K-Means por lo cual el modelo propone al usuario la clasificación en 3 grupos que fue desarrollada en el inciso 9.1.2. A su vez se comprueba como en ambos casos las variables “tempo” y “loudness” son las que mayormente diferencian a los agrupamientos.

10. Conclusiones y Trabajo a Futuro

La música constituye una de las formas de expresión artística más populares a lo largo de la historia y distintas culturas. Su estudio ha sido desarrollado y formalizado en la Teoría Musical, que permite entender los elementos que componen y diferencian distintas obras. Con la llegada de los servicios de streaming de música, el volumen de datos disponibles sobre hábitos de escucha y las características de las canciones creció de manera significativa. Estas plataformas proveen sistemas de recomendación, así como estadísticas sobre los artistas y canciones más escuchadas. Sin embargo, no se proveen detalles acerca de las características de la música que se recomienda o que el usuario escuchó, siendo esta una cuestión de interés para un usuario con conocimientos musicales. Dentro de los distintos servicios de streaming de música, Spotify es uno los más utilizados y conocidos, contando con un gran volumen de usuarios y canciones disponibles. La contribución del presente trabajo fue contribuir a completar la falta de información para un usuario con conocimiento musical acerca de las características y patrones de la música que escucha. El estudio se acotó a la plataforma de Spotify, utilizando las últimas 50 canciones datos de un solo usuario de Argentina, del rango etario de 30 a 35.

El trabajo cumplió con el objetivo general, de desarrollar una herramienta que permita a un usuario de Spotify conocer mejor los patrones de música que escucha. Mediante el proceso de ETL, sumado al procesamiento y presentación de los datos se

provee una interfaz en un Jupyter Notebook que permite comprender como se agrupan las canciones y que variables son las que mayormente contribuyen a estos agrupamientos. Los cuatro objetivos específicos planteados también se cumplieron, ya que se construyó un ETL que obtiene las características de las canciones que escuchó un usuario recientemente, se las agrupa, se presentan las variables que principalmente contribuyen a generar dichos clusters, así como comparar los distintos métodos de aprendizaje automático evaluando los que presentan mejores resultados.

El presente trabajo provee una primera aproximación para poder presentar a un usuario con conocimiento musical con mayor información acerca de los patrones de la música que escucha y sus características. Como mejora futura a este trabajo, resultaría de interés extender el tamaño del dataset, dado que actualmente la Spotify Web API presenta un límite de 50 canciones.

Otra posibilidad a futuro sería la de estudiar agrupamientos para otros conjuntos de datos, como las canciones que un usuario agrega a sus playlists, o las canciones más escuchadas en un país.

Finalmente, como trabajo a futuro se podría también desarrollar una aplicación para el usuario final, proveyendo a una mayor usabilidad y escalabilidad del servicio. Cabe remarcar también que la utilización de tecnologías modernas como Docker y Anaconda permiten mejorar la portabilidad y reutilización del código.

11. Referencias

Copland, A. (1939). *Como escuchar la música.* México: Fondo de Cultura Económica.

Hastie, T., Tibshirani, R. & Friedman, J. (2009). *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* Recuperado de https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12_toc.pdf

Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE*, 78, pp. 1464–1480.

Spotify (2021a). *What is Spotify*. Recuperado de: https://support.spotify.com/us/article/what-is-spotify/

Spotify (2021b). *Company Info*. Recuperado de https://newsroom.spotify.com/company-info/

Spotify (2021c). *Web API Reference*. Recuperado de: https://developer.spotify.com/documentation/web-api/reference

Spotify (2021d). *Web API Reference. Get Recently Played*. Recuperado de: https://developer.spotify.com/documentation/web-api/reference/#/operations/get-recently-played

Spotify (2021e). *Web API Reference. Get Several Audio Features*. Recuperado de: https://developer.spotify.com/documentation/web-api/reference/#/operations/get-several-audio-features

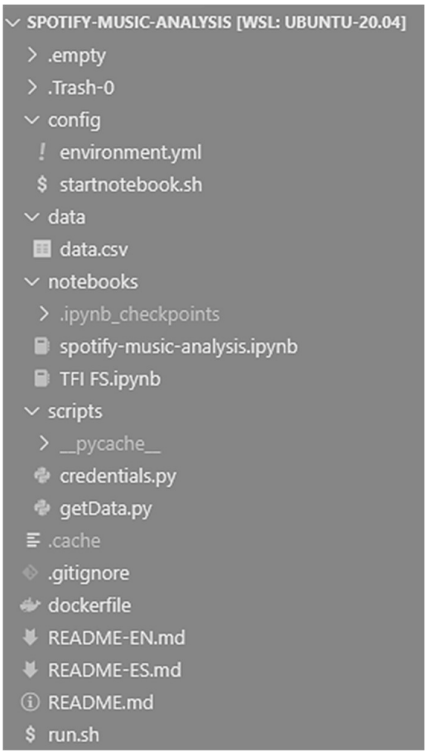
Spotify (2021f). *Authorization.* Recuperado de: https://developer.spotify.com/documentation/general/guides/authorization/

12. Anexos

12.1. Estructura de Carpetas del Repositorio

El repositorio con el código utilizado se encuentra disponible en el siguiente de GitHub: https://github.com/fseguior/spotify-music-analysis-in-python-docker

La estructura de las carpetas del repositorio responde a la siguiente distribución:



Se destacan los siguientes directorios:

- **config:** contiene los archivos de configuración del ambiente de Jupyter Notebooks corriendo en Docker con Anaconda
- **data:** contiene el CSV con el dataset generado por el ETL
- **notebooks:** contiene los notebooks donde se realizan los análisis
- **scripts:** contiene los scripts de Python con las credenciales para autenticarse con la Spotify Web API, y traer los datos generando el CSV en la carpeta “data”

12.2. Configuración y activación del ambiente

En este apartado se presentan los diferentes scripts y archivos de configuración utilizados para configurar y activar el ambiente de Docker con Anaconda, donde corre el Jupyter Notebook

12.2.1. Script de inicialización del ambiente “run.sh” Script “getData.py”

El ambiente se corrió utilizando WSL2 (Windows Subsystem for Linux 2), y la distribución Ubuntu 20.04 LTS. El script run.sh es el que orquestra los archivos de configuración. Crea el container, dentro del cual llama al script “startnotebook.sh” que obtiene los datos de Spotify y abre un notebook corriendo en Docker.

```
docker build -t spotify .
docker run -i -t -p 8888:8888 --mount type=bind,source="$(pwd)",target=/app spotify /bin/bash
-c "bash -i ./config/startnotebook.sh"
```

12.2.2. Script de inicialización del notebook “startnotebook.sh”

```
python ./scripts/getData.py
jupyter notebook --ip='*' --port=8888 --no-browser --allow-root
```

12.2.3. Archivo dockerfile para la creación del container donde corre el notebook

```
FROM continuumio/miniconda3
WORKDIR /app

# Create the environment:
COPY config/environment.yml .
RUN conda env create -f environment.yml
```

```
# Activate environment
RUN echo "conda activate spotify && /opt/conda/bin/conda install jupyter -y --quiet" >>
~/.bashrc
```

12.2.4. Archivo YAML con la configuración del ambiente de Anaconda

```
name: spotify
channels:
  - defaults
dependencies:
  - python
  - pip
  - numpy
  - pandas
  - matplotlib
  - seaborn
  - scikit-learn
  - jupyter
  - psutil
  - pip:
    - sklearn-som
    - spotipy
prefix: /anaconda3/envs/spotify
```

12.2.5. Script “credentials.py”

El siguiente script es utilizado para guardar localmente los tokens de autorización

```
client_ID=''      #Fill in with the value from the Spotify App
client_SECRET=''  #Fill in with the value from the Spotify App
redirect_url='http://localhost:9000'
```

12.2.6. Script “getData.py”

El siguiente script obtiene los datos de la Spotify Web API y los guarda en un archivo CSV, que será el dataset que leerá el Jupyter Notebook.

```
## Import Libraries
import pandas as pd
import spotipy as sp
from spotipy.oauth2 import SpotifyOAuth

#Authentication with Spotify
import credentials as cred

scope = 'user-read-recently-played user-library-read'

sp_auth = sp.Spotify(auth_manager=sp.oauth2.SpotifyOAuth(client_id=cred.client_ID,
client_secret= cred.client_SECRET, redirect_uri=cred.redirect_url, scope=scope))
##API call, recently played tracks

results = sp_auth.current_user_recently_played()['items']
tracks=list()
for item in results:
   tracks.append(item['track']['id'])

a_feats=sp_auth.audio_features(tracks)

featsDF=pd.DataFrame(a_feats)

featsDF=featsDF[["id","danceability","energy","key","loudness","mode", "speechiness",
"acousticness","instrumentalness", "liveness", "valence", "tempo"]]

featsDF

featsCSV=featsDF.to_csv(path_or_buf="/app/data/data.csv", index=False)

print(featsCSV)
```

12.2.7. Estructura JSON del output del método “Get Recently Played Tracks”

Se presenta a continuación la estructura JSON del output del método “Get Recently Played Tracks”. Este fue generado en base a documentación disponible en Spotify (2021d). La variable con la que nos quedamos en el script “getData.py” es “id”, que se encuentra resaltada en la estructura para su más fácil identificación.


```
{
  "href": "string",
  "items": [
                {
                "album": {
                "artists": []
                "available_markets": []
                    "disc_number": 0,
                "duration_ms": 0,
                "explicit": true,
                "external_ids": {}
                "external_urls": {}
                "href": "string",
                "id": "string",
                "is_playable": true,
                "linked_from": {}
                  "restrictions": {}
                  "reason": "string"
             },
                "name": "string",
                "popularity": 0,
                "preview_url": "string",
                "track_number": 0,
                "type": "string",
                "uri": "string",
                "is_local": true
                }
            ],
  "limit": 0,
  "next": "string",
  "cursors": {
```

```
  "after": "string"
  },
  "total": 0
}
```


12.3. Jupyter Notebook con los resultados experimentales

A continuación se presenta el Jupyter Notebook “TFI FS.ipynb” utilizado en el apartado 9 con los resultados experimentales

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import spotipy as sp

featsDF=pd.read_csv("/app/data/data.csv")
print(featsDF.head(10))
```

	id	danceability	energy	key	loudness	mode \
0	3BiiSLo0wyHAY7lpCEO6ge	0.501	0.684	11	-6.706	1
1	2Ldb2ytuC9AxhpAkVEJBI6	0.822	0.3	0	-8.034	1
2	0rt63HYAAIzUZo5O2D0uA6	0.471	0.396	8	-6.654	1
3	1i4F583d9v9RabcfOlKKVL	0.485	0.54	1	-5.4	1
4	0VuDtcT7H4CAljLZEH021x	0.686	0.765	8	-4.778	1
5	2SUxn2O9NHL6GHGQFgwCY0	0.763	0.661	4	-5.592	0
6	5sv0WnUs74Orn6GoPmC5im	0.672	0.73	5	-4.77	1
7	0YbI9lWxUBsCJLb6EBydkr	0.578	0.558	2	-5.438	1
8	3NuLxTXowTRcguTh9RsPmM	0.567	0.211	0	-9.975	1
9	3rmo8F54jFF8OgYsqTxm5d	0.807	0.893	11	-3.745	0

	speechiness	acousticness	instrumentalness	liveness	valence	tempo
0	0.0566	0.00494	0.000478	0.0524	0.442	131.952
1	0.0673	0.882	9.15E-05	0.0665	0.734	125.951
2	0.0336	0.77	0	0.0729	0.263	149.609
3	0.0253	0.668	0	0.0849	0.426	62.631

4	0.118	0.0122	0	0.127	0.743	99.998
5	0.056	0.169	0	0.117	0.632	95.002
6	0.0372	0.113	0	0.0612	0.559	123.969
7	0.0872	0.236	0	0.143	0.54	75.087
8	0.0315	0.952	5.24E-05	0.275	0.289	134.019
9	0.0347	0.0451	2.79E-05	0.366	0.537	126.011

In [2]:

```
#Dataframe para procesar
featsDFData=featsDF.loc[ : ,featsDF.columns != 'id']

#Array para PCA y K Mean featsArray=np.array(featsDFData)

#Feature Labels
featsLabel=["danceability","energy","key","loudness","mode", "speechiness", \
"acousticness", "instrumentalness", "liveness", "valence", "tempo"]
```

- **PCA - Principal Components Analysis**

En primer lugar, se normaliza la muestra y genera el modelo. Luego, se muestra la varianza explicada al ir agregando componentes principales. Finalmente, se presenta un gráfico biplot con cuales son las variables de entrada con mayor incidencia sobre las dos primeras componentes principales. Estas también se presentan con el nombre de la variable.

In [3]:

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

comp=(1,2,3,4,5,6,7,8,9,10,11)

scaler = StandardScaler() scaler.fit(featsArray)
X=scaler.transform(featsArray)     pca =
PCA(n_components=11,
random_state=1) x_new =
pca.fit_transform(featsArray)

expl_variance=(pca.explained_variance_ratio_) print(expl_variance)

plt.scatter(x=comp, y=expl_variance ) plt.plot(comp,
expl_variance )
plt.title("Varianza explicada por cada Componente Principal") plt.show()
```

```
[9.64573958e-01 2.28174365e-02 1.22667964e-02 1.54954643e-04
 7.93494770e-05 4.47143822e-05 3.51090090e-05 1.48713133e-05
 5.73665065e-06 5.07263170e-06 2.00122623e-06]
```

```
## Reference: https://stackoverflow.com/questions/39216897/plot-pca-loadings-and-loading-in-biplot-in-sklearn-like-rs-autoplot

def myplot(score,coeff,labels=None):
    xs = score[:,0]
    ys = score[:,1]
    n = coeff.shape[0]
    scalex = 1.0/(xs.max() - xs.min())
    scaley = 1.0/(ys.max() - ys.min())
    plt.scatter(xs * scalex,ys * scaley)
    for i in range(n):
        plt.arrow(0, 0, coeff[i,0], coeff[i,1],color = 'r',alpha = 0.5)
        if labels is None:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, featsLabel[i],\
```

```
                    color = 'g', ha = 'center', va = 'center', \
                    backgroundcolor = 'white')
        else:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, labels[i], \
                    color = 'g', ha = 'center', va = 'center', \
                    backgroundcolor = 'white')
    plt.xlim(-1,1)
    plt.ylim(-1,1)
    plt.xlabel("PC{}".format(1))
    plt.ylabel("PC{}".format(2))
    plt.grid()
    print(range(n))

myplot(x_new[:,0:2],np.transpose(pca.components_[0:2, :]))
plt.show()
```

In [4]:

range(0, 11)

In [5]:

```
#PC1
comp_PC1=list(np.absolute(pca.components_[0]))

best_comp_PC1=max(comp_PC1)
index_PC1=comp_PC1.index(best_comp_PC1)
main_comp_PC1=featsLabel[index_PC1]

print("La variable con mayor peso en la primera \ componente principal
es '{}'".format(main_comp_PC1))
```

La variable con mayor peso en la primera componente principal es 'tempo'

In [6]:

```
#PC2
comp_PC2=list(np.absolute(pca.components_[1]))

best_comp_PC2=max(comp_PC2)
index_PC2=comp_PC2.index(best_comp_PC2)
main_comp_PC2=featsLabel[index_PC2]

print("La variable con mayor peso en la segunda \ componente
principal es '{}'".format(main_comp_PC2))
```

La variable con mayor peso en la segunda componente principal es 'loudness'

- **K Means Clustering**

In [7]:

```
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score
from matplotlib.colors import ListedColormap

## Genero una lista con lo K-num que voy a probar
num_of_cluster=[]
for x in range(11):
    num_of_cluster.append(x+2)
print(num_of_cluster)

## Itero para los distintos valores de K num
result_silhouette_kmeans=[]

for knum in num_of_cluster:
    kmeans=KMeans(n_clusters=knum, random_state=1)
    y=kmeans.fit_predict(featsDFData)

    score_kmeans = silhouette_score(featsDFData, y)
```

```
    result_silhouette_kmeans.append(score_kmeans)


## Buscamos el mejor valor de número de clusters
result_silhouette_kmeans
best_silhouette_kmeans=max(result_silhouette_kmeans)
index_kmeans=result_silhouette_kmeans.index(best_silhouette_kmeans)

best_knum=num_of_cluster[index_kmeans]

print("El mejor número de clusters es = {}, \
siendo el valor de silhouette {})".format(best_knum, best_silhouette_kmeans))
```

[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

El mejor número de clusters es = 3, siendo el valor de silhouette 0.5812236273769709)

In [8]:

```
kmeans=KMeans(n_clusters=best_knum) model=kmeans.fit(featsDFData)
featsDF["clusterLabel"] = model.labels_

kmeans_centroid=model.cluster_centers_

print("Centroides de los clusters")
print(kmeans_centroid)
```

```
Centroides de los clusters
[[ 4.92688000e-01  4.07718800e-01  3.88000000e+00 -1.24931200e+01
   8.00000000e-01  6.10920000e-02  4.85456000e-01  2.38731796e-01
1.62268000e-01  4.29636000e-01  8.53042800e+01]
 [ 6.12842105e-01  5.82894737e-01  5.94736842e+00 -8.03005263e+00
   7.36842105e-01  4.61473684e-02  2.95437000e-01  8.15898958e-02
1.26078947e-01  5.29789474e-01  1.29362684e+02]
 [ 4.81500000e-01  6.61333333e-01  4.83333333e+00 -8.55783333e+00
   6.66666667e-01  7.90333333e-02  1.47086667e-01  1.80983575e-01
   1.75683333e-01  4.66833333e-01  1.74031500e+02]]
```

Se presentan los agrupamientos utilizando como dimensiones las dos variables con mayor injerencia en las primeras dos componentes principales. Se observa como las observaciones están en dos zonas bien distintas

In [9]:

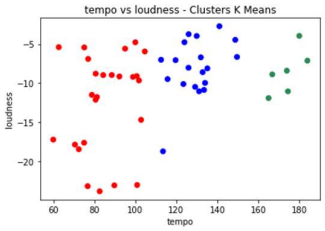
```
colors=['blue', 'yellow', 'cyan', 'seagreen','peru', 'magenta', 'red']

plt.scatter(featsDFData[main_comp_PC1], featsDFData[main_comp_PC2], \
        c=featsDF["clusterLabel"],  cmap=ListedColormap(colors))

plt.title("{}  vs  {}  -  Clusters  K  Means".format(main_comp_PC1,  main_comp_PC2))
plt.xlabel(main_comp_PC1)
plt.ylabel(main_comp_PC2)
```

```
Text(0.5, 1.0, 'tempo vs loudness - Clusters K Means')
```

Out[9]:



El usuario puede también analizar los agrupamientos con otras variables. Se observa como las observaciones están más mezcladas que en el caso anterior

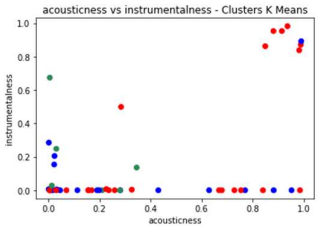
In [10]:

```
#Modificar "var1" y "var2" para comparar los resultados con distintos audio features
var1="acousticness"
var2="instrumentalness"

plt.scatter(featsDFData[var1], featsDFData[var2], \
            c=featsDF["clusterLabel"], cmap=ListedColormap(colors))
plt.title("{} vs {} - Clusters K Means".format(var1, var2))
plt.xlabel(var1)
plt.ylabel(var2)
```

Text(0.5, 1.0, 'acousticness vs instrumentalness - Clusters K Means')

Out[10]:



- **Self Organizing Maps**

In [11]:

```
from sklearn_som.som import SOM
from matplotlib.colors import ListedColormap

## Genero una lista con lo matrix_dim que voy a
probar matrix_dim=[] for x in range(11):
matrix_dim.append(x+2) print(matrix_dim)

## Itero para los distintos valores de matrix_dim result_silhouette_som=[]
 for mtx in
matrix_dim:

    spotify_som = SOM(m=mtx, n=1, dim=11, random_state=1)
    SOMClassification=spotify_som.fit(featsArray)

    y=spotify_som.predict(featsArray)

    score_som = silhouette_score(featsArray, y)
result_silhouette_som.append(score_som)

## Buscamos el mejor valor
result_silhouette_som
best_silhouette_som=max(result_silhouette_som)

index_som=result_silhouette_som.index(best_silhouette_som)
best_som=num_of_cluster[index_som]

print("El mejor matrix dim es = {}, siendo el \
valor de silhouette {})".format(best_som, best_silhouette_som))
```

[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

El mejor matrix dim es = 2, siendo el valor de silhouette 0.5687090095607065)

In [12]:

```
print(best_som)
spotify_som = SOM(m=best_som, n=1, dim=11, random_state=1)
SOMClassification=spotify_som.fit(featsArray)
SOMClass=spotify_som.predict(featsArray)
```

In [13]:

```
som_centroid=spotify_som.cluster_centers_

print("Centorides")
print(som_centroid)
```

```
Centorides
[[[ 5.77862747e-01  5.50478067e-01  6.28762979e+00 -8.76949678e+00
   6.60427541e-01  5.16802861e-02  3.63290050e-01  1.12261980e-01
1.47745125e-01  4.91040411e-01  1.22636046e+02]]

 [[ 5.15159066e-01  5.30868030e-01  5.41322538e+00 -1.05108393e+01
   6.40580209e-01  6.70985229e-02  4.23887513e-01  2.39273321e-01
   1.90910196e-01  4.88778140e-01  9.92479968e+01]]]
```

Se presentan los agrupamientos utilizando como dimensiones las dos variables con mayor injerencia en las primeras dos componentes principales. Se observa como las observaciones están en dos zonas bien distintas. El resultado con SOM es similar al de K-Means

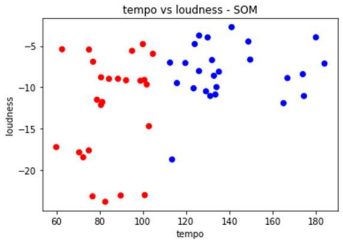
In [14]:

```
plt.scatter(featsArray[0:,index_PC1],featsArray[0:,index_PC2],\
c=SOMClass, cmap=ListedColormap(colors))

plt.title("{} vs {} - SOM".format(main_comp_PC1, main_comp_PC2))
plt.xlabel(main_comp_PC1)
plt.ylabel(main_comp_PC2)
```

Text(0.5, 1.0, 'tempo vs loudness - SOM')

Out[14]:



El usuario puede también analizar los agrupamientos con otras variables. Se observa como las observaciones están más mezcladas que en el caso anterior. El resultado con SOM es similar al de K-Means.

In [15]:

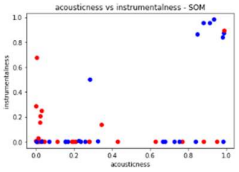
```
#Modificar "var1" y "var2" para comparar los resultados con distintos audio features
var1="acousticness"
var2="instrumentalness"

plt.scatter(featsArray[0:,featsLabel.index(var1)], \
featsArray[0:,featsLabel.index(var2)], \
 c=-SOMClass, cmap=ListedColormap(colors))

plt.title("{} vs {} - SOM".format(var1, var2))
plt.xlabel(var1)
plt.ylabel(var2)
```

Text(0.5, 1.0, 'acousticness vs instrumentalness - SOM')

Out[15]:



- **Comparación K-Means vs SOM**

In [16]:

```
if (best_silhouette_kmeans > best_silhouette_som):
    print("El mejor algoritmo es K-Means, con un número de cluster = {}, \     y un valor de
silhouette {}".format(best_knum, best_silhouette_kmeans))

elif (best_silhouette_kmeans < best_silhouette_som):
    print("El mejor algoritmo es SOM, con un matrix dim = {}, \
    y un valor de silhouette {}".format(best_som, best_silhouette_som))
 else:
    print("K-Means y SOM generan agrupamientos con igual coeficiente \
silhouette de {}. El número de clusters en K-Means es {}, \
    y el matrix dim en SOM es {}".format(best_silhouette_kmeans, best_knum, best_som))
```

El mejor algoritmo para esta aplicación es K-Means, con un número de cluster = 3, y un valor de silhouette 0.