Aplicación de la metodología CRISP-DM en la clasificación automática de géneros musicales utilizando Python con Machine Learning

Emiliano L. Mendez
Instituto Politécnico Nacional Ciudad de México, México

Leonardo H. Loera
Instituto Politécnico Nacional Ciudad de Mexico, México

RESUMEN: Este proyecto tiene como objetivo desarrollar un sistema de clasificación automática de géneros musicales utilizando técnicas de Data Science y algoritmos de Machine Learning. Se sigue la metodología **CRISP-DM** para estructurar el proceso en fases, desde la comprensión del negocio y de los datos hasta el modelamiento y la evaluación. El sistema permitirá al usuario ingresar tres canciones favoritas y, en función de sus características, recomendará cinco canciones adicionales utilizando el algoritmo K-Nearest Neighbors (KNN).

El proyecto aplica herramientas avanzadas como **web scraping** para enriquecer el dataset musical y la librería **librosa** para extraer atributos del audio. Esta implementación tiene aplicaciones relevantes en sistemas de recomendación para plataformas de streaming, mejorando la experiencia del usuario a través de sugerencias personalizadas.

Palabras clave: CRISP-DM, Machine Learning, KNN, web scraping, Data Science, música, Python.

Application of the CRISP-DM methodology in the automatic classification of musical genres using Python with Machine Learning

ABSTRACT: This project aims to develop an automatic classification system for musical genres using Data Science techniques and Machine Learning algorithms. The CRISP-DM methodology is followed to structure the process in phases, from understanding the business and data to modeling and evaluation. The system will allow the user to enter three favorite songs and, based on their characteristics, will recommend five additional songs using the K-Nearest Neighbors (KNN) algorithm.

The project applies advanced tools such as web scraping to enrich the musical dataset and the book library to extract audio attributes. This implementation has relevant applications in recommendation systems for streaming platforms, improving the user experience through personalized suggestions.

Keywords: CRISP-DM, Machine Learning, KNN, web scraping, Data Science, music, Python.

INTRODUCCION

En el contexto de la creciente digitalización y consumo de contenido musical, los sistemas de recomendación han adquirido una importancia crucial para las plataformas de streaming. Este proyecto tiene como objetivo construir un sistema de clasificación automática de géneros musicales, capaz de aprender de los gustos del usuario y recomendarle canciones con características similares. La idea central es que el usuario introduzca tres canciones favoritas, y el sistema devuelva cinco recomendaciones relevantes en función de sus atributos.

La metodología CRISP-DM guía el desarrollo del proyecto, lo que permite estructurar cada fase, desde la comprensión de los datos hasta la evaluación del modelo. El proyecto se desarrollará utilizando **Python** e integrará técnicas de Machine Learning, con el algoritmo **K-Nearest Neighbors (KNN)** como base del modelo de recomendación. Además, utilizaremos **web scraping** para enriquecer la información disponible de las canciones, empleando herramientas como BeautifulSoup y requests.

Este proyecto no solo tiene aplicaciones prácticas en la industria del entretenimiento, sino que también representa una excelente oportunidad de aplicar conceptos fundamentales de ciencia de datos, como la extracción de características y la implementación de modelos supervisados.

MARCO TEORICO

El proyecto se apoya en varios conceptos clave del campo de la ciencia de datos y la inteligencia artificial:

1. **Machine Learning y KNN:** El algoritmo K-Nearest Neighbors (KNN) es una técnica de aprendizaje supervisado utilizada para clasificación. Se basa en la idea de que las instancias similares se encuentran cerca unas de otras en un espacio multidimensional, por lo que al identificar los "vecinos" más cercanos a una canción favorita, podemos recomendar otras canciones con atributos similares.
2. **Metodología CRISP-DM:** La metodología CRISP-DM (Cross Industry Standard Process for Data Mining) divide el proceso de minería de datos en fases bien definidas: comprensión del proyecto, comprensión de los datos, preparación de los datos, modelamiento, evaluación y publicación. Esta estructura flexible facilita un desarrollo ordenado y eficiente.
3. **Web Scraping:** El web scraping es la técnica de extracción de información de sitios web. En este proyecto, se utiliza para complementar el dataset con información adicional de canciones, como etiquetas y descripciones

METODOLOGIA

Metodología CRISP-DM

Para estructurar el desarrollo del proyecto, seguimos la metodología **CRISP-DM** (Cross Industry Standard Process for Data Mining), que proporciona un enfoque claro para cada fase del ciclo de vida del análisis de datos. Esta metodología es flexible, permitiendo adaptar los pasos a las necesidades específicas del proyecto.

Las etapas que abordamos son:

1. **Comprensión del Proyecto:**
   En esta fase definimos los objetivos y requerimientos del sistema de recomendación musical. El objetivo principal es ofrecer recomendaciones de cinco canciones en función de las tres canciones favoritas proporcionadas por el usuario.
2. **Comprensión de los Datos:**
   Identificamos datasets relevantes como el **GTZAN** y las APIs de **Spotify** para obtener atributos musicales como el tempo, la energía, la popularidad y el género. Además, realizamos web scraping desde plataformas como **Last.fm** utilizando herramientas como BeautifulSoup.
3. **Preparación de los Datos:**
   En esta etapa, limpiamos y preprocesamos los datos. Eliminamos valores duplicados o nulos, y normalizamos los valores numéricos para mejorar la precisión del modelo. También integramos la información obtenida por web scraping para enriquecer nuestro dataset.
4. **Modelamiento:**
   Seleccionamos el algoritmo **K-Nearest Neighbors (KNN)** para realizar las recomendaciones. Entrenamos el modelo utilizando los datos preprocesados, divididos en conjuntos de entrenamiento y prueba. Calculamos la similitud entre canciones en función de sus atributos.
5. **Evaluación:**
   Evaluamos el rendimiento del modelo utilizando métricas como **Precision@5** y **Mean Average Precision (MAP)**. Ajustamos los parámetros del modelo para mejorar la precisión en las recomendaciones.
6. **Publicación:**
   La entrega final incluye un notebook en Python con el código documentado y una presentación que muestra los resultados obtenidos, con ejemplos de recomendaciones y gráficos relevantes.

Ciclo de Vida del Proyecto

El ciclo de vida del proyecto se representa en la siguiente figura. Esta muestra cómo los datos se recolectan, limpian y procesan antes de ser utilizados en el entrenamiento del modelo y, finalmente, se visualizan los resultados a través de gráficos interactivos.

Descripción del Ciclo de Vida:

- **Recolección de Datos:** Extraemos la información musical de datasets públicos y mediante web scraping.
- **Limpieza de Datos:** Filtramos los datos para eliminar ruido e inconsistencias.
- **Entrenamiento:** Utilizamos los datos preprocesados para entrenar el modelo KNN.
- **Visualización:** Los resultados se presentan al usuario en un entorno interactivo para probar el sistema de recomendaciones.

Visualización del Proceso Completo

El sistema se diseñará para permitir que los usuarios prueben la recomendación de canciones en tiempo real, ingresando sus canciones favoritas y recibiendo sugerencias personalizadas. Este enfoque garantiza que la solución sea intuitiva y práctica, aplicando todos los conceptos trabajados a lo largo del desarrollo del proyecto.

COMPRENSION DEL PROYECTO

El objetivo es proporcionar recomendaciones musicales personalizadas basadas en las canciones favoritas del usuario. El sistema sugerirá cinco canciones que coincidan con las características de las tres canciones ingresadas por el usuario. Esta solución tiene aplicaciones directas en plataformas de streaming y sistemas de recomendación.

COMPRENSION DE LOS DATOS

Usaremos web scraping junto con la pagina Last.fm para poder sacar la siguiente información de las canciones:

- Género musical
- Popularidad de la canción

Además, realizaremos web scraping para extraer información complementaria desde plataformas como Last.fm y Spotify, empleando BeautifulSoup y requests.

PREPARACION DE LOS DATOS

Durante esta fase, realizaremos las siguientes tareas:

- **Limpieza:** Eliminaremos valores faltantes y duplicados.
- **Preprocesamiento:** Normalizaremos los datos para asegurar la efectividad del modelo.
- **Enriquecimiento:** Integraremos los datos obtenidos mediante web scraping.

MODELAMIENTO

Implementaremos un sistema de recomendación basado en el algoritmo KNN. Los pasos son:

1. Dividimos el dataset en entrenamiento y prueba.
2. Entrenamos el modelo KNN utilizando características como tempo, energía y popularidad.
3. El usuario ingresará sus tres canciones favoritas.
4. El sistema calculará la similitud entre estas canciones y el resto del dataset.
5. Se generarán cinco recomendaciones basadas en las canciones más similares.

EVALUACION

La evaluación del modelo se llevará a cabo utilizando métricas como:

- **Precision@5:** Porcentaje de canciones relevantes dentro de las cinco recomendaciones.
- **Mean Average Precision (MAP):** Promedio de las precisiones en diferentes conjuntos de prueba.

Compararemos también los resultados obtenidos al ajustar diferentes parámetros del modelo KNN, como el número de vecinos.

RESULTADOS Y DISCUSION

Por el momento, con uso de web scraping, la pagina Last.fm nos ayudo mucho y nos facilito mucho el uso y clasificación de los datos. Empecemos con las librearías a usar:

- Pandas
- Requests
    - Beautiful soup
- IPython.display
    - Display

```
import pandas as pd
import requests
from bs4 import BeautifulSoup
from IPython.display import display
```

La librería pandas nos ayudara para poder manejar el uso de los data frames, la librería requests junto con BeautifulSoup nos ayudara para el uso de webscraping y display para mejor ver los data frames.

```
url = "https://www.last.fm/tag/rock/tracks?page=8"
h = {"user-agent" : "Mozilla/5.0 (Windows NT 10.0; Win64;
res = requests.get(url, headers=h)
res.status_code

✓ 0.8s
```

-Se define una URL que apunta a la página 8 de las canciones etiquetadas como "rock" en Last.fm.

-Se configuran los encabezados de la solicitud HTTP para que parezca que proviene de un navegador.

-Se realiza una solicitud GET a la URL.

-Se verifica el estado de la respuesta (200 significa éxito).

- Hicimos cada eso para cada genero de música, así confirmamos que si nos podemos conectarz

```
url = "https://www.last.fm/tag/hip-hop/tracks?page=190"
h = {"user-agent" : "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Ap
```

Hip-hop

```
url = "https://www.last.fm/tag/indie/tracks?page=190"
```

Indie

```
url = "https://www.last.fm/tag/reggae/tracks?page=190"
h = {"user-agent" : "Mozilla/5.0 (Windows NT 10.0; Win64; x
```

Reggae

```
url = "https://www.last.fm/tag/dance/tracks?page=190"
h = {"user-agent" : "Mozilla/5.0 (Windows NT 10.0; Win64;
```

Dance

```
url = "https://www.last.fm/tag/rnb/tracks?page=190"
h = {"user-agent" : "Mozilla/5.0 (Windows NT 10.0; Wi
```

Rnb

```
url = "https://www.last.fm/tag/rap/tracks?page=190"
```

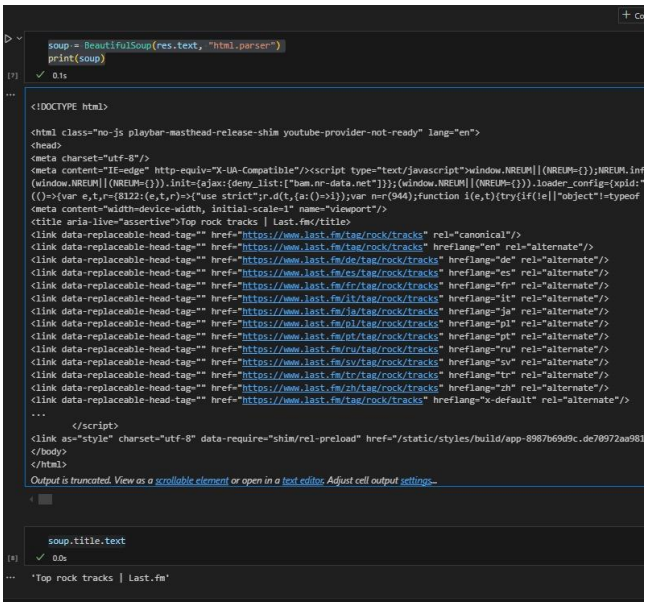
Rap

```
url = "https://www.last.fm/tag/alternative/tracks?page=190"
```

Alternativa

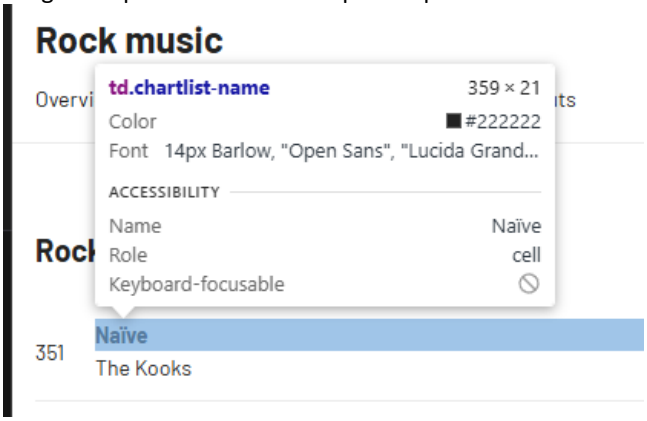
```
url = "https://www.last.fm/tag/electronic/tracks?page=190"
h = {"user-agent" : "Mozilla/5.0 (Windows NT 10.0; Win64; x
```

Electrónica

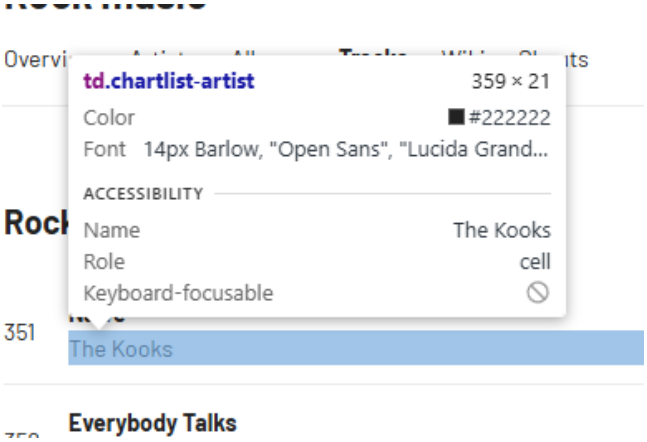


obtenemos la "sopa" de la pagina, y el titulo para configurar si estamos bien

Luego ocupamos ver las etiquetas para cada cancion



El nombre esta en chartlist-name



Mientras que el artista esta en chartlist-artist

De aquí podemos crear un ciclo for que nos extrae las canciones de la pagina junto con sus artistas

```
tracks = soup.find_all('tr', class_='chartlist-row')
for track in tracks:
        # Extraer el nombre de la canción
        song = track.find('td', class_='chartlist-name').find('a').text.strip()

        # Extraer el nombre del artista
        artist = track.find('td', class_='chartlist-artist').find('a').text.strip()

        # Agregar la canción y el artista a la lista
        rock.append({'Canción': song, 'Artista': artist})
```

Extrae, la canción y el nombre del artista, con chartlist-row, se obtiene todas las canciones de esa pagina web y termina hasta que ya no encuentre mas canciones. Hacemos eso para cada genero musical.

Para mas facilidad de manejar los datos, los manejaremos como dataframes, entonces por cada genero hacemos una lista del genero: por ejemplo rock `rock = []` y en el ciclo for, se añade a la lista con append usando pandas

```
# Agregar la canción y el artista a la lista
rock.append({'Canción': song, 'Artista': artist})
        rock.append({'Cancion': song, 'Artista': artist})

rock_df = pd.DataFrame(rock)
display(rock_df)
```

✓ 0.0s

	Canción	Artista
0	Naïve	The Kooks
1	Everybody Talks	Neon Trees
2	Faust Arp	Radiohead
3	Kids With Guns	Gorillaz
4	Piano Man	Billy Joel
5	Don't Stop Me Now - Remastered 2011	Queen
6	LUNCH	Billie Eilish
7	Hold the Line	Toto
8	On a Plain	Nirvana
9	The Final Countdown	Europe
10	Black	Pearl Jam
11	Happier Than Ever	Billie Eilish
12	This Charming Man - 2011 Remaster	The Smiths
13	1979	The Smashing Pumpkins
14	Down Under	Men at Work

Así podemos manejar mejor el uso de las canciones

```
rock_df['Género'] = 'Rock'
display(rock_df)
```

✓ 0.0s

	Canción	Artista	Género
0	Naïve	The Kooks	Rock
1	Everybody Talks	Neon Trees	Rock
2	Faust Arp	Radiohead	Rock
3	Kids With Guns	Gorillaz	Rock

en esa misma lista agremos su genero y listo

Se repitió para cada genero de música

```
dataframes = [rock_df,hip_hop_df,indie_df,reggae_df,dance_df,rnb_df,rap_df,alt_df,elc_df]
universal_df = pd.concat(dataframes, ignore_index=True)
display(universal_df)
```

	Canción	Artista	Género
0	Naïve	The Kooks	Rock
1	Everybody Talks	Neon Trees	Rock
2	Faust Arp	Radiohead	Rock
3	Kids With Guns	Gorillaz	Rock
4	Piano Man	Billy Joel	Rock
...	...	...	...
445	Monster	Starset	Electronic
446	Colours	The Avalanches	Electronic
447	ZVVL	CHVRCHES	Electronic
448	XOXOXO	Black Eyed Peas	Electronic
449	Fade	Alan Walker	Electronic

450 rows × 3 columns

Aquí juntamos todos los datasets para tener nuestro propio data set con que trabajar y facilitar el algoritmo KNN que usaramos para poder mejorar el uso del algoritmo

CONCLUSIONES

REFERENCIAS