# Facultad de Ciencias e Ingeniería


# Sistema de Recomendación

## Integrantes:

Josue Eduardo Huarauya Fabian

Vanesa Nelsy Morales Taipe

Nohereily Kimberly Salazar Berrios

## Docentes:

Nelson Enrique Castro Zarate

Josue Angel Mauricio Salazar

6 de diciembre de 2023

# 1. Introdución

La música es una parte integral de la vida cotidiana de muchas personas. Con la abundancia de canciones y artistas disponibles en la era digital, a menudo es abrumador para los usuarios descubrir nueva música que se alinee con sus gustos. Un sistema de recomendación eficaz puede mejorar significativamente esta experiencia.

Los sistemas de recomendación se originaron a partir de una observación bastante simple: los individuos a menudo se basan en recomendaciones proporcionadas por otros al tomar decisiones diarias y de rutina Resnick y Varian [1997], Resnick y cols. [1994]).

En la música esta herramienta tecnológica permite a los usuarios descubrir nuevas canciones o artistas basándose en sus preferencias y patrones de escucha anteriores. Estos sistemas utilizan diferentes técnicas y algoritmos para proporcionar recomendaciones precisas y personalizadas. La calidad de estas recomendaciones depende en gran medida de la calidad y cantidad de los datos utilizados para alimentar el sistema.

En este informe, describiremos el proceso de obtención, limpieza y tratamiento de un subconjunto de los datos del Million Song Dataset McFee y cols. [2012], y cómo estos datos se pueden utilizar para desarrollar un sistema de recomendación de música basado en filtrado colaborativo y técnicas de reducción de dimensionalidad. Además, abordaremos el desafío del "Long Tail" en la industria musical.

## 1.1. Objetivo General:

Proporcionar a los usuarios recomendaciones de música personalizadas y efectivas basándose en sus patrones de escucha.

# 2. Estado del Arte

Existen estudios realizados de sistemas de recomendación. Uno de ellos es el “Music Recommender System using Autorec Method for Implicit Feedback"(Irawan y Baizal [2023]), que ayuda a los usuarios a encontrar música que se adapte a sus gustos, utilizando el paradigma de filtrado colaborativo. El método utilizado fue el ‘Autoencoder’, que mejora el rendimiento de la factorización de matrices para predecir las valoraciones de los usuarios. Autorec supera al método de descompo-

sición en valores singulares (SVD) en un conjunto de datos de música, con una diferencia de RMSE de 0.7.

Otro estudio titulado "Collaborative Filtering-Based Music Recommendation in View of Negative Feedback System" (Verma y cols. [2022]), menciona que utilizaron un método propio que combina el filtrado colaborativo basado en ítems con un sistema de retroalimentación negativa (NFS) que permite a los usuarios rechazar las canciones que no quieren escuchar. Esto resultó en una nueva serie de recomendaciones basadas solo en las canciones que le gustan al usuario. Gracias al NFS, el usuario puede reconocer fácilmente las recomendaciones con una precisión del 16,78 %. Por lo tanto, el sistema propuesto permite a los usuarios descubrir nuevas recomendaciones cada vez que usan el algoritmo de recomendación NFS y funciona mejor que los antiguos algoritmos basados en el contenido, como los mecanismos de filtrado basados en la popularidad.

Asimismo, el estudio "A Music Recommendation System Based on logistic regression and eXtreme Gradient Boosting" (Tian y cols. [2019]), trata sobre un sistema de recomendación de música que utiliza la regresión logística y el eXtreme Gradient Boosting (xgboost) para predecir las preferencias musicales del usuario. El sistema propuesto es un algoritmo híbrido llamado LX que integra la regresión logística y el xgboost. La regresión logística, que es un modelo lineal, se utiliza como clasificador para predecir las preferencias musicales del usuario. Sin embargo, la regresión logística no maneja muy bien las características de datos no lineales complejas. Para resolver este problema, se propone el uso de xgboost, que es capaz de manejar características no lineales.

## 3. Metodología

### 3.1. Mapa de Metodología

1. Obtención y Tratamiento de la Data
   - Obtención de los datos Million Song Dataset
   - Unir los dos sub-datas
   - Preprocesamiento de datos
2. Filtrado de Contenido y Contexto
   *a*) Filtro 1:
      - Usuarios que hayan escuchado de 90 a más canciones

*b*) Filtro 2:

- Canciones que hayan sido escuchadas por 120 a más usuarios

3. Reducción de Dimensionalidad
   - Aplicación de SVD (Singular Value Decomposition)
   - Utilización de PCA (Principal Component Analysis)
4. Problema Long Tail en Recomendación Musical
   - Consideración de canciones menos populares
   - Estrategias para equilibrar recomendaciones

### 3.2. Definición de las etapas de la investigación

Introducción

Estado del arte

Recopilación de Datos

Análisis de Datos

Interpretación de Resultados

Conclusiones y Discusión

## 4. Experimentación y Resultados

### 4.1. Descripción de la Data

La data se constituye por dos data set que tienen la siguiente descripción:

#### 4.1.1. song_data.csv

- **song_id**: Identificador único de la canción.
- **title**: Título de la canción.
- **release**: Nombre del álbum al que pertenece la canción.
- **artist_name**: Nombre del artista.

- **year**: Año de lanzamiento de la canción.

#### 4.1.2. count_data.csv

- **user_id**: Identificador único del usuario.
- **song_id**: Identificador único de la canción.
- **play_count**: Número de veces que el usuario ha escuchado la canción.

### 4.2. Características de la Data

#### 4.2.1. Características de song_data.csv

- Número total de registros: 1,000,000
- Número de artistas únicos: 72665
- Número de canciones únicas: 999056 (considerando que cada registro es único)
- Valores nulos: Se identificaron 15 valores nulos en la columna "title" y 5 en la columna "release".
- Años de lanzamiento: El conjunto de datos abarca canciones de diferentes años, incluyendo algunos registros sin año especificado (indicado como 0).

#### 4.2.2. Características de count_data.csv

- Número total de registros: 2,000,000
- Número de usuarios únicos: 76353
- Valores nulos: No se identificaron valores nulos.
- Duplicados: 498 registros duplicados
- Estadísticas de *play_count*:

  Media (mean): 3.045, lo que sugiere que, en promedio, una canción es escuchada aproximadamente 3 veces por un usuario.

  Desviación estándar (std): 6.58, lo que indica una variabilidad significativa en el número de veces que los usuarios escuchan canciones. Esto sugiere que, mientras algunas canciones son escuchadas pocas veces, otras son escuchadas muchas veces.

  Mínimo (min): 1, lo que significa que cada registro en este conjunto de datos representa al menos una escucha de una canción por un usuario.

25 % (primer cuartil): 1, lo que indica que el 25 % de las canciones fueron escuchadas solo una vez por los usuarios.

Mediana (50 %): 1, lo que sugiere que al menos el 50 % de las canciones fueron escuchadas solo una vez. Esto respalda la idea de que muchas canciones son escuchadas pocas veces, mientras que un pequeño conjunto de canciones populares tiene un número significativamente alto de reproducciones.

75 % (tercer cuartil): 3, lo que significa que el 75 % de las canciones fueron escuchadas 3 veces o menos por los usuarios.

Máximo (max): 2,213, lo que indica que la canción más escuchada fue reproducida 2,213 veces. Esto destaca la presencia de canciones extremadamente populares en el conjunto de datos.

## 4.3. Unir CSVs

### 4.3.1. Proceso de Combinación

Para combinar eficientemente la información de ambos archivos CSV (referidos en el código como `count` y `song`), utilizamos la columna `song_id`, que es común en ambos archivos. La combinación se realiza de la siguiente manera:

```
#Combinar ambos conjuntos de datos
#usando 'song_id' como clave
combined_data = pd.merge(count, song, on='song_id',
how='left')
```

Posteriormente, llevamos a cabo las siguientes operaciones de limpieza en el conjunto de datos combinado:

- Eliminamos la columna `Unnamed: 0`, que parece ser un índice redundante.
- Descartamos la columna `year`, que contiene información del año de lanzamiento de la canción, ya que no es relevante para el análisis actual.
- Eliminamos registros duplicados para garantizar la integridad de los datos.

El conjunto de datos resultante se guardó en un archivo llamado `combinada.csv`.

### 4.3.2. Análisis del Conjunto de Datos Combinado

**Valores Nulos o Vacíos**: Después de combinar los datos, es esencial verificar la presencia de valores nulos o vacíos. Al hacerlo, encontramos que:

```
nulos = combined_data.isnull().sum()
```

**Búsqueda de Artistas Específicos**: Realizamos búsquedas para tres artistas específicos en el conjunto de datos combinado: 'Sébastien Roch', 'Milton' y 'Remute'. En todos los casos, no se encontraron registros relacionados con estos artistas.

**Cantidad de Artistas Antes de la Combinación**:https://www.overleaf.com/project/65
Antes de combinar los conjuntos de datos, el archivo `song` contenía un cierto número de artistas únicos, que se calculó como:

```
valores_unicos = song['artist_name'].unique()
cantidad_artistas = len(valores_unicos)
```

**Cantidad de Artistas Después de la Combinación**: Tras la combinación, el número total de artistas únicos en el conjunto de datos combinado se determinó de la siguiente manera:

```
valores_unicos = combined_data['artist_name'].unique()
cantidad_artistas = len(valores_unicos)
```

### 4.4. Filtración de la Data Combinada

#### 4.4.1. Filtro 1: Cantidad de Canciones Escuchadas por Usuario

**Sin GroupBy:** Se utilizó un enfoque iterativo para determinar cuántas canciones escuchó cada usuario. Se ordenaron los datos por `user_id` y se iteró a través de ellos, sumando las canciones escuchadas por cada usuario y reiniciando el conteo cuando se encontraba un nuevo usuario. El resultado fue un DataFrame `canciones_usuario` con dos columnas: `user_id` y `cantidad_canciones`. Ejemplo de los datos resultantes:

```
    user_id                                   cantidad_canciones
0   b80344d063b5ccb3212f76538f3d9e43d87dca9e  46
1   85c1f87fea955d09b4bec2e36aee110927aedf9a  1
...
```

**Con GroupBy:** Se utilizó la función `groupby` para agrupar los datos por `user_id` y contar cuántas canciones escuchó cada usuario. El resultado, `conteo_canciones_por_usu` fue similar al obtenido con el método anterior, lo que valida la precisión de ambos enfoques. Ejemplo de los datos resultantes:

```
    user_id                                   canciones_escuchadas
```

```
0   00003a4459f33b92906be11abe0e93efc423c0ff   7
1   00005c6177188f12fb5e2e82cdbd93e8a3f35e64   5
...
```

**Consulta Específica:** Como ejemplo, se consultó cuántas canciones escuchó un usuario específico (`b80344d063b5ccb3212f76538f3d9e43d87dca9e`). Se determinó que este usuario escuchó 46 canciones.

**Aplicación del Filtro**
Se filtró la lista de usuarios para identificar a aquellos que hayan escuchado más de 90 canciones. Se encontraron 3,390 usuarios que cumplen con este criterio. Además, se realizó un histograma del filtro (ver figura 1) que indica un sesgo hacia la derecha.

#### 4.4.2. Filtro 2: Cantidad de canciones que Escucharon los usuarios

**Sin GroupBy:** Se utilizó un enfoque iterativo similar al anterior, pero esta vez se ordenaron los datos por `song_id` y se contaron los usuarios que escucharon cada canción. El resultado fue un DataFrame `usuario_cancion` con dos columnas: `song_id` y `cantidad_usuarios`.

Ejemplo de los datos resultantes:

```
    song_id                 cantidad_usuarios
0   SOAAAGQ12A8C1420C8      66
1   SOAACPJ12A81C21360      147
...
```

**Con GroupBy:** Se utilizó la función `groupby` de pandas para agrupar los datos por `song_id` y contar cuántos usuarios únicos escucharon cada canción. El resultado, `conteo_usuarios_por_cancion`, fue similar al obtenido con el método anterior.

Ejemplo de los datos resultantes:

```
    song_id                 usuarios_diferentes
0   SOAAAGQ12A8C1420C8      66
1   SOAACPJ12A81C21360      147
...
```

**Aplicación del Filtro:** Se filtró la lista de canciones para identificar aquellas que hayan sido escuchadas por más de 120 usuarios. Se encontraron 5,256 canciones

que cumplen con este criterio. Asimismo, se realizó un histograma (ver figura 2) que indica un sesgo hacia la derecha.

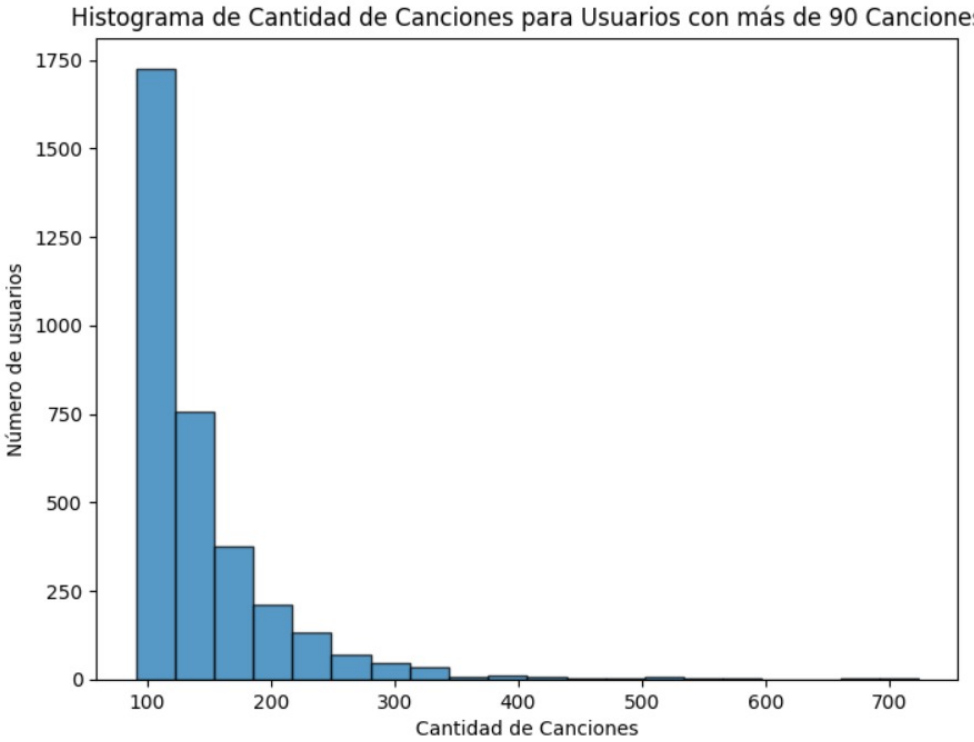


Figura 1: Histograma de cantidad de canciones para usuarios con mas de 90 canciones.

### 4.5. Descripción de la data obtenida despúes de la filtración

Al aplicar ambos filtros a la data combinada eliminamos los usuarios y canciones que no se encontraban dentro de esos valores. Finalmente, se obtuvo una nueva data con las siguientes características:

1. Número total de registros: 386848
2. Número de artistas únicos: 1959
3. Número de usuarios únicos: 3390
4. Número de canciones únicas: 5256
5. Valores nulos: No hay valores nulos.

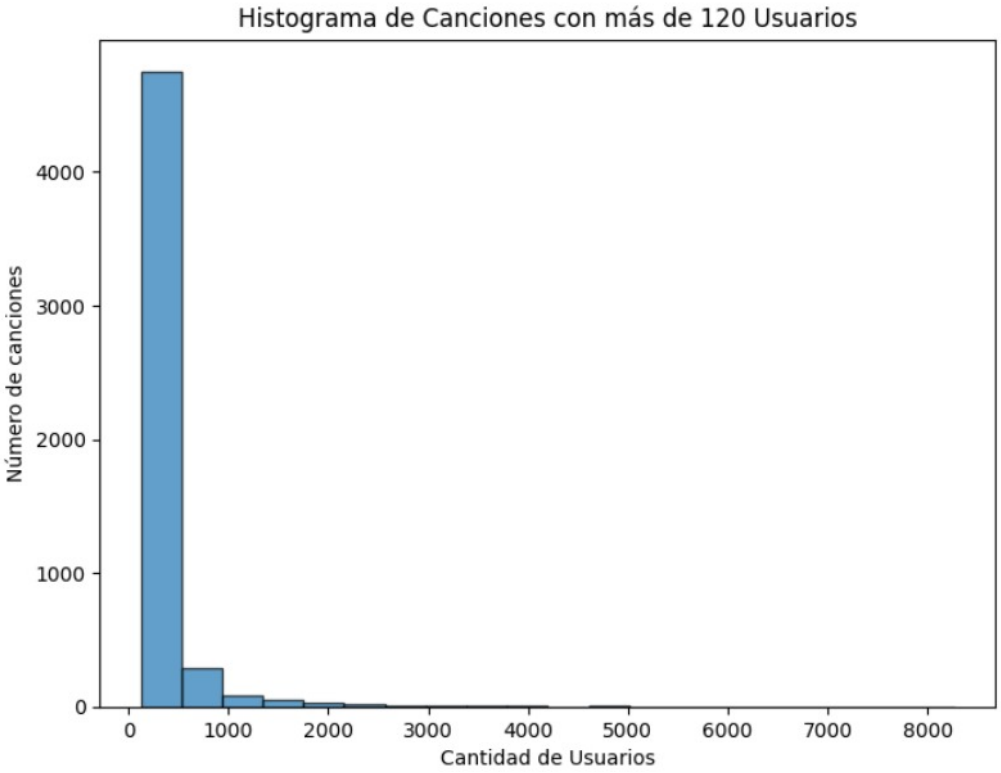


Figura 2: Histograma de cantidad de canciones para usuarios con mas de 120 canciones.

### 4.6. Top 10 Canciones Más Escuchadas

El análisis del Top 10 de canciones más escuchadas nos proporciona una visión clara sobre las canciones que tienen la mayor popularidad entre los usuarios. Para determinar cuáles son estas canciones, se realizó un conteo de cuántos usuarios escucharon cada canción.

#### 4.6.1. En función del ID de la Canción:

Se ordenaron las canciones según la cantidad de usuarios que las escucharon y se seleccionaron las 10 más populares. El resultado se visualizó en un gráfico de barras donde el eje x representa el `song_id` y el eje y muestra la cantidad de usuarios.

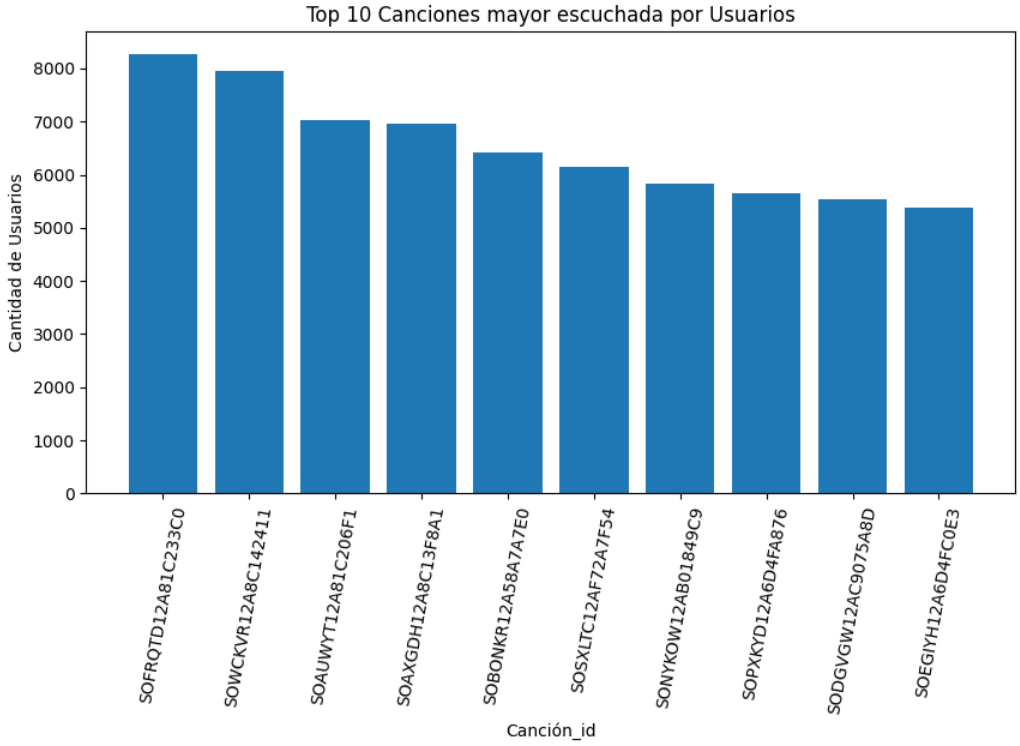


Figura 3: Top 10 canciones en función del ID.

### 4.6.2. En función del Título de la Canción:

Con el fin de proporcionar una visión más intuitiva de los resultados, se procedió a la unión de datos con el conjunto original, permitiendo así obtener el título de cada canción en el Top 10. Luego, se creó un gráfico de barras horizontales donde el eje y representa el título de la canción y el eje x muestra la cantidad de usuarios.

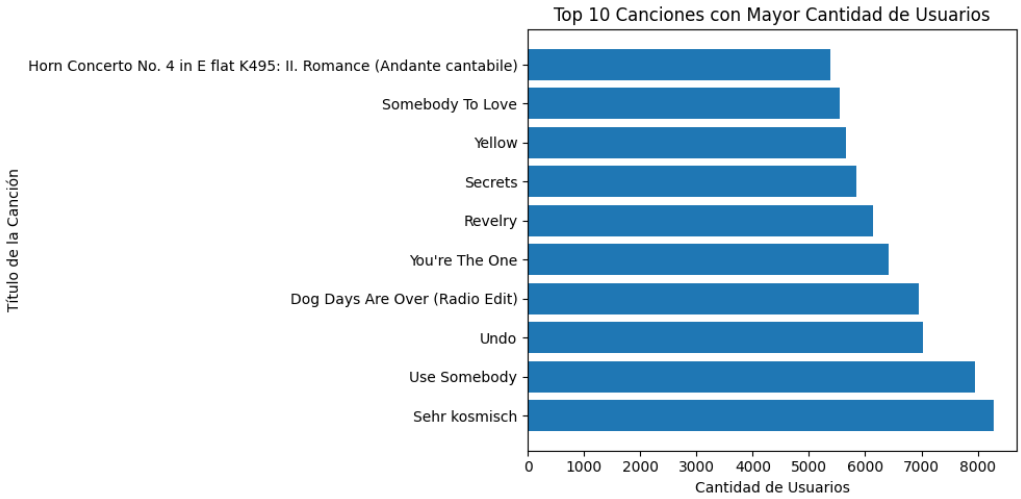


Figura 4: Top 10 canciones en función del título.

#### 4.6.3. En función del número de reproducciones:

Para tener una perspectiva más completa de la popularidad de las canciones, también se identificaron las diez canciones más populares basadas en su recuento total de reproducciones, independientemente del número de usuarios que las hayan escuchado. Se presenta un gráfico de barras que muestra estas canciones con su respectivo recuento de reproducciones.

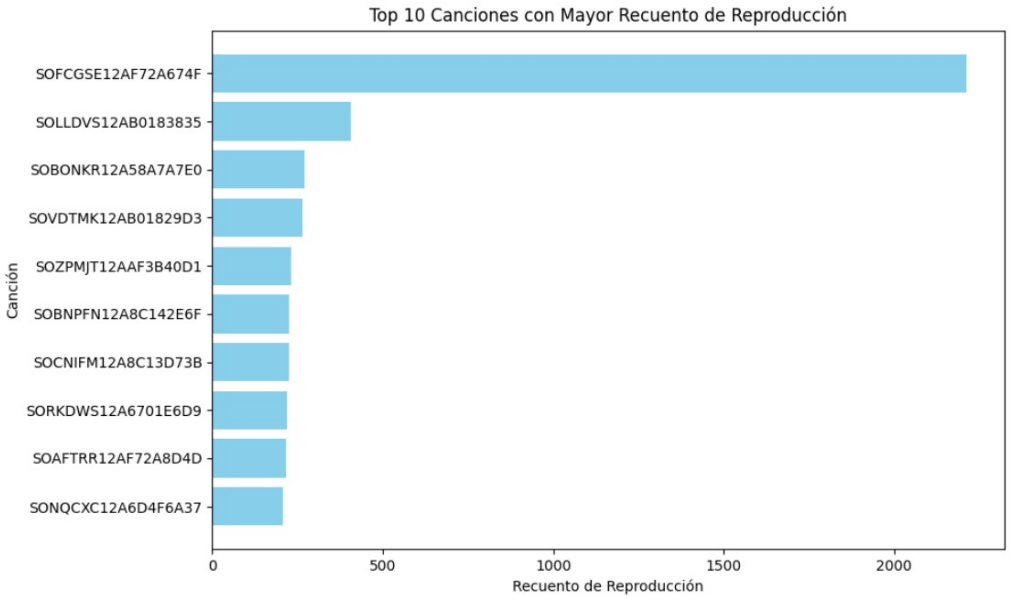


Figura 5: Grafico de barras de play_count con escala logarítmica.

### 4.7. Distribución de play_count

El `play_count` representa cuántas veces se ha escuchado una canción específica. Al visualizar la distribución de este atributo, podemos obtener insights sobre la frecuencia con la que las canciones son reproducidas.

#### 4.7.1. Histograma sin Escala Logarítmica:

Se creó un histograma básico para visualizar la distribución de `play_count`. En este gráfico, el eje x representa el número de reproducciones, mientras que el eje y muestra la frecuencia de canciones con ese número de reproducciones.

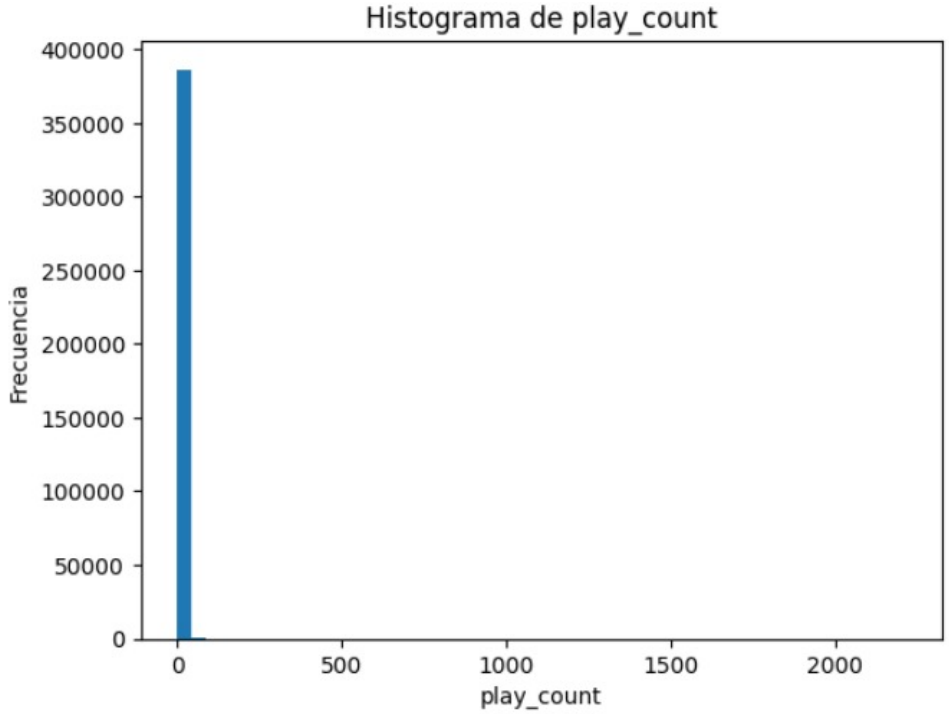


Figura 6: Histograma de play_count.

#### 4.7.2. Histograma con Escala Logarítmica:

Dado que muchos valores de `play_count` son bajos y pocos son muy altos, se utilizó una escala logarítmica para el eje y del histograma, lo que permite una mejor visualización y comprensión de la distribución. En esta visualización, los valores pequeños se expanden, proporcionando una visión clara de la frecuencia de las canciones con menos reproducciones. Asimismo,se observó el sesgo a la derecha respecto al número de reproducciones que hay en la cantidad de registros.

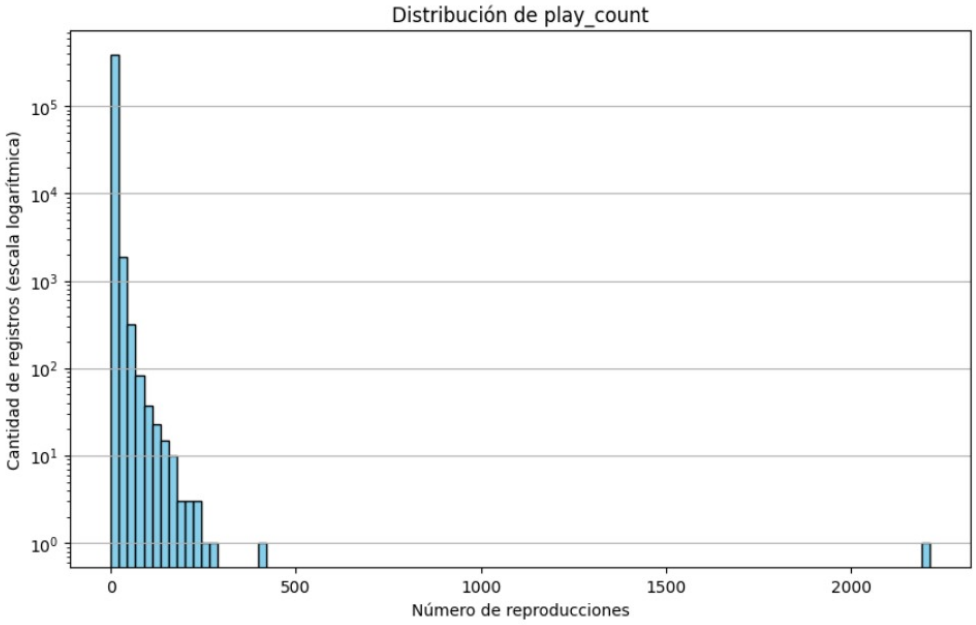


Figura 7: Histograma de play_count con escala logarítmica.

# 5. Experimentación

## 5.1. Filtrado Colaborativo:

El filtrado colaborativo es una técnica común en sistemas de recomendación. En el caso específico de la música, se puede usar para sugerir canciones basadas en la cantidad de veces que han sido reproducidas. Esto se hace mediante un método llamado filtrado colaborativo basado en memoria (user-item collaborative filtering), que utiliza una tabla o matriz con los números de reproducciones de cada usuario para cada canción(ver figura 8).

### 5.1.1. Cálculo de Similitud de Coseno

La similitud entre usuarios o canciones se mide para identificar patrones de escucha similares. Utilizando la matriz de reproducciones, se calcula la similitud de coseno entre los usuarios mediante la función `cosine_similarity` del paquete `sklearn.metrics.pairwise`. Esta función compara los vectores de reproducción de los usuarios y genera una matriz de similitudes que refleja los patrones compartidos. Posteriormente, se forma un conjunto de usuarios afines, conocido como "vecindario", a partir de esta matriz.

### 5.1.2. Similitud de Coseno

Continuando con el análisis de similitud, exploramos más a fondo la métrica de similitud de coseno. Este enfoque matemático es fundamental en la construcción de nuestro sistema de recomendación, ya que permite cuantificar el grado de similitud entre los gustos musicales de los usuarios. La similitud coseno, al ser una medida del coseno del ángulo entre dos vectores, nos ofrece un valor numérico directamente proporcional a las preferencias compartidas entre dos usuarios.

Dado que se ha establecido que los vectores de usuarios se pueden comparar mediante el producto punto, profundizaremos en cómo esta operación se traduce en la similitud de coseno. A continuación, se presenta la fórmula matemática utilizada para calcular esta métrica:

$$\text{Similitud} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Donde $A_i$ y $B_i$ son las componentes de los vectores respectivamente. El rango de la similitud de coseno es de -1 a 1, lo cual indica que valores más cercanos a 1 representan una mayor similitud entre los patrones de escucha de los usuarios.

### 5.1.3. Recomendación de Canciones

En la fase de generación de recomendaciones, se selecciona un usuario de ejemplo y se buscan aquellos con patrones de reproducción similares, utilizando un umbral de similitud establecido, como 0.7. Identificando usuarios con gustos musicales afines a través de la comparación de sus vectores de reproducción, se procede a recomendar canciones que estos usuarios similares han escuchado, pero que aún no han sido reproducidas por el usuario en cuestión. Cabe destacar que estas recomendaciones se ordenan por el total de reproducciones, indicando su popularidad.

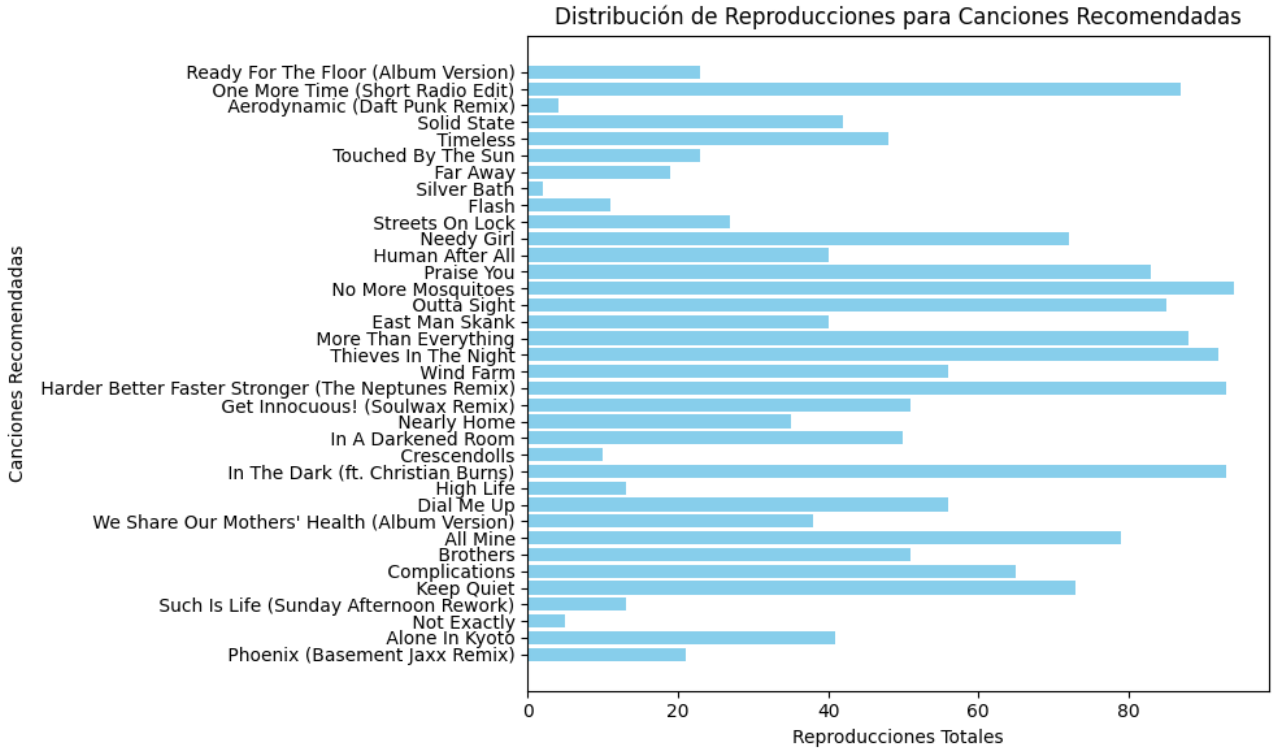


Figura 8: Distribución de Reproducciones para Canciones Recomendadas

## 5.2. SVD CON CUR PARA DATA FILTRADA

El proceso comienza filtrando un conjunto de datos para incluir solo las canciones con un número mínimo de reproducciones, lo que ayuda a mejorar la calidad de las recomendaciones. El SVD se encarga de entender patrones ocultos en los datos de interacciones entre usuarios y elementos, como canciones. Al descomponer esta matriz, el modelo puede identificar preferencias latentes de usuarios y características destacadas de las canciones. La adición de CUR permite manejar matrices grandes de manera más eficiente al seleccionar solo las partes esenciales para el modelado, lo que reduce la complejidad computacional sin sacrificar significativamente la calidad de las recomendaciones. Este enfoque permite que el modelo SVD genere recomendaciones más rápidamente al trabajar con un subconjunto clave de los datos, resultando en una mejora en la escalabilidad del sistema de recomendación.

Asimismo, se utilizó la librería 'surprise', que es una librería para sistemas de recomendación.Uno de sus usos fue el proyecto de Sistema-de-recomendación-de-títulos-musicales", en el cual se usó esta librería para construir y analizar sistemas de recomendación que tratan con datos de calificación explícitos.. En este siste-

ma basado en la memoria, consideraron una tabla dinámica entre los productos y los usuarios y sus valoraciones correspondientes. En el sistema de recomendación basado en la memoria, solo utilizaron esos datos y calcularon la puntuación de similitud recomendando productos similares para un producto en particular. Realizaron una técnica llamada factorización de matrices con la ayuda de una biblioteca llamada Surprise. Usaron GridSearch CV para identificar cuántos factores latentes deben considerarse para la factorización. Una vez que obtienen el valor óptimo del método GridSearch CV, entrenan el modelo y predicen las valoraciones de los productos que el usuario no ha predicho.

Filtramos un data con 90 podemos tener recomendaciones reales. De ello, pudimos hallar el top 10 de canciones escuchada.

$$min\_reproducciones = 90$$

reffig:f1).

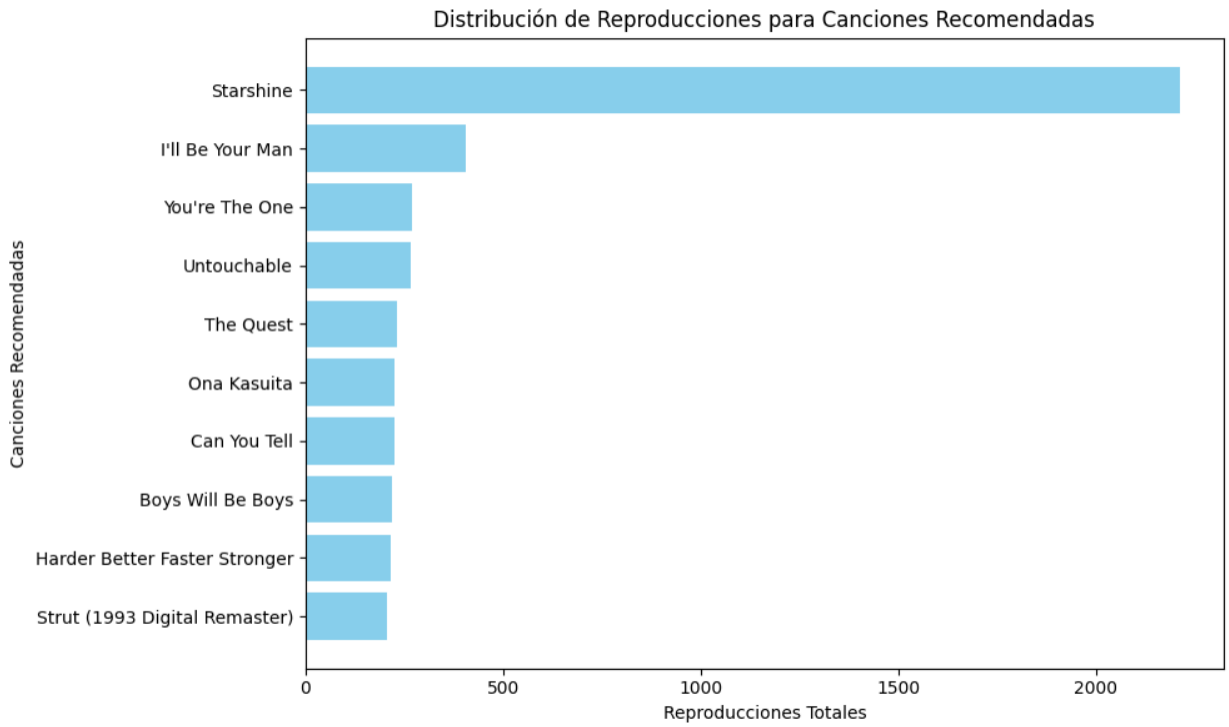


Figura 9: Cantidad de Canciones Únicas Escuchadas por Usuario.

### 5.2.1. Data de entrenamiento y test

En el contexto de modelos de recomendación, el sesgo se refiere a una tendencia sistemática en las predicciones del modelo. En el caso del modelo SVD (Des-

composición en Valores Singulares), el sesgo puede referirse a la incorporación de términos adicionales en la estimación de las valoraciones de los usuarios para corregir o ajustar la predicción.

Cuando se dice entrenar un modelo SVD sin sesgo, significa que el modelo se entrena sin incluir esos términos adicionales o correcciones que podrían introducir una tendencia en las predicciones. En este caso, se utiliza el parámetro biased=False al configurar el modelo SVD.

La idea detrás de entrenar un modelo sin sesgo es simplificar el modelo y evitar introducir sesgos potenciales en las predicciones. Esto puede tener beneficios en ciertos escenarios, especialmente si los datos son suficientemente densos y representativos.

Recuerda que trainset es una estructura de datos especializada diseñada para trabajar con la biblioteca Surprise y no una matriz bidimensional típica, por lo que el concepto de "dimensión" puede no aplicarse de la misma manera que en otras estructuras de datos. El parámetro test-zise insica la proporción del conjunto de datos que se usará para el subconjunto de prueba en este caso el 20 %. El parámetro random-state indica la semilla que se usará para la generación de números aleatorios, en este caso el 42. Esta semilla sirve para que los resultados sean reproducibles.

A esta data filtrada se aplico SVD sin sesgo, se está utilizando una versión simplificada del modelo que no incluye ajustes adicionales en las predicciones, lo que puede ser útil dependiendo del contexto del conjunto de datos y el objetivo de las recomendaciones en la data filtrada con el minimo con reproduciones. Tambien calculamos el error RMSE en el subconjunto de prueba y lo almacena en una variable llamada rmse. El error RMSE es una medida de la diferencia entre las valoraciones reales y las estimadas por el modelo, cuanto menor sea mejor.

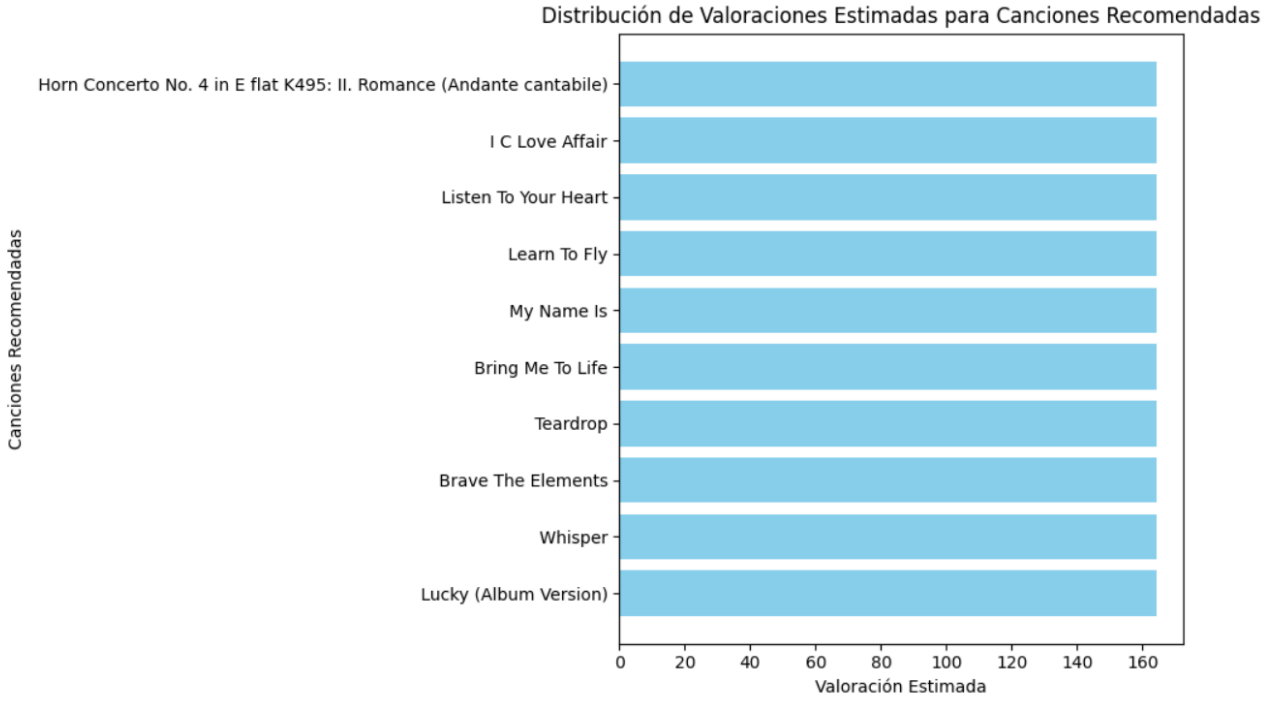


Figura 10: Distribución de Valoraciones Estimadas para Canciones Recomendadas

### 5.2.2. Conteo de Músicas Escuchadas por Usuarios

Se analizó la cantidad de canciones únicas escuchadas por usuario(ver figura 11).

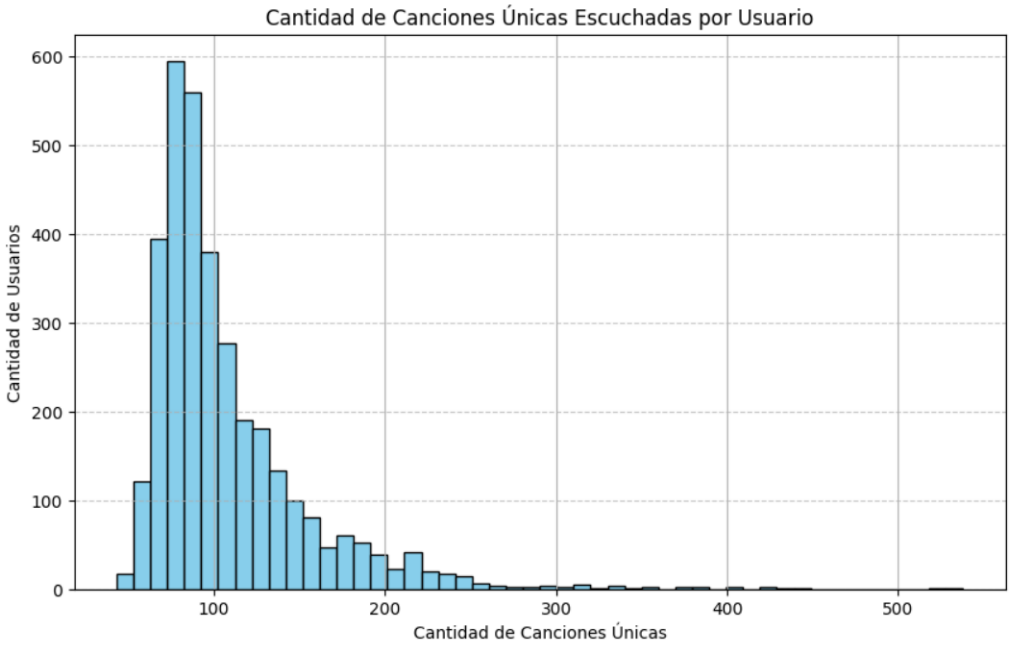


Figura 11: Cantidad de Canciones Únicas Escuchadas por Usuario.

## 6. Conclusiones

La aplicación de técnicas como filtrado colaborativo y SVD con CUR demuestra la versatilidad y eficacia de los sistemas de recomendación en el ámbito musical. El uso de similitud de coseno permite identificar patrones de escucha similares entre usuarios, mejorando la personalización de las recomendaciones. La implementación de SVD sin sesgo simplifica el modelo y puede ser beneficioso en escenarios con datos densos y representativos. El análisis visual y la aplicación de umbrales (por ejemplo, 90 reproducciones) pueden ser útiles para filtrar datos y mejorar la calidad de las recomendaciones.

El problema de long tail, se refiere a que hay una gran cantidad de canciones que son poco conocidas o escuchadas, pero que pueden ser de interés para algunos usuarios. Estas canciones forman parte de la cola larga de la distribución, mientras que las canciones más populares o comerciales forman parte de la cabeza. Un sistema de recomendación musical eficaz debe ser capaz de ofrecer a los usuarios tanto canciones de la cabeza como de la cola, para satisfacer sus gustos y preferencias, y al mismo tiempo descubrirles nuevas opciones.

## 7. Recomendaciones

A partir de los resultados y aprendizajes obtenidos en este proyecto, sugerimos las siguientes recomendaciones para futuras mejoras y desarrollos en sistemas de recomendación musical:

1. Integración de otros Algoritmos: Experimentar con la integración de otros algoritmos podría ofrecer mejoras significativas en la precisión y la diversidad de las recomendaciones. Esto incluye la exploración de algoritmos de filtrado colaborativo basado en modelos, como factorización de matrices y PCA. Se recomienda ajustar SVD con CUR que reduce la complejidad del modelo y puede ser beneficioso en escenarios con datos densos y representativos.
2. Optimización de Umbrales de Filtrado: Continuar experimentando con diferentes umbrales para el filtrado de datos, como el número mínimo de reproducciones, para afinar el equilibrio entre la calidad y la diversidad de las recomendaciones.

## Referencias

Irawan, M. F., y Baizal, Z. (2023). Music recommender system using autorec method for implicit feedback. *JURNAL MEDIA INFORMATIKA BUDIDARMA*, *7*(2), 609–614.

McFee, B., Bertin-Mahieux, T., Ellis, D. P., y Lanckriet, G. R. (2012). The million song dataset challenge. En *Proceedings of the 21st international conference on world wide web* (pp. 909–916). Descargado de http://millionsongdataset.com/tasteprofile/

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., y Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. En *Proceedings of the 1994 acm conference on computer supported cooperative work* (pp. 175–186).

Resnick, P., y Varian, H. (1997). Recommender systems. *Commun. ACM*, *40*(3), 56–58.

Tian, H., Cai, H., Wen, J., Li, S., y Li, Y. (2019). A music recommendation system based on logistic regression and extreme gradient boosting. En *2019 international joint conference on neural networks (ijcnn)* (pp. 1–6).

Verma, J. P., Bhattacharya, P., Rathor, A. S., Shah, J., y Tanwar, S. (2022). Collaborative filtering-based music recommendation in view of negative feedback system. En *Proceedings of third international conference on computing, communications, and cyber-security: Ic4s 2021* (pp. 447–460).