RETE MUSICALE 2.0: EXPLORANDO LA MÚSICA A TRAVÉS DE REDES NEURONALES PROFUNDAS. (diciembre, 2023)

Daza Pereira Juan Pablo y Camargo Sanchez Juan Sebastian.

Fundamentos de la Inteligencia Artificial.

Escuela colombiana de Ingeniería Julio Garavito.

Resumen- El articulo “RETE MUSICALE 2.0: EXPLORANDO LA MÚSICA A TRAVÉS DE REDES NEURONALES PROFUNDAS” se zambulle en la unión entre la música y el mundo de la inteligencia artificial, siendo más precisos, en la exploración de las redes neuronales profundas y su búsqueda de nuevas posibilidades de comprender y crear, a futuro, piezas musicales.

En un mundo donde la música es un lenguaje universal dado que por medio de ella es posible expresar emociones, formas de pensar y sentimientos [1], las redes neuronales las queremos presentar como una herramienta que será capaz de clasificar diversos géneros musicales. Esta se centrará en ser entrenada con diferentes características de entrada, como lo son, por ejemplo, danceability, energy y tempo, para predecir géneros musicales como lo son el pop, rock y R&B.

El articulo mostrará los diferentes desafíos que se llevarán a cabo, como lo son la diversidad de géneros, la subjetividad y ambigüedad en la clasificación musical. Estos desafíos se abordarán por medio de estrategias que más adelante serán explicadas en la metodología de trabajo.

Finalmente, deseamos que a lo largo del artículo se vea la forma en cómo las redes neuronales profundas son capaces de adaptarse a cualquier situación, como lo es en nuestro caso, el ámbito de la música, y como el diseño de patrones permitirá comprender una nueva visión en el lazo de la música y los algoritmos en esta nueva era digital.

Palabras clave **–** Redes neuronales profundas, Clasificación musical, Datos de música, Diversidad de géneros, Subjetividad, Ambigüedad, Características musicales, Entrenamiento de redes neuronales profundas, Géneros musicales (Pop, Rock y R&B), Algoritmos.

I. INTRODUCCIÓN.

La música ha sido una expresión artística que ha evolucionado con el paso de los años, propagándose en varias culturas y lenguas, llegando a tocar las fibras más profundas de la humanidad. Con la nueva era digital, el poder de las notas y melodías se juntan con el poder de la tecnología, dándole paso a un nuevo movimiento entre la creatividad e imaginación humana y el mundo de los algoritmos.

Ese nuevo movimiento, ha llevado a una innovación tecnológica, donde las redes neuronales profundas han sido una herramienta que permite explorar el mundo del arte musical. Estas redes neuronales profundas, que se basan en la complejidad, producción, belleza y exactitud del cerebro humano, que han encontrado en la música un nuevo campo de exploración basado en la clasificación, interpretación y

creación de géneros y sonidos musicales, rompiendo límites y abriendo puertas para que este arte pueda ser aprovechado.

Este artículo se sumergirá en cruzar el arte y los algoritmos, por medio de las redes neuronales profundas, que tendrán la capacidad de aprender, distinguir patrones de aprendizaje y comprender la música en su diversidad capturando no solamente la esencia de sus notas y acordes, sino también las agudezas emocionales y la complejidad de los géneros musicales.

En general, el artículo se centrará en entrenar una red neuronal profunda de clasificación musical, donde la diversidad de géneros, la subjetividad y la ambigüedad se convierten en brechas a superar. Con diez variables de entrada y tres de salida, la red neuronal profunda se ligará con datos musicales, divididos entre los géneros de pop, rock y R&B.

A lo largo de este artículo, los sorprenderemos con esa gran combinación entre la música y los “ritmos” algorítmicos, mostrando nuevas posibilidades y desafiando la percepción actual de la forma de crear y comprender la música.

II. MARCO TEÓRICO.

- Redes Neuronales Profundas

Las redes neuronales profundas, inspiradas en el funcionamiento del cerebro humano, representan una arquitectura de aprendizaje automático que consiste en múltiples capas de unidades interconectadas, llamadas neuronas, que procesan y analizan datos de manera jerárquica. Estas redes han revolucionado diversos campos, desde la visión por computadora hasta el procesamiento del lenguaje

natural, debido a su capacidad para aprender representaciones complejas y extraer características significativas de conjuntos de datos masivos [2].

En nuestro concepto, el de la música, queremos demostrar que las redes neuronales profundas pueden ser una herramienta valiosa para analizar los patrones musicales, clasificar géneros, predecir las preferencias y gustos del usuario, y que a futuro sea una herramienta que sirva para las demás ramas de la música.

- Clasificación musical y sus características (entradas y salidas):

La clasificación musical, como es real en ese arte, asigna etiquetas y categoriza las piezas musicales según sus características. En el estudio que estamos realizando, vamos a tener de base 10 características como variables de entrada: *"danceability", "energy", "key", "loudness", "mode", "speechiness", "acousticness", "instrumentalness", "tempo"* y *"duration_ms"*. Estas métricas nos proporcionan la información necesaria y relevantes de una canción.

Por otro lado, las salidas de la red neuronal se enfocarán en solo tres géneros: *"pop"*, *"rock*" y *"R&B"*. La tarea de la red neuronal será la de asignar correctamente el tipo de género a una canción dadas las características de entradas que se mencionaron anteriormente.

- "Desafíos" en la clasificación musical con redes neuronales profundas:

La diversidad en los géneros musicales, la subjetividad en la percepción y la ambigüedad en ciertas características, son desafíos que nacen a la hora de entrenar redes neuronales profundas dirigidas en la clasificación musical. La

variación de estilos musicales y la evolución de estos dificultan la creación de modelos matemáticos y tecnológicos precisos que sean capaces de abordar toda esa gama musical.

También, la subjetividad humana en la interpretación de la música genera discrepancias en la clasificación, mientras que ciertas características, como la instrumentalidad y la mezcla de varios géneros, suelen se ambiguas y difíciles de clasificar objetivamente.

Por ello, en este estudio, se explorarán estrategias y enfoques para desvanecer esos desafíos, permitiendo realizar una clasificación musical más precisas y adaptable al mundo del arte musical.

III. METODOLOGÍA.

- Recolección y preparación de datos:

Se tomó un conjunto de datos la página Kaggle, de una data llamada “30000 Spotify Songs” [3], que abarca más de 30.000 muestras musicales, y cada una con 10 características mencionadas anteriormente. De todo este conjunto, se seleccionaron específicamente los géneros musicales más relevantes para nuestro estudio: Pop, Rock y R&B.

Para el género de rock tenemos un total de 4.951 canciones, para pop un total de 5.507 canciones y para R&B un total de 5.431 canciones. En total tenemos 12.719 canciones de estos géneros mencionados y otros 20.114 de otros géneros para el entrenamiento y testeo de nuestra red.

De esta data tomamos diez datos que consideramos nosotros importantes para la identificación de géneros musicales, que son:

```
s[['danceability', 'energy','key','loudness','mode','speechiness','acousticness','instrumentalness','tempo','duration_ms']].t
```

Y de los cuales vamos a escoger alguno de ellos para la creación de nuestra red neuronal profunda (si desea saber más información sobre cada atributo y que significa cada uno, puede leerlo en Kaggle donde se encontró el data frame).

- Diseño de la red neuronal:

Se va a empezar con una red neuronal, en la cual se le van a reducir el número de entradas, pero seguirá contando con el mismo número de salidas, tres. Dependiendo del resultado de la precisión y de si este supera el umbral que se definió, se decide si se implementa o no otra alternativa. Si umbral no se cumpliera, se realizarían tres opciones más para reduciendo el número de salidas a un solo género (Rock). La primera opción es usar ingeniería de características para escoger las mejores entradas para la red. La segunda opción es la de seguir usando las mismas 10 entradas, pero para solo un género. Y la tercera opción es apartir de los hechos dado por medio de un experto en música. Todo esto, lo que se hizo y los resultados de la experimentación se verán más adelante en los resultados.

- Ingeniería de selección o de características:

La ingeniería de características implica seleccionar, transformar o crear nuevas características (también llamadas atributos o variables) a partir de los datos originales antes de alimentarlos al modelo de aprendizaje automático [4]. Con la

definición anterior, el objetivo principal es mejorar el rendimiento del modelo “Rete musicale 2.0” al proporcionarle las características más relevantes y significativas para la tarea en cuestión. Por ello, hemos decidido implementar este método de ingeniería de características para mejorar la eficiencia escogiendo las mejores entradas que nos permitan acercarnos al umbral propuesto.

- División de datos y entrenamiento:

El conjunto de datos se dividió en un 80% para entrenamiento (datos reales de cada género) y un 20% para pruebas (datos reales y no reales de cada género). Se utilizó el 80% de los datos para entrenar la red, ajustando los pesos y los sesgos de la red neuronal.

- Evaluación:

Una vez entrenada la red neuronal con el 80% de los datos, se evaluó su rendimiento utilizando el 20% restante reservado para pruebas. Se aplicaron métricas de evaluación para analizar la capacidad del modelo para clasificar correctamente las muestras musicales en los géneros de interés: pop, rock y R&B, considerando la función de activación sigmoide en la capa de salida.

Nuestra metodología se enfoca en la selección especifica de géneros musicales de tantos existentes, la configuración de una arquitectura como modelo de la red neuronal, la división de datos para entrenamiento y pruebas, y la misma evaluación del rendimiento del modelo que se está entrenando. Este enfoque proporciona más detalles del proceso de entrenamiento y la prueba de la red neuronal profunda en el contexto de clasificación musical.

La precisión a la que queremos llegar para este proyecto es del 75%, mejorando la precisión de la primera red neuronal montada (Rete musicale 1.0) y cumpliendo ese porcentaje para dar como conclusión que se implementó una red neuronal profunda efectiva.

IV. RESULTADOS RETE MUSICALE 1.0:

Empezamos a montar la red con perceptrón para ver cómo se comporta. Agarramos el 80 % de cada uno de los géneros y los testeamos. Al montarlo los resultados no fueron los mejores, el costo por cada 1000 iteraciones no estaba bajando y ni se acercaba a 0:

```
(10, 12715)
(3, 12715)
Red 1 de [10,3] con 100 mil iteraciones:
Cost after iteration 0: 36.04142568469426
Cost after iteration 5000: 49.6248679694428
Cost after iteration 10000: 49.62486796938084
Cost after iteration 15000: 49.6248679693601
Cost after iteration 20000: 49.62486796934972
Cost after iteration 25000: 49.624867969343484
Cost after iteration 30000: 49.62486796933933
Cost after iteration 35000: 49.62486796933636
Cost after iteration 40000: 49.62486796933413
Cost after iteration 45000: 49.6248679693324
Cost after iteration 50000: 49.62486796933101
```

Cambiamos la arquitectura de la red para aproximarnos a un menor costo. Esta vez intentamos:

```
41 print(np.shape(allGenresInput))
42 print(np.shape(Y))
43 print("Red 1 de [43] con 100 mil iteraciones:")
44 parameters1=train(allGenresInput, Y, [10,5,4,3], 0.1, 100000, 10000)

(10, 12715)
(3, 12715)
Red 1 de [43] con 100 mil iteraciones:
Cost after iteration 0: 2.080589707975549
Cost after iteration 10000: 1.9079136283496934
Cost after iteration 20000: 1.9079136231698175
Cost after iteration 30000: 1.9079136228932205
Cost after iteration 40000: 1.9079136226148277
```

Aquí el valor 1.907913 va a ser una constante en el costo de nuestras redes ya que, montando diferentes redes, con diferente neurona y con diferentes capas el costo siempre diverge a este valor:

```
44 parameters1=train(allGenresInput, Y, [10,5,4,5,3], 0.1, 100000, 10000)

(10, 12715)
(3, 12715)
Red 1 de [43] con 100 mil iteraciones:
Cost after iteration 0: 2.0787317989443337
Cost after iteration 10000: 1.9079136286645568
Cost after iteration 20000: 1.907913628663264
```

```
44 parameters1=train(allGenresInput, Y, [10,11,4,3], 0.1, 100000, 10000)

(10, 12715)
(3, 12715)
Red 1 de [43] con 100 mil iteraciones:
Cost after iteration 0: 2.076533984907169
Cost after iteration 10000: 1.907913608173706
```

```
44 parameters1=train(allGenresInput, Y, [10,6,8,4,3], 0.1, 100000, 10000)

(10, 12715)
(3, 12715)
Red 1 de [43] con 100 mil iteraciones:
Cost after iteration 0: 2.0848027755420238
Cost after iteration 10000: 1.9079136286739085
Cost after iteration 20000: 1.9079136286739085
```

Cada una de estas redes además de tener un valor constante tenían tiempos de ejecución de horas por lo que nos ahorrábamos que acabara el número total de ejecuciones cuando sabíamos que iba a divergir a ese valor.

Incluso intentamos reducir las entradas de nuestra red para ver si podíamos generar algún cambio en el costo:

```
43 parameters1=train(allGenresInput, Y, [7,8,5,4,3], 0.1, 100000, 5000)

(7, 12715)
(3, 12715)
Red 1 de [43] con 100 mil iteraciones:
Cost after iteration 0: 2.0863526025962935
Cost after iteration 5000: 1.907913631540566
Cost after iteration 10000: 1.9079136298150996
Cost after iteration 15000: 1.9079136281055262
Cost after iteration 20000: 1.9079136264115428
Cost after iteration 25000: 1.9079136247323616
Cost after iteration 30000: 1.9079136230666522
Cost after iteration 35000: 1.9079136214125296
Cost after iteration 40000: 1.9079136197675781
Cost after iteration 45000: 1.907913618128905
Cost after iteration 50000: 1.9079136164932056
Cost after iteration 55000: 1.907913614856837
Cost after iteration 60000: 1.9079136132158836
Cost after iteration 65000: 1.9079136115662165
Cost after iteration 70000: 1.9079136099035383
Cost after iteration 75000: 1.9079136082234163
Cost after iteration 80000: 1.907913606521301
Cost after iteration 85000: 1.9079136047925354
Cost after iteration 90000: 1.9079136030323518
Cost after iteration 95000: 1.907913601235862
Cost after iteration 100000: 1.9079135993984107
```

No resulto, por lo que decidimos escoger una red en la que no tome mucho tiempo de ejecución y tenga el valor de costo constante.

```
Red 1 de [10,6,3] con 100 mil iteraciones:
0.6666666666666666
```

Aplicando nuestra función de precisión observamos que lo máximo que pudimos lograr entrenar la red fue de un 0.66.

Creemos que esto se debe a que o no pudimos encontrar una arquitectura apropiada para entrenar nuestra red o que factores tal como la veracidad en la base de datos no estuvieran correctos. Tenemos también la sospecha que puede que haya un sobre entrenamiento de nuestra red.

V. RESULTADOS RETE MUSICALE 2.0 (CON LA RED ORIGINAL Y USANDO INGENIERÍA DE CARACTERÍSTICAS):

Para la preparación de datos usamos la ingeniería de características de datos usando sklearn para python. Quitamos las columnas que no queremos que analice, que sería lo del nombre, álbum, etc. Después de eso le decimos que el enfoque de los datos debe estar orientado al tracklist_genre con esto la herramienta nos sacara un resultado numérico según la importancia de cada columna:

```python
import pandas as pd
from sklearn.ensemble import RandomForestClassifier

# Load your dataset (replace 'your_dataset.csv' with your actual file name)
df = pd.read_csv('/content/sample_data/spotify_songs.csv')

# Separate features and target variable
X = df.drop(['track_id', 'track_name', 'track_artist', 'track_popularity','track_album_id', 'track_album_name',
y = df['playlist_genre']  # Target variable column

# Create a random forest classifier
clf = RandomForestClassifier(random_state=42)

# Fit the classifier to your data
clf.fit(X, y)

# Get feature importances from the trained model
feature_importances = clf.feature_importances_

# Create a DataFrame to display feature importances
feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': feature_importances
})

# Sort the DataFrame by importance in descending order
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

# Print the feature importance DataFrame
print("Feature Importance:")
print(feature_importance_df)
```

Resultado del proceso:

```
Feature Importance:
              Feature  Importance
10              tempo    0.122339
5         speechiness    0.117409
0        danceability    0.113161
1              energy    0.096658
11        duration_ms    0.090055
9             valence    0.089920
6        acousticness    0.089616
3            loudness    0.084848
7    instrumentalness    0.073664
8            liveness    0.065454
2                 key    0.043470
4                mode    0.013405
```

Ahora hacemos la selección de los datos más importantes, se redujo a 3, y lo aplicamos en nuestra red neuronal para empezar a experimentar el costo.

Intentamos varias redes con diferentes capas y neuronas, pero sigue siendo persistente el problema de la red pasada 1.0. La red no se está entrenando correctamente o está en alguna clase de sobre entrenamiento. El valor de costo es constante para cualquier red que experimentemos y sigue dando como costo el valor 1.097 aproximadamente.

Aquí alguno de los resultados de las redes:

```
38 parameters1=train(allGenresInput, Y, [3,4,3], 0.1, 10000, 5000)

(3, 12715)
(3, 12715)
Cost after iteration 0: 2.0824554538748408
Cost after iteration 5000: 1.9078366035720582
Cost after iteration 10000: 1.907916996442692
```

```
38 parameters1=train(allGenresInput, Y, [3,4,3], 0.1, 10000, 5000)

(3, 12715)
(3, 12715)
Cost after iteration 0: 2.0824554538748408
Cost after iteration 5000: 1.9078366035720582
Cost after iteration 10000: 1.907916996442692
```

```
38 parameters1=train(allGenresInput, Y, [3,2,4,2,5,3], 0.1, 10000, 5000)

(3, 12715)
(3, 12715)
Cost after iteration 0: 2.086466593154675
Cost after iteration 5000: 1.9079136286867258
```

```
38 parameters1=train(allGenresInput, Y, [3,12,3], 0.1, 10000, 5000)

(3, 12715)
(3, 12715)
Cost after iteration 0: 2.0815460917848605
Cost after iteration 5000: 1.9060936679529776
Cost after iteration 10000: 1.9073687277981004
```

Abandonamos la posibilidad de conseguir alguna mejoría en las redes incluso si las segmentamos en una red para cada género.

También se omite los test ya que la mejoría fue nula y la precisión de esta nueva red será de 0.6666 sin conseguir la mejoría esperada de 0.8.

VI. RESULTADOS RETE MUSICALES 2.0 (INGENIERIA DE CARACTERISTICAS Y UNA SALIDA):

Como se mencionó anteriormente en la metodología se iban a implementar 3 opciones si no se llegará a cumplir con el umbral que se propuso. La primera opción es la que vamos a presentar a continuación. Se va a implementar una red neuronal, pero haciendo uso de la ingeniería de características y reduciendo e l numero de entradas a solo un género. La ingeniería de características fue

realizada en el numeral V de este y a partir de esos resultados escogimos los datos más relevantes para las entradas de nuestra red neuronal, y esas entradas escogidas a partir de la ingeniería de características, son las siguientes: “danceability”, “speechiness” y “tempo”. Ya con estas entradas entramos a realizar la arquitectura de la red neuronal con sus tres respectivas entradas y su salida que será la del género de Rock. Los resultados de implementación de la primera red neuronal con las opciones establecidas son las siguientes:

```
31 parameters1=train(allGenresInput, Y, [3,4,2,3,1], 0.1, 10000, 5000)

(3, 8308)
(1, 8308)
Cost after iteration 0: 0.6930420711636058
Cost after iteration 5000: 0.6920786313215734
```

```
31 parameters1=train(allGenresInput, Y, [3,5,1], 0.1, 30000, 5000)

(3, 8308)
(1, 8308)
Cost after iteration 0: 0.6932549967432736
Cost after iteration 5000: 0.6921193086212397
Cost after iteration 10000: 0.6922158107622745
Cost after iteration 15000: 0.6922115575412898
```

```
31 parameters1=train(allGenresInput, Y, [3,5,3,1], 0.1, 30000, 5000)

(3, 8308)
(1, 8308)
Cost after iteration 0: 0.6931659451174854
Cost after iteration 5000: 0.6920665512543724
Cost after iteration 10000: 0.6919832924223126
Cost after iteration 15000: 0.6912839642055825
Cost after iteration 20000: 0.6831020889654782
Cost after iteration 25000: 0.6919891995118043
Cost after iteration 30000: 0.6905346127884977
```

Aplicando nuestra función de precisión observamos que lo máximo que pudimos lograr entrenar la red fue de un 0.84.

Analizando esta primera opción, como primer dato, se rompe con el umbral propuesto al inicio de este artículo superándolo un 9%, un resultado satisfactorio. Y lo segundo es que se puede ver que, usando la ingeniería de características, reduciendo el numero de entradas y salidas, la red neuronal se entrenó muy bien, el costo de iteraciones mejoró, también como lo fue el tiempo de ejecución que es menor a las demás redes montadas.

```
Red 1 de [3,5,3,1] con 5 mil iteraciones:
0.8410478945676632
```

VII. RESULTADOS RETE MUSICALE 2.0 USANDO OPCIÓN 2 (EXPERTO Y UNA SALIDA):

Haciendo uso del conocimiento de un experto en el arte de la música, esté nos dio a conocer un concepto que se usa en este arte, el concepto del metrónomo. El metrónomo es una herramienta necesaria para los músicos porque con esa herramienta es que ellos pueden mantener las mediciones en la música. Este metrónomo se encarga de medir variables musicales como el sonido, ritmo, tiempo y armonía. Gracias al conocimiento del experto y a la data que tenemos, pudimos hacer uso de esas cuatro variables para implementar la segunda opción de la red neuronal basada en el conocimiento del experto. Por ello, la arquitectura de la red queda con cuatro entradas (Danceability, Tempo, Acousticness, Loudness), que son las cuatro variables del metrónomo, y la misma salida que se uso en el numeral VI (Rock), obteniendo así los siguientes resultados:

```
Red 1 de [3,5,3,1] con 5 mil iteraciones:
0.8410478945676632
```

Nuevamente, como en los numerales VI y VIII, los resultados los mismos a pesar de cambiar las entradas, sigue sucediendo lo mismo, lo único raro es que varia el tiempo de ejecución.

VIII. RESULTADOS RETE MUSICALE 2.0 USANDO OPCIÓN 3 (10 ENTRADAS Y UNA SALIDA):

Ya como ultima opción para ver las diferencias de resultados, se tomó la misma red neuronal del numeral VI, pero se le realizaron cambios. Los cambios que se realizaron son en las entradas, se usan las mismas entradas de la versión 1.0 de la Rete Musicale, manteniendo, nuevamente, la arquitectura de la red neuronal del numeral VI. Con estos cambios los resultados son los siguientes:

```
31 parameters1=train(allGenresInput, Y, [10,5,3,1], 0.1, 10000, 5000)

(10, 8308)
(1, 8308)
Cost after iteration 0: 0.6931716926052857
Cost after iteration 5000: 0.6920786321184966
Cost after iteration 10000: 0.6920786321184966
```

```
Red 1 de [3,5,3,1] con 5 mil iteraciones:
0.8410478945676632
```

Nuevamente se tiene la misma precisión que la red del numeral VI, manteniendo el mismo costo de iteraciones, pero en lo único que varía en es el tiempo de ejecución, siendo esta red más demorada que la red del numeral VI.

IX. CONCLUSIONES.

Con todos los resultados anteriores podemos llegar a la conclusión de que siempre van a variar los resultados de entrenamiento con respecto a la

arquitectura, porque al entrenar una red neuronal con diferentes números de entradas y salidas implica consideraciones específicas en términos de arquitectura, manejo de datos y objetivos de la red. Es decir, la dimensionalidad de los datos es relevante, a mayores entradas se trabaja con conjunto de datos que tienen múltiples variables o características, y a menor entradas, el proceso de entrenamiento y modelado es más especifico y simple. Ahora para la parte de la arquitectura, tener muchas variables de entrada y salida implica tener una red con más capas o neuronas en comparación con un modelo con menos entradas y salida siendo esta más simple de implementar en términos de arquitectura. Y en la parte de procesamiento de datos para la red más compleja (versión 1.0 y numeral V) era necesario un preprocesamiento de datos aplicando técnicas de selección para reducir la dimensionalidad y mejorar el rendimiento y eficiencia de la red, por ello se usó la ingeniería de características, la ayuda de un experto, aplicando reducción de entradas y salidas, teniendo una red más eficiente.

En resumen, la diferencia principal radica en la complejidad y el manejo de la información que entra y sale de la red neuronal. Una mayor dimensionalidad de entrada y salida puede requerir modelos más complejos y un preprocesamiento más detallado de los datos, mientras que una menor dimensionalidad puede permitir una arquitectura más simple y un enfoque más directo en características específicas. Por ello se pudo observar que haciendo una red dirigida hacia un solo género fue eficiente (84% de eficiencia) a comparación que una red de muchas entradas dirigidas a tres géneros.

X. REFERENCIAS.

[1]: La música como medio de expresión. Disponible en:

https://www.upb.edu.co/es/noticias/la-musica-como-medio-de-expresi%C3%B3n#:~:text=La%20m%C3%BAsica%2C%20adem%C3%A1s%2C%20es%20un,Arist%C3%B3teles%2C%20%E2%80%9CLa%20m%C3%BAsica%20expresa%20los

[2]: ¿Qué es una red neuronal? Disponible en:

https://aws.amazon.com/es/what-is/neural-network/#:~:text=Las%20redes%20neuronales%20profundas%2C%20o,entre%20un%20nodo%20y%20otro

[3]: 30000 Spotify Songs. Disponible en:

https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs/data?select=spotify_songs.csv

[4]: Ingeniería de características. Disponible en:

https://osm3rr.medium.com/en-data-science-qu%C3%A9-es-la-ingenier%C3%ADa-de-caracter%C3%ADsticas-432c32b88901

[5] Implementación de sklearn:

ChatGPT con la consulta de: "Como usar sklearn para una tabla donde elija las mejores característica de una tabla con su importancia”