Clasificación de géneros musicales mediante técnicas de Machine Learning

I) Introducción

El objetivo de este proyecto es comparar varios algoritmos de aprendizaje automático en su capacidad para clasificar automáticamente extractos de canciones en el género musical correcto. La música es hoy en día una parte importante del contenido de Internet: la red es probablemente la fuente más importante de piezas musicales y esto ha llevado al crecimiento exponencial de varios sitios dedicados a difundir, distribuir y comercializar música; estos grandes servicios de transmisión han reconocido que el análisis manual y la anotación de canciones individuales no parece una solución sostenible a largo plazo. Hasta ahora, el procedimiento estándar para organizar el contenido musical es el uso manual de meta-etiquetas de información, estas incluyen el título de la canción, el álbum, el año, el número de la pista y el **género musical**, así como otros detalles específicos del contenido del archivo. Sin embargo, la información sobre el género musical suele ser incompleta e inexacta, ya que depende de la subjetividad humana. Como no existe un estándar industrial, surge más confusión: por ejemplo, la misma pieza musical se indica como perteneciente a diferentes géneros musicales, debido a diferentes interpretaciones y / o diferentes esquemas de codificación.

El dataset escogido para el desarrollo del proyecto es el conjunto de datos de la colección de géneros GTZAN que se recopiló entre 2000-2001. Consiste en 1000 archivos de audio cada uno con una duración de 30 segundos. Hay 10 clases (10 géneros musicales) cada una con 100 pistas de audio. Cada pista está en formato .wav. Contiene archivos de audio de los siguientes 10 géneros: Blues, Clásico, Country, Disco, Hip hop, Jazz, Metal, Pop, Reggae y Rock.

El proceso a seguir es el siguiente: primero, se extraen varias características acústicas (coeficientes cepstrales de frecuencia de mel, centroide espectral, tasa de cruce cero, frecuencias cromáticas, reducción espectral) para generar un conjunto de características, luego de esto, sigue una descripción de los modelos de aprendizaje automático, así como una presentación de los resultados de cada clasificador. Posteriormente, se realiza una interpretación de los resultados y finalmente se concluye con las principales lecciones aprendidas y recomendaciones para futuras investigaciones.

II) Desarrollo

1) Extracción de características

La extracción de características es una parte muy importante para analizar y encontrar relaciones entre diferentes cosas. Los modelos no pueden entender los datos proporcionados de audio directamente para convertirlos en un formato comprensible. Se utiliza la extracción de características. Es un proceso que explica la mayoría de los datos, pero de una forma

comprensible. La extracción de características es necesaria para los algoritmos de clasificación, predicción y recomendación.

Para la extracción de las características se utilizará la librería *librosa*, esta es un paquete de Python para análisis de audio y música. Proporciona los componentes básicos necesarios para crear sistemas de recuperación de información musical. Para representar las pistas numéricamente, se extrajeron 28 características de audio de cada pista. Estos valores representan el promedio de toda la pista (es decir, el valor MFCC1 de una pista se definió como el valor medio de MFCC1 en todas las ventanas de esa pista). Cada característica desempeñara un papel equivalente cuando se someta a los clasificadores de aprendizaje automático. Sin embargo, no todas las características fueron igualmente informativas, como se discutirá en la sección de análisis de este informe.

El vector de características final se describe en la Tabla 1.

Feature #	**Descripcion**
1	chroma_stft
2	rmse
3	spectral_centroid
4	spectral_bandwidth
5	rolloff
6	zero_crossing_rate
7	contrast
8	tonnetz
9	1 st. MFCC mean
10	2 nd. MFCC mean
11	3 rd. MFCC mean
12	4 th. MFCC mean
13	5 th. MFCC mean
14	6 th. MFCC mean
15	7 th. MFCC mean
16	8 th. MFCC mean
17	9 th. MFCC mean
18	10 th. MFCC mean
19	11 th. MFCC mean
20	12 th. MFCC mean
21	13 th. MFCC mean
22	14 th. MFCC mean
23	15 th. MFCC mean
24	16th. MFCC mean
25	17 th. MFCC mean
26	18 th. MFCC mean
27	19 th. MFCC mean
28	20 th. MFCC mean

Tabla 1. Vector de características

2) Modelos

En la siguiente sección primero se implementó dos modelos más simples como medidas de referencia de desempeño, para luego avanzar a más modelos complicados para aumentar la precisión. Se implementó Logistic Regression, Naive Bayes, k-nearest neighbors, Support Vector Machines y por ultimo una red neuronal.

2.1) Análisis de componentes principales

PCA es un método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conserva su información. Para este caso se hizo una descomposición en dos componentes principales para visualizar los posibles grupos de géneros y tener alguna idea de la complejidad de la clasificación. A simple vista se puede concluir que los clasificadores tendrán cierta dificultad en diferenciar las canciones de Rock y Country ya que estos dos géneros tienen características muy similares; en su contra parte el género clásico tendrá muy buenos resultados porque sus puntajes promedio para una serie de características son significativamente diferentes a los de otros géneros.

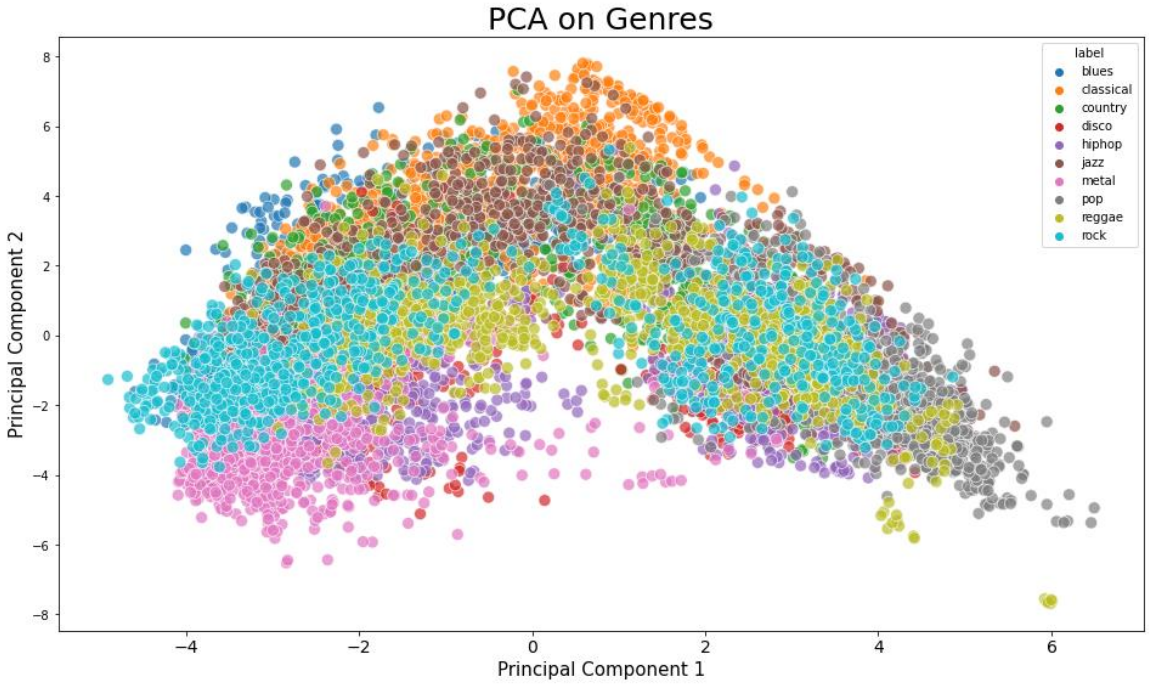

Figura 1. Descomposición en componentes principales

2.2) Train-Test Split

Se generan conjuntos de entrenamiento y prueba para desarrollar y probar nuestro clasificador. Usamos el 75% de los datos para entrenamiento y reservamos el 25% restante para pruebas. La función *shuffle* mezcla aleatoriamente los datos. Se hizo esto antes de dividir los datos en train y test para aleatorizarlos. Se utilizo la función *train_test_split* de Scikit-learn para realizar la división 80-20. *train_test_split* garantiza que todas las clases estén representadas por igual.

2.3) Logistic Regression

A la función que relaciona la variable dependiente con las independientes también se le llama función sigmoidea. La función sigmoidea es una curva en forma de S que puede tomar cualquier valor entre 0 y 1, pero nunca valores fuera de estos límites. La ecuación que define la función sigmoidea es

$$f(x) = \frac{1}{1 + e^{-x}}$$

donde x es un número real. En la ecuación se puede ver que cuando x tiene a menos infinito el cociente tiende a cero. Por otro lado, cuando x tiende a infinito el cociente tiende a la unidad. En la siguiente figura se muestra una representación gráfica de la función logística (función sigmoide). [1]

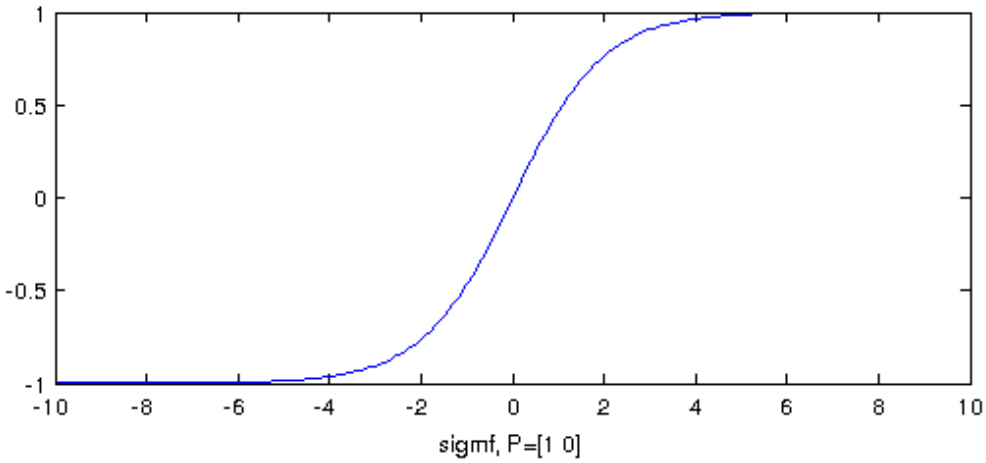

Figura 2. Función sigmoide

El proceso de entrenamiento de una función logística se puede realizar maximizando la probabilidad de que los puntos de un conjunto de datos clasifiquen correctamente. Lo que se conoce como estimación de máxima verosimilitud. La estimación de máxima verisimilitud es un enfoque genérico para la estimación de parámetros en modelos estadísticos. La maximización se puede realizar utilizando diferentes métodos de optimización como el descenso de gradiente. [1]

La implementación de este clasificador se hizo con el método *LogisticRegression de sklearn.linear_model.* Cabe aclarar que las regresiones logísticas son solo clasificadores binarios, lo que significa que no pueden manejar vectores objetivo con más de dos clases por lo tanto se implementó el esquema one-vs-rest para usar el clasificador como multiclase. En la regresión logística de uno contra el resto (OVR) se entrena un modelo separado para cada clase y se predice si una observación es esa clase o no (lo que lo convierte en un problema de clasificación binaria). Asume que cada problema de clasificación (por ejemplo, clase 0 o no) es independiente.

2.4) Naive Bayes

El algoritmo de **Naive Bayes** se fundamenta en los conocidos como **métodos bayesianos**, un conjunto de principios matemáticos fundamentales desarrollados por **Thomas Bayes** cuyo fin es la descripción de eventos probabilísticos. Sobre esta base los clasificadores basados en los métodos bayesianos utilizan datos de entrenamiento para calcular una probabilidad observada de cada clase basada en los valores de las variables. Cuando el clasificador se utiliza posteriormente en datos no etiquetados, utiliza las probabilidades observadas para predecir la clase más probable de los nuevos datos dados para estas variables. Este tipo de clasificación se muestra en la Figura 3.

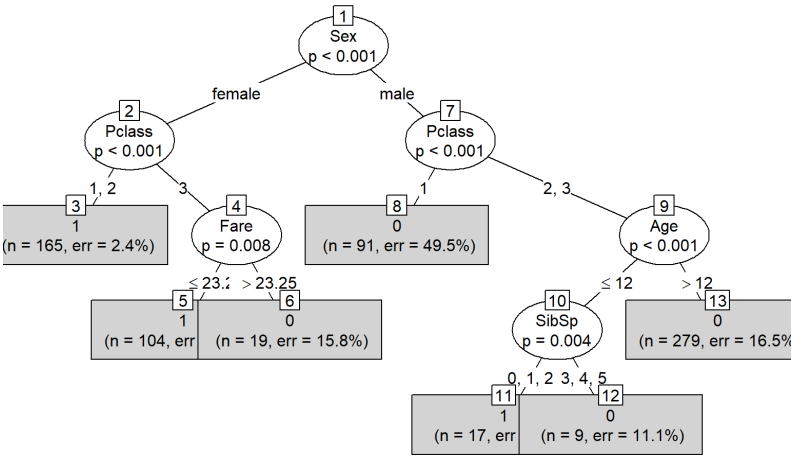

Figura 3. Clasificación utilizando Naive Bayes

La implementación de este clasificador se hizo con el método *GaussianNB() de sklearn.naive_bayes.*

2.6) k-nearest neighbors

Se calcula la distancia euclidiana entre los vectores almacenados en el entrenamiento y el nuevo vector de entrada, y se seleccionan las k muestras más cercanas. La nueva muestra se clasifica con la clase que más se repite en los vectores seleccionados. Este tipo de clasificación se muestra en la Figura 4. [10]

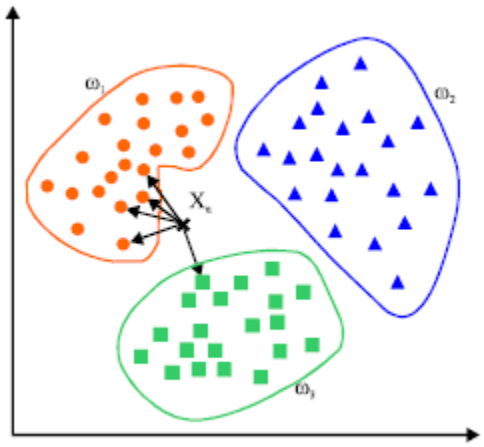

Figura 4. Clasificación utilizando KNN

El clasificador KNN se implementó con *KNeighborsClassifie*r de *sklearn*. Después de varias pruebas, se eligieron los siguientes hiperparámetros: k = 3, weights='distance', metric=distance, algorithm='brute'; estos demostraron ser los mejores encontrados.

2.7) Support Vector Machines

Si se grafican los datos, en caso de ser posible, este tipo de clasificador normaliza los datos para que las dos muestras más cercanas al hiperplano que separa las dos clases queden a una distancia de 1. Esta distancia entre las dos muestras más cercanas se le llama el margen y las dos muestras se llaman vectores de soporte. La clasificación optima (error mínimo en la clasificación) se obtiene cuando este margen sea máximo, entonces, si el margen es $\boldsymbol{Margin} = {}^{\boldsymbol{2}}/_{||\boldsymbol{W}||}$ siendo $\boldsymbol{W}$ el vector normal al hiperplano, se minimiza $\boldsymbol{W}$**.** Esta idea se muestra en la Figura 5. [10]

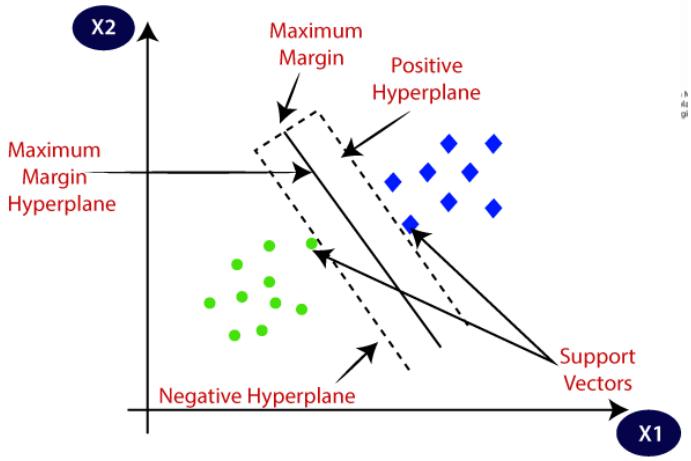

Figura 5. Clasificación de un SVM

Cuando los datos no son linealmente separables, se utiliza la técnica del Kernel, donde se llevan los datos a una dimensión *n* (Kernel de grado *n*) donde sean linealmente separables por un hiperplano y se calcula la

clasificación en esa dimensión. Lo anterior se muestra la Figura 6 donde se pasan los datos de $\boldsymbol{R^2}$ a $\boldsymbol{R^3}$ para así generar una clasificación. [10]

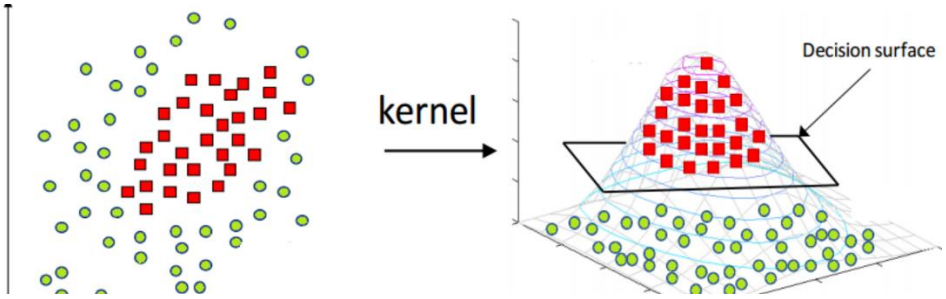

Figura 6. Cambio de los datos de $\boldsymbol{R^2}$ a $\boldsymbol{R^3}$

La máquina de vectores de soporte se implementó con SVC de *sklearn.svm.SVC*. Después de varias pruebas, se eligieron los siguientes hiperparámetros: kernel = rbf, C = 5000, gamma = 0.1, shrinking = True; estos demostraron ser los mejores encontrados.

2.8) Neural Network

Cada neurona hace una combinación lineal de las componentes del vector y las multiplica por una constante, esta constante se halla para cada neurona tal que minimice el error cuadrático medio utilizando mínimos cuadrados mostrado en la ecuación (1) donde el error es la diferencia entre la etiqueta $\boldsymbol{t_i}$ y el resultado del clasificador $\boldsymbol{y_i(X^k)}$. La función de activación escala los datos en el intervalo (-1,1). Esta idea se muestra en la Figura. 7. [10]

$$\boldsymbol{\epsilon = \frac{1}{2}\sum_i \left(y_i(X^k) - t_i\right)^2} \quad (\mathbf{1})$$

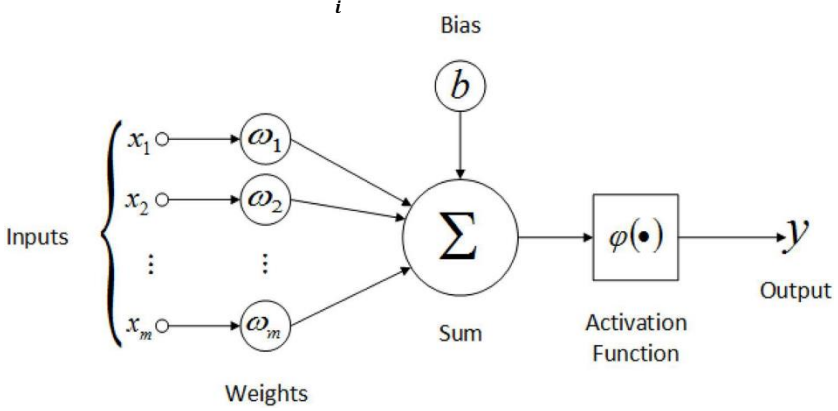

Figura 7. Modelo de Red neuronal de una sola capa.

Cuando los datos no son linealmente separables, se deben agregar más capas para que la salida de cada neurona sea la entrada de otra. Haciendo que se pueda generar una clasificación adecuada. Lo anterior se muestra en la Figura. 8. [10]

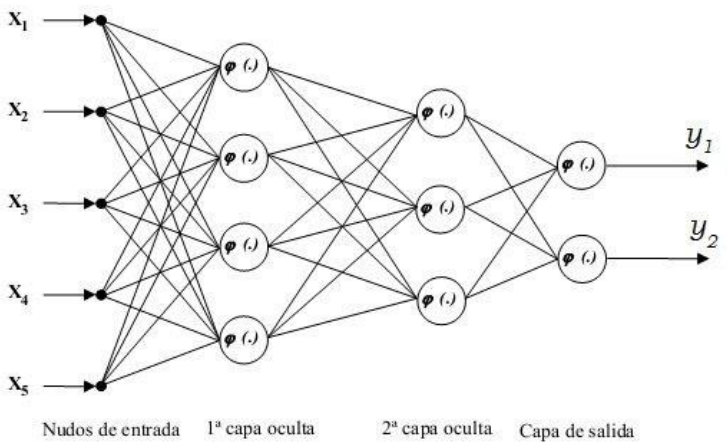

Figura 8. Modelo de Red neuronal multi-capa.

Para la implementación de la red neuronal se utilizó el modelo *MLPClassifier* de *sklearn.neural_network* ya que es un modelo sencillo que es relativamente fácil de ajustar, y esto permitió dedicar más tiempo a ajustar los otros clasificadores. Se seleccionaron 5000 capas ocultas con función de activación Relu, con estos hiperparametros se obtuvieron los mejores resultados.

III) Resultados

Las principales métricas cuantitativas que se usaron para evaluar los modelos implementados son la precisión y el coeficiente de correlación de Matthews, y la principal forma de visualizar el rendimiento de cada modelo es a través de su respectiva matriz de confusión como se ve en la Figura 9. Debido a que el etiquetado se distribuyó uniformemente en nuestros datos estas matrices de confusión ofrecen no solo una forma de visualizar nuestros datos, sino información más específica que la que ofrecen los valores de precisión, todos los valores de la matriz de confusión se han normalizado y redondeado.

	MCC	Accuracy
Logistic Regression	0.59	0.63
Naive Bayes	0.35	0.41
k-nearest neighbors	0.91	0.92
Support Vector Machines	0.92	0.93
Neural Network	0.88	0.89

Tabla 2. MCC y Accuracy de cada modelo

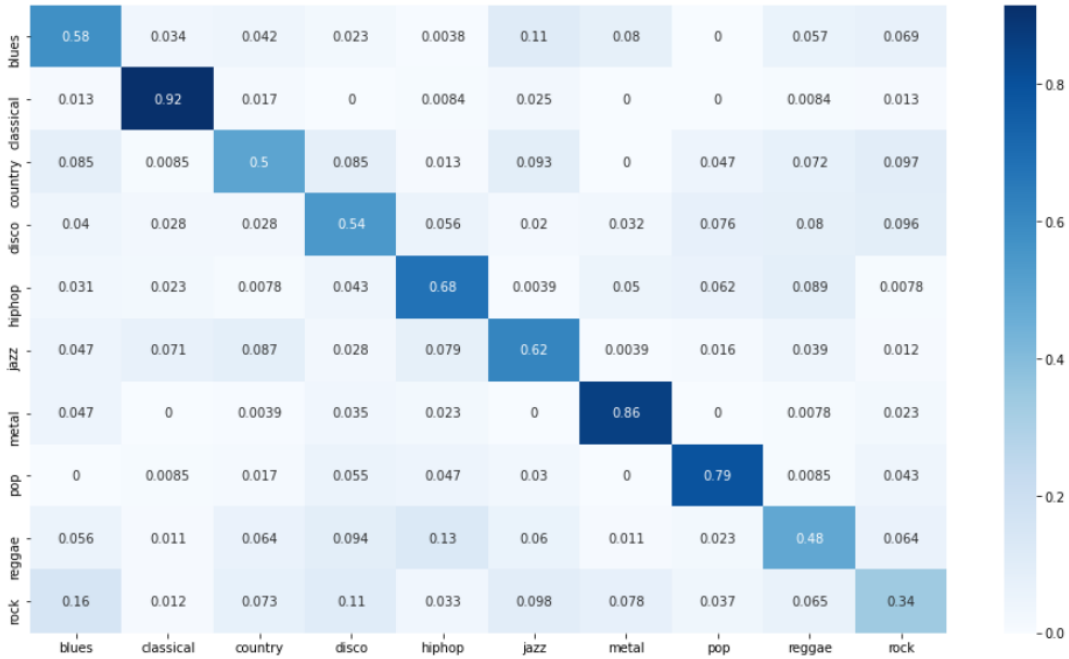

Figura 9. Matriz de confusión para las predicciones de Logistic Regression

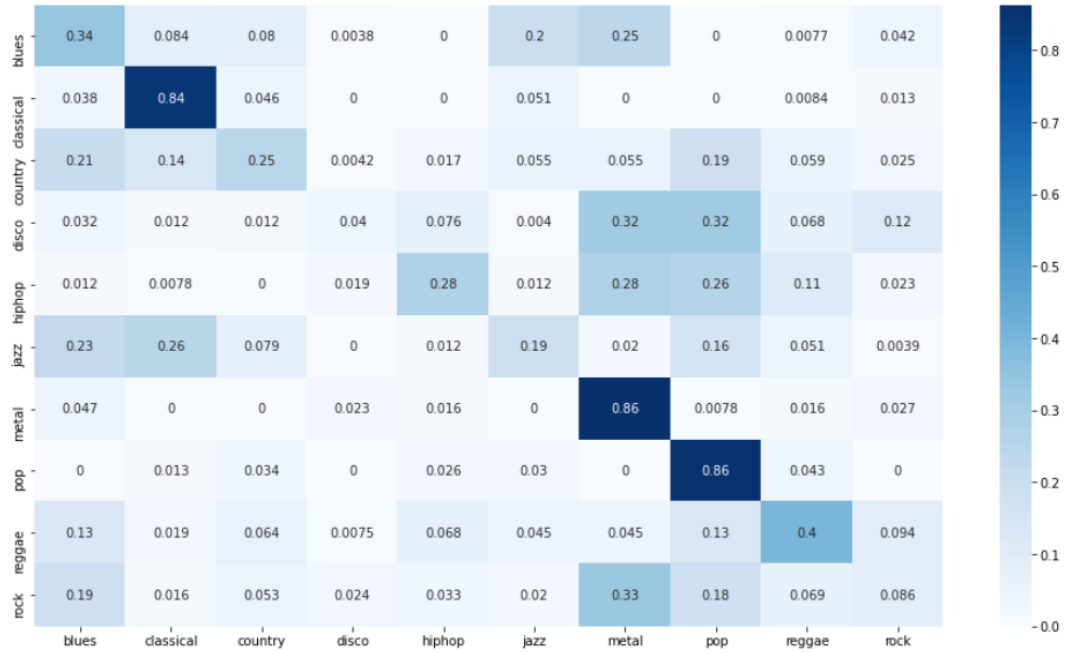

Figura 10. Matriz de confusión para las predicciones de Naive Bayes

Figura 11. Matriz de confusión para las predicciones de KNN

Figura 12. Matriz de confusión para las predicciones de SVM

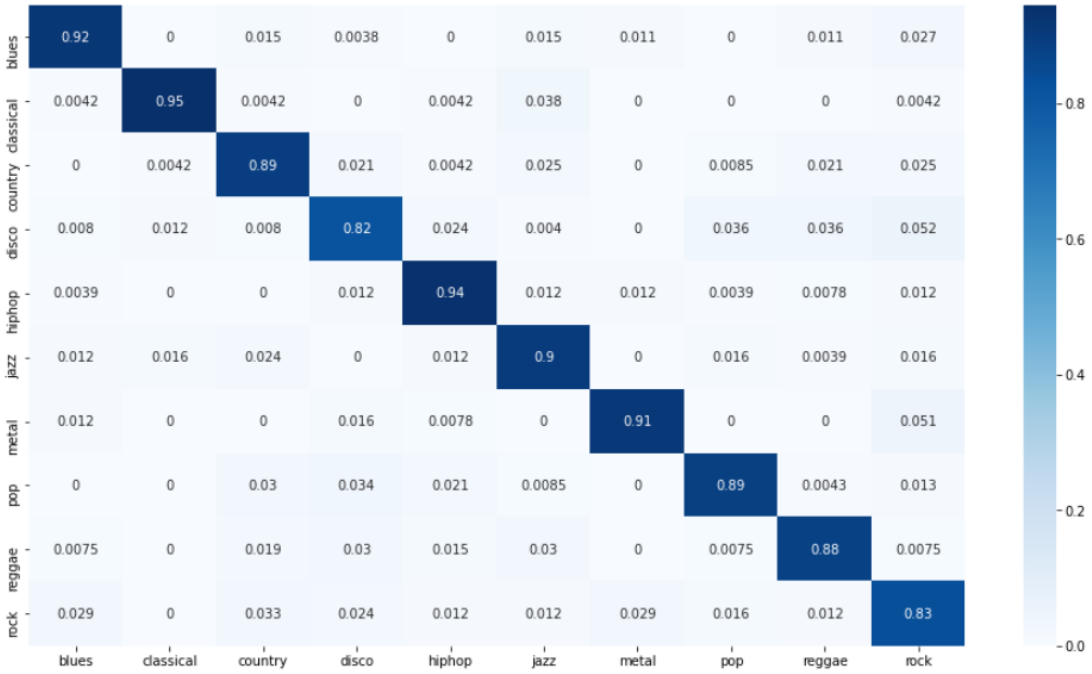

Figura 13. Matriz de confusión para las predicciones de Neural Network

El objetivo principal de este proyecto fue comparar los algoritmos de aprendizaje automático en términos de cuán adecuados eran para el problema de la clasificación de géneros musicales. En términos de MCC y precisión, la clasificación final de mejor a peor es: **SVM** (92% y 93%), KNN (91% y 92%), Neural Network (88% y 89%), LR (59% y 63%), Naive Bayes (35% y 41%).

IV) Conclusiones

- Dado que las señales de audio cambian constantemente, fue necesario dividir estas señales en cuadros más pequeños para obtener mejores resultados. Cada cuadro tiene una longitud de 3 seg de esta manera se aumentó 10 veces la cantidad de datos que se incorporaron a los modelos de clasificación.
- Conforme aumenta el valor de k para el clasificador de K- vecinos cercanos disminuye la precisión, ya que, al tomar una porción mayor de muestras para la comparación, se obtienen más de la clase a la que no pertenece la muestra evaluada, generando una clasificación errada.
- En general, las redes neuronales ofrecen más "personalización" que los otros clasificadores analizados (es decir, es posible construirlas con diferentes números de nodos, capas, etc.) sin embargo los clasificadores SVM y K- vecinos cercanos obtuvieron mejores resultados, además fue el clasificador mas lento comparado con los otros cuatro modelos.
- Los puntajes promedio de género en el conjunto de datos estándar fueron los siguientes: Blues: 74.4%, Clásica: 93%, Country: 68%, Disco: 65.2%, Hip-hop: 75.6%, Jazz: 69.6%, Metal: 91.2%, Pop: 88%, Reggae: 72.4%, Rock: 59%.
- Aunque el rendimiento varió con cada algoritmo de clasificación, se identificaron tendencias similares en los resultados de cada uno. Por

ejemplo, en la mayoría de los experimentos, el Metal, Pop y la música Clásica, se clasificaron con mayor precisión.

- El género clásico puede haber sido uno de los géneros clasificados con más éxito porque sus puntajes promedio para una serie de características eran significativamente diferentes a los de otros géneros, esto implica la presencia de diferencias sónicas que se detectaron con éxito en el entrenamiento de los algoritmos.
- En conclusión, el modelo SVM es el mejor clasificador para este tipo de problema además es el más rápido comparado con los clasificadores más cercanos a los resultados obtenidos, lo que aumenta su ventaja.

V) Referencias

[1] https://www.analyticslane.com/2018/07/23/la-regresion-logistica/

[2] http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-65002008000300002

[3] https://data-flair.training/blogs/python-project-music-genre-classification/

[4] https://www.kaggle.com/andradaolteanu/work-w-audio-data-visualise-classify-recommend/notebook

[5] https://towardsdatascience.com/extract-features-of-music-75a3f9bc265d

[6] B_Lansdown_Machine_Learning_For_Music_Genre_Classification.pdf

[7] https://www.mathworks.com/help/wavelet/ug/music-genre-classification-using-wavelet-scattering.html

[8] http://cs229.stanford.edu/proj2018/report/21.pdf

[9] https://www.kaggle.com/ashishpatel26/feature-extraction-from-audio

[10] https://repository.javeriana.edu.co/bitstream/handle/10554/38692/Francisco%20Andr%c3%a9s%20Reales%20Castro.pdf?sequence=1&isAllowed=y