Reconocimiento sonoro de instrumentos musicales

|  |  |  |
| --- | --- | --- |
| TITULACIÓN:  Máster Inteligencia Artificial  Curso académico:  2021-2022  Lugar de residencia, mes y año: Medellín, Colombia 30-05-2022 | Alumno/a:  Rodriguez Villamizar, William Steve  D.N.I:  1098685961  Director:  Escrig Pérez, Helio | Convocatoria:  Segunda  Orientación:  Virtual  Créditos:  12 ECTS |

Tabla de contenido

[1. Introducción 4](#__RefHeading___Toc680_2582143497)

[1.1. Objetivos 4](#__RefHeading___Toc682_2582143497)

[2. Inteligencia artificial 6](#__RefHeading___Toc2286_2582143497)

[3. Estado del arte 7](#__RefHeading___Toc684_2582143497)

[4. Dataset 8](#__RefHeading___Toc686_2582143497)

[5. Metodología CRISP-DM 10](#__RefHeading___Toc688_2582143497)

[6. Preparación de los datos 12](#__RefHeading___Toc690_2582143497)

[7. Preprocesado de los datos 15](#__RefHeading___Toc2592_2582143497)

[7.1. Extracción de características 15](#__RefHeading___Toc694_2582143497)

[8. PCA 19](#__RefHeading___Toc696_2582143497)

[9. Modelado 22](#__RefHeading___Toc698_2582143497)

[10. Producccion 25](#__RefHeading___Toc700_2582143497)

Indice de tablas

[Tabla 1: Resumen dataset 9](#Tabla!0|sequence)

[Tabla 2: Vector de salida luego de extraer las características del audio 17](#Tabla!1|sequence)

Indice de imagenes

[Imagen 1: Metodología CRISP-DM 10](#Imagen!0|sequence)

[Imagen 2: Cantidad datos según split tiempo a los audios 13](#Imagen!1|sequence)

[Imagen 3: Tamaño vector entrada según split tiempo a audios 13](#Imagen!2|sequence)

[Imagen 4: Fragmento de audio ampliado para identificar los cruces por cero, tomado de [3] 16](#Imagen!3|sequence)

[Imagen 5: Cantidad de información que contiene cada componente 19](#Imagen!4|sequence)

[Imagen 6: Porcentaje de la información para las nuevas componentes 20](#Imagen!5|sequence)

[Imagen 7: Comparación de nuevas componentes vs porcentaje de información 21](#Imagen!6|sequence)

[Imagen 8: Porcentaje Aciertos luego de train según el split en el algoritmo RandomForest (algoritmo elegido solo para generar el grafico) 22](#Imagen!7|sequence)

# Introducción

El presente trabajo tiene como fin tomar un conjunto de datos de composiciones musicales del género clásico y poder clasificar los diferentes instrumentos que intervienen durante la canción, para ello se aplicará la metodología CRISP-DM como base atacar este tipo de problemas de inteligencia artificial, al final se entrenan varios modelos de IA y se comparan los resultados obtenidos.

## **Objetivos**

La finalidad de este proyecto es entrenar un modelo de inteligencia que pueda definir que instrumentos musicales están sonando en un audio, se pretende reconocer 11 instrumentos, estos son:

* Piano
* Violin
* Viola
* Violonchelo
* Clarinete
* Fagot
* Bocina
* Oboe
* Vlauta
* Clave
* Contrabajo

Para ello se usara un dataset de kaggle, una web que contiene confiables y amplios datasets para diversos proyectos de ciencia de datos, big data y machine learning, en especifico se usara la base de datos de musinet1.

A su vez se procederá a aplicar sobre los audios de entrada del modelo una extracción de características, según lo propone Essentia2, en esto solo se elegirán unas características a usar, a fortuna [Sanket Doshi3](https://medium.com/@sdoshi579?source=post_page-----75a3f9bc265d--------------------------------), nos define las características mas relevantes y que en su trabajo dieron importantes resultados, y al igual que El, se usará la librería librosa4, la cual es una librería para extraer características de audios.

Luego de la extracción de características se creara un archivo .csv en el cual se incorporan estas características mencionadas, archivo que se usara para el modelo.

Finalmente se evaluara el modelo y se definirá la confianza, precisión y exactitud del mismo.

# **Inteligencia artificial**

Poner teoria

Que es machine learning

tipos de algoritmos

supervisado

no supervisado

PCA

Cuales son los algoritmos clasicos?

Random Forest

Regresion logistica

arbol de decision

dummy clasificación

k neigbors

Gausian NB

redes neuronales

definicion

funciones de activacion

proceso de aprendizaje

# Estado del arte

Durante una exhaustiva investigación de trabajos relacionados se ha revisado entre otros el trabajo de Aurora Salgado5, donde en su trabajo usa un dataset que contiene audios de instrumentos específicos (violín, piano, etc), cada audio tiene solo un instrumento, pero su proceso de tratamiento de audios es un punto de partida crucial que se nos permite aprovechar su experiencia para nuestro tratamiento de datos.

Seguidamente se estudio el trabajo realizado por la universidad de Washington6, donde recopilaron y clasificaron un amplio dataset (musinet1) y entrenaron varios modelos usando scratch7 y redes neuronales convolucionales, funciones de activación ReLu, maxpooling, entre otras consideraciones logran resultados mas que admirables, pero en particular bastante llamativo como aplicar el PCA a conjunto de datos de tipo audio, PCA que nos viene bien y que usaremos mas adelante dado que en este trabajo demostró resultado no menos que excelentes.

En particular revisando también el trabajo de JUAN SEBASTIÁN MÉNDEZ HERNÁNDEZ8, donde con las características de audio de MFCC, demuestran que esta característica es fundamental para la clasificación de instrumentos, para el cual le prestaremos especial atención a la extracción de esta característica, la cual se explicara a mas detenimiento en capitulo posterior.

Finalmente usaremos lo aprendido por Juan Sebastian Gómez Cañón9, que le en experiencia luego de crear su clasificador de instrumentos musicales usando redes neuronales, confirma el porque a mayor cantidad de datos se obtienen mejores métricas positivas en el entrenamiento y test de modelos de IA, en especial en dataset de audios, experiencia que nos viene bien pues solo tenemos 330 audios (muestras), mientras que en todas las menciones anteriores se contaban con mas de 5,000 muestras, por lo cual mas adelante usaremos una técnica para aumentar la cantidad de muestras.

Cabe resaltar que en temas de IA, el tratamiento de datos no ha sido tan profundizado en temas de audio como lo ha sido vasta-mente investigado para temas de visión por computadora (por poner un ejemplo), sin embargo se ataca el problema teniendo como base el estado del arte y se propone unos ajustes que no aparecen en la documentación original.

# Dataset

Para poder llevar a cabo este trabajo se ha hecho una búsqueda de un dataset que estuviera etiquetado, pero sobre todo que fuera confiable en su fuente para poder usarlo como medio de este trabajo, en esta búsqueda varios trabajos, entre ellos, como el realizado por John Thickstun, Zaid Harchaoui y Sham Kakade6 donde usaron el dataset de musinet y Scratch para enseñar a un modelo a aprender diferentes características de audios, que si bien en este trabajo no se usara scratch o se usara el mismo enfoque, fue suficiente para entender que este es dataset adecuado para este trabajo.

En esto hay una comunidad dedicada a reunir diferentes datasets para diferentes usos de inteligencia artificial (IA para abreviar), en esto se hace referencia a [kaggle](https://www.kaggle.com/datasets) que es una web altamente enriquecida con variedad de paquetes de datos para modelados de diferentes algoritmos.

En kaggle se ha logrado encontrar un conjunto de datos que es preciso al proporcionar un dataset apropiado para este trabajo, este es llamado como el conjunto de datos de [musinet1](https://www.kaggle.com/datasets/imsparsh/musicnet-dataset), el cual está constituido por 330 grabaciones de música clásica con licencia libre, junto con más de 1 millón de etiquetas anotadas que indican el tiempo preciso de cada nota en cada grabación, el instrumento que toca cada nota y la posición de la nota en la estructura métrica de la composición. Las etiquetas se adquieren a partir de partituras musicales alineadas con grabaciones mediantes deformación dinámica del tiempo. Las etiquetas son verificadas por músicos calificados, donde se estima una tasa de error de etiquetado del 4%, a continuación un resumen del dataset:

Tabla 1.

*Resumen datos del dataset musinet*

|  |  |  |
| --- | --- | --- |
| **Compositor** | **cantidad de audios** | **tiempo total (segundos)** |
| Beethoven | 157 | 65149 |
| Bach | 67 | 11041 |
| Schubert | 30 | 15188 |
| Mozart | 24 | 9386 |
| Brahms | 24 | 11531 |
| Cambini | 9 | 2577 |
| Dvorak | 8 | 3343 |
| Ravel | 4 | 1643 |
| Faure | 4 | 1963 |
| Haydn | 3 | 888 |
| **TOTAL** | **330** | **122709** |

Tabla 1: Resumen dataset

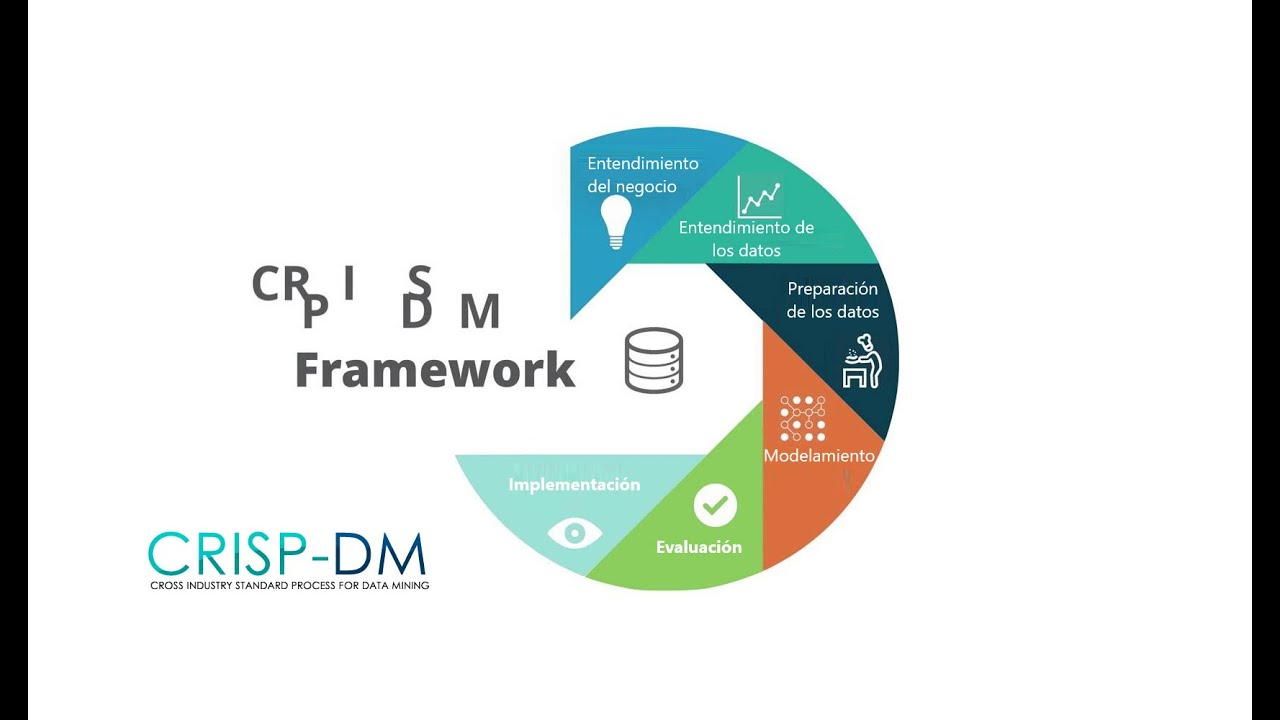
Con esto se puede observar que como valor adicional a la descripción oficial del dataset tenemos 122.709 segundos de grabación, esta es una nota importante pues jugaremos con este dato en el momento de preparar los datos, esto debido a que hasta el momento solo se tienen 330 registros de diferentes autores, que si bien es un numero de composiciones que en si, es un valor importante, para entrenar nuestros futuros modelos se va a incrementar la cantidad de registros con una técnica propuesta en este trabajo para que el modelo pueda generalizar mejor el aprendizaje de los datos.

También se resumen los instrumentos con sus id:

1. Piano
2. Violín
3. Viola
4. Violonchelo
5. Clarinete
6. Fagot
7. Bocina
8. Oboe
9. Flauta
10. Clave
11. Contrabajo

# Metodología CRISP-DM

Antes de empezar a atacar los objetivos de este trabajo vamos a entender el proceso de cómo se va a buscar la solución al problema. Para ello se usará la metodología CRISP-DM, el cual es una forma perfecta para atacar un problema de ciencia de datos para modelados de algoritmos de IA, lo cual viene perfecto para este problema.

Imagen 1: Metodología CRISP-DM

El **entendimiento del negocio** hace referencia a entender los objetivos, entender el problema y tener claro la meta que se deberá alcanzar al final de la implementación, para este trabajo, como se ha mencionado anteriormente se quiere entregar un audio al modelo y que este en respuesta indique que instrumentos musicales han sonado en ese audio.

El **entendimiento de los datos** es un análisis del dataset, revisar cada dato, revisar las etiquetas, ver que datos aparecen con ruido o alguna anomalía que se pueda identificar, etc.

La **preparación de los datos** nos conlleva a trabajar sobre los datos para solucionar lo detectado en el punto anterior, separar datos, hacer incremento de la información con alguna técnica de data-incrementation, normalizar o estandarizar la información, extraer las características de los datos, quizás, hacer reducción de dimensionalidad, extraer los componentes principales, quitar datos null, ausentes y por supuesto convertir a vectores, matrices y tensores los datos (paso crucial pues los modelos trabajan con matrices o tensores).

El **modelado**, es quizás la parte mas llamativa del proceso, acá es donde modificamos el algoritmo, dejamos fluir la creatividad para tratar de crear un modelo de inteligencia artificial que pueda aprender de los datos, se entrena ese modelo, se valida y se pone a prueba las habilidades del programador al momento de diseñar el algoritmo.

La **evaluación**, es usar del dataset el conjunto de datos de test, el cual es una pequeña información aislada del resto para poner a prueba el modelo diseñado y entrenado en el paso anterior, con el fin de entregar al modelo datos nuevos, frescos, desconocidos y ver qué métricas entrega, y así evaluar la eficiencia del mismo, esta evaluación define si el modelo está listo o no para pasar a producción.

**Implementación** es precisamente ese paso a producción cuando el modelo ha demostrado que es capaz de procesar efectivamente nuevos datos y se pone en un entorno de producción, generalmente a través de una API, ya sea en AWS o en un servidor local.

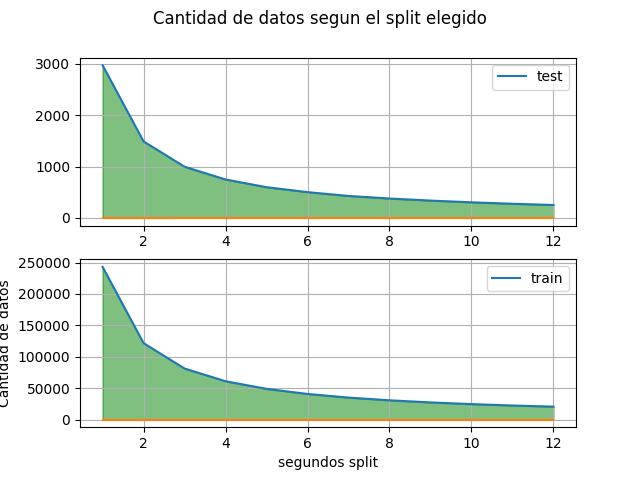
# Preparación de los datos

Ya se ha expuesto que se tienen 330 muestras de sonido, pero esa cantidad de registros es muy pobre para entrenar los algoritmos de aprendizaje supervisado, por ello se presenta una forma de aumentar las muestras mediante un particionado de los datos, es decir, se tomara una canción X y la dividiremos en fragmentos de n segundos

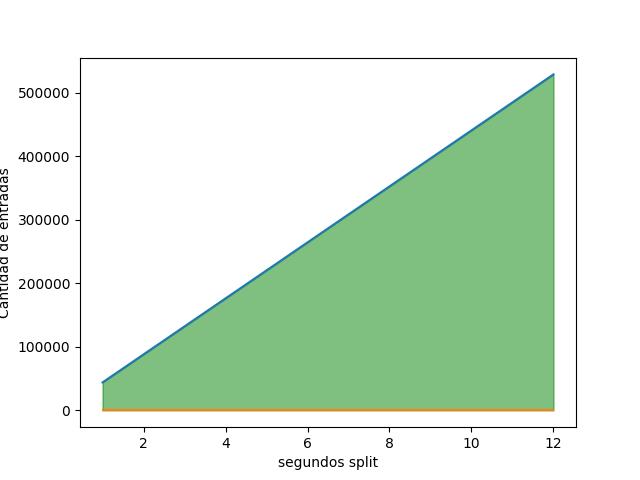
A su vez para cada fragmento de audio se buscara en el correspondiente banco de etiquetas, la etiqueta que corresponda para cada trozo del audio original, por ejemplo, si dividimos en secciones de 2 segundos un audio de 100 segundos se tendrán 50 muestras de audio.

Haciendo esto también se reduce el tamaño del vector que representa el audio, y para cada uno de esos 50 muestras se tendrán 11 salidas etiquetadas, para 11 instrumentos, una por cada etiqueta posible según el dataset, cabe aclarar que para algunas de estas salidas simplemente será cero, es decir, este instrumento no suena en este audio, esto se hace para poder estandarizar la cantidad de salidas igual para todas las muestras.

Al aplicar este proceso a todos los audios se tendrá un incremento de muestras de forma significativa, esto aplica tanto para los datos train como para los datos de test, así:

Imagen 2: Cantidad datos según split tiempo a los audios

Esta división también afecta el tamaño del vector que representa cada audio, que en términos generales representa la misma cantidad de entradas que se podrá usar para el modelo, una relación se vería con el siguiente grafico:

Imagen 3: Tamaño vector entrada según split tiempo a audios

En esto se logra intuir que entre mayor sea el split, mayor tamaño tendrá el vector de entrada del modelo.

En ello se puede ver que al hacer el split pequeño, no solo se aumentan las muestras, sino que también se reduce el tamaño del vector, ahora solo falta definir cual es el valor del split de tiempo que sea mas apropiado para el entrenamiento de los modelos.

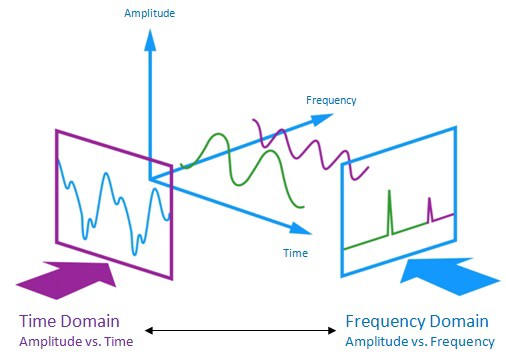
En este punto de este trabajo aun no se definirá el valor del split a usar, esto se hace en el apartado de modelado, por lo que por ahora centraremos la atención en extraer las características de cada fragmento de audio, en esto hay varios métodos para extraer las características de audios.

## 

## **Preprocesado de los datos**

La extracción de características es una parte muy importante para analizar y encontrar relaciones entre diferentes cosas. Los datos proporcionados de audio no pueden ser entendidos por los modelos directamente para convertirlos en un formato comprensible. Se utiliza la extracción de características. Es un proceso que explica la mayor parte de los datos de forma comprensible. La extracción de características es necesaria para los algoritmos de clasificación, predicción y recomendación.

En la literatura de tratamiento de señales, un audio es una señal tridimensional en la que tres ejes representan el tiempo, la amplitud y la frecuencia.



En los siguientes puntos de este capitulo se va a profundizar en diferentes características que se pueden extraer a un audio

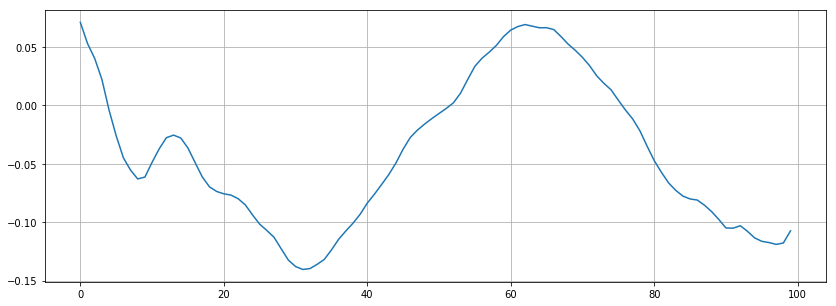
## **Extracción de características**

Se usara Librosa, una librería de python para analizar y extraer características de una señal de audio

### Taza cruce por cero

Una señal de audio es una variación donde la señal pasa varias veces entre valores positivos y negativos, estos cambios es lo que define cada tonalidad e intensidad del sonido, esta es la tasa de cambio de signo a lo largo de una señal, es decir cambia de positiva a negativa y viceversa, esto es muy usado para el reconocimiento de voz, a gran percusión en valor es mayor, como es el caso del metal o rock.

Por ejemplo, para el siguiente audio se puede identificar que hay tres cruces por cero

Imagen 4: Fragmento de audio ampliado para identificar los cruces por cero, tomado de 3

### Centroide espectral

Este inicia done se encuentra el “centro de masa”[[1]](#footnote-2) de un sonido, y se calcula como la media ponderada de las frecuencias presentes en el sonido. Si no hay variación de frecuencias entonces el centroide espectral estaría alrededor de un centro. [10](https://es.wikipedia.org/wiki/Masa_de_sonido)

### Reducción espectral

Es la frecuencia por debajo de la cual se encuentra un porcentaje especifico de energía espectral.

### RMS

Es el valor de la raíz cuadrada media (RMS) para cada cuadro, ya sea de las muestras de audio y/o de un espectrograma, que se emplea para conocer la media aproximada de potencia 11.

### croma

Se define el espectro de un sonido como la representación de la distribución de energía sonora de dicho sonido en función de la frecuencia. El espectro es importante porque la percepción auditiva del sonido es de naturaleza predominantemente espectral.12

### ancho de banda

El ancho de banda espectral es la longitud de la extensión de frecuencias, medida en hercios (Hz), en la que se concentra la mayor potencia de la señal13.

### MFCC — Coeficientes centrales de frecuencia Mel

Esta característica es uno de los métodos más importantes para extraer una característica de una señal de audio y se usa principalmente cuando se trabaja con señales de audio. Los coeficientes centrales de frecuencia de MEL (MFCC) de una señal son un pequeño conjunto de características (generalmente entre 10 y 20) que describen de manera concisa la forma general de una envolvente espectral14.

### Resumen extracción

Con esto ya se ha extraído las características de la señal musical.

Para resumir: para cada audio se tendrá el siguiente vector de respuesta:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **croma** | **rms** | **centroide espectral** | **ancho banda** | **reducción espectral** | **cruces por cero** | **MFCC** |
| 1 valor | 1 valor | 1 valor | 1 valor | 1 valor | 1 valor | 20 valores |

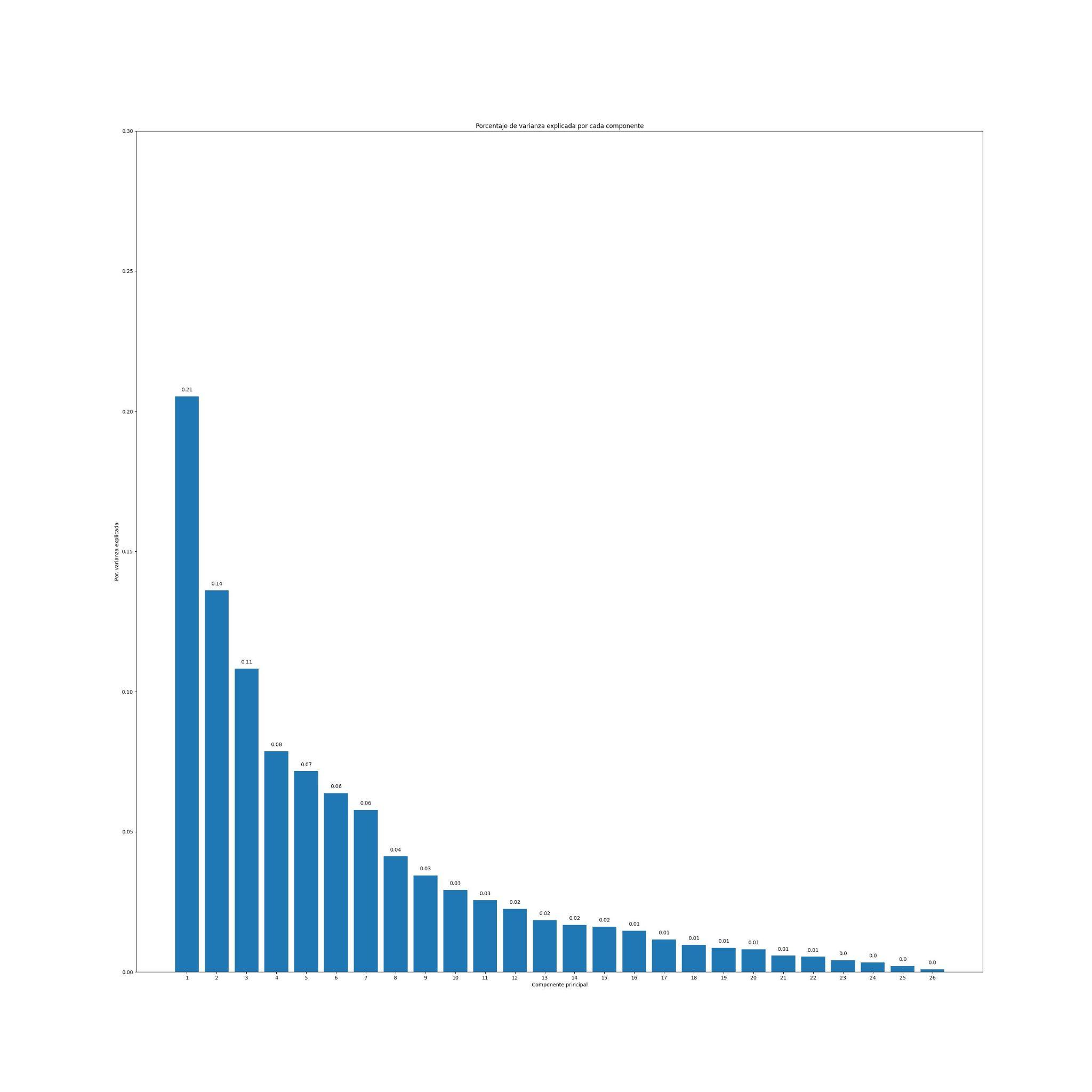
Tabla 2: Vector de salida luego de extraer las características del audio

Es decir para cada audio, sin importar el tamaño del vector de entrada, se convertirá luego de la extracción de características en un vector de tamaño 26 valores.

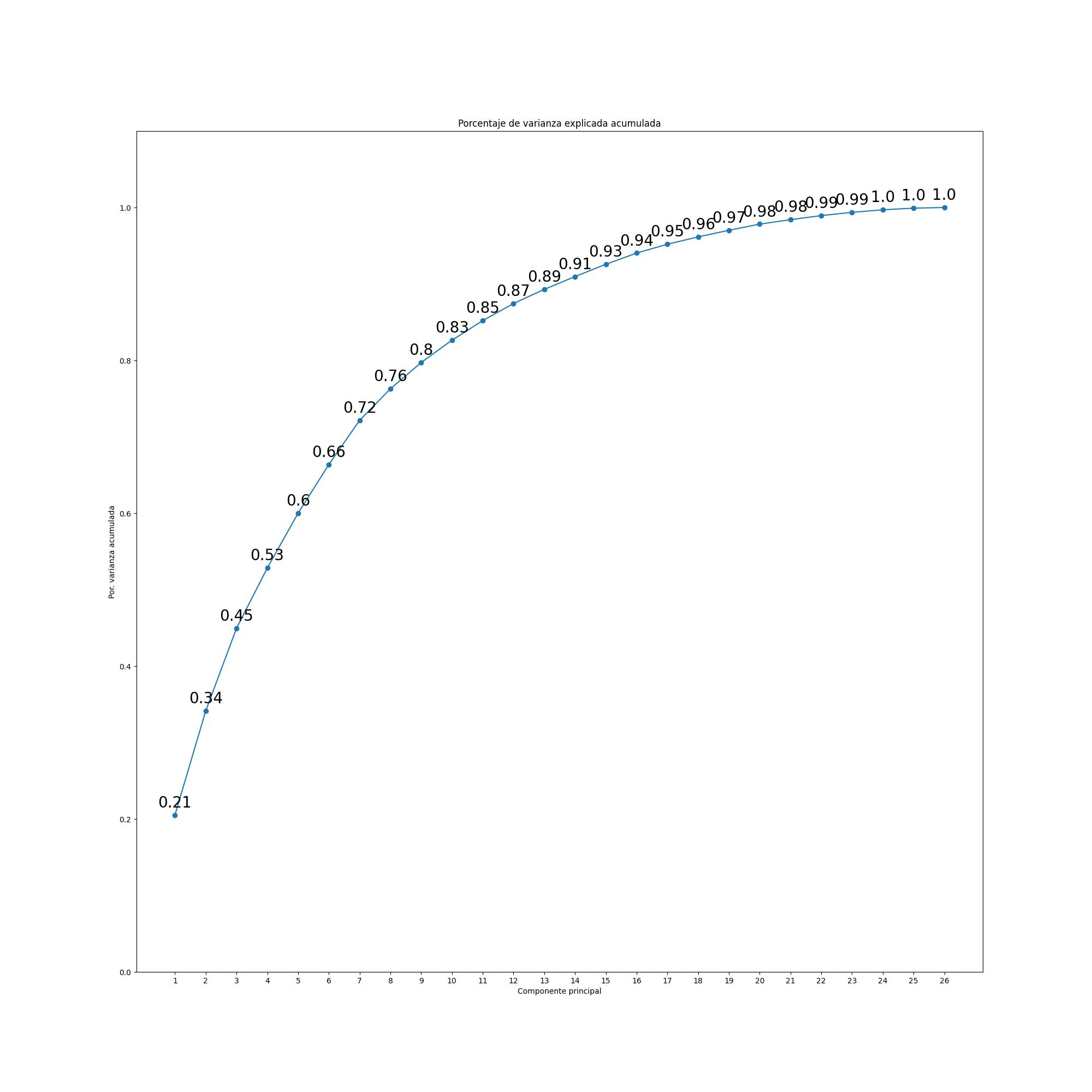
Como el tamaño del vector de entrada no define el tamaño del vector de salida (resultado de la extracción de características), entonces ya el tamaño del vector no es un factor que tenga preocupación para entrenar el modelo, lo único aún por resolver es la cantidad de datos, tema que se solucionará en el modelado.

## **PCA**

Luego de extraer las características a todos los audios se tendrán 26 características, pero para fines prácticos, aun es un vector algo amplio, por lo que se realizó sobre los datos un análisis de cada una de esas características, también llamado componentes, dando el siguiente resultado:

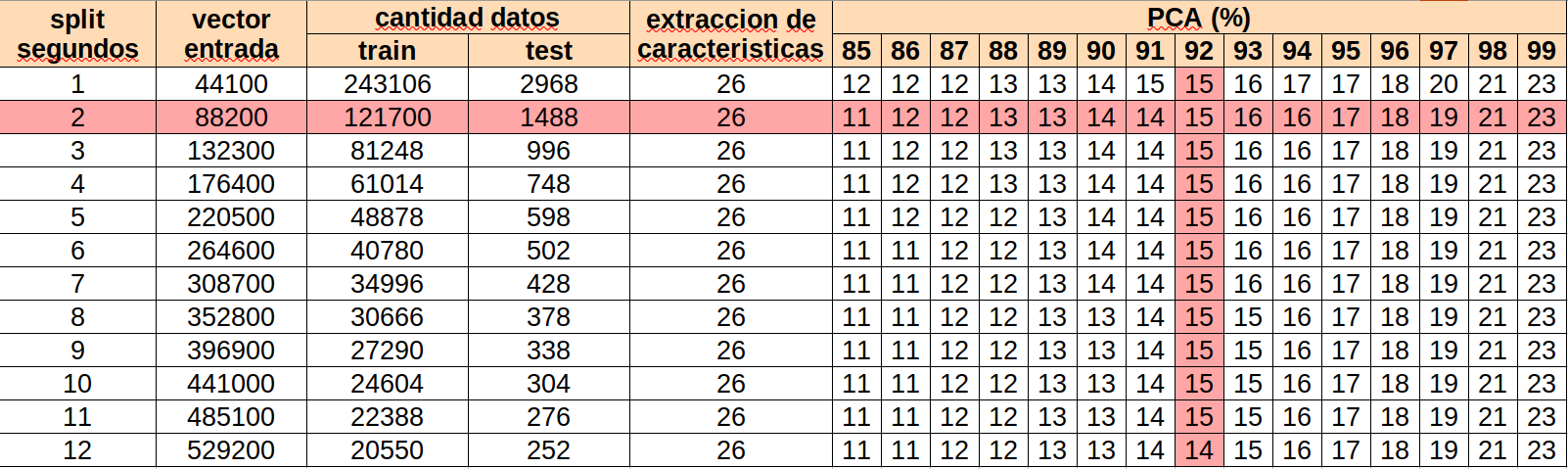
Imagen 5: Cantidad de información que contiene cada componente

Como se puede observar, las ultimas componentes contienen muy poco de la información original, así que con el PCA, evaluemos que valor seria mas apropiado como nueva cantidad de componentes, en pocas palabras el vector 26 posiciones lo vamos a reducir a un valor menor, por ejemplo 21 posiciones, de esta manera al modelo que se entrene le resultara mas fácil aprender la generalidad de los datos y dará mejores métricas positivas en la evaluación del modelo, para ello se debe sacrificar un poco de la información, buscando que sea muy poco o mantener al menos un 90% o mas de la información, pero cual seria el porcentaje de la información que se mantendría luego de una reducción de dimensionalidad, para facilitar esta decisión se ha construido el siguiente grafico:

Imagen 6: Porcentaje de la información para las nuevas componentes

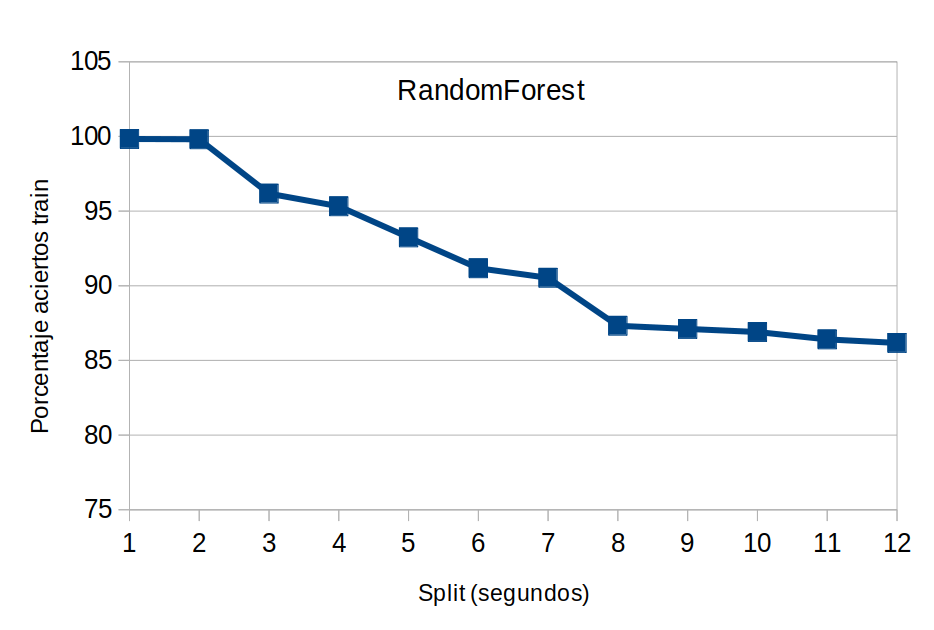
Con este análisis de resultados de reducción de dimensionalidad, se puede observar que al mantener las 15 primeras características se puede conservar cerca del 93% del total de la información, 15 nuevas componentes diferentes a las componentes originales, dicho de otra manera: al reducir 42% el tamaño del vector de 26 posiciones (tamaño original=26, tamaño luego de PCA=15), puedo mantener el 93% de la información, es decir, sacrificando un aproximado al 7% de la información puedo reducir la cantidad de entradas a casi la mitad, lo que es un buen negocio, y que va a beneficiar exponencialmente al modelo a entrenar.

En resumen, se tendría lo siguiente:

Imagen 7: Comparación de nuevas componentes vs porcentaje de información

# Modelado

Ahora ha llegado el momento de tomar la decisión crucial que va a definir cual es el split a usar, para ello no se uso una decisión aleatoria o una de alguna bibliografía debido a que esta división para incrementar los datos surgió durante conversación con el tutor, por lo que se uso el método de ensayo y error, anteriormente en la imagen 7, se mostró la posibilidad de usar de 1 a 12 como valor de segundos para el split, para ello se muestra el siguiente gráfico que resume un entrenamiento para cada split:

Imagen 8: Porcentaje Aciertos luego de train según el split en el algoritmo RandomForest (algoritmo elegido solo para generar el grafico)

El anterior grafico es el resultado de varios entrenamientos, para ello, para cada audio se ha hecho cada split expuesto y cada vector resultante se aplicó la extracción de características y PCA (vector de 15 posiciones resultante para cada audio), luego se creó un archivo csv que contenía en las filas cada vector de cada audio, con esto, se realizo un train usando el algoritmo RandomForest (con hiperparametros por default) con cada archivo de cada split y con los resultados de cada entrenamiento se genero el grafico anterior, esto para determinar cual split era el mas adecuado.

Se puede observar que hay dos puntos de inflexión, el primero es el split a 7 segundos donde se obtiene un porcentaje de aciertos de 90,53% y otro punto de inflexión en el split a 2 segundos donde se logra un porcentaje de aciertos de 98,6%.

Teniendo 330 audios iniciales en un total de 122,709 segundos, se ha decidido en partir cada audio en mini-audios de 2 segundos, donde teníamos un porcentaje de aciertos superior al 95% (por ello se eligen los dos segundos) y cantidad necesaria de datos para lograr un porcentaje mayor, con esto, las 330 muestras se convierten en 121,700 muestras, lo que le da un valor de conveniencia para lograr en el entrenamiento del modelo una generalidad de los datos bastante alta.

Cada dato sera un vector de 88200 valores, que seguido de la extracción de características se tendrán 26 valores y finalmente luego del PCA se tendrá 15 valores, es decir, para el entrenamiento, se tendrá una matriz de 15 columnas y 121,700 muestras, con esto ahora ya se puede pasar a entrenar el modelo de inteligencia artificial.

Hasta ahora no se ha hablado de accuracy, Recall u otros, esto debido a que sciki-learn para las salidas multiclase no dispone de unas métricas que se puedan usar con facilidad, ni siquiera la matriz de confusión (para multiclase), por ello se creo una función que contabiliza la cantidad de aciertos (cuando la predicción es clase X y la salida etiquetada es también clase X) y los no aciertos como falsos, entonces se tiene una balanza donde en un extremo están los aciertos y en el otro las equivocaciones, función que devuelve el valor en el lado de los aciertos en forma de porcentaje.

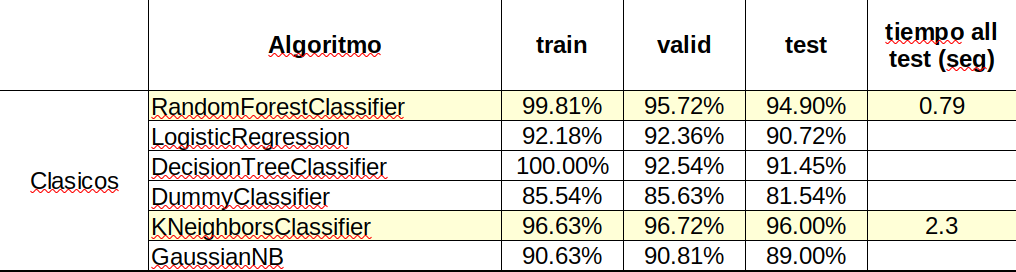
Kaggle en sus datasets siempre entrega dividido el set de train y el set de test, pero a mi gusto, siempre que trabajo con kaggle me gusta tener mi propio set de test para que el test dado por kaggle sean mis datos que validen mi pre-producción, en esto mi set de train le realizo un split de datos dejando el 80% como mínimo para el train el 20% (generalmente) para validación y el set de datos que kaggle destina para test, usarlo para el test, siendo así se tiene (con split a 2 segundos):

* Train: 102,160 muestras
* Valid: 24,340 muestras
* Test: 744 muestras (dadas por kaggle)

\*\*\*\*\*\*\*\*\*\*\* Explicar los hiperparametros usados

\*\*\*\*\*\*\*\*\*\*\* Explicar porque se usa cada uno de los algoritmos

Para entrenar el modelo se ha optado por usar algoritmos tanto clásico de machine learning como de deep learning, obteniéndose los siguientes resultados para los clásicos:



Notándose unas importantes métricas, las cuales se han resaltado para facilitar su deduccion, se resalta aun mas que se ha conseguido en el test, se hace relacion especialmente al test debido a que es la verdadera evaluacion del modelo con datos desconocidos en el entrenamiento, resultados por encima del 94% en el test.

# Producccion

1. Referencias

Bibliography

1: kaggle, MusicNet Dataset, 2020, https://www.kaggle.com/datasets/imsparsh/musicnet-dataset

2: Music Technology Group - Universitat Pompeu Fabra, music extractor features, 2021

3: Sanket Doshi, Music Feature Extraction in Python, 2018, https://towardsdatascience.com/extract-features-of-music-75a3f9bc265d

4: McFee, Brian, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg y Oriol Nieto, librosa: análisis de señales de audio y música en python, 2015

5: Aurora Salgado Díaz del Río, Reconocimiento automático de instrumentos  
mediante aprendizaje máquina, 2019, https://biblus.us.es/bibing/proyectos/abreproy/92353/descargar\_fichero/TFG-2353-SALGADO.pdf

6: John Thickstun, Zaid Harchaoui & Sham M. Kakade, LEARNING FEATURES OF MUSIC FROM SCRATCH, 2017

7: MIT, NSF, , SFE, Lego, scratch, , https://scratch.mit.edu/

8: JUAN SEBASTIÁN MÉNDEZ HERNÁNDEZ, IDENTIFICACIÓN DE INSTRUMENTOS MUSICALES A PARTIR DEL  
ALGORITMO MFCC, 2020

9: Juan Sebasti ́an G ́omez Ca ̃n ́, Automatic Instrument Recognition using DeepConvolutional Neural Networks, 2018

10: , Masa sonido, , https://es.wikipedia.org/wiki/Masa\_de\_sonido

11: , Rmse audios, , https://www.euronics.es/blog/rms/#:~:text=RMS%20(Root%20Mean%20Squared)%20o,calidad%20del%20sonido%20que%20escucharemos)

12: , , , https://www.granada.org/inet/sonidos.nsf/d483b298c3f6a1b9c1257cdd00384c53/3fdfc36a7489b607c1257cde0024bb34!OpenDocument#:~:text=El%20espectrograma%20es%20una%20representaci%C3%B3n,representa%20en%20el%20eje%20horizontal.

13: , , , https://www.studio-22.com/blog/enciclopedia/ancho-de-banda#:~:text=Ancho%20de%20banda%20%2D%20Referido%20al,es%20mayor%20a%203%20dB)

14: , , , https://towardsdatascience.com/extract-features-of-music-75a3f9bc265d

1. [↑](#footnote-ref-2)