Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

Máquinas de aprendizaje

Reporte: Proyecto final

Docente: Abraham Sánchez López

Alumno	Matrícula
Taisen Romero Bañuelos	202055209

Music Genre Classification

Nuestro trabajo es clasificar canciones en “x” género según sus características. Mi primera impresión sobre los datasets es que antes que nada había que entender qué significaban algunas variables. Class es la variable que nos indica el género, sin embargo, no tenemos una lista que asocie el número al género, por lo que habrá que hacer una lista para asociar el número de la clase a un género. También, no tengo muy claro qué significan las variables key, mode, valence y time_signature. Tampoco tengo certeza sobre cómo se miden las escalas de loudness, instrumentales entre otras.

```
> head(train)
            Artist.Name                          Track.Name Popularity danceability energy key loudness mode speechiness acousticness
1     Bruno Mars That's What I Like (feat. Gucci Mane)         60        0.854  0.564   1   -4.964    1       0.0485      0.017100
2                Boston                        Hitch a Ride         54        0.382  0.814   3   -7.230    1       0.0406      0.001100
3          The Raincoats                   No Side to Fall In         35        0.434  0.614   6   -8.334    1       0.0525      0.486000
4                   Deno          Lingo (feat. J.I & Chunkz)         66        0.853  0.597  10   -6.528    0       0.0555      0.021200
5 Red Hot Chili Peppers     Nobody Weird Like Me - Remastered         53        0.167  0.975   2   -4.279    1       0.2160      0.000169
6           The Stooges     Search and Destroy - Iggy Pop Mix         53        0.235  0.977   6    0.878    1       0.1070      0.003530
  instrumentalness liveness valence   tempo duration_in.min.ms time_signature Class
1               NA   0.0849  0.8990 134.071            234596              4     5
2         0.004010   0.1010  0.5690 116.454            251733              4    10
3         0.000196   0.3940  0.7870 147.681            109667              4     6
4               NA   0.1220  0.5690 107.033            173968              4     5
5         0.016100   0.1720  0.0918 199.060            229960              4    10
6         0.006040   0.1720  0.2410 152.952            208133              4     6
```

Y también veo necesario convertir las medidas de tiempo a un formato más entendible.

duration_in.min.ms	time_signature
2.345960e+05	4
2.517330e+05	4
1.096670e+05	4
1.739680e+05	4
2.299600e+05	4

Entonces, así quedaría la información actualizada (para no tener ni una duda de qué significan los datos):

Columna	Descripción	Tipo/Rango	Unidad/Escala
Artist.Name	Nombre del artista o banda.	Txt	—
Track.Name	Título de la canción.	Txt	—
Popularity	Popularidad de la canción en Spotify. Basado en reproducciones y reciente actividad.	Numérica (0–100)	Entero
danceability	Medida de qué tan bailable es la canción.	Numérica (0–1)	Proporción

energy	Medida de intensidad y actividad.	Numérica (0–1)	Proporción
key	Tono musical, representado como número entero (C=0, C#/Db=1, ..., B=11). La escala musical.	Entero (0–11)	Nota musical codificada. C=0 C♯/D ♭ =1 D=2 D♯/E ♭ =3 E=4 F=5 F♯/G ♭ =6 G=7 G♯/A ♭ =8 A=9 A♯/B ♭ =10 B=11
loudness	Volumen promedio en decibelios (dB). Valores negativos = más silencioso.	Numérica (negativo) **Posible conversión a positivos si da mejor interpretabilidad**	Decibelios (dB)
mode	Modo de la canción: mayor (1) o menor (0).	Binaria (0 o 1)	—
speechiness	Presencia de palabras habladas.	Numérica (0–1)	Proporción
acousticness	Probabilidad de que sea una pista acústica. O qué tan acústica es la canción.	Numérica (0–1)	Proporción
instrumentalness	Predicción de que no haya voz. Valores altos → instrumental.	Numérica (0–1)	Proporción
liveness	Probabilidad de que sea una grabación en vivo.	Numérica (0–1)	Proporción
valence	Medida de positividad de la canción. Valores altos → más alegre.	Numérica (0–1)	Proporción
tempo	Velocidad de la canción. Lo que marca el ritmo.	Numérica (> 0)	BPM (beats por minuto)
duration_in.min.ms	Duración de la canción.	Numérica (> 0)	Milisegundos. **Posible conversión a segundos**
time_signature	Compás musical (por	Entero (usualmente	—

	ejemplo 3/4 o 4/4).	3–5)	
Class	Género musical (valor entre 0 y 10).	Entero (0–10)	Categoría (variable objetivo)

Class value	Género musical
0	Acoustic/Folk
1	Alt_Music
2	Blues
3	Bollywood
4	Country
5	HipHop
6	Indie/Alt
7	Instrumental
8	Metal
9	Pop
10	Rock

Ahora, hay un detalle importante a considerar, el dataset submission contiene las predicciones de las canciones de test. Este dataset es particularmente problemático porque cuenta con observaciones sin clasificar, pues, tienen cero en todas sus celdas (están con la notación one-holt encoding).

	Acoustic.Folk_0	Alt_Music_1	Blues_2	Bollywood_3	Country_4	HipHop_5	Indie.Alt_6	Instrumental_7	Metal_8	Pop_9	Rock_10
13	0	0	0	0	0	0	0	0	0	0	0

```
> submission[rowSums(submission) == 0, ]
    Acoustic.Folk_0 Alt_Music_1 Blues_2 Bollywood_3 Country_4 HipHop_5 Indie.Alt_6 Instrumental_7 Metal_8 Pop_9 Rock_10
12                0           0       0           0         0        0          0              0       0      0       0
13                0           0       0           0         0        0          0              0       0      0       0
14                0           0       0           0         0        0          0              0       0      0       0
15                0           0       0           0         0        0          0              0       0      0       0
16                0           0       0           0         0        0          0              0       0      0       0
17                0           0       0           0         0        0          0              0       0      0       0
18                0           0       0           0         0        0          0              0       0      0       0
19                0           0       0           0         0        0          0              0       0      0       0
20                0           0       0           0         0        0          0              0       0      0       0
21                0           0       0           0         0        0          0              0       0      0       0
22                0           0       0           0         0        0          0              0       0      0       0
23                0           0       0           0         0        0          0              0       0      0       0
24                0           0       0           0         0        0          0              0       0      0       0
```

En total, hay 7,702 canciones sin clasificar.

```
> filas_ceros
[1] 7702
```

Aquí hay varias opciones para empezar a trabajar, podríamos tratar este dataset considerando esas observaciones como datos nulos y así aplicar técnicas como los datos sintéticos o podemos dejarlo un poco aparte para al final compararlo con mis predicciones. Personalmente, me siento más interesado en la segunda opción, aunque no por ello no voy a analizar el dataset de submission, ya que los otros datasets contienen muchos valores nulos, por lo que considero importante hacer un análisis de estos datos nulos y descubrir si tienen algo que ver en los datos no clasificados de submission.

Mi primera impresión al comparar el número de datos nulos es que no hay una aparente correlación directa dadas las diferencias del conteo (consideraré como datos nulos las observaciones de submission que no tienen clasificación). De hecho, podemos notar que la gran mayoría de observaciones de submission son observaciones "nulas". Esto me hace pensar que debe haber una razón por la cual tantas observaciones no fueron clasificadas, pero quizás sea más interesante preguntarse por qué la minoría si fueron clasificadas, y no solo eso, si la clasificación fue correcta.

```
> str(submission)
'data.frame':   7713 obs. of  11 variables:
```

```
> sum(is.na(submission))
[1] 0
> filas_ceros
[1] 7702
> sum(is.na(train))
[1] 6819
> sum(is.na(test))
[1] 2944
```

Para empezar, veamos cuántas observaciones son válidas y cuántas no

```
> table(rowSums(submission))  #Cuántas filas tienen todo ceros o un 1

   0    1
7702   11
```

Como suponíamos, la gran mayoría de observaciones son inválidas, pero ¿Qué tienen de especial estas 11 canciones que sí fueron clasificadas correctamente por el modelo, en comparación con las otras 7702 que no lo fueron? además, ¿Fue correcta la clasificación? Para responder lo segundo necesitaríamos saber la clase verdadera de las once canciones, lo cual no está en el dataset "test" (que es el dataset a partir del cual se hicieron las predicciones). Pero también podemos revisar si las canciones

tienen características particulares que las hacen fáciles de clasificar, así que hagamos un análisis de ello.

```
> summary(test_clasificadas)
 Artist.Name         Track.Name          Popularity     danceability        energy            key           loudness           mode       
 Length:11          Length:11          Min.   :20.00   Min.   :0.3100   Min.   :0.4030   Min.   : 1.000   Min.   :-13.664   Min.   :0.0000  
 Class :character   Class :character   1st Qu.:35.00   1st Qu.:0.3630   1st Qu.:0.6730   1st Qu.: 7.000   1st Qu.: -7.112   1st Qu.:0.5000  
 Mode  :character   Mode  :character   Median :56.00   Median :0.4300   Median :0.7420   Median : 7.000   Median : -5.059   Median :1.0000  
                                       Mean   :51.36   Mean   :0.4862   Mean   :0.7659   Mean   : 6.889   Mean   : -6.507   Mean   :0.7273  
                                       3rd Qu.:66.50   3rd Qu.:0.6210   3rd Qu.:0.9355   3rd Qu.:10.000   3rd Qu.: -4.833   3rd Qu.:1.0000  
                                       Max.   :80.00   Max.   :0.7000   Max.   :0.9790   Max.   :11.000   Max.   : -3.008   Max.   :1.0000  
                                                                                         NA's   :2                                          
  speechiness       acousticness      instrumentalness       liveness         valence           tempo        duration_in.min.ms time_signature
 Min.   :0.02650   Min.   :0.000166   Min.   :0.0000087   Min.   :0.0289   Min.   :0.1320   Min.   : 74.03   Min.   :     4.1   Min.   :4     
 1st Qu.:0.04520   1st Qu.:0.001895   1st Qu.:0.0000180   1st Qu.:0.1095   1st Qu.:0.3390   1st Qu.:102.39   1st Qu.:187826.5   1st Qu.:4     
 Median :0.06040   Median :0.004210   Median :0.0000439   Median :0.1490   Median :0.5060   Median :129.98   Median :220413.0   Median :4     
 Mean   :0.07853   Mean   :0.058657   Mean   :0.0919258   Mean   :0.1602   Mean   :0.4957   Mean   :126.39   Mean   :211290.6   Mean   :4     
 3rd Qu.:0.08760   3rd Qu.:0.101800   3rd Qu.:0.0090400   3rd Qu.:0.2130   3rd Qu.:0.6210   3rd Qu.:145.92   3rd Qu.:241927.5   3rd Qu.:4     
 Max.   :0.21200   Max.   :0.218000   Max.   :0.7680000   Max.   :0.3150   Max.   :0.9620   Max.   :173.83   Max.   :348859.0   Max.   :4     
                                       NA's   :2                                                                                              
```

Aquí podemos observar que, en promedio, las canciones presentan una popularidad moderadamente alta (≈51), superior a la media general, lo que sugiere que el modelo que usaron tuvo mayor confianza en canciones conocidas. Además, también destacan por valores elevados de energía (≈0.77) y tempo (≈126 BPM), atributos que indican intensidad y ritmo marcado (podría significar alguna preferencia por algún género en específico, pero es apresurado para concluir eso). Además, la mayoría muestra valores muy bajos de instrumentalness, salvo una excepción que claramente es instrumental, lo que podría significar que el modelo distingue mejor las canciones con voz, y también tuvo una "preferencia" por las canciones con un compás de 4/4, pero creo que esto más bien es el reflejo de un sesgo en los datos ya que la mayoría de las canciones están hechas en 4/4. En fin, al parecer el modelo solo etiquetó aquellas canciones con patrones fuertes, ignorando la mayoría con características más neutras o difusas. Esto me hace pensar que el modelo que generó las predicciones de submission fue un modelo muy conservador que solo clasifica observaciones con alta certeza (aquellas que presentan patrones muy claros o extremos), pero creo que hay que hacer un análisis más antes de continuar.

```
> submission[rowSums(submission) == 1, ] #Ver a qué géneros corresponden
   Acoustic.Folk_0 Alt_Music_1 Blues_2 Bollywood_3 Country_4 HipHop_5 Indie.Alt_6 Instrumental_7 Metal_8 Pop_9 Rock_10
1                1           0       0           0         0        0           0              0       0     0       0
2                0           1       0           0         0        0           0              0       0     0       0
3                0           0       1           0         0        0           0              0       0     0       0
4                0           0       0           1         0        0           0              0       0     0       0
5                0           0       0           0         1        0           0              0       0     0       0
6                0           0       0           0         0        1           0              0       0     0       0
7                0           0       0           0         0        0           1              0       0     0       0
8                0           0       0           0         0        0           0              1       0     0       0
9                0           0       0           0         0        0           0              0       1     0       0
10               0           0       0           0         0        0           0              0       0     1       0
11               0           0       0           0         0        0           0              0       0     0       1
```

Al observar a qué género pertenece cada canción me di cuenta que estas once observaciones simulan una matriz identidad, lo que significa que **el modelo sólo identificó una canción por género**. De hecho, ahora tiene más sentido por qué sólo once observaciones fueron válidas, pues, este número coincide con el número de géneros. Sin duda alguna esto es **un detalle revelador** ya que sugiere que el modelo no solo fue extremadamente conservador al clasificar, sino que además parece haber entregado una sola predicción por clase, como si buscara representar cada género con

su ejemplo más claro o extremo. Por tanto, más que un error de datos, este comportamiento apunta a una limitación estructural del modelo o del código que generó las predicciones: priorizó exactitud extrema sobre cobertura, lo que resultó en una clasificación mínima. Este detalle es importante no sólo por lo revelador que es por sí mismo, sino que también es una razón contundente para no centrarme en el manejo de este dataset. De hecho, pensándolo con detenimiento, quizás la persona que hizo ese modelo lo hizo así para garantizar no equivocarse en sus predicciones y de esa forma ganar el concurso en el que participaba. Si esto es así, hice ingeniería inversa sin querer gracias a hacer las preguntas adecuadas.

Muy bien, una vez resuelto este acertijo pasemos al KDD de nuestros datos.

Knowledge Discovery in Databases (KDD)

Para nuestro KDD y el EDA tenemos la opción de juntar los datos de entrenamiento y prueba para tratarlos y analizarlos como un conjunto y después los volvemos a separar respetando sus proporciones originales, o bien podríamos tratar cada uno de forma independiente, pero tratándose del análisis de datos veo necesario juntar todos los datos para un análisis más profundo. Esta fusión se justifica porque, aunque el conjunto de prueba no incluye la variable objetivo Class, sí contiene la mayoría de las características necesarias para detectar tendencias generales, clusters naturales o relaciones entre variables acústicas que podrían ser útiles tanto para predicción como para comprensión musical.

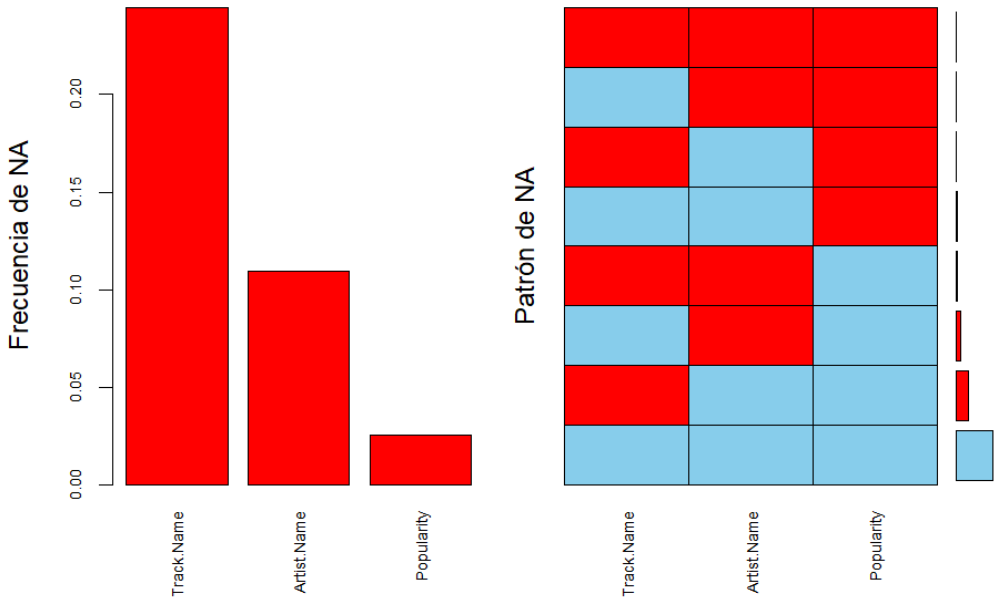
Bueno, al inicio del KDD no hay mucho que comentar, uní los datasets y convertí algunas variables a factor. Lo interesante empezó al decidir qué hacer con los valores faltantes (NA).

```
> #Ver proporción de NAs
> sum(is.na(datos_completos))
[1] 17476
> colSums(is.na(datos_completos))
       Artist.Name         Track.Name         Popularity       danceability             energy                key           loudness 
                 0                  0                655                  0                  0               2822                  0 
              mode        speechiness       acousticness   instrumentalness           liveness            valence              tempo 
                 0                  0                  0               6286                  0                  0                  0 
duration_in.min.ms     time_signature              Class             source 
                 0                  0               7713                  0 
```

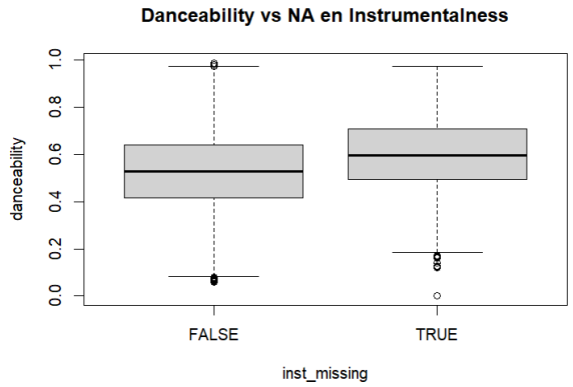
Los NA de Class no deben generar preocupación ya que nosotros los pusimos debido a que test no tenía dicha variable. Pero respecto a las demás variables podemos notar que instrumentalness tiene el porcentaje de NA's más alto (24.45%). Este porcentaje es alto para tratarse de valores nulos, por lo que usar una imputación simple podría no ser la mejor opción. Sería interesante probar el uso de datos sintéticos, pero creo que me reservaré ese gusto para la parte de mejora del modelo para observar qué impacto tiene en las predicciones (en parte porque la mayoría de variables con NA's tienen una

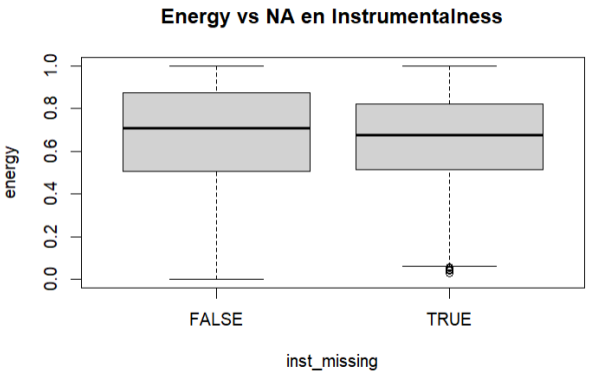
baja proporción de datos faltantes con respecto al total, por lo que sería más prudente usar técnicas que se ajusten mejor a estas proporciones como la imputación). Pero antes de que pensemos qué método usaremos para tratar los datos faltantes, mejor descubramos qué tipo de datos faltantes representa cada variable; si se trata de datos faltantes MCAR (Missing completely at random), MAR (Missing at random) o MNAR (Missing not at random). Después, con base en los resultados obtenidos podríamos elegir qué método es más adecuado.

Para esta labor empecé creando un gráfico que nos muestra la frecuencia y el patrón de valores faltantes en las variables Track.Name, Artist.Name y Popularity usando la función aggr() del paquete VIM. Las gráficas de barras de la izquierda nos indican lo que ya sabíamos, la proporción de NA's por variable. En las celdas de la derecha (patrones de NA), el color rojo indica presencia de NA, y el azul indica valores disponibles. Se observan varias combinaciones de ausencias, como que muchas filas tienen ambos campos textuales faltantes (Track.Name y Artist.Name). Otras combinan la falta de campos textuales con ausencia de Popularity. Solo un pequeño subconjunto tiene los tres valores faltantes simultáneamente. Estos patrones indican que la ausencia de datos no es completamente aleatoria (no-MCAR) y sugiere dependencias entre las variables faltantes, especialmente entre Track.Name y Artist.Name. Esto ya nos indica la necesidad de emplear métodos de imputación que consideren la relación entre variables, como missForest.



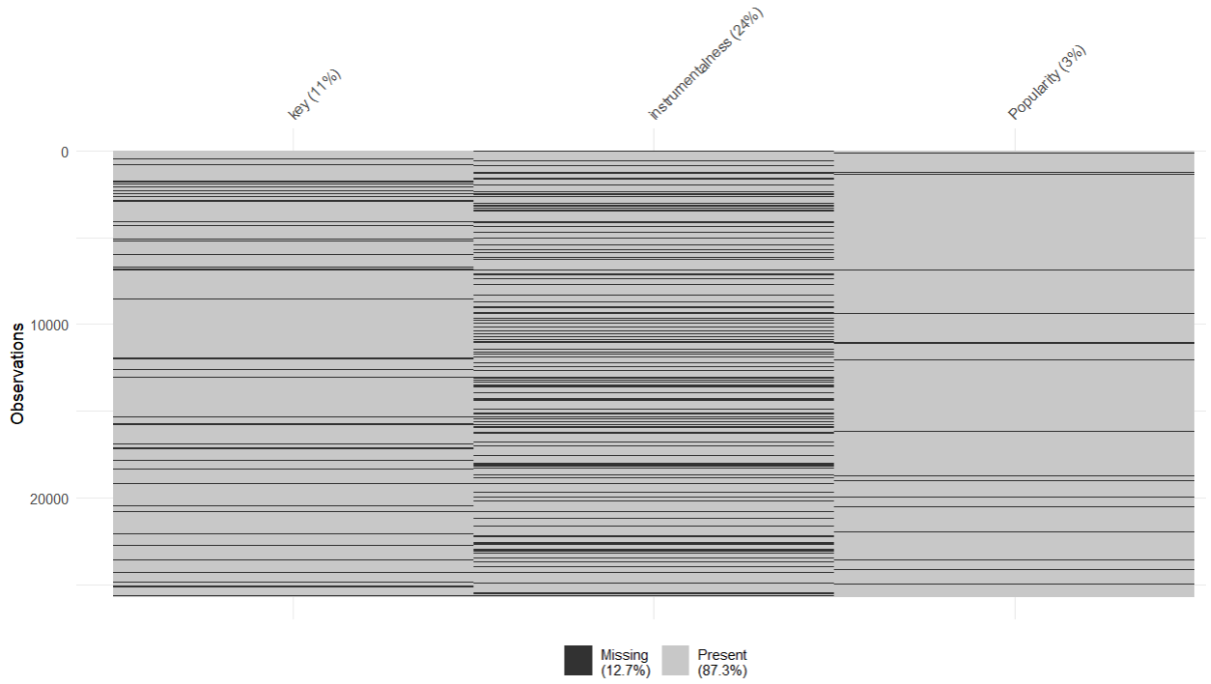
El siguiente análisis de los valores faltantes es mediante la generación de dos boxplots. A primera vista notamos la presencia de **outliers** (valores atípicos), por lo que probablemente tengamos que hacer algo al respecto como parte del KDD, pero de momento continuemos con el tratamiento de datos faltantes.





Los boxplots comparan la distribución de dos variables numéricas (danceability y energy) según la presencia o ausencia de datos faltantes en instrumentalness. Para ello se creó una variable binaria inst_missing, que indica si una fila tiene (TRUE) o no (FALSE) un valor NA en instrumentalness. Entonces, considerando esto, ¿Qué podemos deducir a partir de lo que vemos en los boxplots?, pues que las canciones con instrumentalness faltante (grupo TRUE) tienden a tener una mayor mediana de danceability que aquellas con valores presentes. Además, el rango intercuartílico es ligeramente más alto, lo que indica más dispersión en la capacidad de baile cuando instrumentalness no está disponible. Esta diferencia sugiere que la falta de datos podría ser **no aleatoria**, quizá instrumentalness tiende a faltar en canciones más bailables (como canciones con voz dominante). Y con respecto al segundo boxplot podemos decir algo similar ya que ambas distribuciones (Energy vs NA en instrumentalness) son relativamente similares, pero aún así hay una ligera reducción en la mediana de energía en el grupo con instrumentalness faltante. Esto respalda la idea de que los datos son MNAR o MAR, y por ende se deben imputar con métodos que consideren múltiples variables (otro punto para missForest).

Finalmente, usé otra librería (naniar) para generar otro plot de patrones y frecuencias de datos faltantes. Este nuevo gráfico nos corrobora de nuevo la proporción de NA's de cada variable, pero más interesante aún se observan patrones en los que ciertas observaciones presentan más de un valor faltante simultáneamente. Este comportamiento sugiere que los datos no faltan completamente al azar (MCAR). Entonces, basándonos en la evidencia visual y el pseudo análisis exploratorio la hipótesis de que los datos son al menos MAR (Missing At Random) se ve reforzada, lo cual justifica el uso de métodos de imputación multivariados como missForest.



Al aplicar missForest descubrí que es un método que consume demasiados recursos, por lo que tardó mucho en ejecutarse en mi equipo de cómputo (alrededor de hora y media). Para este trabajo hice una ejecución completa de missForest, pero para pruebas futuras quizá sea más viable limitar el número de iteraciones (en el código ya está esa versión alterna). Los resultados fueron favorables, aunque pareciera que el PFC del 29% es alto, hay que considerar que este tipo de imputación se basa en predicción automática, y que variables como Class tienen alta cardinalidad y son difíciles de predecir sin contexto musical completo. Quizás sería interesante comparar los resultados del OOBerror si quitamos la columna Class, pero debido a los altos tiempos de espera no ahondaré en ello.

```
> imputacion$OOBerror  # error promedio de imputación (0 = perfecto)
    NRMSE       PFC
0.1085671 0.2910861
```

Ahora, luego de esta mini odisea podemos continuar con el KDD. Para continuar convertí el tiempo de milisegundos a minutos. Además, decidí explorar el asunto de los outliers que quedó pendiente. Consideré buena idea hacer un escalado Z ya que muchas variables están en escalas distintas (loudness va de -40 a +1 (decibeles); tempo, alrededor de 70–200 (bpm), etc.) y eso puede ser perjudicial si queremos aplicar PCA para mejorar los tiempos de ejecución o si queremos usar redes neuronales. La ventaja de usar Z-score es que centraría y normalizaría los datos, sin embargo, debido a que no elimina los efectos de los outliers habría que trabajarlos por aparte (pero antes de aplicar Z-score, ya que los outliers influyen fuertemente en la media y desviación estándar).

Para tratar los outliers apliqué la estrategia llamada Winsorización. Esta técnica reemplaza valores extremos por percentiles (por ejemplo, 1% y 99%), lo que nos permite conservar todos nuestros datos pero limitando su impacto negativo (esto es útil porque quiero usar PCA, y este es un método sensible). Empecé buscando las variables con outliers usando la función summary().

```
> summary(datos_imputados) #Buscar vars. con outliers
   Popularity      danceability        energy              key          loudness          mode         speechiness       acousticness    
 Min.   :  1.00   Min.   :0.0000   Min.   :2.02e-05   2      :3479   Min.   :-39.952   Minor: 9265   Min.   :0.00000   Min.   :0.00000  
 1st Qu.: 33.00   1st Qu.:0.4340   1st Qu.:5.09e-01   7      :3436   1st Qu.: -9.532   Major:16444   1st Qu.:0.03480   1st Qu.:0.00437  
 Median : 44.00   Median :0.5460   Median :7.00e-01   9      :3170   Median : -6.988                 Median :0.04730   Median :0.07980  
 Mean   : 44.64   Mean   :0.5448   Mean   :6.63e-01   1      :2736   Mean   : -7.890                 Mean   :0.07981   Mean   :0.24538  
 3rd Qu.: 56.00   3rd Qu.:0.6600   3rd Qu.:8.61e-01   4      :2393   3rd Qu.: -5.188                 3rd Qu.:0.08310   3rd Qu.:0.42500  
 Max.   :100.00   Max.   :0.9890   Max.   :1.00e+00   11     :2317   Max.   :  1.355                 Max.   :0.96000   Max.   :0.99600  
                                                      (Other):8178                                                                       
 instrumentalness     liveness         valence            tempo          time_signature      Class      duration_minutes  
 Min.   :0.000001   Min.   :0.0119   Min.   :0.0000   Min.   :  0.00   0:   1        Rock      :6900   Min.   : 0.000008  
 1st Qu.:0.000305   1st Qu.:0.0977   1st Qu.:0.3000   1st Qu.: 99.63   1:  167        Pop       :3649   1st Qu.: 2.768200  
 Median :0.020151   Median :0.1300   Median :0.4820   Median :120.26   3: 1769        Indie/Alt:3606   Median : 3.484517  
 Mean   :0.149152   Mean   :0.1968   Mean   :0.4873   Mean   :122.73   4:23497        Metal     :2667   Mean   : 3.339141  
 3rd Qu.:0.117000   3rd Qu.:0.2580   3rd Qu.:0.6720   3rd Qu.:141.98   5:  275        HipHop    :2140   3rd Qu.: 4.202217  
 Max.   :0.996000   Max.   :1.0000   Max.   :0.9860   Max.   :249.44                  Alt_Music:1893   Max.   :26.018883  
                                                                                      (Other)   :4854                      
```

Luego, seleccioné las variables Popularity, loudness, instrumentalness, tempo, y duration_minutes. Finalmente apliqué la winsorización y revisé si el proceso alteró la distribución central (lo que significaría que algo salió mal) pero todo marchaba bien. Los outliers se habían tratado adecuadamente y la distribución no se alteró gravemente.

```
> summary(data_clean)
   Popularity      danceability        energy              key          loudness          mode         speechiness       acousticness    
 Min.   : 4.00   Min.   :0.0000   Min.   :2.02e-05   2      :3479   Min.   :-22.556   Minor: 9265   Min.   :0.00000   Min.   :0.00000  
 1st Qu.:33.00   1st Qu.:0.4340   1st Qu.:5.09e-01   7      :3436   1st Qu.: -9.532   Major:16444   1st Qu.:0.03480   1st Qu.:0.00437  
 Median :44.00   Median :0.5460   Median :7.00e-01   9      :3170   Median : -6.988                 Median :0.04730   Median :0.07980  
 Mean   :44.61   Mean   :0.5448   Mean   :6.63e-01   1      :2736   Mean   : -7.860                 Mean   :0.07981   Mean   :0.24538  
 3rd Qu.:56.00   3rd Qu.:0.6600   3rd Qu.:8.61e-01   4      :2393   3rd Qu.: -5.188                 3rd Qu.:0.08310   3rd Qu.:0.42500  
 Max.   :84.00   Max.   :0.9890   Max.   :1.00e+00   11     :2317   Max.   : -2.256                 Max.   :0.96000   Max.   :0.99600  
                                                     (Other):8178                                                                       
 instrumentalness     liveness         valence            tempo          time_signature      Class      duration_minutes  
 Min.   :0.0000013   Min.   :0.0119   Min.   :0.0000   Min.   : 70.82   0:   1        Rock      :6900   Min.   :0.000041  
 1st Qu.:0.0003050   1st Qu.:0.0977   1st Qu.:0.3000   1st Qu.: 99.63   1:  167        Pop       :3649   1st Qu.:2.768200  
 Median :0.0201506   Median :0.1300   Median :0.4820   Median :120.26   3: 1769        Indie/Alt:3606   Median :3.484517  
 Mean   :0.1490069   Mean   :0.1968   Mean   :0.4873   Mean   :122.73   4:23497        Metal     :2667   Mean   :3.312535  
 3rd Qu.:0.1170000   3rd Qu.:0.2580   3rd Qu.:0.6720   3rd Qu.:141.98   5:  275        HipHop    :2140   3rd Qu.:4.202217  
 Max.   :0.9339200   Max.   :1.0000   Max.   :0.9860   Max.   :194.80                  Alt_Music:1893   Max.   :8.469236  
                                                                                      (Other)   :4854                     
```

Ya sólo queda hablar de la aplicación de Z-score. Como muestra el summary() siguiente, todas las variables numéricas ahora oscilan en torno a 0 y sus valores extremos son más manejables.

```
> summary(data_z)
   Popularity       danceability          energy              key          loudness          mode         speechiness       acousticness    
 Min.   :-2.37388   Min.   :-3.284689   Min.   :-2.8187   2      :3479   Min.   :-3.8113   Minor: 9265   Min.   :-0.95272   Min.   :-0.7919  
 1st Qu.:-0.67876   1st Qu.:-0.668207   1st Qu.:-0.6547   7      :3436   1st Qu.:-0.4337   Major:16444   1st Qu.:-0.53731   1st Qu.:-0.7778  
 Median :-0.03578   Median : 0.007015   Median : 0.1574   9      :3170   Median : 0.2260                 Median :-0.38809   Median :-0.5344  
 Mean   : 0.00000   Mean   : 0.000000   Mean   : 0.0000   1      :2736   Mean   : 0.0000                 Mean   : 0.00000   Mean   : 0.0000  
 3rd Qu.: 0.66566   3rd Qu.: 0.694294   3rd Qu.: 0.8419   4      :2393   3rd Qu.: 0.6928                 3rd Qu.: 0.03927   3rd Qu.: 0.5797  
 Max.   : 2.30233   Max.   : 2.677756   Max.   : 1.4329   11     :2317   Max.   : 1.4532                 Max.   :10.50711   Max.   : 2.4225  
                                                          (Other):8178                                                                       
 instrumentalness     liveness          valence             tempo          time_signature      Class      duration_minutes  
 Min.   :-0.5552   Min.   :-1.1567   Min.   :-2.03425   Min.   :-1.77312   0:   1        Rock      :6900   Min.   :-1.89983  
 1st Qu.:-0.5540   1st Qu.:-0.6200   1st Qu.:-0.78184   1st Qu.:-0.78901   1:  167        Pop       :3649   1st Qu.:-0.31220  
 Median :-0.4801   Median :-0.4180   Median :-0.02205   Median :-0.08446   3: 1769        Indie/Alt:3606   Median : 0.09864  
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   4:23497        Metal     :2667   Mean   : 0.00000  
 3rd Qu.:-0.1193   3rd Qu.: 0.3827   3rd Qu.: 0.77114   3rd Qu.: 0.65744   5:  275        HipHop    :2140   3rd Qu.: 0.51026  
 Max.   : 2.9245   Max.   : 5.0239   Max.   : 2.08199   Max.   : 2.46161                  Alt_Music:1893   Max.   : 2.95755  
                                                                                          (Other)   :4854                     
```

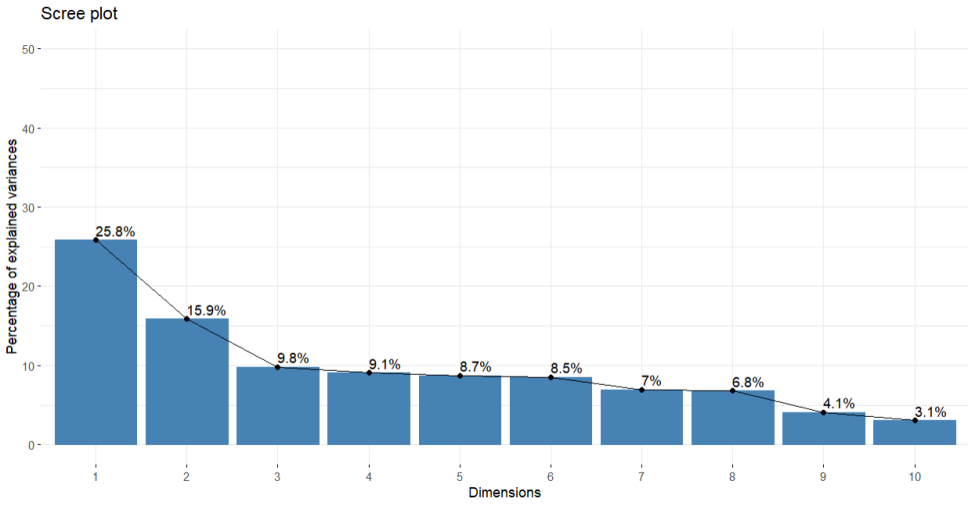
Ahora ya sólo queda separar los datos en entrenamiento y prueba para después aplicar PCA. No aplico PCA antes porque PCA aprende los componentes principales con

base a las varianzas del conjunto, entonces si lo hago PCA sobre el conjunto completo se filtra información del test en el entrenamiento, lo cual rompe la premisa de evaluación honesta (el modelo "vería" datos que no debería conocer). Cabe mencionar que al separar de nuevo los datos no coinciden en el número de filas porque eliminé las columnas ‘source’ y ‘track name’, esta última no la incluí porque no parecía aportar información relevante para un modelado de ML.

```
> dim(train)
[1] 17996    18
> dim(test)
[1] 7713   18
>
> dim(train_clean)
[1] 17996    16
> dim(test_clean)
[1] 7713   16
```

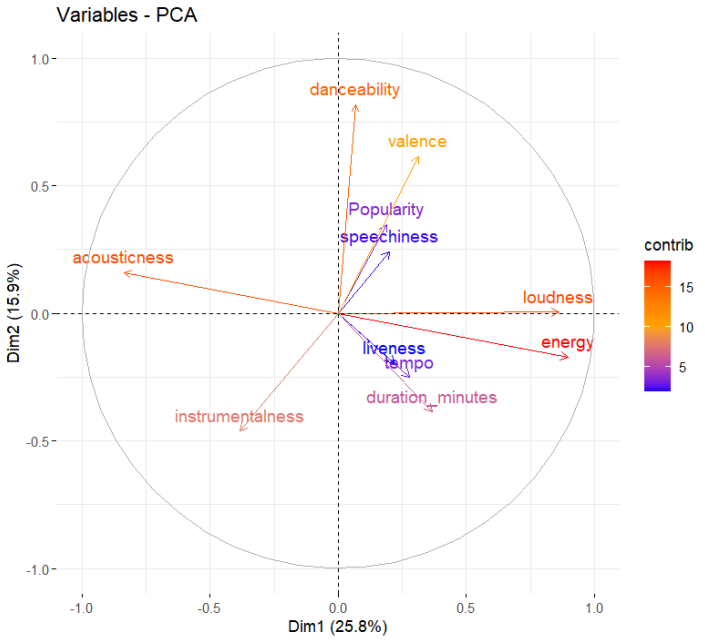
Para aplicar PCA vi la oportunidad de hacerlo de modo que sea más natural la introducción al EDA, hay dos paquetes que facilitan la comprensión de los resultados de PCA (FactoMineR y factoextra), aunque no son tan compatibles para el modelado como el PCA que hace caret, por lo que haré PCA 2 veces, uno como parte del EDA y otro para el modelado.

Nuestro primer gráfico (scree plot) muestra que a partir del componente principal número 6 o 7, la ganancia marginal de información disminuye notablemente (la varianza extra explicada por cada componente; disminuye cuando los nuevos componentes aportan poca información), lo que sugiere que entre 6 y 8 componentes serían suficientes para conservar más del 90% de la varianza original. Esto marca el llamado codo de la curva, el cual es un criterio para decidir cuántos componentes mantener tras aplicar PCA.



Luego, en nuestro precioso gráfico de observaciones por género musical podemos notar cómo se distribuyen las canciones en el nuevo espacio reducido (PC1 vs PC2). Algunos géneros como Instrumental, HipHop, Metal y Rock muestran cierta separación espacial, lo cual indica que poseen características musicales distintivas en comparación con el resto. Otros géneros presentan más superposición, sugiriendo que sus atributos son más similares. Esto nos confirma que usar PCA sería muy bueno para compactar la información útil.

El siguiente gráfico pretende mostrar cómo las variables originales están asociadas con los dos primeros componentes. Flechas más largas y de color rojo representan una mayor contribución, por lo que serán nuestro primer target a observar. Notamos que variables como energy y loudness están fuertemente correlacionadas con el primer componente (PC1), mientras que danceability y valence lo están más con el segundo (PC2). Lo más interesante es que acousticness e instrumentalness apuntan en dirección opuesta a energy y loudness, lo que sugiere que capturan propiedades contrarias en las canciones (más suaves, acústicas o sin voz dominante, en contraste con las canciones con mucho ruido y energía). Las variables con flechas más cortas tienen menor impacto en los primeros componentes, por lo que me parece razonable centrar el análisis en las más destacadas.



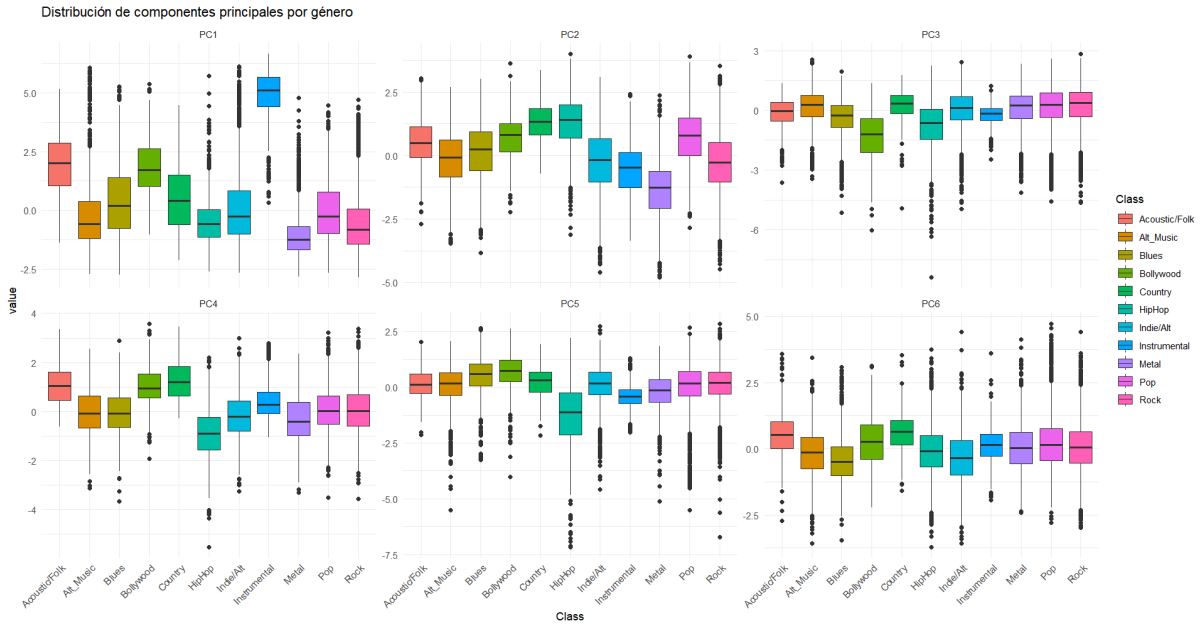
Dicho lo dicho, pasemos al EDA de manera formal.

Nota: Para este punto ya apliqué PCA con caret para tener una versión más compatible con el modelado. Lo apliqué al conjunto de entrenamiento y prueba por separado.

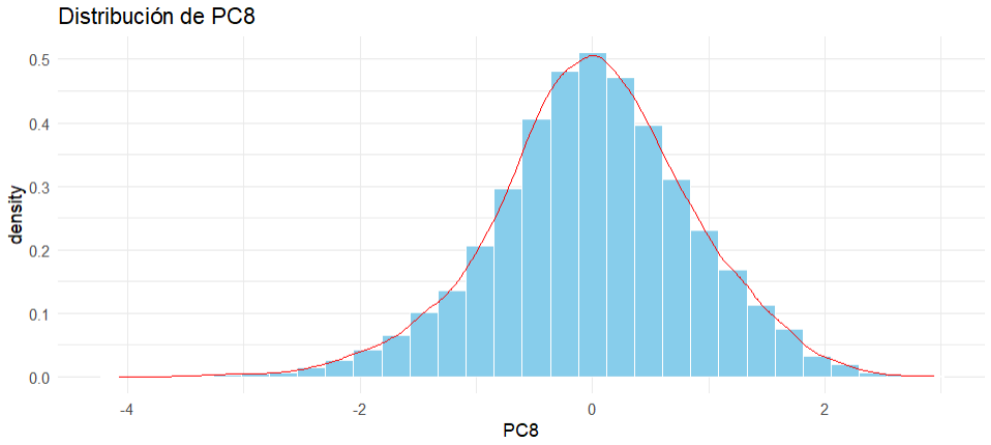
Exploratory data analysis (EDA)

Para empezar quisiera recordar que la mayoría de nuestras variables se transformaron en PC's (componentes principales), por lo que habrá que tener en cuenta ese detalle para generar y entender los ploteos siguientes.

A continuación vemos algunos boxplots de los componentes principales (PC1 a PC6) agrupados por género (Class). En PC1 podemos observar una separación clara entre Pop, Rock, Acoustic/Folk, y HipHop (por ejemplo, Rock y Pop tienden a valores más altos). Algo similar se ve en PC2, pues los géneros como Pop y Rock tienden a valores más altos que Metal o Instrumental. Esta diferenciación entre géneros continúa en todos los PC's, lo que significa que muestran patrones distinguibles. Esto es de utilidad pues es una señal de que aplicar PCA si fue de ayuda para separar géneros. Lo único "malo" es que hay superposición entre géneros, pero esto no debe preocuparnos mucho porque para generar este boxplot no se consideraron otras variables que sí están presentes en el dataset de trabajo, lo cual mejorará este detalle sin duda alguna.



El histograma de distribución de PC8 nos muestra lo que esperaba después de normalizar, una distribución aproximadamente normal. Aunque vemos que no tiene colas muy pesadas ni asimetrías marcadas por lo que puede ser un componente "menos informativo" o más neutral respecto a la separación de clases, pero esto no es necesariamente malo ya que lo convierte en un PC seguro para usarlo en modelos que asuman normalidad.



También hice una matriz de correlación pero me salió un resultado inesperado pero normal si entendemos que las componentes principales (PC1 a PC8) obtenidas tras aplicar PCA estandariza y ortogonaliza las variables (las PCs son ortogonales entre sí, lo que implica correlación cero entre componentes distintas). Pero esto no debe apaciguar nuestros ánimos, podemos hacer una matriz de correlación de los datos antes de aplicarle PCA, aunque por razones didácticas no profundizaré mucho en su interpretación.

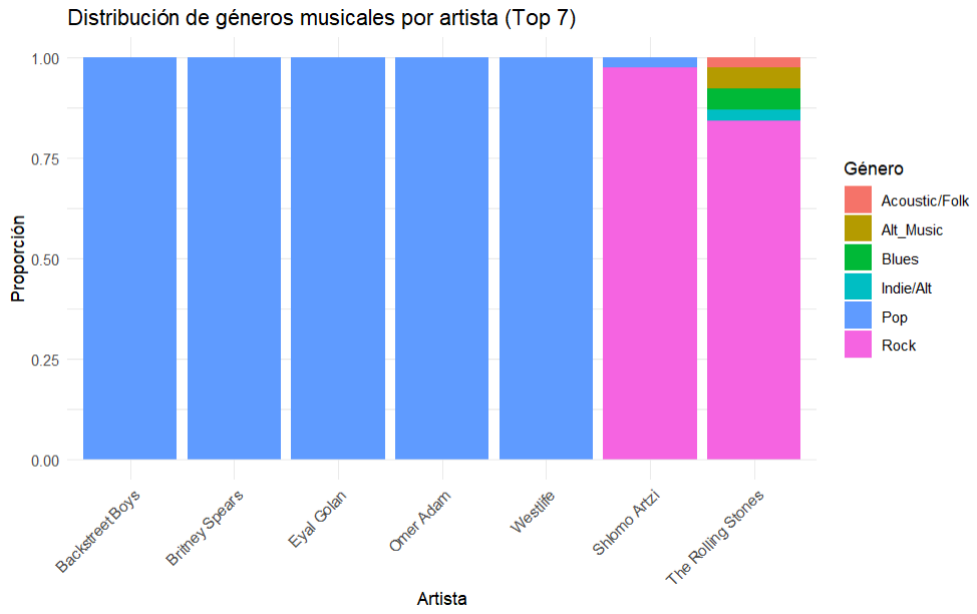
```
> print(round(cor_matrix, 3))
    PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8
PC1   1   0   0   0   0   0   0   0
PC2   0   1   0   0   0   0   0   0
PC3   0   0   1   0   0   0   0   0
PC4   0   0   0   1   0   0   0   0
PC5   0   0   0   0   1   0   0   0
PC6   0   0   0   0   0   1   0   0
PC7   0   0   0   0   0   0   1   0
PC8   0   0   0   0   0   0   0   1
```

La nueva matriz de correlación muestra algo sorprendente para mí, la popularidad no tiene una correlación fuerte con ninguna variable, lo cual me resulta inesperado, pues generalmente las canciones que se vuelven virales lo hacen porque siguen ciertas estructuras musicales que se sabe que funcionan (canciones pegajosas). Posiblemente sería otra historia si incluímos Class en la matriz, pero no lo veo necesario ya que estas mismas estructuras musicales son las variables que aparecen en la matriz (obviamente no todas, pero si algunas).

```
> print(round(cor_matrix_original, 2))
                Popularity danceability energy loudness speechiness acousticness instrumentalness liveness valence tempo duration_minutes
Popularity            1.00         0.17   0.05     0.13        0.03        -0.13           -0.19    -0.07    0.06 -0.01            -0.02
danceability          0.17         1.00  -0.09     0.05        0.20         0.01           -0.21    -0.11    0.44 -0.19            -0.11
energy                0.05        -0.09   1.00     0.78        0.13        -0.75           -0.17     0.20    0.22  0.21             0.27
loudness              0.13         0.05   0.78     1.00        0.10        -0.62           -0.35     0.11    0.17  0.16             0.19
speechiness           0.03         0.20   0.13     0.10        1.00        -0.09           -0.07     0.07    0.05  0.05             0.01
acousticness         -0.13         0.01  -0.75    -0.62       -0.09         1.00            0.16    -0.11   -0.12 -0.17            -0.35
instrumentalness     -0.19        -0.21  -0.17    -0.35       -0.07         0.16            1.00    -0.04   -0.22 -0.03             0.01
liveness             -0.07        -0.11   0.20     0.11        0.07        -0.11           -0.04     1.00    0.02  0.03             0.05
valence               0.06         0.44   0.22     0.17        0.05        -0.12           -0.22     0.02    1.00  0.05            -0.09
tempo                -0.01        -0.19   0.21     0.16        0.05        -0.17           -0.03     0.03    0.05  1.00             0.05
duration_minutes     -0.02        -0.11   0.27     0.19        0.01        -0.35            0.01     0.05   -0.09  0.05             1.00
```

Ahora bien, pasemos al que posiblemente sea el punto más importante del EDA. Descubrir a qué géneros se suele asociar a cada artista puede ayudarnos a crear una variable que sea un predictor fuerte, considerando que para los datos de prueba no se sabrá el género de la canción (pero si se sabría qué géneros suele hacer "x" artista). Muchos artistas tienen trayectorias definidas por un género, y las características acústicas (PCs) no distinguen bien entre ciertos géneros, pero el artista sí, por lo que crear una nueva variable surge como una oportunidad de oro. Por si sola esta explicación es contundente y fundamenta bien la creación de una nueva variable, pero para darle más peso a esta decisión analicemos la relación que hay entre los 7 artistas más conocidos del dataset (basándose en la frecuencia con que aparecen) y los géneros disponibles.

Como podemos ver, Backstreet Boys, Britney Spears, Eyal Golan, Omer Adam y Westlife están fuertemente asociados exclusivamente al género "Pop". Y The Rolling Stones tiene más diversidad, aunque también predominan en "Rock", con algunos ejemplos de "Indie/Alt", "Alt_Music", "Blues", "Acoustic/Folk". Todo esto coincide con el conocimiento popular presente, por lo que funciona como evidencia de que la nueva variable main_genre puede ser un predictor muy fuerte.



Este análisis exploratorio, aunque más breve que el KDD, resulta fructífero por la comprensión entre variables más allá de lo que un modelo o una matriz de correlación nos puede decir a primera instancia.

Estrategia de solución (modelado)

Bien, como propuesta de solución me he planteado usar reglas de asociación y Naïve Bayes usando 7-Fold CV y Kappa para evaluar el rendimiento. Usar reglas de asociación como modelo de clasificación ofrece interpretabilidad, permitiendo entender con claridad bajo qué condiciones se predice cada clase (aunque lo malo es que tenemos más variables numéricas por lo que habría que discretizar). Sin embargo, tienden a generar reglas específicas con bajo soporte, lo que limita su capacidad de generalización. Como alternativa, proponemos usar Naïve Bayes, un modelo simple pero robusto que asume independencia entre variables y ofrece buena capacidad de generalización, especialmente en dominios con alta dimensionalidad y clases desbalanceadas, por lo que se acopla bien a nuestro dataset. Como ya se mencionó tuve que aplicar discretización a los datos. Pero aquí hay un tema del que hablar, los tiempos de ejecución fueron demasiado altos por lo que aplicar la estrategia de "prueba y error" para refinar las reglas parece inviable, por ello pasaré a usar Bayes, pero antes de eso expliquemos lo que obtuve con las reglas.

Para empezar usé el dataset de prueba para hacer predicciones, después quise sacar su matriz de confusión y kappa para observar el rendimiento pero olvidé que el dataset de prueba no indicaba a qué clase pertenecían las canciones. Pero de todos modos fue posible la inspección de las reglas generadas. Por ejemplo, el análisis de las 10 reglas

con mayor confianza muestra un patrón recurrente: muchas de ellas involucran combinaciones simples como main_genre = Bollywood o agregan una sola variable más, como key o alguna componente principal discretizada. Estas reglas alcanzan una confianza perfecta (1.0) y un lift superior a 44, lo que indica que cuando estas condiciones se cumplen, la predicción es prácticamente certera y mucho mejor que una asignación aleatoria, ¿Cierto? Pues sí, pero tampoco estamos frente a la panacea de nuestro problema. Estas reglas presentan un soporte relativamente bajo (alrededor del 0.1% al 2% del total de transacciones), lo que implica que cubren solo subconjuntos muy específicos del dataset. De hecho, la mediana del soporte global es de apenas 0.0018, mientras que la mediana del número de instancias cubiertas por regla es de solo 32. La longitud media de las reglas es de 7.2 condiciones, lo cual sugiere que muchas son altamente especializadas y poco generalizables.

```
> inspect(head(sort(reglas_utiles, by = "confidence"), 10)) #Las top 10 por confianza
     lhs                                          rhs              support     confidence coverage    lift     count
[1]  {main_genre=Bollywood}                    => {Class=Bollywood} 0.022338297 1          0.022338297 44.76617 402
[2]  {main_genre=HipHop, bin_PC5=1}            => {Class=HipHop}    0.001111358 1          0.001111358 12.43677  20
[3]  {time_signature=5, main_genre=Indie/Alt}  => {Class=Indie/Alt} 0.001444766 1          0.001444766  6.95632  26
[4]  {main_genre=Bollywood, bin_PC3=2}         => {Class=Bollywood} 0.002833963 1          0.002833963 44.76617  51
[5]  {time_signature=3, main_genre=Country}    => {Class=Country}   0.001389198 1          0.001389198 46.50129  25
[6]  {key=3, main_genre=Bollywood}             => {Class=Bollywood} 0.002000445 1          0.002000445 44.76617  36
[7]  {main_genre=Bollywood, bin_PC7=1}         => {Class=Bollywood} 0.004278729 1          0.004278729 44.76617  77
[8]  {time_signature=3, main_genre=Bollywood}  => {Class=Bollywood} 0.003723050 1          0.003723050 44.76617  67
[9]  {key=8, main_genre=Bollywood}             => {Class=Bollywood} 0.001889309 1          0.001889309 44.76617  34
[10] {key=6, main_genre=Bollywood}             => {Class=Bollywood} 0.001889309 1          0.001889309 44.76617  34
> summary(reglas)
set of 299263 rules

rule length distribution (lhs + rhs):sizes
    2     3     4     5     6     7     8     9    10    11    12    13
   12   434  4849 23166 57931 83942 73423 39671 13104  2489   235     7

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   6.000   7.000   7.276   8.000  13.000

summary of quality measures:
    support           confidence        coverage             lift            count
 Min.   :0.001000   Min.   :0.6000   Min.   :0.001000   Min.   : 2.182   Min.   :  18.00
 1st Qu.:0.001278   1st Qu.:0.8182   1st Qu.:0.001445   1st Qu.: 3.471   1st Qu.:  23.00
 Median :0.001778   Median :0.8876   Median :0.002056   Median : 6.820   Median :  32.00
 Mean   :0.003116   Mean   :0.8751   Mean   :0.003619   Mean   :10.368   Mean   :  56.08
 3rd Qu.:0.003056   3rd Qu.:0.9500   3rd Qu.:0.003556   3rd Qu.:11.594   3rd Qu.:  55.00
 Max.   :0.220216   Max.   :1.0000   Max.   :0.261725   Max.   :46.501   Max.   :3963.00

mining info:
         data ntransactions support confidence
 transacciones         17996   0.001        0.6

call
 apriori(data = transacciones, parameter = list(supp = 0.001, conf = 0.6, minlen = 2, target = "rules"), appearance = list(rhs = grep("^Class=", itemLab
els(transacciones), value = TRUE), default = "lhs"), maxlen = 15)
```

Además, el análisis de las reglas usando k-Fold amarga todavía más el gusto. Para esta ocasión usé el dataset de train para usar k-Fold y poder ver su rendimiento, pero es posible que haberlo hecho como lo hice no fue la mejor opción. El kappa obtenido refleja que el modelo basado en reglas no logra generalizar de forma efectiva a nuevos subconjuntos de datos.

```
> mean_kappa
[1] 0.00196733
```

Esto sugiere que, si bien las reglas captaron patrones específicos con alta precisión en los datos de entrenamiento, su cobertura es muy limitada, y la mayoría de las predicciones terminan recurriendo al respaldo por main_genre, lo que no aporta una mejora sustancial respecto al azar (forcé al código a que use main_genre para que siempre produzca una predicción, incluso si ninguna regla se aplica). Aunque

main_genre es un predictor semánticamente informativo (ya que condensa el historial típico del artista), su poder discriminativo por sí solo es limitado, especialmente en casos donde los artistas son versátiles o están asociados a múltiples géneros. Esto impacta negativamente en la generalización, ya que muchas instancias quedan cubiertas únicamente por esta estrategia de respaldo, sin el soporte adicional de una regla específica. Como consecuencia, el modelo basado en reglas termina funcionando como un clasificador degenerado que predice el género más frecuente por artista, sin aprovechar plenamente las variables acústicas ni capturar patrones robustos. Esta situación evidencia que usar Bayes podría ser bueno no sólo porque puede generalizar mejor la información, sino que tiene la ventaja de que asigna probabilidades a cada clase incluso en contextos con información parcial o ambigua.

Para aplicar Bayes tengo que mencionar algo importante que recién descubrí. Al parecer usar k-Fold luego de aplicar PCA implica un riesgo potencial porque las componentes principales fueron calculadas usando todo el conjunto completo (train_pca), lo que incluye información de los futuros folds de validación. Para tratar este problema usaré los datos antes de aplicar PCA (ventajas de hacer varias copias de seguridad de los datasets) y luego aplicaré PCA dentro de cada fold. También, aprovecharé y usaré el sub-muestreo estratificado del 50% para acelerar los tiempos de ejecución.

Este enfoque fue un rotundo éxito en cuanto a tiempo de ejecución, pasé de casi dos horas de ejecución a apenas cinco o diez segundos de espera. Además, el kappa se vio beneficiado (aunque sigue siendo relativamente bajo).

```
Fold 1 - Kappa: 0.3367
Fold 2 - Kappa: 0.3587
Fold 3 - Kappa: 0.3493
Fold 4 - Kappa: 0.374
Fold 5 - Kappa: 0.3572
> mean_kappa <- mean(kappas)
> cat("\nKappa promedio (Naïve Bayes, PCA, submuestreo 50%):", round(mean_kappa, 4), "\n")

Kappa promedio (Naïve Bayes, PCA, submuestreo 50%): 0.3552
```

Sin duda alguna estos resultados parecen prometedores por lo que probaré a refinar la estrategia removiendo PCA y aumentando el número de folds a diez.

```
Fold 1 - Kappa: 0.3406
Fold 2 - Kappa: 0.3263
Fold 3 - Kappa: 0.3445
Fold 4 - Kappa: 0.3292
Fold 5 - Kappa: 0.3494
Fold 6 - Kappa: 0.3363
Fold 7 - Kappa: 0.3464
Fold 8 - Kappa: 0.3198
Fold 9 - Kappa: 0.3415
Fold 10 - Kappa: 0.3431
> mean_kappa_no_pca <- mean(kappas_no_pca)
> cat("\nKappa promedio (Naïve Bayes, sin PCA, 10-fold):",

Kappa promedio (Naïve Bayes, sin PCA, 10-fold): 0.3377
```

Mejoras al modelo

Bueno, los resultados anteriores no fueron extremadamente mejores como me hubiera gustado pero sin duda alguna fueron mejores. Esto posiblemente se deba a que sólo trabaje con las variables numéricas, pues convertir las otras categorías de factor (o char) a notación one-holt encodeing podría resultar contraproducente ya que hay una inmensa cantidad de artistas, por ejemplo. Además, olvidé que el dataset de respaldo no tenía main_genre, por lo que el siguiente modelo considera estos puntos.

```
Fold 1 - Kappa: 0.7275
Fold 2 - Kappa: 0.7103
Fold 3 - Kappa: 0.7334
Fold 4 - Kappa: 0.6865
Fold 5 - Kappa: 0.7013
Fold 6 - Kappa: 0.6834
Fold 7 - Kappa: 0.6976
Fold 8 - Kappa: 0.7224
Fold 9 - Kappa: 0.7159
Fold 10 - Kappa: 0.7002
> mean_kappa_nb_cat <- mean(kappas_nb_cat)
> cat("\nKappa promedio (Naïve Bayes, categóricas incluidas, 10-fold):", round(mean_kappa_nb_cat, 4), "\n")

Kappa promedio (Naïve Bayes, categóricas incluidas, 10-fold): 0.7078
```

Como vemos, los resultados fueron excelentes esta vez, tardó menos de cinco segundos en ejecutarse y devolvió un kappa bueno. Aunque simple, Bayes funciona muy bien cuando hay columnas categóricas predictivas bien distribuidas. El centering/scaling que apliqué solo para variables numéricas fue un éxito. La mejora respecto a PCA sin categóricas muestra que la selección de variables puede ser más importante que la complejidad del modelo. Y hablando de modelos, creo pertinente demostrar con un ejemplo práctico cómo el uso de dummy vars podría complicar el modelado. Con Bayes no hubo mucho problema ya que este modelo puede trabajar directamente con variables categóricas, sin embargo, SVM necesita variables numéricas por lo que durante la ejecución tuve varias advertencias.

```
+   cat("Fold", i, "- Kappa:", round(kappas_svm_cat[i], 4), "\n")
+ }
Aviso en preProcess.default(train_encoded, method = c("center", "scale")) :
  These variables have zero variances: time_signature.0
Fold 1 - Kappa: 0.8097
Aviso en preProcess.default(train_encoded, method = c("center", "scale")) :
  These variables have zero variances: time_signature.0
Fold 2 - Kappa: 0.8082
Aviso en preProcess.default(train_encoded, method = c("center", "scale")) :
  These variables have zero variances: time_signature.0
Fold 3 - Kappa: 0.8173
Aviso en preProcess.default(train_encoded, method = c("center", "scale")) :
  These variables have zero variances: time_signature.0
Fold 4 - Kappa: 0.7932
Aviso en preProcess.default(train_encoded, method = c("center", "scale")) :
  These variables have zero variances: time_signature.0
Fold 5 - Kappa: 0.8226
Aviso en preProcess.default(train_encoded, method = c("center", "scale")) :
  These variables have zero variances: time_signature.0
Fold 6 - Kappa: 0.8083
Aviso en preProcess.default(train_encoded, method = c("center", "scale")) :
  These variables have zero variances: time_signature.0
Fold 7 - Kappa: 0.8022
Aviso en preProcess.default(train_encoded, method = c("center", "scale")) :
  These variables have zero variances: time_signature.0
Fold 8 - Kappa: 0.8231
Aviso en preProcess.default(train_encoded, method = c("center", "scale")) :
  These variables have zero variances: time_signature.0
Fold 9 - Kappa: 0.8221
Aviso en preProcess.default(train_encoded, method = c("center", "scale")) :
  These variables have zero variances: time_signature.0
Fold 10 - Kappa: 0.8107
Hubo 40 avisos (use warnings() para verlos)
> mean_kappa_svm_cat <- mean(kappas_svm_cat)
> cat("\nKappa promedio (SVM, categóricas incluidas, 10-fold):", round(mean_kappa_svm_cat, 4), "\n")

Kappa promedio (SVM, categóricas incluidas, 10-fold): 0.8117
```

Aún así obtuvimos un buen kappa pero considero más enriquecedor el pensamiento estratégico que seguimos en Bayes para mejorarlo poquito a poquito en lugar de pasar directamente con los modelos más sofisticados. Veamos una tabla comparativa de nuestro trabajo hasta ahora.

Modelo	Kappa promedio
Reglas de asociación	0.0019
Naïve Bayes (solo numéricas, sin PCA)	0.3377
Naïve Bayes (con PCA)	0.3552
Naïve Bayes (numéricas + categóricas)	0.7078
SVM	**0.8117**

Nota: recordar que desde el primer modelo Bayesiano en adelante usé sub-muestreo estratificado.

La estructura que seguí para aplicar SVM fue la siguiente:

- Sub-muestreo estratificado,
- Eliminación de varianza cero,
- Normalización adecuada por fold,
- Evaluación robusta sin leakage.

El modelo SVM con categóricas codificadas como dummies logra un salto de +10 puntos de Kappa respecto a Naïve Bayes. Lo que demuestra que las variables categóricas como main_genre, key, mode, time_signature son altamente informativas, y SVM aprovecha mejor la interacción entre variables y la no linealidad del espacio.

Conclusión

La capacidad del SVM para modelar relaciones no lineales y aprovechar la interacción entre variables numéricas y categóricas fue clave para dar un mejor kappa, aunque no porque SVM fue el mejor hay que olvidarnos de los otros modelos. Bayes nos aportó el esqueleto que posteriormente usaría con SVM y las reglas nos dieron una nueva variable que fue un predictor fuerte (género principal). Este trabajo evidencia que, más allá del tipo de modelo, una adecuada selección y transformación de variables puede marcar una diferencia sustancial en la capacidad predictiva de los algoritmos aplicados.