
Universidad de
SanAndrés

# Aprendizaje Supervisado
# Clasificación de gusto musical

Docentes: Marcela Svarc y Facundo Carrillo
Alumnos: Ezequiel Talamona, Raul Urrizaga, Tomas Pernas

# Índice

# 1. Introducción

En el siguiente proyecto vamos a trabajar con un dataset que contiene información y características sobre 750 canciones y una columna llamada “label” que especifica si cada canción le gusta o no a un determinado usuario. El objetivo principal es generar un mejor entendimiento de la información contenida en el dataset y lograr predecir en función de los features de una canción si al usuario le va a gustar o no un nuevo tema. A su vez y para poder medir el resultado final del modelo, definimos una métrica de éxito y comparamos los diferentes modelos desarrollados en función de esta.

# 2. Análisis y pre procesamiento de datos

El análisis exploratorio de datos, también conocido por sus siglas AED, nos sirve para analizar y conocer el dataset en profundidad. Podemos obtener insights interesantes y también preparar el conjunto de datos para adecuarlo según el algoritmo que se requiera para resolver el problema que se esté abordando.

- **Limpieza de datos**

Como primer paso realizamos una revisión y limpieza de datos. Eliminamos las 14 instancias duplicadas y revisamos que no hayan valores nulos en el dataset. A su vez procederemos a revisar la existencia de posibles outliers. Igualmente destacamos que en líneas generales encontramos un dataset limpio y listo para ser utilizado.

- **Histogramas y boxplot**

Utilizamos las herramientas de histogramas y boxplot para analizar con mayor detalle tanto la distribución como los outliers que presentan las distintas variables dentro del dataset.

Al momento de realizar los histogramas, nos resultó interesante comentar particularmente el de las variables “duration”, “tempo” y “key”. Cuando observamos el histograma de la variable “duration” podemos dar cuenta que la gran mayoría de los temas que escucha el usuario analizado rondan entre los 3:00 y 4:00 minutos de duración promedio, mientras que para la variable “tempo” los temas en su gran mayoría se encuentran entre los 120 y 140 BPM promedio.

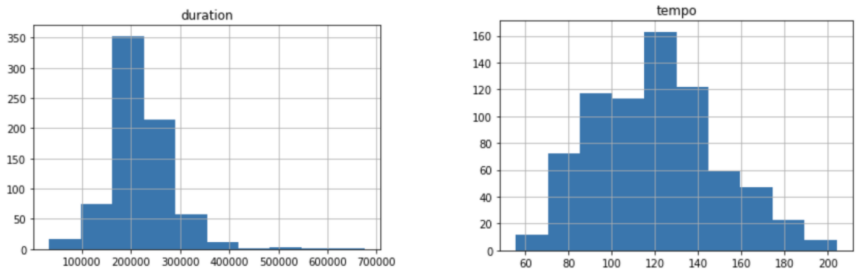


Figura 1: Histogramas de tiempo y duración

Podemos generalizar y establecer que cada género musical se encuentra asociado a un tempo determinado. Entonces teniendo en cuenta que del análisis del histograma se desprende que el BPM ronda entre 120 y 140 BPM diremos entonces que el usuario analizado prefiere escuchar canciones relacionadas al género musical electrónico.

Por último sobre el feature ”key”, el cual hace referencia a la tonalidad en la que se encuentra la canción, y filtrando por el label = 1 , es decir, aquellas canciones que si le gusta a la persona,

podemos observar que el tono más disfrutado por la persona al momento de escuchar música es el DO.

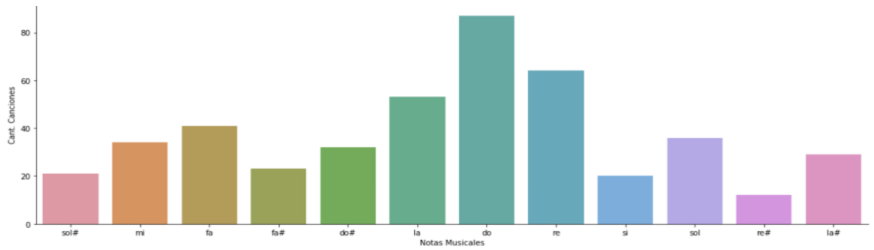


Figura 2: Notas músicales por canciones favoritas del usuario.

Con esto nos resulta interesante mencionar un artículo de la web donde nos menciona: "Kenny Ning, analista de datos Spotify, se encargó de analizar la tonalidad de cada track en la plataforma para determinar la frecuencia con que aparecen. Los resultados arrojados muestran, que en general, las tonalidades mayores son mucho más usadas que las menores, lo que tiene mucho sentido pues a la gente le gustan los sonidos felices o positivos. Así la tonalidad de Sol Mayor obtuvo la cifra más alta de popularidad entre las usadas seguida por las tonalidades de Do Mayor y Re Mayor. "

Teniendo en cuenta el artículo mencionado, podemos aproximarnos a una idea relacionada a los gustos del usuario y decir que prefiere escuchar canciones con tonalidades alegres sobre las tristes.

Con respecto a los boxplot, no notamos valores atípicos en ninguna de sus variables.

- **Análisis de Correlación**

Realizamos un análisis de correlación para observar en un alto nivel si existe algún tipo de relación entre las variables del dataset. Encontramos que las variables con mayor correlación entre sí son "loudness" y "energy". Como así también "valence" y "danceability", son dos variables con alta correlación. Las primeras dos van a ser de gran importancia en el desarrollo y análisis de los modelos de clasificación.

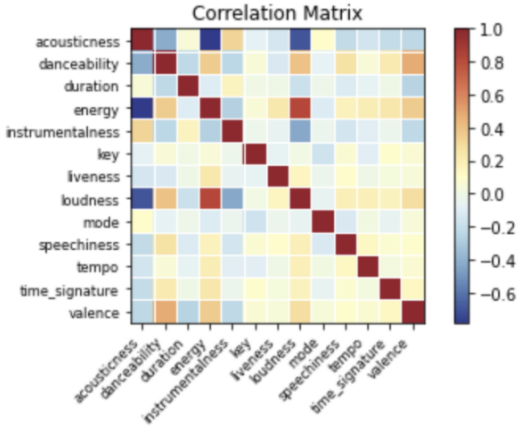


Figura 3: Matriz de correlación.

- **Estandarización**

Generamos un dataset alternativo para correr un modelo de PCA y reducirlo a sólo dos variables para futuras pruebas. Para esto estandarizamos este dataset alternativo utilizando la feature StandardScaler del paquete preprocessing de scikit learn. Standard Scale genera un nuevo dataset con media 0 y desvío standard 1, lo que nos permite ejecutar un modelo de reducción de dimensionalidad con PCA posteriormente.

# 3. Descripción de modelos utilizados

- **Random Forest**

Comenzamos utilizando un modelo de clasificación ensamblado de árboles de decisión llamado Random Forest. El modelo utiliza un proceso de bootstrapping llamado bagging, en el que genera "n" cantidad de árboles, en dicho proceso y a diferencia de un proceso de bagging convencional, en cada iteración realiza una extracción de variable lo que hace que los modelos generados sean un diferentes entre sí y de esta forma poder disminuir la varianza en el momento de la votación a la clase ganadora.

Utilizamos también una función de scikit learn llamada Grid Search CV que nos permite iterar el modelo con diferentes parámetros y adicionalmente ejecutar un método de cross validation (nfolds: 50) durante el proceso. Los parámetros que utilizamos en el Grid Search para las diferentes iteraciones y para obtener un resultado óptimo son: maxdepth o altura del árbol (1 a 5) / maxfeatures"auto", "log2" - Es el número de variables a considerar para realizar el siguiente split en cada iteración.

Ajustamos y ejecutamos el modelo sobre una muestra de train del dataset de tamaño del 70 porciento del total, utilizando como parámetro random state igual 4. Para realizar el split utilizamos el método de train, test y split de scikit learn.

Luego de ejecutar y ajustar el modelo bajo las condiciones anteriormente detalladas, encontramos que el mejor estimador (utilizando el atributo de best estimator de Grid Search) se ajusta con los siguientes parámetros: maxdepth=4 / maxfeatures='log2' / criterion='gini' / nestimators=100.

El accuracy de la predicción sobre el set de entrenamiento es 0.86, mientras que en el set de test es de 0.83. Vamos a tomar como métrica de éxito a lo largo del trabajo el accuracy para analizar los diferentes modelos propuestos.

Utilizamos el aprendizaje de este modelo para entender cuáles son las variables con mayor impacto a la hora de tomar una decisión sobre nuestro "label" objetivo. Para eso utilizamos unas funciones descritas en el código que nos permitieron llegar al siguiente gráfico:

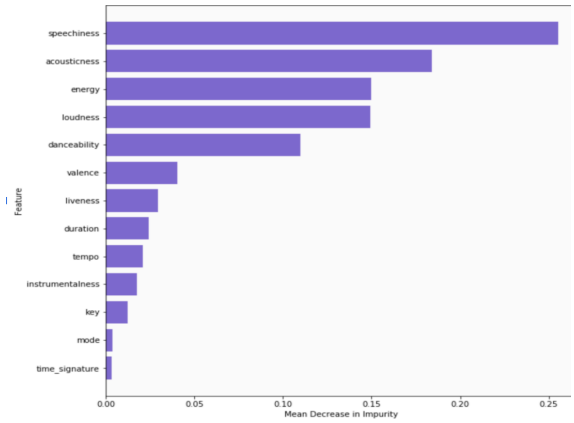


Figura 4: Feature importance for Random Forest model.

En él podemos observar en el eje Y todas las features que posee el dataset analizado y en el eje X la magnitud del "Mean Decrease Impurity". Está métrica es un indicador, también llamado importancia de Gini, que representa la importancia en términos de promedio de cada feature a la hora de separar a todas las samples de un dataset en cada nodo. Entonces, a partir del anterior gráfico obtenemos como información para input de los próximos modelos que las features 'speechiness', 'acousticness', 'energy', 'loudness' y 'danceability' son de gran importancia a la hora de la predicción.

- **Support vector machine**

El siguiente modelo utilizado es SVM. El mismo puede ser utilizado tanto para clasificación como para regresión, aunque su uso más habitual es para clasificar. Se basa en la idea de dividir en 2 clases el set de datos mediante un hiperplano o conjunto de hiperplanos ( en alta dimensionalidad) de separación. Una buena separación entre las clases permitirá una correcta clasificación. Llamaremos "Support Vector" a los puntos que están más cerca del hiperplano y que si fueran removidos, alterarían la posición del mismo, constituyendo los elementos críticos del set de datos.

Decidimos correr este modelo con tres enfoques diferentes: Utilizando el dataset en "crudo" y con sus parámetros por default. Utilizando solo las 5 features más importantes anteriormente detalladas, con parámetros por default. Utilizando solo las 5 features más importantes anteriormente detalladas, con parámetros por Grid Search y cross validation.

Primera etapa: Ajustamos y corrimos el modelo sobre una muestra de train del dataset de tamaño del 70 porciento del total, utilizando como parámetro random state igual 4. Para hacer el split utilizamos el método de train, test y split de scikit learn. El accuracy de este modelo sobre la muestra de test es 0.60.

Segunda etapa: Tomamos como input las variables con mayor índice de importancia de Gini del modelo de Random Forest que corrimos anteriormente. Utilizando sólo estas 5 variables 'speechiness', 'acousticness', 'energy', 'loudness' y 'danceability'. A su vez, utilizamos los mismos parámetros que vienen por default que usamos en la primera etapa. Solo haciendo feature selection y reduciendo la cantidad de variables con la que se entrena el modelo logramos aumentar en un +34 porciento el accuracy, obteniendo 0.80 en la muestra de test.

Tercera etapa: Utilizamos solo las 5 features más importantes anteriormente detalladas, con variación de parámetros por Grid Search y cross validation. Utilizamos Grid Search CV con diferentes parámetros y cross validation con nfolds: 50. Los parámetros que utilizamos en el Grid Search para las diferentes iteraciones y para obtener un resultado óptimo son: 'C': [0.1, 1, 10, 100, 1000] - Cantidad de errores (regularization parameter) 'gamma': [1, 0.1, 0.01, 0.001, 0.0001] - Coeficiente kernel 'kernel': ['linear', 'poly', 'rbf', 'sigmoid'] - Kernel type Luego de ejecutar y ajustar el modelo bajo las condiciones anteriormente detalladas, encontramos que el mejor estimador (utilizando el atributo de best estimator de Grid Search) se ajusta con estos parámetros: 'C': 100 / 'gamma': 0.1.

No observamos grandes diferencias respecto al modelo anterior, arrojando como resultado un accuracy de 0.82 sobre los datos de test.

- **KNN**

Es un método de aprendizaje supervisado que busca las observaciones más cercanas a la que se está tratando de predecir y clasificar el punto de interés basado en la mayoría de datos que lo rodean. El algoritmo memoriza las instancias de entrenamiento que son usadas como "base de conocimiento" para la fase de predicción. Básicamente lo que hace es calcular la distancia entre el ítem a clasificar y el resto de ítems del dataset de entrenamiento, seleccionar los k elementos más cercanos (con menor distancia, según la función que se use) y realizar una "votación de mayoría" entre los k puntos. Los puntos de una clase que dominen decidirán su clasificación final. Para ejecutar y probar el modelo KNN vamos a mantener una estrategia similar a la anterior, pero cambiamos ligeramente el enfoque.

Decidimos correr este modelo con tres enfoques diferentes: Utilizando el dataset con todas las features y con Grid Search cross validation. Utilizando Grid Search, cross validation y solo las 5 features seleccionadas anteriormente. Utilizamos una técnica de reducción de dimensionalidad llamada Principal Component Analysis (PCA) y ejecutamos KNN sobre dos nuevas features.

Primera etapa: Ajustamos y corrimos el modelo sobre una muestra de train del dataset de tamaño del 70 porciento del total, utilizando como parámetro random state igual 4. Para hacer el split utilizamos el método de train, test y split de scikit learn. Utilizamos Grid Search CV con diferentes parámetros y cross validation con nfolds: 50. Los parámetros que utilizamos en el Grid Search para las diferentes iteraciones y para obtener un resultado óptimo son: 'nneighbors': np.arange(1,50,1) El accuracy de este modelo sobre la muestra de test es 0.61.

Segunda etapa: Utilizamos solo las 5 features más importantes anteriormente detalladas ('speechiness', 'acousticness', 'energy', 'loudness' y 'danceability'), con variación de parámetros por Grid Search y cross validation. El accuracy con este método aumenta un +32 porciento, dando como resultado 0.80 sobre los datos de test.

Tercera etapa: Utilizamos el dataset estandarizado con media 0 y desvío estándar 1 que habíamos generado y mencionado al principio del trabajo (utilizando el método standard scaler de scikit learn). Reducimos el dataset a dos componentes, utilizando PCA, pasamos de un dataset de 13 features a uno con 2 features que capturan la mayor variabilidad. En el gráfico de la derecha observamos un scatter plot de ambos componentes y sus respectivos labels, representados con los colores violeta y amarrillo. Se observa cierta simetría en la distribución, pero aún así mucha contaminación de varios puntos entre ambos grupos. Ejecutamos y entrenamos el modelo KNN con este nuevo dataset, utilizando los mismos parámetros que en las primeras dos etapas hasta aquí mencionadas (utilizando grid search y cross validation). El accuracy del modelo sobre los datos de test es de 0.78.

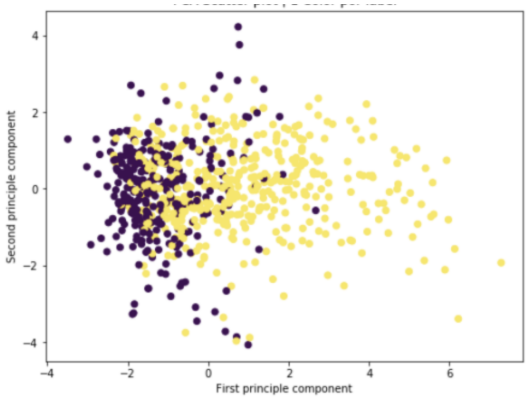


Figura 5: PCA Scatter plot.

# 4. Conclusiones

A lo largo del trabajo fuimos descubriendo diferentes puntos importantes y hallazgos sobre la problemática inicial. La primera conclusión es que tomando accuracy como métrica de éxito, el modelo que mejor se ajusta y generaliza con los datos es el de Random Forest con grid search y cross validation. Otro punto muy importante a la hora de probar otros modelos es el impacto positivo que tiene la selección de variables, observamos un impacto positivo de más del 30 porciento en accuracy seleccionado solamente las variables más importantes del modelo. También observamos que la reducción de dimensionalidad es de gran ayuda para simplificar un problema e intentar observar los datos desde otra perspectiva, sin embargo, nos quedamos con el modelo de random forest, dado que es el modelo más simple y el que arrojó mejores resultados. Como conclusión entonces diremos que el modelo elegido fue Random Forest con los siguientes parámetros y métricas de

éxito: bootstrap=True / criterion='gini' / maxdepth=3 / maxfeatures='log2' / nestimators=100 / accuracy: 0.83.

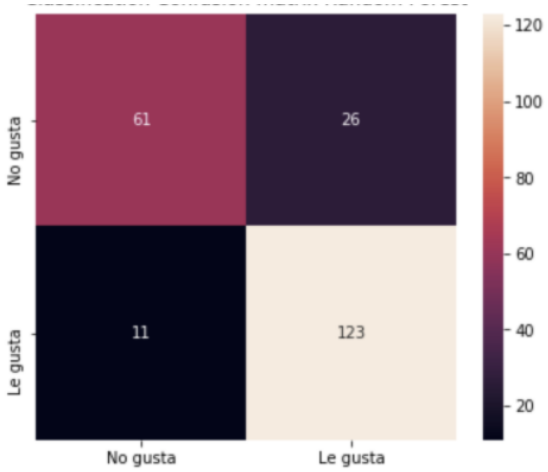


Figura 6: Matriz de confusión.