Tecnológico de Monterrey

Módulo 2 Análisis y Reporte sobre el desempeño del modelo.

Inteligencia artificial avanzada para la ciencia de datos I

Alumno

Jackeline Conant Rubalcava A01280544

Profesores

Antonio Carlos Bento

Alfredo Esquivel Jaramillo

Mauricio González Soto

Julio Antonio Juárez Jiménez

Frumencio Olivas Álvarez

Jesús Adrián Rodríguez Rocha

Hugo Terashima Marín

07 de septiembre del 2024

Introducción a Base de Datos y Modelos de aprendizaje automático con árboles de decisión

Para esta actividad, utilicé una base de datos llamada **“Dataset of Songs in Spotify”**, en la cual se muestran varios aspectos sobre diferentes canciones y sus géneros. Por esta razón, me decidí en crear modelos de predicción para poder determinar qué tipo de género se está escuchando en cada canción.

La base de datos cuenta con **42305 registros y 22 columnas** originalmente, donde se pueden ver:

Datos	Descripción	Tipo de valor
danceability	Bailabilidad del audio	Floating-point values
energy	Energía del audio	Floating-point values
key	Keys musicales	Integer values
loudness	Intensidad	Floating-point values
mode	Mode del audio	Integer values
speechiness	Palabras del audio	Floating-point values
acousticness	Acústicalidad	Floating-point values
instrumentalness	Instrumentalidad	Floating-point values
liveness	Vitalidad	Floating-point values
valence	Valencia	Floating-point values
tempo	Tempo	Floating-point values
type	Tipo de audio	String value
id	Identificación única del audio	String value
uri	Identificación única del audio de spotify	String value
track_href	Link referencia del audio	String value

analysis_url	Link de análisis	String value
duration_ms	Duración del audio en milisegundos	Integer values
time_signature	Indicación del compás	Integer values
genre	Género musical	String value
song_name	Nombre del audio	String value
Unnamed: 0	Variable de llave Unnamed: 0	Floating-point values
title	Título del álbum	String value

Posteriormente se hizo una limpieza de los datos que serían apropiados para los modelos a utilizar y la base de datos terminó con **12 columnas**, de las cuales 11 serían las características y 1 sería la etiqueta. Las 11 características serían *danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration_ms, time_signature y genre*. La etiqueta sería *genre.*

Datos	**Descripción**	**Tipo de valor**
danceability	Bailabilidad del audio	Floating-point values
energy	Energía del audio	Floating-point values
loudness	Intensidad	Floating-point values
speechiness	Palabras del audio	Floating-point values
acousticness	Acústicalidad	Floating-point values
instrumentalness	Instrumentalidad	Floating-point values
liveness	Vitalidad	Floating-point values
valence	Valencia	Floating-point values
tempo	Tempo	Floating-point values
duration_ms	Duración del audio en milisegundos	Integer values

time_signature	Indicación del compás	Integer values
genre	Género musical	String value

Antes de hacer la división de los valores, utilicé la función label "reprocessing.LabelEncoder", para tomar los 15 géneros que existen en la variable género y asignarles un número dentro de la misma variable. Lo que da el resultado de:

Género	**Asignación**	**Cantidad de audios**
Dark Trap	0	4578
Emo	1	1680
Hiphop	2	3028
Pop	3	461
Rap	4	1848
Rnb	5	2099
Trap Metal	6	1956
Underground Rap	7	5875
Dnb	8	2966
Hardstyle	9	2936
Psytrance	10	2961
Techhouse	11	2975
Techno	12	2956
Trance	13	2999
Trap	14	2987

Previo a implementar los modelos con los valores de mi dataset, dividí mis características y etiquetas utilizando la función **train_test_split**. En esta división, el conjunto de entrenamiento(test set) representa el **80%** de mi dataset, mientras que el conjunto de prueba

representa el otro **20%** restante. También se utilizó **random_state** para mezclar los datos y asegurarse de que no estuvieran en su orden original.

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

La principal razón por la que me he decidido en utilizar árboles de búsqueda y boosting de árboles de búsqueda fue porque buscaba aprender más sobre los árboles de decisión para el aprendizaje automático. Por lo que utilicé frameworks de Xgboost y sklearn para la creación de algoritmos para Machine Learning: **XG Boost y Decision Tree**.

Inicialmente, yo buscaba utilizar únicamente XG Boost, pero al ver la complejidad que conlleva este modelo, ya que utiliza muchos árboles de decisión, decidí también crear un solo árbol de decisión para mostrarlo como ejemplo y comparar los resultados de sesgo y varianza que se obtendrían con ambos modelos.

XG Boost

Es un algoritmo de boosting basado en árboles de decisión. El boosting implica construir varios modelos de forma secuencial, donde cada modelo corrige los errores del anterior, buscando encontrar una solución más precisa, en este caso, la predicción que más se acerque a lo que deseamos encontrar.

Los parámetros que utilicé primeramente ya están algo tuneados, ya que ya estuve experimentando con ellos previamente, para otras actividades:

Parámetros iniciales:

- n_estimators = Número de árboles de búsqueda = 200
- max_depth = Máxima profundidad de búsqueda = 12
- learning_rate = Tasa de aprendizaje(alpha) = 0.1
- objective = objetivo del modelo = 'multi:softprob' = busqueda de clasificación multiclase para encontrar la probabilidad de distribución de cada clase.

Posteriormente obtuvimos los siguientes resultados:

Muestras mal clasificadas: 2768

Género	precision	recall	fl-score	support
Dark Trap = 0	0.54	0.47	0.50	970
Emo = 1	0.73	0.74	0.73	341
Hiphop = 2	0.43	0.40	0.41	621
Pop = 3	0.34	0.11	0.15	98
Rap = 4	0.45	0.32	0.37	341
Rnb = 5	0.40	0.35	0.37	396
Trap Metal = 6	0.34	0.30	0.32	384
Underground Rap = 7	0.39	0.50	0.44	1192
Dnb = 8	0.96	0.98	0.97	599
Hardstyle = 9	0.91	0.94	0.93	619
Psytrance = 10	0.95	0.93	0.94	598
Techhouse = 11	0.89	0.91	0.90	568
Techno = 12	0.88	0.87	0.88	590
Trance = 13	0.84	0.89	0.86	562
Trap = 14	0.85	0.87	0.87	582

Obteniendo un promedio de exactitud de 67.29%

Después de esto para poder analizar más claramente el modelo utilicé cross validation, para poder analizar las habilidades de mi modelo y ver si había alguna mejora.

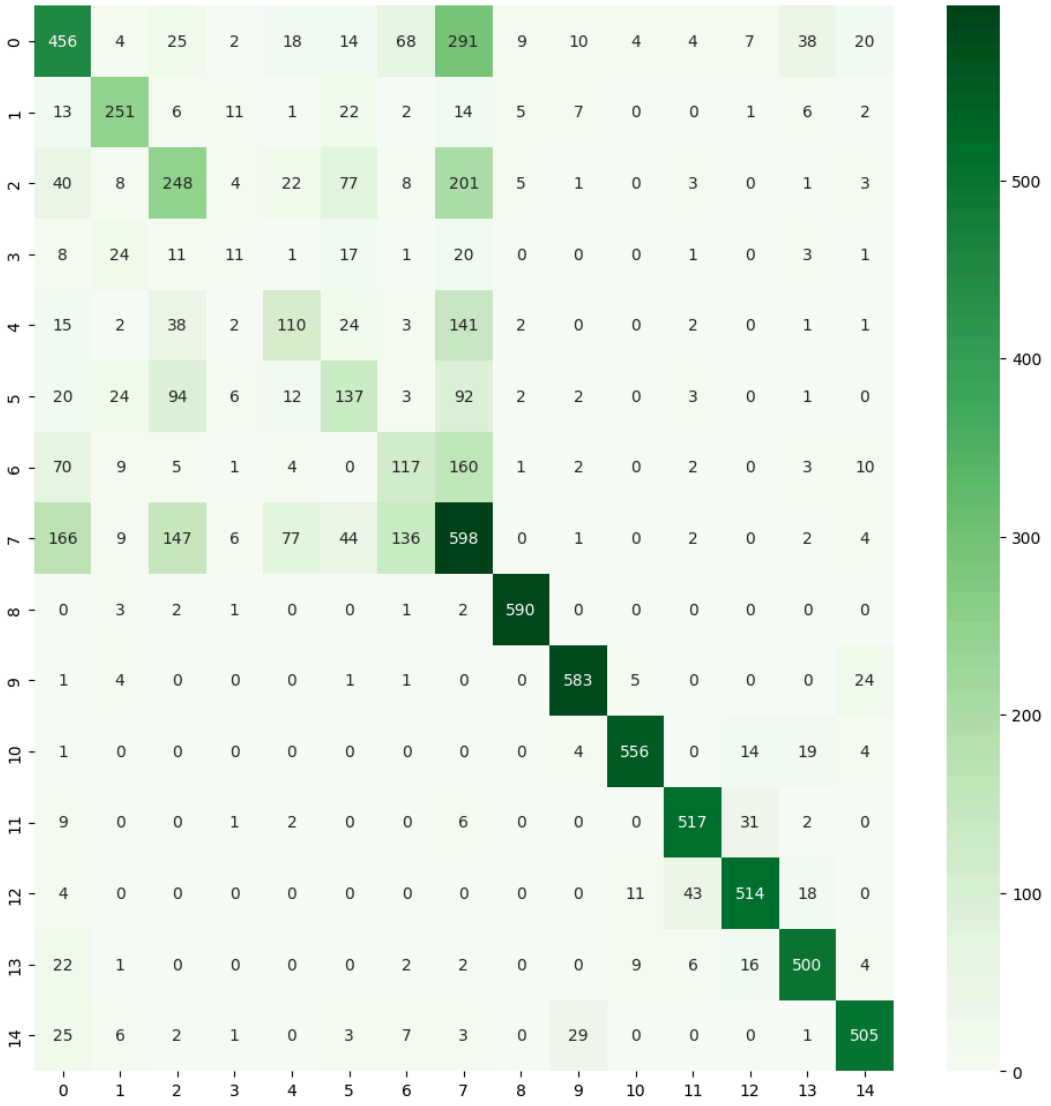
Al ejecutar el modelo con la función de cross validation, utilicé **10 folds**, lo que dividirá el Train set en 10 partes, donde la primera servirá como mi Validation set para los siguientes folds.

Cross-Validation Scores: [0.67501773, 0.66154573, 0.67099976, 0.66320019, 0.66769085, 0.66099291, 0.65886525, 0.66879433, 0.65271868, 0.67281324]

Mean Score: 0.6652638663294058

Standard Deviation Score: 0.00661841529228662

Finalmente hicimos una gráfica de confusión para poder comparar los valores reales del set de evaluación y los que fueron predichos por el modelo de boosting XG Boost:



Decision Tree

Algoritmo de ejemplo que utilizaremos para la clasificación de nuestros valores con árboles de decisión. El propósito de este modelo es predecir utilizando un árbol con nodos, donde cada nodo evaluará una característica diferente hasta llegar a una predicción final.
Aunque había utilizado un algoritmo de boosting de árboles de decisión, nunca me había tomado el tiempo para utilizar un árbol de decisión solamente, lo que me ayudó a comprender mejor este tipo de algoritmos.

Parámetros iniciales:

- max_depth = Máxima profundidad de búsqueda = 14
- random_state = número de orden aleatorio = 42

Posteriormente obtuvimos los siguientes resultados:

Muestras mal clasificadas: 3167

Género	**precision**	**recall**	**fl-score**	**support**
Dark Trap = 0	0.46	0.40	0.43	970
Emo = 1	0.58	0.56	0.57	341
Hiphop = 2	0.39	0.39	0.39	621
Pop = 3	0.10	0.08	0.09	98
Rap = 4	0.58	0.30	0.39	341
Rnb = 5	0.34	0.28	0.30	396
Trap Metal = 6	0.33	0.22	0.26	384
Underground Rap = 7	0.41	0.60	0.48	1192
Dnb = 8	0.96	0.94	0.95	599
Hardstyle = 9	0.84	0.87	0.85	619
Psytrance = 10	0.92	0.87	0.90	598
Techhouse = 11	0.84	0.84	0.84	568
Techno = 12	0.81	0.80	0.81	590
Trance = 13	0.77	0.80	0.79	562

Trap = 14	0.80	0.76	78	562

Obteniendo un promedio de exactitud de 62.57%

Después de esto para poder analizar más claramente el modelo utilicé cross validation, para poder analizar las habilidades de mi modelo y ver si había alguna mejora.

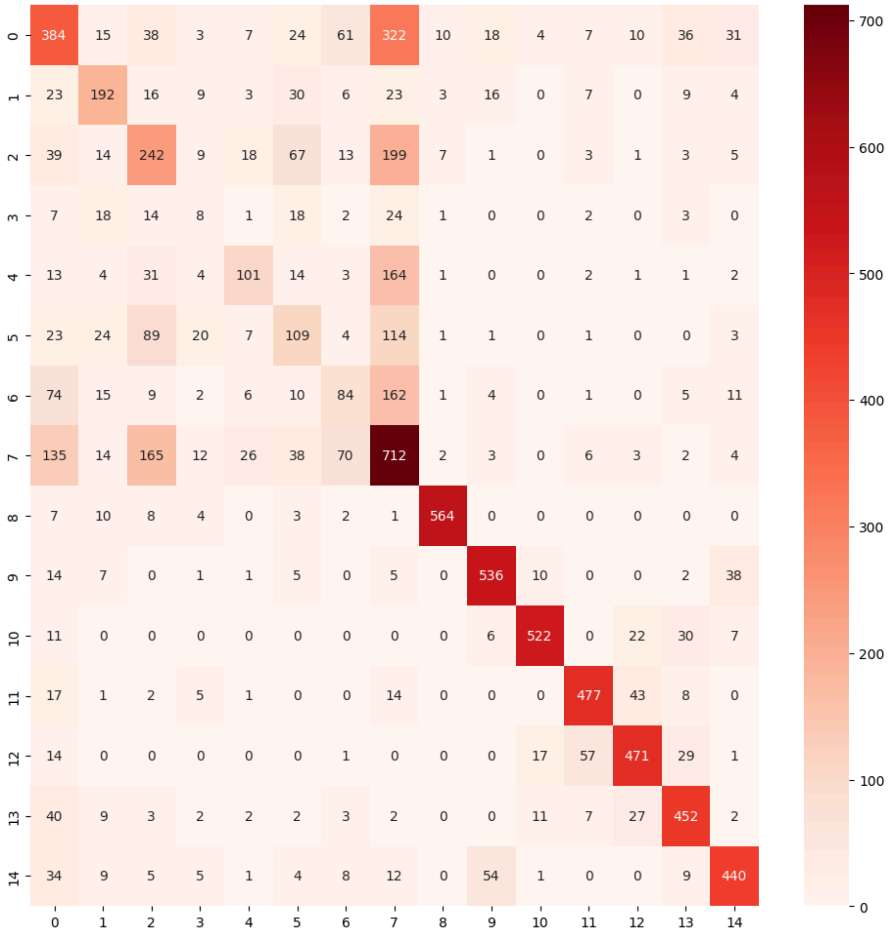
Al ejecutar el modelo con la función de cross validation, utilicé **10 folds**, lo que dividirá el Train set en 10 partes, donde la primera servirá como mi Validation set para los siguientes folds.

Cross-Validation Scores: [0.63932876, 0.62372961, 0.62798393, 0.63034744, 0.63223824, 0.62174941, 0.63120567, 0.62624113, 0.6144208, 0.62269504]

Mean Score: 0.6269940040665738

Standard Deviation Score: 0.006518491119475002

Finalmente hicimos una gráfica de confusión para poder comparar los valores reales del set de evaluación y los que fueron predichos por el modelo de boosting XG Boost:



Diagnóstico y explicación el grado de bias o sesgo:

XG Boost

El diagnóstico nos mostró que el bias que tiene nuestro modelo es bajo. Esto en comparación con el Decision Tree puede demostrar que este modelo ha mejorado, ya que al disminuir el error o bias, esto significa que se está ajustando de mejor manera al Train set. Normalmente podemos ver esto con algoritmos con mayor complejidad como XG Boost.

Bias: 1.1286865557862478

Decision Tree

El diagnóstico nos mostró que el bias que tiene nuestro modelo es relativamente alto. Al ser solamente un árbol y no tener todos sus parámetros afinados, el error puede llegar a ser mayor(especialmente si la profundidad de mi árbol también es muy alta). Demostrando que necesita ajustarse mejor al Train set.

Bias: 2.4622489510963903

Diagnóstico y explicación el grado de varianza:

XG Boost

El diagnóstico nos mostró que la varianza que tiene nuestro modelo es relativamente alta. Aunque se tunearon un poco los parámetros, al tener muchas clases hay una dificultad para el modelo al predecir los valores de forma correcta, lo que causa dentro de la gráfica gran varianza, pero no tanta como el Decision Tree.

Varianza: 3.07

Decision Tree

El diagnóstico nos mostró que la varianza que tiene nuestro modelo es alta. Esto significa que el resultado de las predicciones pueden llegar a ser bastante diferentes, además que se puede ver dentro de la gráfica anterior que hay una gran cantidad de géneros mal predecidos.

Varianza: 3.51

Diagnóstico y explicación el nivel de ajuste del modelo:

XG Boost

Este algoritmo nos ha mostrado que tenemos un gran caso de **overfitting**. Inicialmente al codificar mi modelo XG Boost yo utilicé mi Training set para comparar las predicciones de mi modelo, demostrando que su exactitud era muy alta, casi llegaba a los 100, pero al empezar a hacer predicciones con el Test set, la exactitud bajo más del 33%. Esta prueba, además de tener un bias bajo y una varianza relativamente alta, demuestra que tiene overfitting. Hay que tener en cuenta que XG Boost no es un modelo muy simple y puede ser bastante complejo en su utilización, ya que cuesta más tiempo en ejecutar que un solo árbol de decisión.

Decision Tree

A diferencia del modelo anterior, este algoritmo muestra un caso común de **underfitting**. La razón detrás de esta declaración es porque solamente se está utilizando un árbol de decisión para buscar la solución, lo que no deja campo para mejoramiento en comparación con el modelo de XG Boost que usa el método de boosting para corregir sus errores. En base a lo dicho previamente, el modelo cuenta con un bias relativamente alto y una varianza relativamente alta, lo que demuestra que tiene underfitting.

Detalles Finales para mejoramiento de modelos XGBoost y Decision Tree

Para los algoritmos de XGBoost y Decision Tree utilicé la función de Grid Search para afinar mis hiper parámetros y así poder llevarlos a la mayor precisión que pueden tener actualmente. Lo que me dió como resultado el máximo de exactitud al que pueden llegar mis dos modelos.

XG Boost

GS_XgBoost Classifier Score: 69.51%

Decision Tree

GS_DecisionTree Classifier Score: 62.89%

Bibliografía:

- Interview Query. (n.d.). Regression datasets and projects. Retrieved September 7, 2024, from https://www.interviewquery.com/p/regression-datasets-and-projects
- Kaggle. (n.d.). Dataset of songs in Spotify. Retrieved September 7, 2024, from https://www.kaggle.com/datasets/
- Wohlwend, B. (2020, October 15). Decision tree, random forest, and XGBoost: An exploration into the heart of machine learning. Medium. Retrieved September 7, 2024, from https://medium.com/@brandonwohlwend/decision-tree-random-forest-and-xgboost-an-exploration-into-the-heart-of-machine-learning-78c29c8d7b3b
- XGBoost Development Team. (2018). XGBoost parameters — xgboost 0.90 documentation. Retrieved September 7, 2024, from https://xgboost.readthedocs.io/en/latest/parameter.html