**MBD - Data mining**

**Birds songs data set**

**Sebastián Cuevas**

**Pol Gràcia**

**Introducción**

El análisis de datos es una herramienta fundamental en la era digital para extraer información valiosa a partir de grandes conjuntos de datos. En particular, el data mining se enfoca en identificar patrones y relaciones en los datos para obtener conocimientos relevantes. En este contexto, el presente trabajo de máster tiene como objetivo desarrollar un pipeline de entrenamiento de un modelo de machine learning y deep learning para predecir la especie de pájaro de un dataset basado en sonido.

Para alcanzar este objetivo, se ha realizado un análisis exploratorio de los datos (EDA) con el fin de comprender la distribución de las variables y detectar posibles valores atípicos o datos faltantes. A continuación, se ha desarrollado una pipeline de entrenamiento que incluye la selección de características relevantes, la división del conjunto de datos en entrenamiento y prueba, y la elección del algoritmo de aprendizaje automático más adecuado.

El modelo de machine learning y deep learning desarrollado utiliza como entrada características extraídas del sonido de los pájaros y se ha entrenado y evaluado utilizando técnicas de validación cruzada para garantizar su rendimiento en datos no vistos. Los resultados obtenidos en DL muestran una alta precisión en la predicción de la especie de pájaro, lo que demuestra la efectividad del modelo en la tarea de clasificación de especies de pájaros a partir de grabaciones de sonido.

El trabajo se ha realizado en lenguaje de programación Python, con las librerías de scikit learn para ML y pytorch para DL.

Se ha trabajado en grupo mediante el uso de github y e puede encontrar todo el código del proyecto en el repositorio:

<https://github.com/pgraciae/BirdsSong>

Cabe destacar que aunque en esta documentación se da una explicación genérica de todos los pasos realizados en la práctica, los notebooks también se encuentran con comentarios y detalles de las tareas realizadas.

**Dataset**

El dataset provisto consta de los siguientes datos:

* **metadata.csv:**

Un fichero tipo csv que contiene las especies, el nombre del audio, el nombre del spectograma y el segundo de inicio y fin del sonido del pájaro en cuestión.

No se ha usado ya hemos detectado errores de inconsistencia en él (p.e. hay valores que no existen en el dataset).

* **Audio\_files:**

Carpeta que contiene 20 subdirectorios con el nombre de cada especie. Dentro de cada subdirectorio encontramos dos tipos de fichero:

* **.mp3:** contiene el archivo de audio dónde se ha registrado un canto.
* **.txt:** contiene el segundo de incidio y el segundo de fin en el que un pájaro produce un canto. Siempre se relaciona con un fichero de audio.
* **spectrograms:**

Carpeta que contiene 20 subdirectorios con el nombre de cada especie. Dentro de cada subdirectorio encontramos ficheros de tipo npy correspondiente a un espectrograma de sonido.

Para trabajar con los datos, hemos creado dos clases ‘Dataset’ (una para ML y otra para DL) que nos permitirán leer los datos secuencialmente sin tener que cargar en memoria todos los archivos.

Para hacer esto, cúando se crea el objeto dataset, se crea un diccionario de metadatos que recoge todos los registros con los nombres de la especie y la ruta hacia el espectograma o audio. En el momento en que se hace un ‘getitem’, se llaman las relativas funciones para cargar el fichero en cuestión.

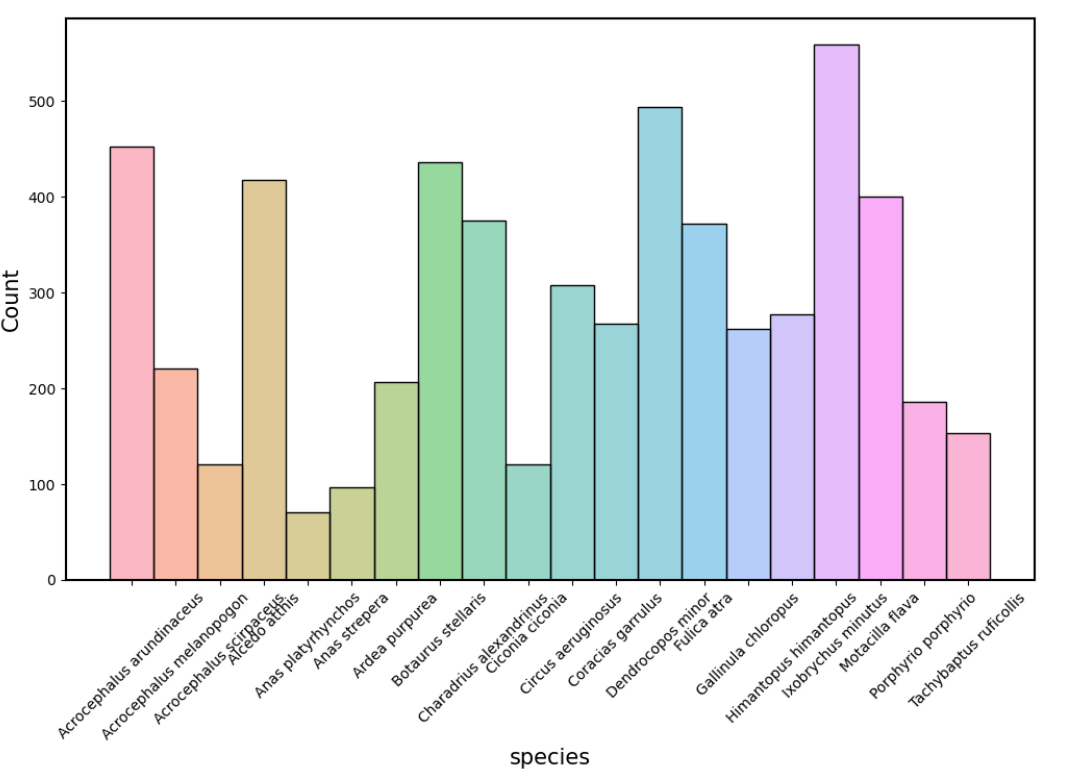
Para el modelo de DL también se ha creado un dataloader que permite separar el dataset en train, test y validation. Siempre se usa un stratified random split para poder asegurar que tenemos datos de todas las calses en todos los dataloaders.

**EDA**

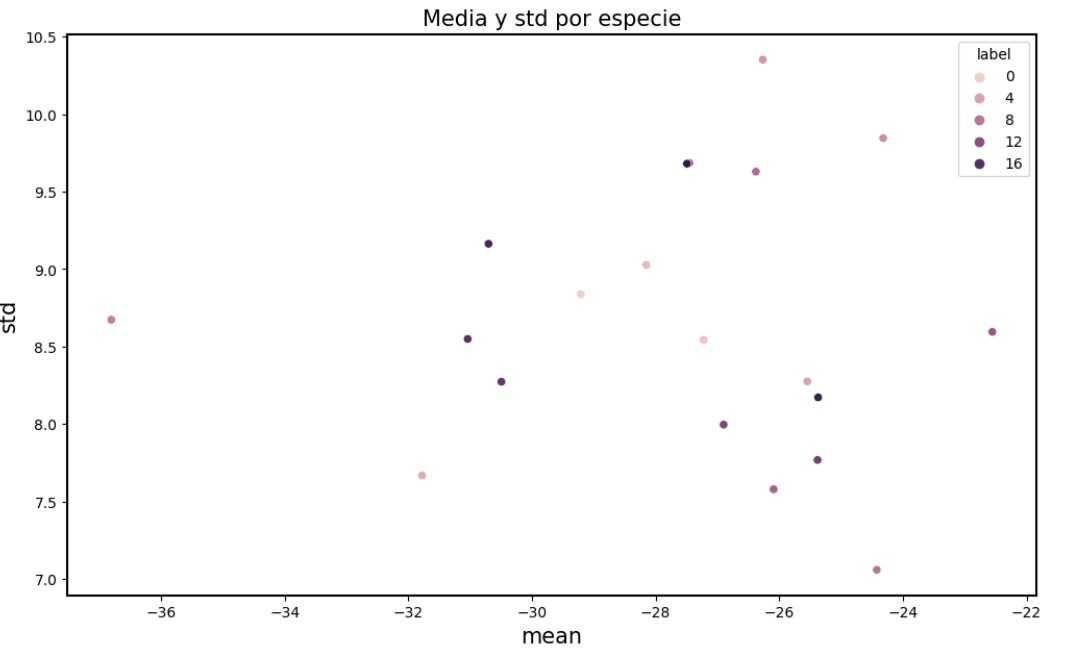
Se ha realizado un notebook de EDA para hacer un estudio de la distribución de las clases. Se puede encontrar una descripción más detallada de todo el EDA en los comentarios del mismo notebook ‘src/EDA.ipynb’

No se han realizado las típicas operaciones de busca de nans o outliers en el dataset debido a la naturaleza del mismo y se ha asumido que no hay.

Nuestro foco en el EDA ha sido entonces encontrar la distribución de las clases y los parámetros para la normalización del dataset.



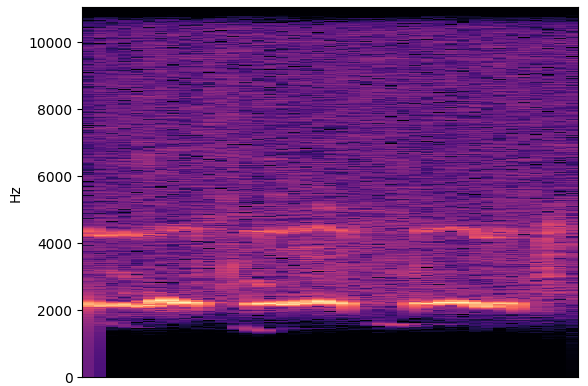
También se ha realizado el cálculo de la media y la std genérica de todo el dataset para hacer la normalización y se ha hecho un ejemplo de la normalización sobre una imagen. También se ha estudiado si estas medias y std son distintas por especie.



1. **Machine Learning**

Inicialmente para el ML se iba a trabjar con audios y espectros creados aparti de los tiempos de inicio que ofrecia los datos y dacion 1 segundo pero se econtro con una problemática de que el tamaño de espectro iba a varia en determinados audios debido a su duracion corta y complicaria a normalizar y procesar la informacion.

Se aprecia una imagen del espectro que ofrecia con los terminos mencionados.



El tamaño 41, va a variar en algunos audios.



Por tal motivo se procedió a trabajar con los espectros directamente.

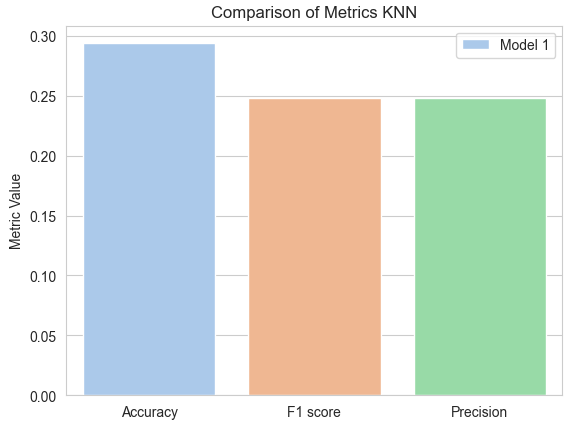
Para aplicar algún modelo de machine learning previamente se tratará los datos. Inicialmente se creó un dataset para train y otro para test con el factor del nombre del audio y el nombre del pájaro para que en ello mas adelante los espectros de un audio no sean divididos y forman parte ya sea del grupo train o del grupo test.

Una vez ya teniendo se para cada audio se procede a crear los dataset de train y test con los espectros.

Después, se balancea el dataset de train para con el método undersampling para que los tipos de pájaros con gran cantidad de espectros no tenga un peso muy desproporcional al grupo minoritario.

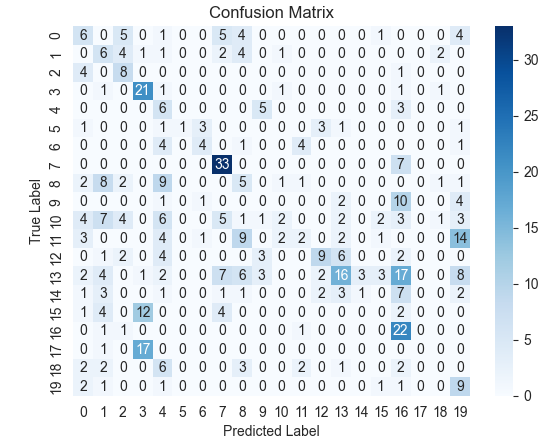
El modelo para machine learning aplicamos un KNN con una constante de k =5 con mejores resultados, aunque no sean muy elevados con un accurancy de 0.29.

En el siguiente grafico se muestra la comparación de las métricas obtenidas:



Se puede considerar como un punto de partida para seguir trabajando en la mejora del modelo. Es importante tener en cuenta que el accuracy depende de varios factores, como la cantidad y calidad de los datos de entrenamiento, la elección del modelo y los parámetros utilizados. Por lo tanto, es posible mejorar el accuracy del modelo realizando ajustes en estas áreas. En resumen, el modelo KNN utilizado en este informe tiene un margen de mejora, y se pueden explorar diferentes técnicas para mejorar su desempeño en la clasificación de imágenes.

También se muestra su matriz de confusión para apreciar la relación de la predicción con la prueba:

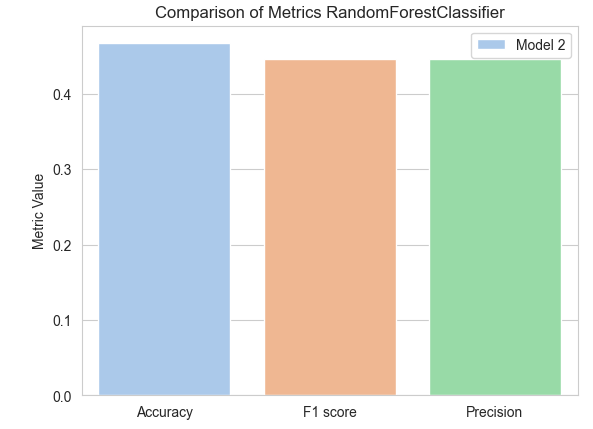


Mas detalle se lo puede encontrar en el repositorio: <https://github.com/pgraciae/BirdsSong/blob/master/Opcion_audios.ipynb>

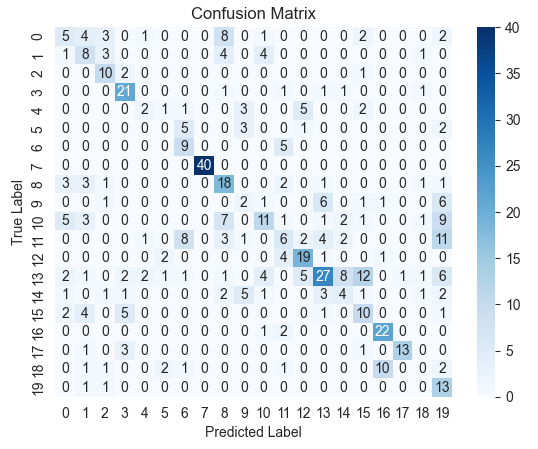
Se concluye que el modelo no es lo bastante robusto para realizar una predicción satisfactoria por lo que se continua con otro modelo.

1. **Ensemble**

En modelo se realiza un Random Forest con sklearn para apreciar un mejor resultado.



Se evidencia un mejor resultado con el detalle, pero no logra hacer lo bastante efectivo debido a la gran cantidad de información por lo que para imágenes y audios es recomendable utilizar Deep learning.



Con la matriz de correlación la predicción es mas acertada pero no a nivel requerido, aunque se aprecia la mejora a comparación del modelo anterior. El detalle se encuentra en el código:

<https://github.com/pgraciae/BirdsSong/blob/master/Espectros_ML_Ensemble.ipynb>

1. **Deep Learning**

Para entrenar el modelo de Deep learning, se usarán los espectrogramas del dataset, ya que modelos como las Redes Neuronales Convolucionales (CNN), están diseñados para extraer automáticamente características significativas de las imágenes. Dado que los espectrogramas son esencialmente imágenes, las CNN pueden aprender patrones complejos en los datos y utilizarlos para hacer predicciones precisas.

Se ha usado la librería pytorch y en concreto pytorch lightning para realizar el entrenamiento.

La conexión entre las partes es compleja y la exhaustiva descripción de todas las componentes y pipeline podría ser una tarea de tfm así que en este apartado se dará una explicación más genérica y en la evaluación de entrevista se puede entrar en tanto detalle cómo el evaluador desee.

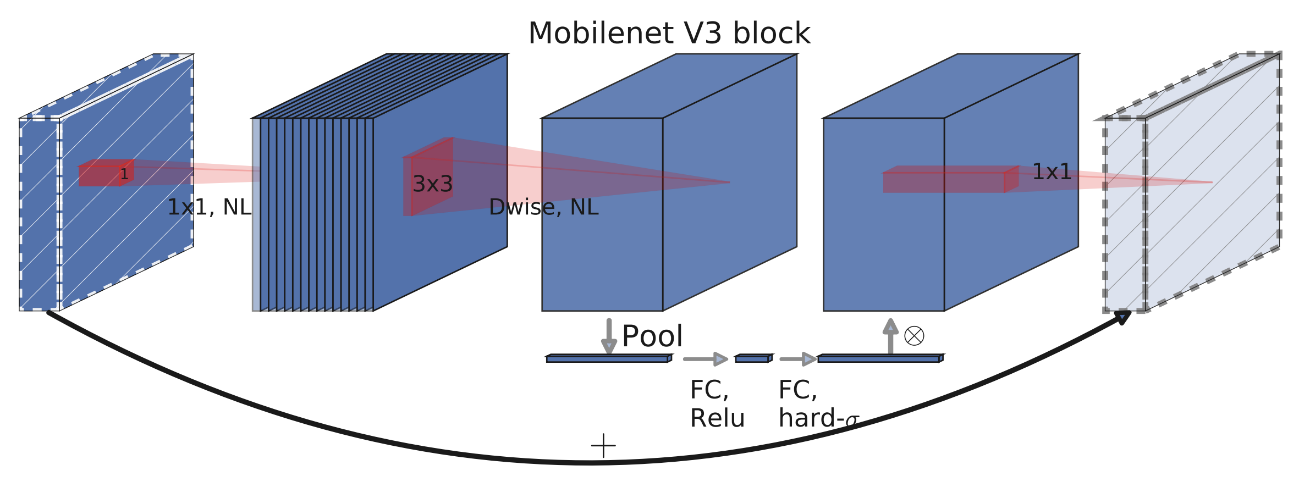
Los objetos para el entrenamiento se han organizado de la siguiente manera:

* **Dataset:** Tal y cómo se ha explicado anteriormente, carga los datos de un espectrograma y su label cúando se produce una llamada a un item .
* **Datamodule:** Sirve para separar el dataset en train, test y validation y obtener un dataloader para cada. Se llamará recurrentemente con el trainer.
* **LightingModule:** Modelo de entrenamiento. En este caso se ha realizado una abstracción que perimte cargar y usar cualquier modelo del módulo timm (Pytorch Image Models).
* **Callbacks:** Módulo que permite registrar en un logger (este caso wandb.ai) las métricas definidas. En este caso es la ROC y una confusión matrix.
* **Trainer:** Objeto de pytroch lighning que se encarga del bucle de entrenamiento y llamada a todos los objetos definidos.

En el pipeline de entrenamiento, en primer lugar se define el datamodule, que crea los dataloaders de training, test y validation. También se definen transformations, que se usan para normalizar los datos, cogiendo los parámetros extraídos en el EDA. No se han definido más transformaciones cómo rotaciones, crops o negativos ya que los espectrogramas siempre mantienen la misma estructura de imagen. El batchsize definido del modelo es de 64, se ha visto que a medida que se augmentaba el batch size, mejoravan todos los parámetros y el modelo convergía más rápidamente, pero por potencia computacional no se ha podido analizar batchs más grandes.

A continuación, se define el modelo. En este caso hemos usado la red ‘mobilenetv3\_small\_100’ preenetrenada y haremos finetunning sobre ella, reentrenando todas sus features pero aprovechando los pesos iniciales. Se cambia la classification layer y se imputa una linear con 20 terminaciones (por las 20 clases de pájaros). No hemos usado redes más grandes ya que por cuestiones técnicas no disponíamos de GPU ni TPU para entrenar los modelos y hemos preferido utilizar un modelo pequeño.

La arquitectura es la siguiente:



Es una arquitectura relativamente pequeña con 1.5 M de parámetros a entrenar que se caracteriza por su veloz inferencia en la predicción incluso en cpu.

Una vez el modelo esta definido, se crean los callbacks. Se define un checkpoint, para guardar un snapshot del modelo en cada epoch y se define el modelo custom ‘ImagePredictionLogger’ que hemos creado y registra la roc y la confusión matrix.

A continuación, se define el trainer. Se definen 15 epochs de entrenamiento y los callbacks y logger.

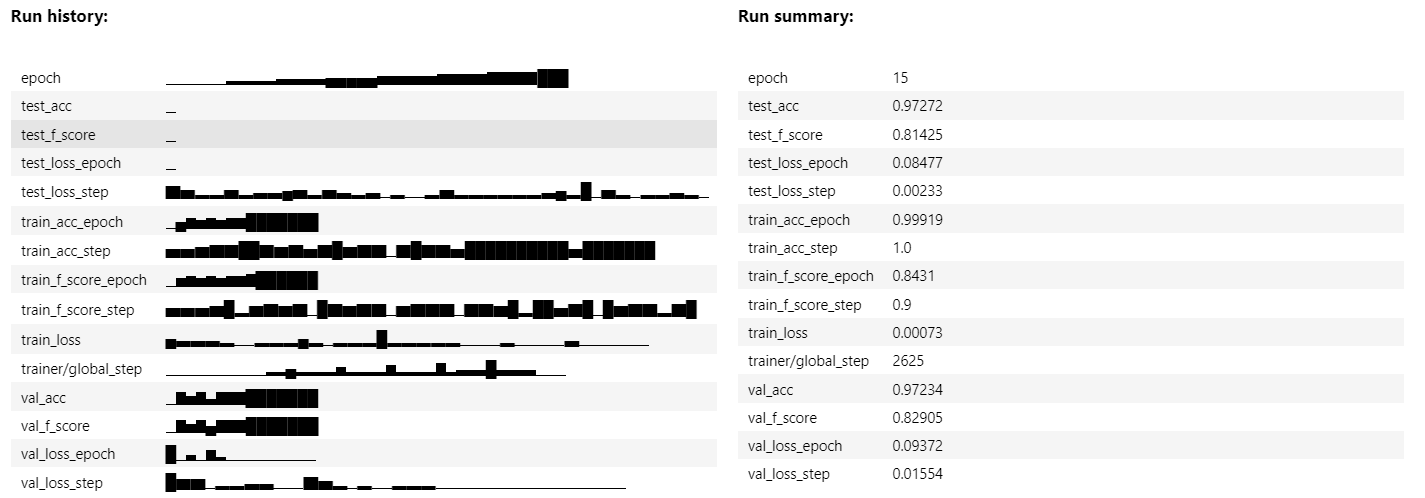
Finalmente, se hace un fit del modelo con el data module y se analizan los resultados en el logger (wandb). Recomendamos acceder para ver mejor los resultados y de manera interactiva.

**Resultados**

A continuación se presentan los resultados obtenidos con el entrenamiento. Se recomienda mirar directamente los resultados y gráficos interactivos en el link:

<https://wandb.ai/pgraciae/birds/runs/>

Al final se han obtenido los siguientes resultados:



Se ha obtenido un muy buen resultado de accuracy, tanto en el train cómo en el validation cómo en el test.

En cuanto a la f1-score, se ha obtenido un muy buen resultado tanto de test cómo de validation, lo que indica que el modelo no está muy overfitted.

En conclusión, estamos contentos con la performance del modelo y creemos que con más potencia computacional se habrían podido usar modelos más grandes y quizá se habrían obtenido incluso mejores resultados.

Es necesario mencionar que en los plots de la validación (roc, CM) los datos representados sólo son los primeros 64 elementos correspondientes al primer batch del test.

