Team 16

Etienne Leroy	Pablo Bertaud - Velten	Matei Atodiresei
- Dataset Visualisation - Basic CNN - Complex CNN - Hyperparameter Tuning	- Data Preprocessing - Dense Model - Decision Tree (2nd dataset)	- Decision Tree (1st dataset) - KNN models - Random Forests

Highlights (4–6):

1. Every dataset is different which often leads to unexpected results, thus we need to keep an open mind and try different models.
2. Overfitting is a big issue that arises from either too small of a dataset or too complex of a model.
3. Decision trees were much more accurate on tabular data (good for splitting, especially with features extracted from the audio) than image data from spectrograms.
4. CNN performed worse than the decision tree.
5. Deep learning should be the last solution to a problem due to its intensity and complexity.

Brief Report:

The scope of the project was to implement and determine which of the following algorithms in either their basic or complex form; **K-Nearest Neighbors, Decision Trees**, **Random Forests , Dense neural network**, or **Convolutional neural network**, performed best in **classifying** the GTZAN dataset.

We proceeded with the project by **preprocessing** the data into datasets, and then **Numpy** arrays or Pandas **dataframes** to be used as input for each of our algorithms. We were left with two main datasets to be used, a spectrogram dataset, used for the neural networks, k-Nearest Neighbors, and decision trees, and a tabular dataset, used for Decision Trees, Random Forests.
Once we had the data ready to use, we implemented our various models.

The K-NN is a simple model that performed pretty well compared to the other models after we implemented some improvements: using **PCA**, using **HOG** (Histogram of Oriented Gradients) and choosing K to be 1. We believe this choice of K is best because the dataset is **noisy** and it captures the **complex boundaries** better.

Decision trees on both datasets showed very different accuracies: up to 42% for the first dataset, but 65% for the tabular data. This is due to the fact that the tabular data had **precise features** that were "handcrafted" directly from the songs, and helped to capture the characteristics of the songs. On the other hand the image data was **high-dimensiona**l and so splitting on features was hard for decision trees.

The Random Forests helped with the accuracy of the tabular data as they **prevented overfitting** that could arise from a single tree. Our best accuracies came from Forests using tabular data (86%), which combines a dataset suitable for trees with the good **generalization** that comes from forests. This improvement was not present on the first data set, as it was too small to make the decision tree overfit.

The **Dense neural network** served as a baseline for all the other **CNN's**, a point to compare the more complex CNN's. The first basic CNN used one **convolutional layer** with **ReLu** activation, followed by **batch normalization**, **max pooling**, and **dropout**. The complex CNN used 3 convolutional layers of the same format. All NN's had **'softmax'** activation on the final dense layer, and were compiled using the **'Adam'** optimizer with **categorical cross-entropy** loss, and the **accuracy** metric. Contrary to expectation, the CNN's performed very poorly, which we attribute to the small size of our dataset and the overfitting of the more complex model. This is illustrated by the fact that even after **hyperparameter tuning**, the complex CNN performed worse than the basic CNN. Regardless of how many methods we used to prevent overfitting, accuracy barely increased. Training time is also a problem since these complex models take hours to train.

References:

https://scikit-learn.org/stable/
https://www.tensorflow.org/api_docs/python/tf
https://keras.io/api/
https://keras.io/api/layers/regularization_layers/dropout/
https://keras.io/api/layers/normalization_layers/batch_normalization/
https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f
https://keras.io/keras_tuner/