# Age prediction from file audio

Giorgio Bono
*Politecnico di Torino*
Turin, Italy
s343572@studenti.polito.it

Claudio Camolese
*Politecnico di Torino*
Turin, Italy
s344788@studenti.polito.it

***Abstract*—This project is about predicting a person's age from their voice using machine learning. Features representing sound characteristics are extracted from each audio file. Different machine learning methods are tested. The goal is to find the best features and models for accurate age prediction.**

## I. PROBLEM OVERVIEW

In this project, it is used a dataset of audio files. The audio files are subsequently divided into two folders: a development folder and an evaluation folder.

The development folder contains the audio files that our model will be trained on, while the evaluation folder contains the audio files that our model will be tested on.

The training dataset consists of 2,933 audio files, while the testing dataset contains 691 audio files.

Some information regarding the audio files has already been preprocessed, and the extracted data is contained within a $development.csv$ file for the train set and in a $evaluation.csv$ for the test set.

Each audio file belongs to a different person and has a different duration.

Inside this file, there are informations obtained from the audio files, such as the gender, ethnicity, and some values derived from preprocessing using the *librosa* library.

In the development file, no null values were find, so no method for filling null has been used.

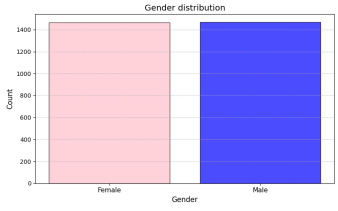


Fig. 1. Gender distribution

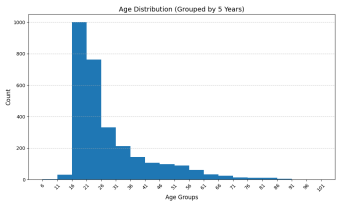


Fig. 2. Age distribution

As can be seen from the graph (1), the gender distribution within the dataset is fairly uniform.

The same cannot be said for age (2) where there is a high number of audio files belonging to teenagers (ages between 16 and 21).

A correlation matrix is plotted in (3). From this plot it can notice that the most of the features are not correlated. However, some of them are strongly correlated. These features are the ones related to *silence duration* and *number of words/characters* due to their implicit correlation.

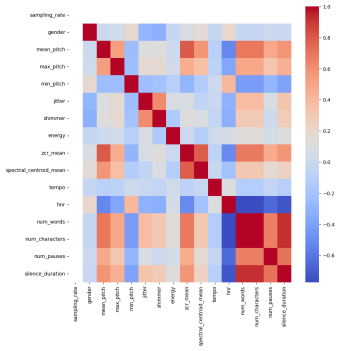


Fig. 3. Initial correlation

Another singular feature of our initial dataset is the *ethnicity*. It contains 165 different values. Analyzing this feature is possible to plot the following distribution:

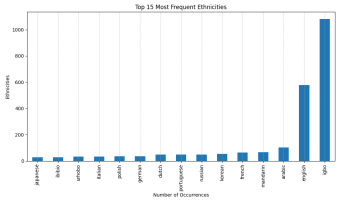


Fig. 4. Ethnicity

It can observe that there are many different ethnicities; however, only some of them have a frequency high enough to be considered relevant.

Now the processing of outliers is presented.
In the following plots, the features with the most variance are shown, both for train and test set.

As we can observe from the figures (5) and (6), both the train and test sets exhibit the same distribution of outliers. This

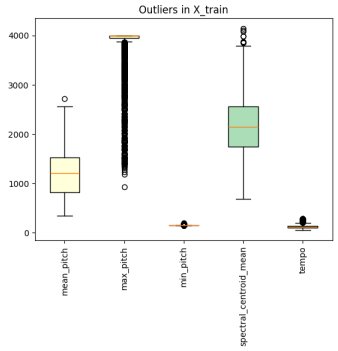


Fig. 5. Train outliers

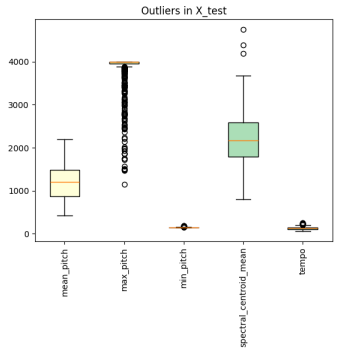


Fig. 6. Test outliers

indicates that these outliers cannot be ignored, as disregarding them could lead to overfitting the training data and result in a model that lacks generalization.

## II. PROPOSED APPROACH

### A. Preprocessing

To train a machine learning model on data, it is necessary to convert strings into numbers. For the ethnicity column, this was achieved using *OneHotEncoding*. This method for each unique category, creates a binary vector where the position corresponding to the category is marked as 1 all the other positions are marked as 0.

However, since we have a very large number of ethnicities, and most of them have very low frequencies, all possible ethnicity values that do not appear among the top 15 most frequent ones are categorized as an *ethnicity_other*.
Other features have been extracted from audio files in order to have more categories to work with.

The feature selected are:

- Spectrogram: representing the overall intensity of frequencies over time.
- Bandwidth: indicating the width of the spectrum in which the majority of the signal's energy is concentrated.
- Contrast: contrast, which measures the difference in amplitude between peaks and valleys in a sound spectrum.
- Flatness: quantifying how noise-like the signal is (higher values indicate noisier signals).
- Rolloff: indicating the frequency below which a given percentage of the total spectral energy lies.
- Chroma: Average chroma feature, which captures the harmonic content of the audio signal.
- Mfcc: which represent the short-term power spectrum of sound, commonly used for speech and audio processing.
- $\delta_{\text{mfcc}}$: representing the rate of change of the spectral features.
- Freq: is the Fast Fourier Trasform of the signal.
- Skew: describing the asymmetry of the signal's frequency content.
- kurtosis: representing the "tailedness" or sharpness of the frequency content.

Particularly important is the contribution of mfcc and $\delta_{\text{mfcc}}$. Specifically, for the mfcc feature, the first 100 coefficients were extracted, while for $\delta_{\text{mfcc}}$, we extended up to the second order.

This means that for MFCC we retain 100 coefficients from the MFCC computation, capturing more details from audio file. Talking about $\delta_{\text{mfcc}}$, using infos till second order, means we want to capture how MFCC values are changing in consecutive frames and the accelaration of this changes.
This informations play a crucial role since younger speakers tend to have more flexible vocal cords, resulting in faster and more varied changes in speech dynamics, which are reflected in distinct $\delta_{\text{mfcc}}$ patterns. On the other hand, older speakers exhibit slower, smoother, and less abrupt transitions in vocal dynamics.

A scaler is then applied to the dataset to distribute the variance among the data. There are two possible alternatives for preprocessing: StandardScaler and MinMaxScaler.

StandardScaler centers the mean of the data at 0 and sets the standard deviation to 1, while MinMaxScaler rescales the data so that they fall within a specific range.

MinMaxScaler is often used to standardize ranked values in a dataset. On the other hand, StandardScaler works very well for datasets with many features that have different levels of variance, as in our case.

From StandardScaler, the next step is to analyze the data using Principal Component Analysis (PCA). PCA is a technique used to reduce the number of features while preserving as much information as possible. It does this by representing the dataset through new components derived from a linear combination of the original features, effectively suppressing noisy components.

However, as shown in the figure, too many principal components (PCs) would still be required to explain at least 90% of the data. This means that we would not actually be achieving significantly better results.

### B. Model selection

Several regressors were considered for training, including *RandomForest*, *DecisionTree*, *MLP*, *SV*, *GradientBoosting*, *HistGradientBoosting* and various linear regressors.

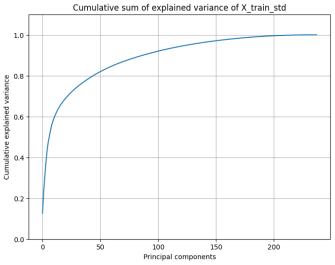


Fig. 7. PCA on standardized dataset

Initially, they were tested with default values. This allowed us to understand which models would be more performant for potential fine-tuning.

To test these models in a more robust way and improving generalization, *K-fold* is used.

*K-fold* cross-validation is a resampling technique used to evaluate machine learning models more reliably than a simple train-test split. The dataset is divided into k equal-sized subsets, or folds. The model is then trained on k-1 of these folds and evaluated on the remaining fold. This process is repeated k times, with each fold serving as the evaluation set once, while the others are used for training. For each run in the kfold we select the model with the best metrics, in our case the $RMSE$.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{n}} \tag{1}$$

where $\widehat{y_i}$ is the prediction and $y_i$ is the ground truth.

The value of $k$ to choose for performing *K-Fold* is selected by studying how the $RMSE$ changes as $k$ varies.

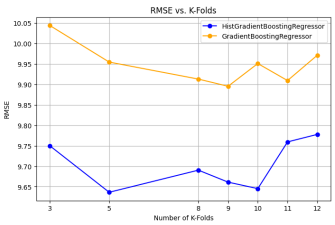


Fig. 8. k in *K-Fold*

As can been seen from (8), the value of $k$ was chosen by calculating which $k$ would result in the best RMSE. As a result, for the two models we selected, we obtained for *HistGradientBoostRegressor* $k = 5$ and for *GradientBoost-Regressor* $k = 9$.

Below, the models will be analysed in detail based on their functionality.

- *DecisionTreeRegressor*: starts by dividing the data into subsets, using a feature that best reduces the error. This process continues until it reaches a point where no further splits can improve the prediction, or certain limits are reached. The final prediction is made by averaging the values of the data in the leaf nodes.
- *GradientBoostingRegressor*: builds an ensemble of models sequentially, aiming to minimize the error of previous models. It does this by adding weak learners, usually decision trees, to correct residuals.
- *HistGradientBoostingRegressor*: is a variant designed for efficiency, particularly when working with large datasets. It follows a similar approach to the GradientBoostingRegressor but uses histogram-based methods. Instead of processing continuous features directly, it discretizes them into bins, creating histograms to speed up computations.
- *MLPRegressor* is a regression model based on neural networks in scikit-learn, built to learn complex, non-linear patterns in data. The model has an input layer, one or more hidden layers with functions to transform data, and an output layer that gives continuous predictions. It improves by adjusting the connections between layers using backpropagation and optimization techniques like Adam or SGD.

### C. Hyperparameters tuning

*Fine-tuning* refers to the process of making small, careful adjustments to a model that has already been trained on a general task, to enhance its performance for a specific task or dataset.

This part is extremely important since specific parameters associated with a particular model could lead to highly overfitted results on our test dataset and reduced generalization of the model.

The grid search did not yield optimal results on the leaderboard, as it led to overfitting of the training set. Therefore, the best choice to ensure the generalization of the results was to implement the model with the default parameters.

It is important to consider that although numerous features were extracted from the audio files, only a subset of these actually contributes to the task of predicting a person's age. Consequently, a selection of the extracted features is made to perform the regression.

## III. RESULTS

In our code, a random state was added to ensure data reproducibility. The best-performing model was `HistGradientBoostingRegressor`, achieving an RMSE of 9.278 after removing certain features from the dataset, as they were noisy and had minimal influence.

The `HistGradientBoostingRegressor` with all the features extracted from audio files achieved an RMSE of 9.510. This experimentally demonstrates the importance of feature selection from the audio files.

## IV. DISCUSSION

It has been experimentally demonstrated that the proposed approach outperforms the baseline for this specific task. However, the problem of predicting a person's age using a regressor is particularly challenging without resorting to more advanced methods, such as deep neural networks or pre-trained models. In our experience, not only is the choice

of model and its parameters crucial, but so is the selection of features used to train the model. Even a different combination of features can lead to completely different results.

We also noticed that a model that performs better in our local evaluation does not always yield better results than a model with slightly lower performance. This is likely because the model is able to learn very well from the training data, but the ”hidden” data in the leaderboard contains values that differ significantly from the data on which the model was trained, leading to worse performance. This is the case, for example, with the HuberRegressor, which achieved an RMSE of 7 locally but performed poorly on the leaderboard.

Among all the features extracted, the most significant ones were MFCC and its temporal variations. Additional features could have been extracted from the audio files to provide a more complete view of the data.

Expanding the grid search for hyperparameter tuning could allow for a broader exploration of parameter selection, but this must be done carefully to avoid overfitting.

Considering other regression models could also provide insights into how different approaches perform. The combination of these three strategies could potentially lead to better results than those we achieved.

In the end, we could have found just a sub-optimal solution using a discrete range of features and models.