Age Prediction Using Audio and Demographic Features

Geard Koci, Davide Omento
Politecnico di Torino
Geard Koci s328626, geard.koci@studenti.polito.it
Davide Omento s330764, davide.omento@studenti.polito.it

***Abstract*—In this report, we present a potential approach to building a regression model for predicting a speaker's age based on their vocal characteristics. A key challenge we encountered was the preprocessing of the dataset, which played a crucial role in the model's performance. By applying effective preprocessing techniques, we achieved significant improvements in the model's accuracy and predictive capabilities.**

I. Problem overview

The proposed project focuses on estimating the age of a speaker from an audio file containing a spoken sentence, using the information extracted from the audio.

The dataset consists in 3624 samples divided in:

- a *development set* containing 2933 samples and for each one of them the target age;
- an *evaluation set* containing 691 samples.

The first step is to analyze the development set, as this is the dataset we will use to train the regression model. During this analysis, we should identify any anomalies, such as missing values. The development set contains 19 columns, along with an additional column for age, while the evaluation set includes only the 18 columns.

Let us analyze these features in detail:

- *Id*: an increasing number for every row, we will use it as the index during preprocessing;
- *Sampling rate*: fixed at 22050 Hz for all recordings in both the development and evaluation sets;
- *Gender, Ethnicity*: the gender and ethnicity of the speaker, representing the only two categorical features in the dataset;
- *Mean, Min, and Max Pitch*: the average, minimum, and maximum pitch of the speech signal, measured in Hz;
- *Jitter, Shimmer, Energy, ZCR Mean, Spectral Centroid Mean, Tempo, HNR*: audio-related features capturing variations, amplitude, and other characteristics of the speech signal;
- *Number of Words, Number of Characters, Number of Pauses, Silence Duration*: these features quantify the linguistic content of the audio, such as the total number of words, characters, pauses, and the duration of silence;
- *Path*: the file path to the corresponding audio file in .wav format.

Here we propose the distributions of the energy and shimmer in the development set:

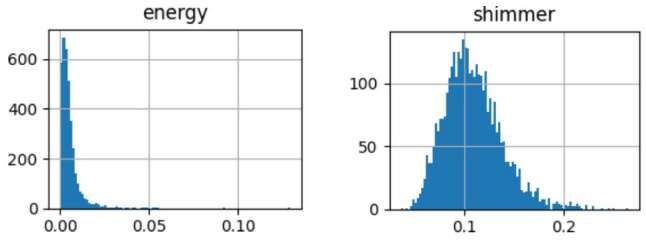

Fig. 1. Shimmer and energy histograms.

II. Proposed approach

A. Preprocessing

We first saved the development dataset in a pandas dataframe, then we defined a function to extract additional features directly from the audio file, complementing the ones already available. In particular, using the *librosa* library, we computed several summary statistics from the spectrogram (mean and standard deviation, dividing the Mel spectrogram into 40 bands), the number of words per second, the mean and stds of 13 formant frequencies, respectively *mfcc_mean* and *mfcc_std*, and the voiced/unvoiced ratio. These features were then added to the dataframe.

Here we can see for example the words per second and the standard deviation of the first mfcc:

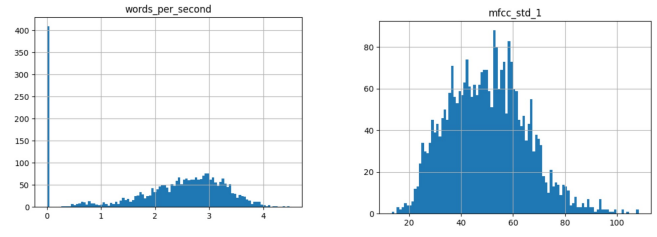

Fig. 2. Words per second and mfcc histograms.

Subsequently, we proceeded to clean up the dataset. One of the features was the path of the audio file. However, after extracting the relevant features from the file, we removed this column, as it was no longer useful. There were no missing values in the dataset, so no rows were dropped from the original.

Next, we encoded the *gender* feature using *One Hot Encoding* through the *get dummies* function and converted all

the elements in the dataframe to numeric format, if they were not already. We also removed any brackets present in the dataset, ensuring that all columns contained only floating point numbers. Note that during this first encoding we did not encode the ethnicity column, because we will use it later on.

We repeated the processes for the test set and furthermore, we corrected a typo in the evaluation dataset, where the word *gender_female* was incorrectly spelled *gender_famale*.

We then proceeded to compute an initial approximation, without the *ethnicity* feature, of our target variable, the speaker's age, using a *RandomForestRegressor* with the following parameters: {*criterion = 'poisson'*, *n_estimators = 250*, *max_depth = 20*, *max_features = None*}. Thanks to this, we were able to include the *age* feature in the evaluation set, even if only as an estimation. This step was important because without the addition of the *age* feature, it would not have been possible to concatenate the two datasets and simplify further processing.

One of the biggest challenges was managing the categorical feature *ethnicity*, due to the high variability in the dataset, which included 221 different ethnicities. This feature plays a crucial role in age estimation, as people from the same region of the world tend to have similar speech patterns. As a result, *ethnicity* becomes a valuable factor in capturing regional and cultural variations in speech that are related to age.

Given its importance, we decided to explore *ethnicity* further by deriving a new feature, called *ethnicity_behavior*.

To achieve this, we made a key enhancement by deriving new features from the existing ones. Since the goal is to estimate the speaker's age, for each sample, we computed the mean and the standard deviation of every non-categorical feature in rows where the *age* value was within a distance of 10 from the age of the current sample. These new features were then added to the dataset, with the new columns named *feature_mean* and *feature_std*, where *feature* represents the name of the respective non-categorical feature.

From these newly derived features, we were able to extract an additional feature called *deviation*. This feature is computed, for each non-categorical feature and for each row, as follows:

$$deviation(f, s) = \frac{x_f^s - f\_mean}{f\_std}$$

where x_f^s represents the value of the feature f for the sample s, and *f_mean* and *f_std* are the corresponding features computed in the previous step. This allowed us to standardize the features relative to their local distribution, providing a normalized measure of deviation for each sample.

The feature *ethnicity_behavior* was then computed as the average deviation of other features aggregated by ethnicity. This allowed us to capture underlying patterns of variation associated with ethnicity in a way that complements the original feature. By leveraging these patterns, we aimed to improve the accuracy and robustness of the age estimation model. All the columns generated for computing the *ethnicity_behavior* have now been dropped to prevent overfitting in our models.

In addition to the large number of different ethnicities in the dataset, we observed that most of the ethnicities in the evaluation set did not appear in the development set. This could pose a significant challenge for our models, as they may be poorly trained on features, such as ethnicity, that they have never encountered before.

Therefore, a key step was to group similar ethnicities together, ensuring that all ethnicities, including those in the evaluation set, were considered in the process. To address this, we created clusters of ethnicities based on their similarities, assigning each ethnicity to one of these subgroups, and ultimately to one of the main groups. The resulting groupings are shown in the table below.

Group	Subgroup
European	Romance, Germanic, Slavic, Baltic
African	Chadic, Niger-Congo, Omotic, Bantu
Asian	Chinese Languages, Tibetic, Other Asian, Indo-Asian
Middle-Eastern	Indo-Iranian, Turkic, Semitic
Oceanian	Austronesian, Indigenous

TABLE I
GROUPS AND SUBGROUPS OF ETHNICITIES

Based on this grouping, we created a new feature named after each group and subgroup. For each sample, if its ethnicity belongs to a particular group or subgroup, the corresponding feature is assigned a value of 1.

Finally, we encoded the *ethnicity* feature using *One Hot Encoding*. As a result, all features in the dataframe are now non-categorical, allowing us to directly apply mathematical operations on them.

We then examined the numeric features of the dataset and worked on processing them to achieve better results. Below, we show the distribution graphs of some of these features.

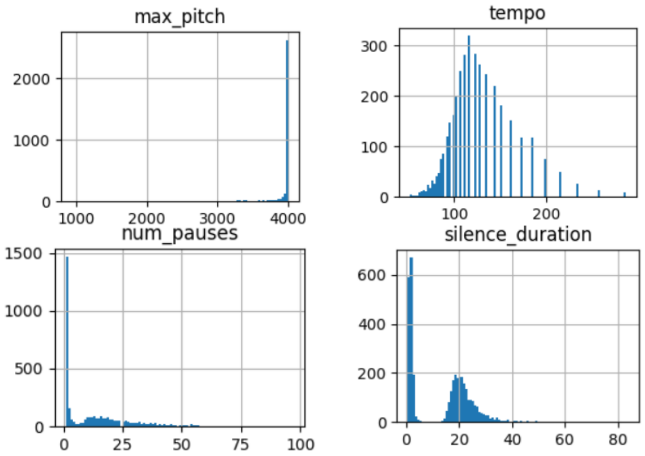

Fig. 3. Non transformed features histograms.

Upon inspecting the distribution of the data, it becomes evident that the features exhibit significant skewness, with some values being much larger than others. Additionally, the presence of extreme values (outliers) further amplifies the variability and could distort the analysis. To address these issues, applying a logarithmic transformation helps to

compress the range of large values, making the distribution more symmetric and stabilizing the variance. This can improve model performance and make the data more suitable for statistical analysis. Below, we present the distribution graphs after applying the logarithmic transformation for the selected features.

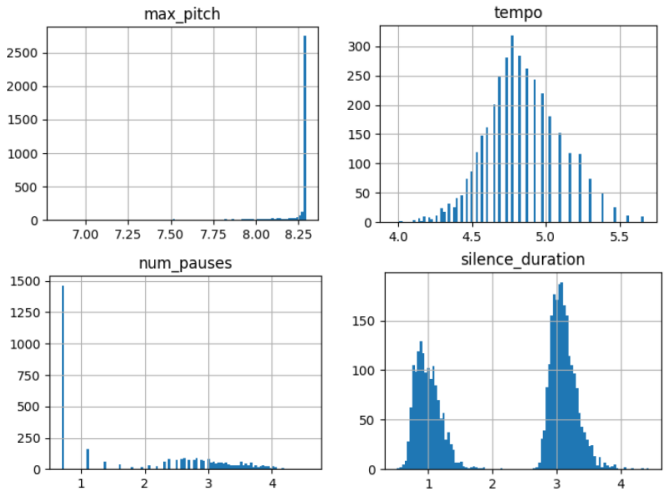

Fig. 4. Log transformed features histograms.

The trasformation has been applied to all these features:

1) max pitch;
2) min pitch;
3) jitter;
4) shimmer;
5) energy;
6) tempo;
7) number of pauses;
8) silence duration.

The features not selected for transformation are those that do not exhibit particularly skewed distributions, such as the mean and standard deviation features.

B. Model selection

We have tested the following algorithms:

- **Random Forest**: this algorithm uses multiple decision trees, each trained on different subsets of data and features, to make predictions. By combining the results of these individual trees, it typically reduces the overfitting issues that are common with single decision trees while still maintaining a certain level of interpretability. The performance of a random forest improves as the number of estimators increases, up to a certain point. Random forests can be tuned using a similar set of parameters as single decision trees. Like decision trees, random forests process one feature at a time, so no normalization is required. We chose to use random forests because they have been shown to perform well on audio signal classification tasks. [1]
- **Extreme Gradient Boosting (XGBoost)**: XGBoost is an advanced ensemble learning algorithm that builds a predictive model by sequentially combining the outputs of weak learners, typically decision trees. It enhances traditional Gradient Boosting([2]) by introducing regularization techniques (L1 and L2) to prevent overfitting and improve generalization. XGBoost also uses a more efficient leaf-wise tree construction strategy, handles missing values automatically, and implements parallelization during training, making it faster and more scalable. These improvements make XGBoost highly effective for large and complex datasets, offering superior performance in both classification and regression tasks. We chose XGBoost due to its proven success and efficiency in machine learning applications.

For both classifiers, the best-working configuration of hyperparameters has been identified through a grid search, as explained in the following section.

C. Hyperparameters tuning

Model	Parameter	Values
Random Forest Regressor	criterion	{'poisson', 'squared_error'}
	n_estimators	{100, 250, 500}
	max_depth	{None, 20}
	max_features	{'sqrt', None}
Gradient Boosting	n_estimators	{150, 250, 300}
	max_depth	{2, 3, 5}
	learning_rate	{0.05, 0.1, 0.2}
	gamma	{1, 7, 15}
	reg_alpha	{0.1, 0.2}
	reg_lambda	{2, 5}

TABLE II
HYPERPARAMETERS CONSIDERED

The sets of hyperparameters to be tuned are those of the Random Forest and Gradient Boosting methods. The relevant parameters and their possible values are summarized in Table II. In this work, we employed *Grid Search* to fine-tune the hyperparameters of our models. Grid Search is a comprehensive and systematic approach that exhaustively searches through a predefined hyperparameter space to find the optimal combination of parameters. By using cross-validation, Grid Search ensures the evaluation of each set of hyperparameters, thus providing a reliable estimate of the model's performance.

The content of this section is based on [3].

III. RESULTS

For the *RandomForestRegressor*, the best combination of hyperparameters was found to be {*criterion = 'poisson'*, *n_estimators = 500*, *max_depth = None*, *max_features = None*}. , resulting in an R^2 score of 0.416. For the *XGBRegressor*, the optimal parameters were {*n_estimators = 250*, *max_depth = 3*, *learning_rate = 0.1*, *gamma = 15*, *reg_alpha = 0.2*, and *reg_lambda = 5*}, achieving a competitive performance.

By applying Grid Search, we ensured that both models were optimally tuned, thus maximizing their predictive power

and improving the generalization to unseen data. This method provided valuable insights into the most important hyperparameters that affect the model's performance and allowed us to fine-tune our models in a rigorous and methodical way.

Using the in-build function of the RandomForestRegressor we can see what are the most important features to predict the age, listed as follows:

1) silence_duration;
2) words_per_second;
3) ethnicity_english;
4) jitter;
5) mfcc_mean_6.

Now we compared the results on a scatter plot:

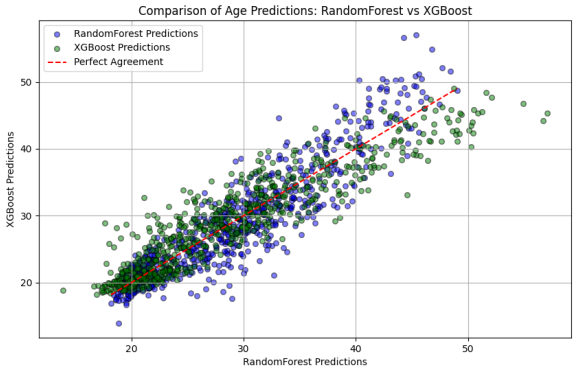

Fig. 5. Result of the predictions.

We observed that many predictions were quite similar, with the majority clustered in the 20-30 year range, while predictions became more spread out around the 50-year mark. This pattern suggested the potential benefit of using an ensemble approach, where we could combine our models with appropriate weights.

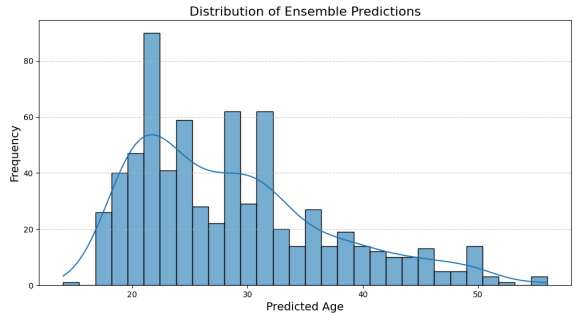

Fig. 6. Ensembled results of the predictions.

To determine the optimal weights for the ensemble, we relied on the performance metrics from the submission platform. From the outset, we noticed that the ExtremeGradientBoosting model yielded a significantly lower RMSE, indicating it should have a higher weight in the ensemble. Ultimately, the optimal weights we determined were 0.95 for the ExtremeGradientBoosting model and 0.05 for the RandomForestRegressor.

IV. Discussion

In this work, we explored various preprocessing techniques and machine learning models to enhance the performance of our system. While some approaches showed promise, others did not lead to noticeable improvements, highlighting the complexity of the problem and suggesting areas for further exploration.

We tested standardization methods using the *Standard Scaler* and *MinMax Scaler*, but these did not significantly improve the model's performance. This is likely due to the fact that models like *Random Forest Regressor* and *Extreme Gradient Boosting* make decisions based on threshold values and are not affected by the scale of the variables.

We also experimented with feature selection and outlier removal, but these methods did not yield substantial improvements. As a result, we propose the following potential areas for enhancement:

- **Try alternative regression models:** Models like Support Vector Regression, Gaussian Processes, or Neural Networks may better capture the data's underlying structure. [4]
- **Explore advanced feature extraction:** Techniques like automated feature extraction via Convolutional Neural Networks (CNNs) could reveal more useful representations.
- **Refine hyperparameter optimization:** A more extensive and targeted search space for hyperparameter tuning could lead to improved results.
- **Improve handling of categorical features:** Using advanced encoding techniques or models like *CatBoost* could enhance performance, especially with important categorical features.

References

[1] M. Kleiman, *Hands-On Random Forests: Building robust machine learning models using random forests and decision trees*. Packt Publishing, 2020.

[2] S. Fuchs, *Hands-On Gradient Boosting with XGBoost and scikit-learn*. Packt Publishing, 2020.

[3] F. Giobergia, "Slides," 2025. Slides from the course Data Science Lab, University Poltecnico di Torino.

[4] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2019.