Speaker Age Estimation Problem

Romain Emanuele Salvi
Politecnico di Torino
Student ID: s339304
s339304@studenti.polito.it

Mattia Sernia
Politecnico di Torino
Student ID: s339829
s339829@studenti.polito.it

***Abstract*—This paper presents a solution to the age regression problem using a dataset of audio recordings. We implemented a Data Science pipeline to predict the speaker's age. The report describes the key stages of the pipeline from data exploration to result discussion. The proposed approach achieves a competitive public score, demonstrating its effectiveness.**

I. Problem Overview

The task assigned to us was to estimate the age of speakers based on their speech. The dataset on which we applied our data science pipeline is a collection of audio recordings, and each data sample corresponds to a speech signal representing a spoken sentence. This is how the samples are subdivided:

- 2,933 samples for the *development set*. These came with *age*, the target label.
- 691 samples for the *evaluation set*.

From each sample, a variety of acoustic and linguistic features were extracted from the speech signal. These features also included the path to the *WAV* recording of the file. Being an uncompressed audio format that stores data in PCM (Pulse Code Modulation), compared to compressed formats like *MP3*, *WAV* files are much larger. During the first step of the pipeline, data exploration, we noticed it follows abnormal patterns under many aspects. First, we remarked the unusual distribution of the *number of words* attribute of the data. The most common number of words is 69, which accounts for 59.0% of the dataset. However, given that we are solving an age estimation task based on speech, we were not expecting 13.38% of the samples to have 0 words in them. What is interesting is that the vast majority (90.93%) of the samples having 0 words are from the "igbo" ethnicity. This would have been a minor inconvenience if the dataset did not contain many recordings from igbo people, but they are the most common ethnicity in both the development and the evaluation dataset. The ethnicity distribution is displayed in Table I. The clear difference in top ethnicities between the development and evaluation set will be handled in the data preprocessing part of the pipeline.

TABLE I
Top 5 Ethnicities comparison in the development and evaluation set (Percentages)

development set		evaluation set	
Ethnicity	**Percentage (%)**	**Ethnicity**	**Percentage (%)**
Igbo	36.86	Igbo	33.57
English	19.74	Spanish	23.44
Arabic	3.48	Turkish	5.35
Mandarin	2.22	Vietnamese	3.18
French	2.15	Swedish	2.89

We manually looked through the instances of recordings of igbo people with a number of words equal to 0. To find out whether that was because the recordings actually did not contain any words in them, we found and plotted an instance having "igbo" ethnicity and 0 words spoken, according to the data we were provided with. This plot is represented in Figure 1.

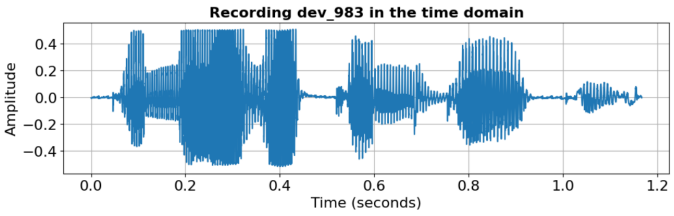

Fig. 1. Representation in the time domain of the selected data point.

As can be both seen from the amplitude variations in the plot and heard listening to the audio recording, the person pronounces some words. This is the case for many recordings with ethnicity "igbo". This means that the attribute describing the number of words does not actually give information about the number of words spoken in general, but the number of english words pronounced. Another interesting trend regarding the samples having igbo as ethnicity is their duration: the average duration of the recordings that compose the dataset is 17.12 seconds. However, recordings from igbo people are rather short. This is shown in Figure 2. The y-axis, which represents the occurrences, is displayed on a logarithmic scale to better compare the distributions in the two datasets and to identify outliers (which have 1 occurrence). It can be seen from the image that these recordings are all much shorter than the average, but they are similarly distributed in the two sets of data, development and evaluation. This suggests a correct

division of the ”igbo” ethnicity between the dataset that will be used to train our model and the one used to test it.

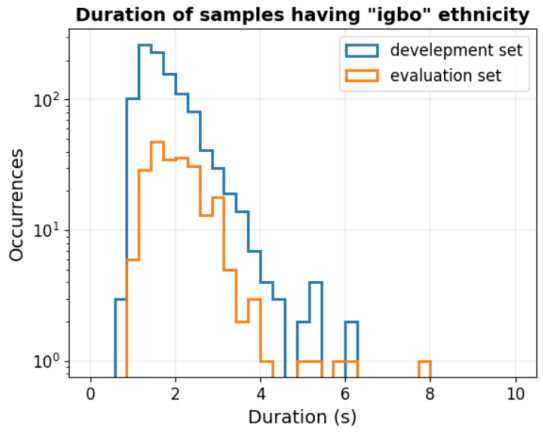

Fig. 2. Duration of samples having ”igbo” ethnicity.

II. Proposed Approach

A. Preprocessing

Data preprocessing is a critical component of the pipeline, as it is not possible to extract meaningful insights from a dataset that has not undergone appropriate preprocessing. The first step was to analyze the feature types in order to know where we would need to act. These are described in Table II.

TABLE II
Feature Types

Feature Name	Type	Feature Name	Type
Id	int64	energy	float64
sampling_rate	int64	zcr_mean	float64
age	float64	spectral centroid_mean	float64
gender	object	tempo	object
ethnicity	object	hnr	float64
mean_pitch	float64	num_words	int64
max_pitch	float64	num_characters	int64
min_pitch	float64	num_pauses	int64
jitter	float64	silence_duration	float64
shimmer	float64	path	object

- **Ethnicity** The ethnicity feature is categorical and, as such, cannot be directly analyzed without proper encoding. There are 165 distinct ethnicities in the development dataset. As illustrated in Table I, the Igbo ethnicity is the most represented, comprising approximately 37% of the whole data. In the development set, the second most common ethnicity is English, accounting for 20%, while all other ethnicity contribute less than 5% each. Given this distribution, it was reasonable to encode the three most common ethnicity as distinct categories, while grouping the remaining ethnicity under a single category labeled as ”Other”. To avoid introducing an implicit ranking, we encoded it using one-hot encoding.
- **Gender** Gender is another categorical feature, so it must be encoded as well. This feature has only two possible values: *male* and *female*. Analyzing both *development* and *evaluation* sets, we noticed an error in the second dataset. Specifically, in the *evaluation set*, the number of unique values for gender was three: *male*, *female*, and *famale*. It was reasonable to assume that the third value was a typing error. Listening at the corresponding audio we confirmed it. So, we decided to replace it with *female*. The feature was later encoded using a label encoder, as the possible values were limited to just two.
- **Tempo** This feature represents the estimated speaking rate in beats per minute (BPM). Since the feature should contain only numerical values, we were expecting it to be a numeric type. However all entries in this feature are stored as strings representing arrays with a single element. These elements are the numerical values we need. To address this, we stripped the strings, extracted the numbers and converted the feature to the appropriate format: a floating point number.
- **Path** This feature contains the file path to the audio recording. We used a Python library to extract meaningful information from the audio files. The extracted features are summarized in Table III. Many of these features are generated in the form of matrices. To reduce the number of features, we computed the mean and standard deviation across all rows of each matrix, creating new features for the dataset. Once all necessary data were extracted, the path column was no longer useful, so we removed it from the dataset.

TABLE III
Features Extracted and Their Descriptions.

Feature	Description
mfcc	Mel-Frequency Cepstral Coefficients (MFCCs), which represent the short-term power spectrum of the audio on a mel scale.
chroma	Chromagram representing the energy distribution across 12 pitch classes.
spectral_contrast	Spectral contrast, measuring the difference in amplitude between peaks and valleys in the spectrum.
beat_frames	The estimated beats (temporal positions) in frames.
rms	Root Mean Square (RMS) energy, a measure of the signal’s loudness.
spectral_bandwidth	Spectral bandwidth, indicating the width of the frequency band.
spectral_rolloff	Spectral roll-off, the frequency below which a certain percentage of the total spectral energy is contained.
duration	Duration representing the duration in seconds of the audio signal.

- **ch/second** This is a feature we computed. It represents the number of characters per second in the recording. The feature was calculated by dividing the *num_characters* feature by the *duration* feature. We were expecting a person of older age to speak slower thus having a lower value in this feature.
- **Id, sampling_rate** These two columns aren’t useful to our analysis anymore, so we decided to remove them.

The sampling rate was used to extract the set of features with the library.

After extracting all the necessary features, we initially aimed to identify highly correlated features. To achieve this, we computed the correlation matrix for the features. The highest correlated pair of features was *num_words* and *num_characters*, with a correlation of 0.999890. We considered dropping one of these two columns to reduce redundancy, but this led to a decrease in score of the model. Therefore, we decided to keep both features.

Another important step in the data preprocessing process is the identification of outliers. We focused on removing data points having unusual values in more than one feature. Similarly to the correlation analysis, removing a small number of outliers resulted in worse performance for the regression model. As a result, we chose to keep all the rows in the dataset.

B. Model selection

Starting from the previously described features, we began testing this set of algorithms:

- **Random Forest** This algorithm is based on multiple decision trees. Decision trees are prone to overfitting, the deeper they are the more they overfit the data. However, a random forest regressor combines multiple trees trained on random subsets of the data,thus reducing overfitting and improving generalization to unseen data. This model is more resistent to outliers, which we decided to keep, and performs well even in a high-dimensional space, which is our case. The hyperparameters we decided to tune for this regressor are:
  - *n_estimators* This parameter defines the number of trees in the forest. We chose a high number of trees, as to reduce overfitting: a larger number of trees averages out individual errors, enhancing the model's robustness.
  - *max_depth* This parameter specifies the maximum depth of a tree. Excessively short trees may result in underfitting; on the contrary, trees that are too long might overfit the data, as explained before.
  - *max_features* This parameter defines the number of features to consider when searching for the best split, at each split.
- **Linear Regressor** This is the simplest regressor of all. It works by computing a set of coefficients for all the features, aiming to minimize the residual sum of squares between the target and the predictions. The only parameter tuned for this regressor was *fit_intercept*. We did not expect much from the results of this regressor, but we included it to compare its performance with respect to other regressors.
- **Lasso** Lasso regression works similarly to linear regression. It penalizes all non-zero weights linearly, introducing a penalty based on the absolute value of the coefficients. The hyperparameters tuned for this regressor are:
  - *alpha* . This is the regularization strength, controlling the magnitude of the penalty applied to the coefficients.
  - *fit_intercept* This parameter determines whether the intercept should be fitted as part of the model or not.
- **Ridge** Ridge regression, also known as L2 regularization, is a version of linear regression that adds the squared magnitude of the coefficients as a penalty. It works by penalizing features whose coefficients deviate significantly from 0. The hyperparameters tuned for this regressor were the same as the ones tuned in the lasso regression: *alpha* and *fit_intercept*.

To ensure correct performance of the regressors (except Random Forest Regressor, which does not need input data to be scaled), we needed to scale the data. As result of some preventive runs of the regressors, we found Min-Max scaler and Standard scaler to be among the best scaling techniques for our data. Given the wide range of values across the features, we hypothesized that the Min-Max scaler would work better. Also, we did not want the total variance to be spread across too many features, losing important information from our most meaningful features. To verify this intuition, we set up two pipelines to test both on each regressor. The second step of said pipelines was the application of polynomial features on our data. This was done to introduce more flexibility in representing relationships between features. This increases model capacity, but also raises the risk of overfitting. To avoid it, we only used polynomial features up to the second degree.

C. Hyperparameter tuning

To test all possible combinations of hyperparameters for each model, we applied a grid search with cross-validation of the model learned. This allowed us to find optimal hyperparameters configuration of a model, finding the one which works best for our task. The cross validation (more specifically, k-folds validation) allows to evaluate the model numerous times on distinct subsets of the data, resulting in a more trustworthy estimate of performance. This helps us avoid overfitting. A 5-fold cross-validation approach was implemented, ensuring that each fold would be of comparable size to the evaluation set. Table IV shows the combinations of hyperparameters tried.

TABLE IV
POSSIBLE HYPERPARAMETERS CONFIGURATION

Regressor	Hyperparameter	Possible Values
Random Forest	n_estimator	50,100
	max_depth	5,10,50,100
	max_features	'auto', 'log2', 'sqrt'
Linear Regressor	fit_intercept	True, False
Ridge	alpha	1, 2, 4.5, 5, 5.5, 10
	fit_intercept	True, False
Lasso	alpha	0.1,0.3, 1, 10
	fit_intercept	True, False

III. Results

Having run all configurations of hyperparameters we selected for each regressor, we could analyze their results in terms of Root Mean Squared Error (RMSE). Figure 3 shows, for each classifier and scaling technique applied, the best possible result. The scaling technique applied to the models had a significant impact on their performance. For instance, the performance difference between the best Lasso configuration trained on data scaled using the standard scaling technique and that scaled with Min-Max scaling is not negligible, amounting to 0.7452. As expected, the Linear Regressor did not rank among the best-performing models. This result aligns with our initial expectations, given its limited capacity to capture non-linear relationships and interactions between features. However, what was unexpected is that the Random Forest Regressor performed worse than both Lasso and Ridge regression. Ridge and Lasso regressions performed better, possibly due to their regularization capabilities, which penalize overly complex models and prevent overfitting. The best possible configuration, in terms of RMSE, was obtained using a Ridge regressor applied on data scaled with the Min-Max technique. This result confirms our previous hypothesis. The best score and hyperparameters are shown Table V. The table shows the top three model configurations in terms of RMSE when data is scaled with the Min-Max scaler. An interesting observation is that changing the *fit_intercept* parameter has minimal impact on the overall result. On the other hand, the inclusion of the *Polynomial Feature* set to 2 proves to be highly beneficial, significantly improving the model's performance. None of the configurations with polynomial features degree set to 1 appear in the table.

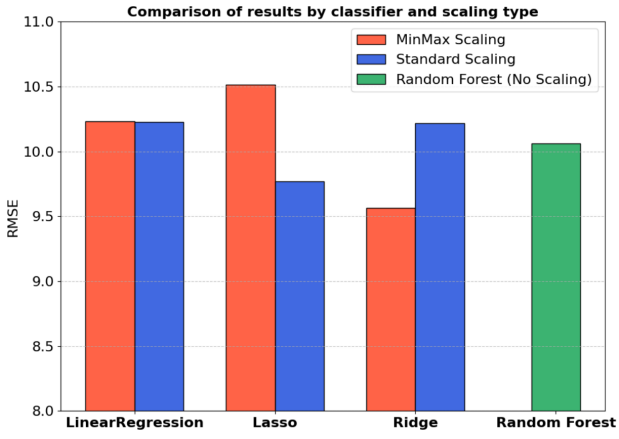

Fig. 3. Comparison of the best results for each scaler and classifier.

TABLE V
THE THREE BEST CONFIGURATIONS USING MIN-MAX SCALED DATA

Regressor	alpha	fit_intercept	Polynomial Feature	RMSE
Ridge	5.5	False	2	9.5629
Ridge	5	False	2	9.5642
Ridge	5.5	True	2	9.5650

The Regressor chosen for the discussion of the results is the Ridge regressor using *alpha*= 5.5, *fit_intercept*= False and *Polynomial Feature*= 2. It is the one that obtained the best score.

IV. Discussion

The Ridge object allows us to inspect the values of the coefficients learned, and nine out of the top ten features were extracted or computed during the preprocessing step. This indicates that the extracted audio features and the encoding applied are the primary determinants for the age estimation task. The only feature in Figure 3 that was originally present in the dataset is *gender*, which was encoded. As expected, the feature encoding the "igbo" ethnicity turned out to be important, as it placed sixth. Figure 3 shows the ten most valuable features in terms of absolute value of the coefficient learned.

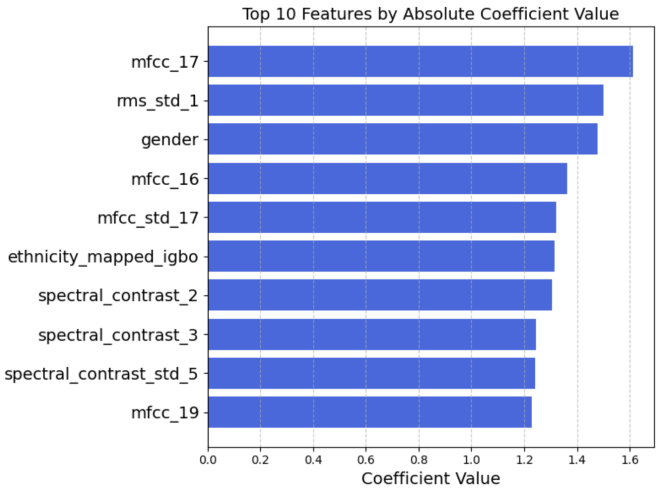

Fig. 4. Ten most valuable features by absolute coefficient value

We can come to the conclusion that our pipeline performed notably well compared to those of our peers, as the RMSE of 9.056 achieved on the Public dataset ranks among the best. However, there is still room for improvement: manually reviewing the audio files, we noticed that many of them (particularly the ones having "igbo" ehtnicity) were of poor quality. Many recordings contained significant noise and interference. Such factors likely impacted the overall data quality. Perhaps cleaning those recordings removing the noise would have improved the performance of our model. Another limitation to the solution of this age estimation task is that the databases we were provided with are not fully representative of all the potential diversity in human voices. Access to a more comprehensive dataset would likely have resulted in more accurate age estimations.

References

[1] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. Librosa: Audio and music signal analysis in python. In *Proceedings of the 14th Python in Science Conference*, pages 18–25, 2015.

[2] Damian Kwasny and Daria Hemmerling. Gender and age estimation methods based on speech using deep neural networks. *Sensors*, 21(14):4785, 2021.