Age Estimation from Speech Dataset Regression Problem

Arian Mohammadi S346278
Politecnico di Torino
s346278@studenti.polito.it

***Abstract*—This report presents a regression approach for estimating a speaker's age from speech recordings. Numerical features are standardized and categorical variables are encoded using a preprocessing pipeline implemented with `scikit-learn`. Ensemble regression models, namely Random Forest and Gradient Boosting, are trained and optimized through cross-validation. The proposed method achieves reliable predictive accuracy and good generalization on unseen data.**

I. Problem Overview

The proposed task is a regression problem on the *Age Estimation from Speech Dataset*, consisting of audio recordings and associated features for speakers. The objective is to predict the continuous age of speakers based on acoustic and linguistic characteristics extracted from their speech.

The dataset is divided into two parts:

- **Development Set:** 2,933 samples with target ages for training and validation.
- **Evaluation Set:** 691 samples without target ages for generating predictions.

The features include:

- Acoustic features: sampling rate, pitch metrics, jitter, shimmer, energy, tempo, Zero-Crossing Rate mean, spectral centroid mean, Harmonics-to-Noise Ratio.
- Linguistic features: gender ,ethnicity, silence duration, number of pauses, number of words, and number of characters.

We use the development set to train a regression model that predicts the continuous target variable (age) for the evaluation set.

The sampling rate for all the audio signals in the dataset is specified as 22,050 Hz, which means 22,050 audio samples are recorded every second. This value is sufficient for most speech processing tasks because the Nyquist-Shannon sampling theorem [1] states that the sampling rate should be at least twice the highest frequency present in the signal. Human speech typically falls within the range of 0–11 kHz, so a sampling rate of 22,050 Hz ensures adequate representation [2].

Tempo feature in the dataset represents the estimated speaking rate of the audio signal, measured in beats per minute (BPM). Its distribution shows a central tendency with variations, reflecting diverse speaking styles and rhythms [3].

Figure 1 shows the distribution of tempo values in the dataset, indicating the variety of speaking rates across different speakers.

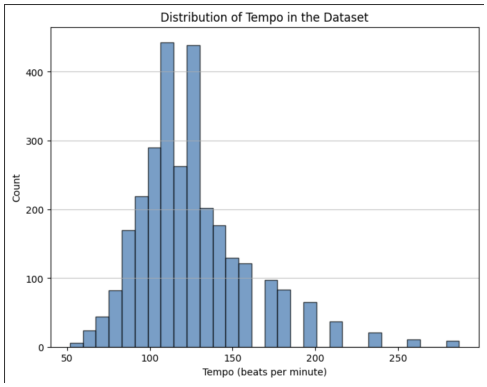

Fig. 1. Distribution of features in the dataset.

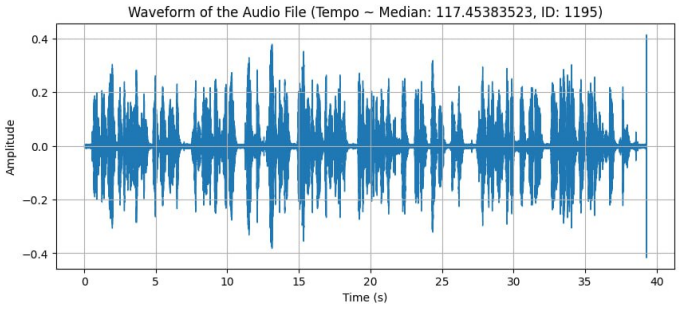

Fig. 2. Representation of an audio with median tempo in the time domain.

The graph shows the waveform of the audio signal, representing changes in amplitude over time. Peaks and troughs indicate variations in loudness, while flat sections represent silence.

II. Proposed Approach

A. Data Preprocessing

The dataset used in this project consists of a combination of numerical and categorical features that provide valuable information regarding the recordings. To ensure the dataset is in a format suitable for machine learning, we preprocess both feature types using specialized techniques.

1) Handling Features: The dataset consists of both numerical and categorical features, each requiring specific preprocessing techniques to ensure compatibility with machine learning models.

- **Numerical Features:** Continuous features, such as *tempo*, *mean_pitch*, and *energy*, were standardized to have a mean of 0 and a standard deviation of 1, ensuring comparability across scales [3]. Missing values were imputed using the median of each column, a robust approach to outliers that minimizes bias [4].
- **Categorical Features:** Features like *gender* and *ethnicity* were encoded using one-hot encoding, transforming each category into binary vectors to avoid introducing an ordinal relationship. Missing values in these features were filled with the most frequent category to maintain consistency [5].

2) Handling Data Imbalance: The categorical feature *ethnicity* exhibits a strong imbalance, where a small number of groups contain most of the samples while many categories appear only a few times. Such imbalance may bias the model toward majority groups and reduce predictive performance for minority classes.

To mitigate this issue, sample weighting was applied during training. Specifically, `compute_sample_weight(class_weight="balanced")` from `scikit-learn` was used to assign higher importance to underrepresented groups. This strategy ensures that all demographic categories contribute more equally to the learning process and improves fairness and generalization.

Figures 3 and 4 illustrate key characteristics of the dataset: the skewed age distribution and variations in mean age across the top 10 ethnicities, respectively. These insights highlight the diversity in the data and the importance of appropriate preprocessing to handle imbalances and heterogeneity.

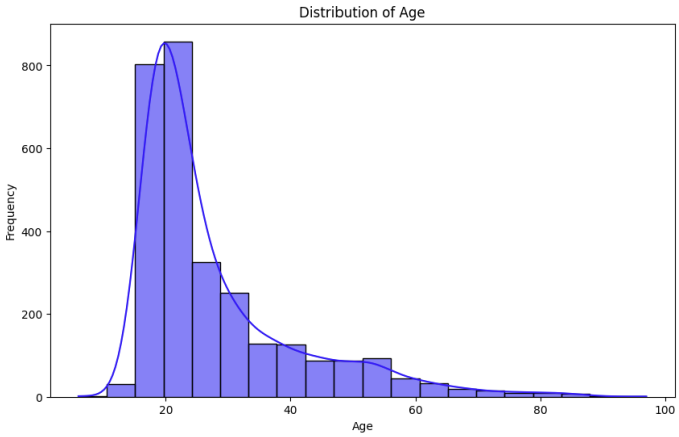

Fig. 3. Age distribution in the dataset. The majority of samples belong to the 15–33 years range, highlighting the dataset's skew toward younger individuals. This imbalance emphasizes the need for stratified sampling during model training.

3) Combined Preprocessing: The preprocessing of numerical and categorical features was automated using the `ColumnTransformer` from `scikit-learn` [6]. Numerical features were standardized using `StandardScaler` to ensure comparability and imputed with their mean to handle missing values [4]. Categorical features were one-hot encoded and imputed with the most frequent category to maintain consistency [5]. This pipeline ensures efficiency and reduces errors during preprocessing.

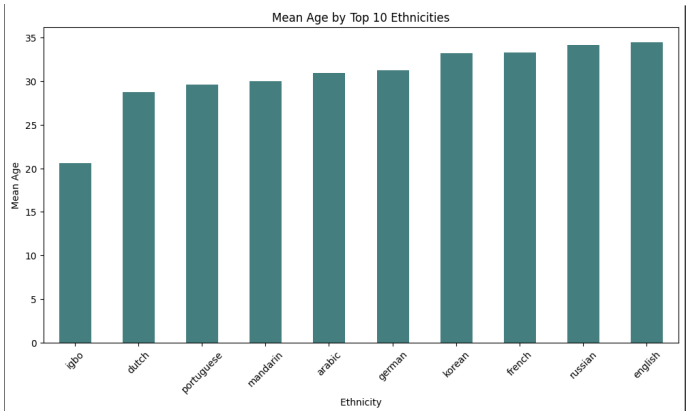

Fig. 4. Mean age across the top 10 ethnicities in the dataset. The plot reveals that some ethnic groups, such as Igbo, have significantly younger mean ages, while others, like English or Russian, have older averages. These differences can influence model predictions.

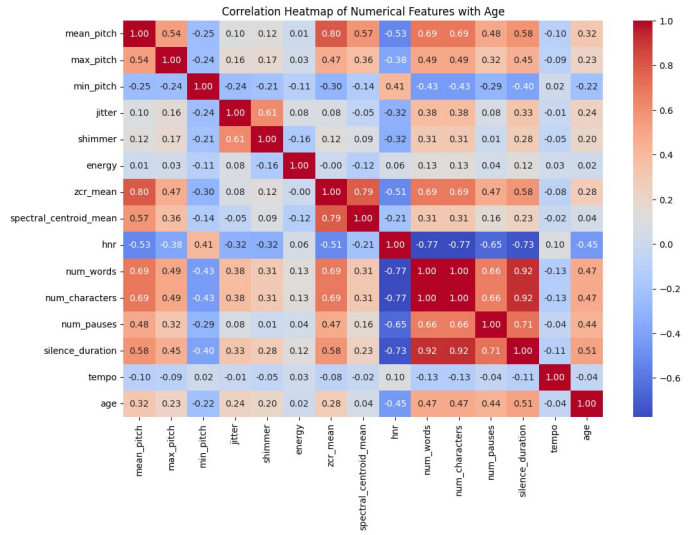

Fig. 5. Correlation heatmap of numerical features with age. Features like *num_characters* and *num_words* show strong positive correlations, while *hnr* exhibits a negative correlation, reflecting its inverse relationship with age.

Figure 5 highlights the relationships between features and age. Strong positive correlations, such as those for *num_words*, suggest that older speakers tend to produce more complex speech. Negative correlations, like *hnr*, may indicate changes in vocal characteristics with age.

B. Model Selection

Two machine learning algorithms are implemented to predict the target variable (*age*):

1) Random Forest Regressor: The Random Forest Regressor is an ensemble learning method that constructs a collection of decision trees during training and outputs the average prediction of the trees to improve accuracy and control overfitting [7]. Key advantages include:

- **No Scaling Required:** Random Forest can handle both numerical and categorical features directly, eliminating the need for normalization or scaling of data.
- **Overfitting Reduction:** By averaging multiple trees, it reduces the risk of overfitting compared to a single decision tree.
- **Hyperparameter Flexibility:** Important parameters include:
  - `n_estimators`: The number of trees in the forest, with higher values often improving stability and accuracy.
  - `max_depth`: Controls the depth of each tree, preventing overly complex models.
  - `min_samples_split` and `min_samples_leaf`: Define the minimum number of samples required to split a node or remain as a leaf, ensuring trees generalize better.

Random Forest is well-suited for datasets with a mix of numerical and categorical features and performs robustly with minimal tuning. However, it may not capture complex relationships in data as efficiently as boosting algorithms [7].

2) Gradient Boosting Regressor: Gradient Boosting is a sequential ensemble technique that builds models iteratively, with each new tree learning to correct the errors of the previous ones. This makes it highly effective for improving predictive performance [8]. Key characteristics include:

- **Learning Rate:** The parameter `learning_rate` determines how much each tree contributes to the overall model. Smaller values often yield better generalization but require more trees.
- **Tree Depth:** The parameter `max_depth` controls the complexity of individual trees. Shallower trees help prevent overfitting and focus on correcting residual errors.
- **Subsampling:** A `subsample` fraction introduces randomness by training each tree on a random subset of the data, improving generalization.
- **Fine-Grained Control:** Parameters such as `n_estimators` (number of trees) and `min_samples_leaf` offer precise control over the model's complexity and performance.

Gradient Boosting excels at handling structured data and can model complex, non-linear relationships effectively. However, it requires careful tuning of hyperparameters to avoid overfitting or underfitting. Additionally, it is computationally more intensive than Random Forest due to its sequential nature [8].

3) Comparison of Models: Random Forest is preferred for its simplicity, robustness, and lower risk of overfitting when computational resources are limited. In contrast, Gradient Boosting delivers superior performance on structured data by iteratively improving its predictions, making it an excellent choice for tasks requiring high accuracy at the cost of increased training time [7], [8].

C. Hyperparameter Tuning

Hyperparameters for both models were optimized using `GridSearchCV` with 3-fold cross-validation. This exhaustive search evaluates multiple parameter combinations and selects the configuration that minimizes the cross-validation mean squared error.

Key parameters tuned include the number of trees (`n_estimators`), tree depth (`max_depth`), minimum samples per split/leaf, learning rate, and subsampling ratio. This systematic approach improves model robustness and reduces both overfitting and underfitting.

- **Number of Trees (`n_estimators`):** Values from 100 to 500 were tested for both models, balancing accuracy and training time.
- **Maximum Depth (`max_depth`):** Explored values include 10, 20, 30, and `None` for Random Forest, and 3, 5, 7, and 9 for Gradient Boosting to control model complexity and prevent overfitting.
- **Minimum Samples Split (`min_samples_split`):** For Random Forest, values of 2, 5, and 10 were tested to adjust tree branching and generalization.
- **Minimum Samples Leaf (`min_samples_leaf`):** Tested values (1, 2, 4) in Random Forest ensured stable, adequately sized leaf nodes.
- **Learning Rate (`learning_rate`):** For Gradient Boosting, rates of 0.01 to 0.2 were tested, with smaller rates paired with more trees for better generalization.
- **Subsample Fraction (`subsample`):** Gradient Boosting used values of 0.8 and 1.0 to add randomness and reduce overfitting.

The best parameter combinations for both models were selected based on cross-validation scores, achieving strong results with improved accuracy and generalization, as detailed in the Results section [7], [8].

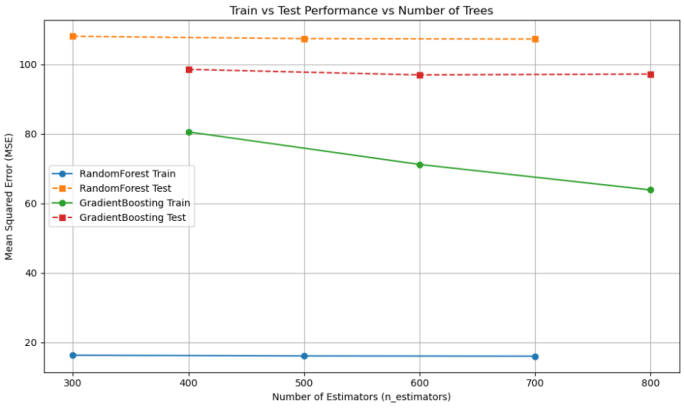

Fig. 6. Model performance comparison across different values of `n_estimators` for Random Forest and Gradient Boosting models. The plot highlights the mean squared error (MSE) trends for train and test datasets.

Figure 6 shows that Random Forest maintains low and stable MSE for both train and test sets, while Gradient Boosting reduces training MSE as `n_estimators` increases but struggles to improve test MSE.

D. Cross-Validation

The hyperparameter combinations were evaluated using a randomized search strategy with 3-fold cross-validation [9], [10]. This approach efficiently explores the parameter space and prevents overfitting by testing on different splits of the data.

III. Results

Both models were evaluated using Root Mean Squared Error (RMSE) on the training and held-out test sets. The evaluation was performed before retraining the final model on the complete dataset to avoid data leakage.

A. Best Hyperparameters

TABLE I
BEST HYPERPARAMETERS FOUND VIA GRID SEARCH

Model	Best Configuration
Random Forest	$n_{est} = 300$, $depth = 30$, $split = 2$, $leaf = 1$, $feat =$ sqrt
Gradient Boosting	$n_{est} = 300$, $lr = 0.05$, $depth = 5$, $leaf = 2$, $sub = 0.8$

B. Performance Comparison

TABLE II
MODEL PERFORMANCE COMPARISON (RMSE)

Model	Train RMSE	Test RMSE
Random Forest	3.96	10.12
Gradient Boosting	7.252	9.895

The results indicate that Random Forest achieves lower training error but exhibits a larger gap between train and test performance, suggesting slight overfitting. Gradient Boosting, although having higher training error, achieves better test performance and thus generalizes more effectively to unseen data.

Based on these results, Gradient Boosting was selected as the final model and retrained on the full development dataset before generating predictions for the evaluation set.

C. Comparison of Models

The comparison between the two models shows that Gradient Boosting outperforms Random Forest in terms of test RMSE, reflecting better generalization to unseen data. Random Forest, on the other hand, exhibited lower train RMSE, indicating that it fits the training data more closely but at the risk of overfitting.

D. Summary of Results

The Gradient Boosting model, with its tuned parameters, was identified as the best overall model for this task, demonstrating strong generalization and balanced performance on both the training and test sets.

IV. Discussion

The proposed preprocessing and modeling pipeline addresses missing values, feature scaling, categorical encoding, and class imbalance. Sample weighting was applied to compensate for the uneven distribution of ethnic groups, ensuring that minority categories contributed equally during training.

Gradient Boosting achieved the best trade-off between bias and variance, resulting in the lowest test RMSE and superior generalization. In contrast, Random Forest obtained lower training error but exhibited mild overfitting.

Future improvements may include additional acoustic feature engineering, larger hyperparameter searches, or deep learning approaches such as neural networks and convolutional architectures to further enhance predictive performance.

References

[1] Richard G. Lyons, *Understanding Digital Signal Processing*, 3rd ed., Pearson Education, 2004.

[2] Ben Gold and Nelson Morgan, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, John Wiley & Sons, 2009.

[3] Ben Gold and Nelson Morgan, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, John Wiley & Sons, 2009.

[4] Timo Hämäläinen and Petri L. W. B. Oksanen, *Data Cleaning and Imputation: A Practical Guide*, Springer, 2020.

[5] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed., Morgan Kaufmann, 2016.

[6] Fabian Pedregosa, et al., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, 12, 2825-2830, 2011.

[7] Leo Breiman, *Random Forests*, Machine Learning, 45(1), 5-32, 2001.

[8] J.H. Friedman, *Greedy Function Approximation: A Gradient Boosting Machine*, Annals of Statistics, 29(5), 1189-1232, 2001.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning*, 2nd ed., Springer, 2009.

[10] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, Springer, 2013.