Speaker Identification Using Voice Features

Can Machine Learning Distinguish Between Brothers with Similar Voice?

CS 438/638 Machine Learning - Fall 2025

Giorgi Karazanashvili

1. Introduction and Problem Description

1.1 Research Question

This project investigates speaker identification: Can voice features learned from a small and diverse set of speakers effectively distinguish between two individuals with similar voices?

1.2 Why This Problem is Interesting

Speaker identification systems are often utilized for security, forensics, voice assistance, etc. It is important that these systems don't misclassify people with similar voices: people with similar accents, family members, or people from similar demographics. Being able to understand how well machine learning models can handle these edge cases carries real practical value.

1.3 Output and Input Features

The output feature is speaker identity, with 7 possible classes: dani, giorgi, khoi, murad, niko, shani, and teymur.

Input Features: 64 acoustic features extracted from audio recordings, including:

- **MFCCs (52 features):** 13 Mel-Frequency Cepstral Coefficients × 4 statistics (mean, std, min, max). MFCCs are widely used in voice recognition because they capture the overall shape of speech spectrum.
- **Spectral Features (8 features):** Centroid, rolloff, bandwidth, and zero-crossing rate × 2 statistics each. These capture the frequency distribution and texture of the voice.
- **Pitch Features (4 features):** Fundamental frequency (F0) × 4 statistics. Pitch distinguishes male/female voices and captures personal speaking patterns.

Relationship between features and output: Each person's voice is different due to their vocal tract shape and how they speak. MFCCs capture these differences numerically, which makes them useful for classification

1.4 Data Collection

Audio data was collected from 7 speakers with different genders and accents reading a standardized set of sentences. Each speaker recorded 11-50 sentences in format using consistent recording conditions. Figure 1 shows the distribution of samples across speakers.

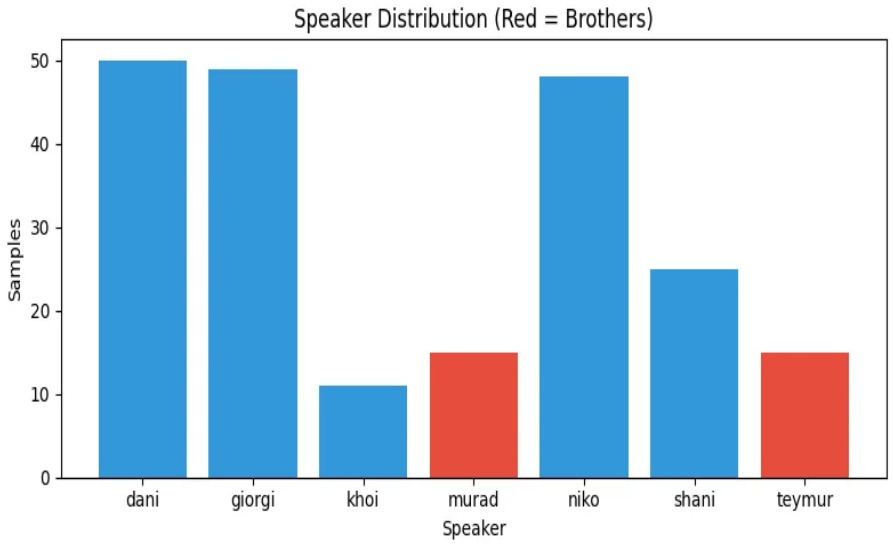


Figure 1: Distribution of audio samples across speakers. Red bars indicate the two brothers (Murad and Teymur) who are the focus of our research question.

Speaker	Recordings	Role
Dani	50	Diverse training speaker
Giorgi	49	Diverse training speaker
Niko	48	Diverse training speaker
Shani	25	Diverse training speaker
Khoi	11	Diverse training speaker
Murad	15	Brother (test subject)
Teymur	15	Brother (test subject)

Total: 213 audio samples

1.5 Audio Preprocessing with Audacity

To ensure higher agreement rate there were no requirements regarding the recordings, which led to most of the sentences being captured as continuous audio sessions. To extract individual sentence clips for feature analysis, I used Audacity's audio editing capabilities.

Workflow:

- Loaded the full recording session into Audacity
- Used “Analyze” → “Label Sounds” to automatically detect sound regions based on amplitude thresholds
- Manually corrected label boundaries to ensure each sentence was properly isolated
- Renamed labels with speaker prefix and sentence number (speaker_01.wav)
- Used File → Export Multiple to export all labeled regions as separate WAV files

Figures 2 and 3 show the Audacity interface during the sentence segmentation process.

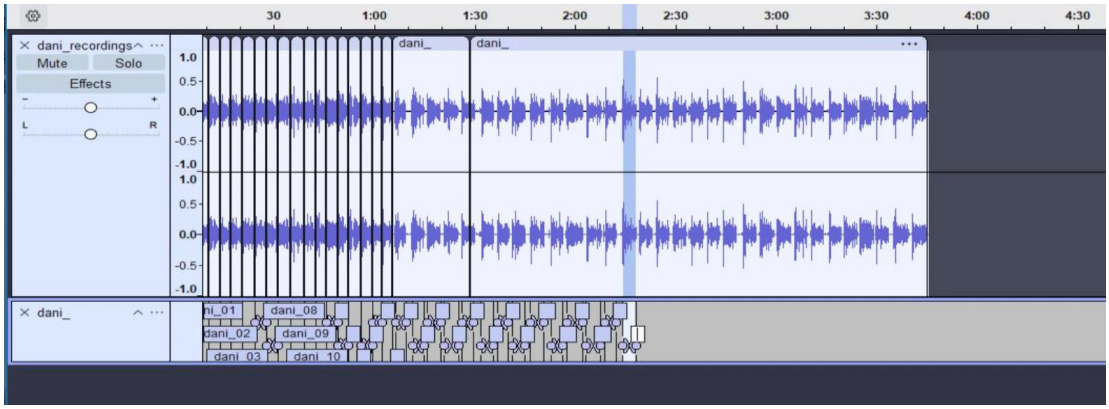


Figure 2: a full recording session with sentence labels (dani_01, dani_02, etc.) marking each segment for export.

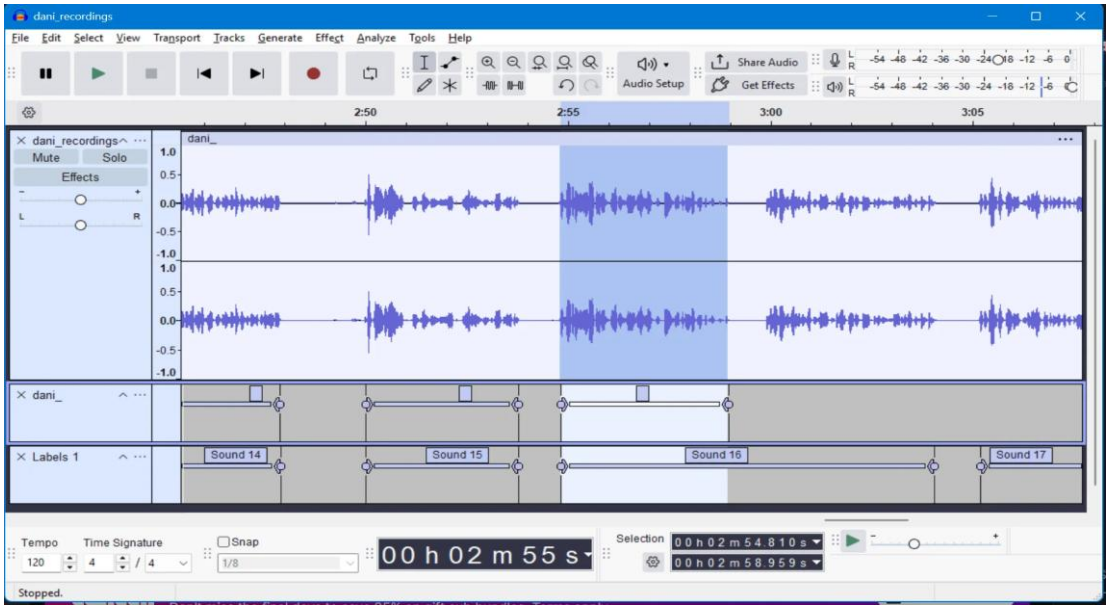


Figure 3: Detailed view showing individual sentence boundaries in the label track. Each label defines the start and end points for a sentence clip.

1.6 Feature Generation Process

Raw audio files were processed using Python's librosa library to extract the 64 acoustic features. Librosa analyzes audio files in small frames (time slices), and since the audio files varied in length, I computed statistics (mean, std, min, max) across the frames in the clip, producing a fixed 64 vector per file.

2. Data Analysis and Preprocessing

2.1 Feature Distribution Analysis

Before training, I analyzed how feature values are distributed across the dataset. This revealed an important issue: features have dramatically different scales as shown in Figure 4.

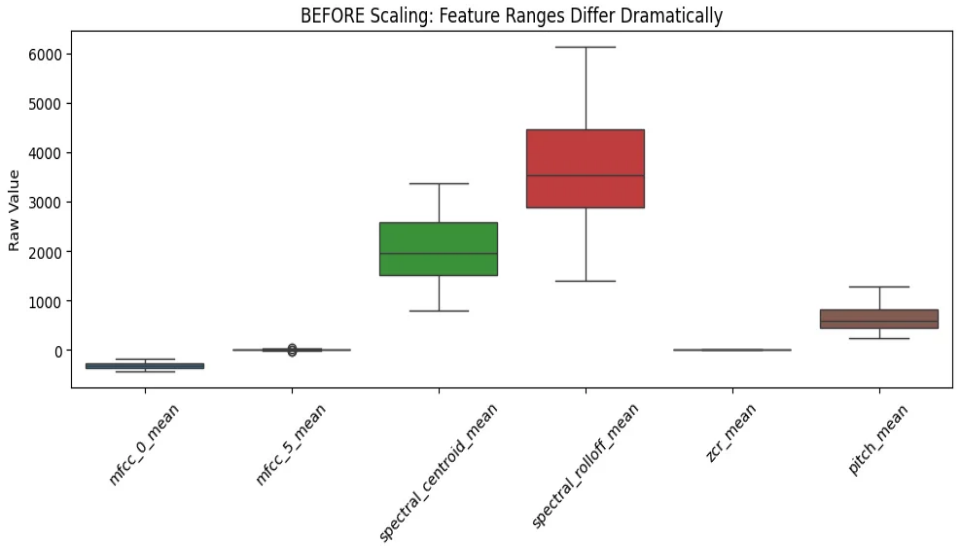


Figure 4: Feature value distributions before scaling.

This disparity would cause gradient-based optimization to be dominated by large-scale features while ignoring informative small-scale features.

2.2 Feature Scaling

To address the issue, I used StandardScaler normalization to transform each feature to have mean=0 and standard deviation=1. This ensures all features contribute equally during model training. The scaling parameters were computed on the training set only, then applied to validation and test sets.

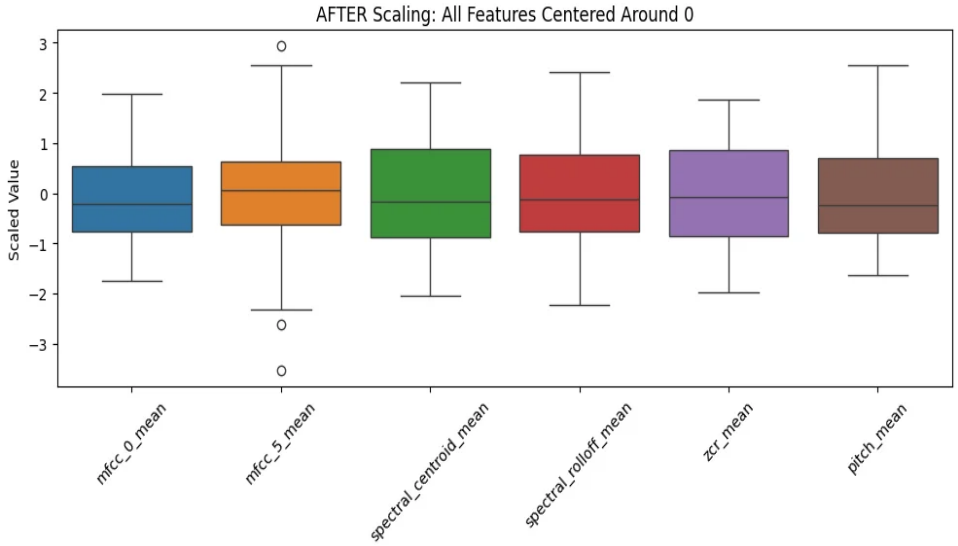


Figure 5: Feature distributions after StandardScaler normalization. All features are now centered around 0 with comparable spreads, enabling fair comparison during model training.

2.3 Correlation Analysis

Many audio features are inherently correlated. For example, spectral_centroid_mean and spectral_rolloff_mean had a correlation of 0.984 because both measure where frequency energy is concentrated. Highly correlated features provide redundant information.

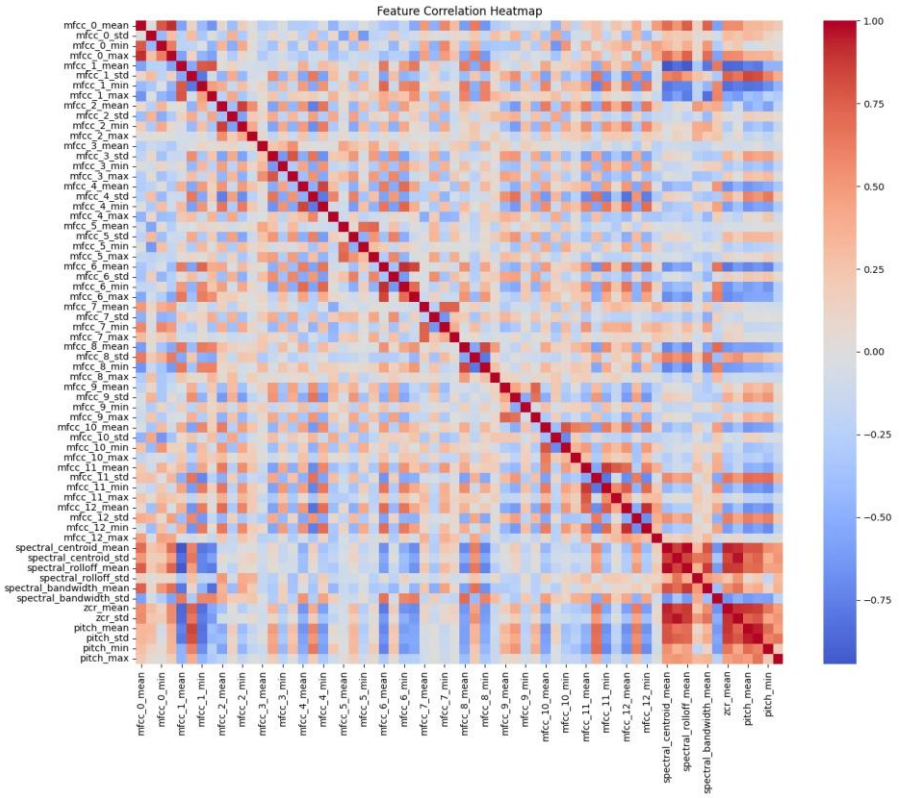


Figure 6: Feature correlation heatmap.

After scaling, I removed 2 features with correlation > 0.95, reducing the feature set from 64 to 62 features for initial modeling.

2.4 Train/Validation/Test Split

A standard 70/15/15 split would leave only ~2 brother samples in the test set, making evaluation unreliable. Therefore, I decided to implement a custom stratified split that guarantees adequate brother representation:

Split	Total	Murad	Teymur
Training	142	7	7
Validation	33	3	3
Test	38	5	5

This ensures we have enough brother samples in the test set to meaningfully evaluate the purpose of the research.

3. Feature Selection

With 62 features and only 142 training samples, we risk overfitting. Thus employed two different feature selection strategies to create alternative models:

3.1 Method 1: Lasso Regularization (L1)

Lasso regression adds a penalty that drives irrelevant feature coefficients to exactly zero, performing automatic feature selection. Unlike manual selection, Lasso considers feature interactions and predictive power simultaneously.

Implementation: trained Logistic Regression with L1 penalty using various regularization (C) strength. Lower C means stronger regularization.

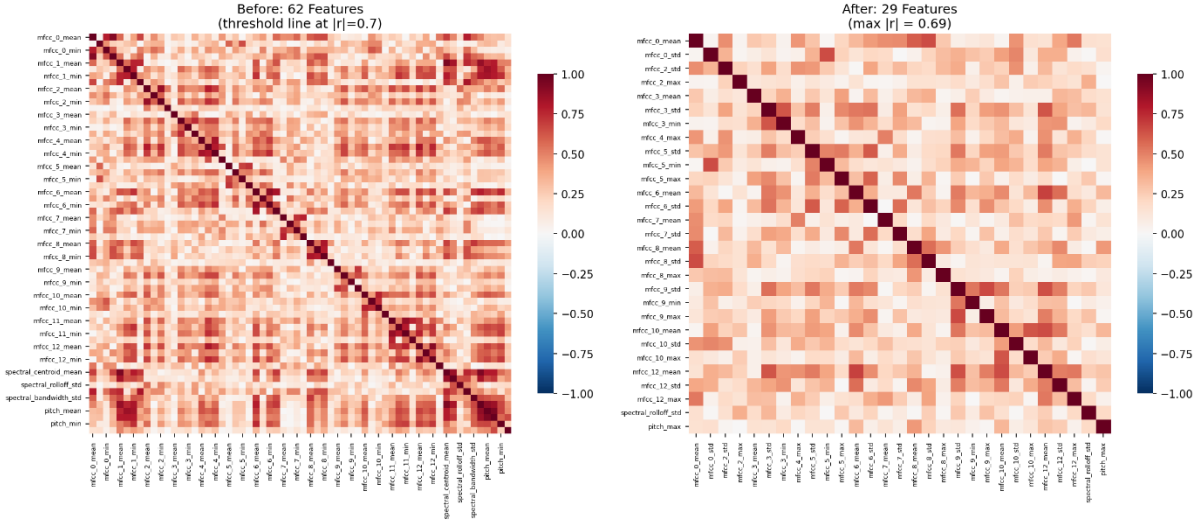
Result: C=0.25 → 43 features. This provides good validation performance while reducing dimensions from 62 to 43 features.

3.2 Method 2: Correlation-Based Reduction

Remove one feature from each highly correlated pair, keeping the one with higher discriminative power (F-score). This creates a more diverse feature set with minimal redundancy.

Implementation: For every pair of features with |correlation| > 0.7, keep the feature with the higher F-score and remove the other.

Result: Reduced from 62 to 29 features by removing 33 redundant features as shown in figure 7.



4. Model Training and Parameter Tuning

4.1 Algorithm Selection

For the final model I picked Logistic Regression with L2 regularization. Its interpretable, fast and works well when classes are linearly separable. The L2 penalty (Ridge) prevents overfitting without eliminating features.

Note: L1 (Lasso) was used for feature selection (Section 3.1), while L2 (Ridge) is used for the final model.

4.2 Hyperparameter Tuning

Tuned the regularization parameter C by evaluating precision and recall on the validation set. Figure 8 shows the tuning results for Model 1 and Figure 9 shows tuning results for Model 2.

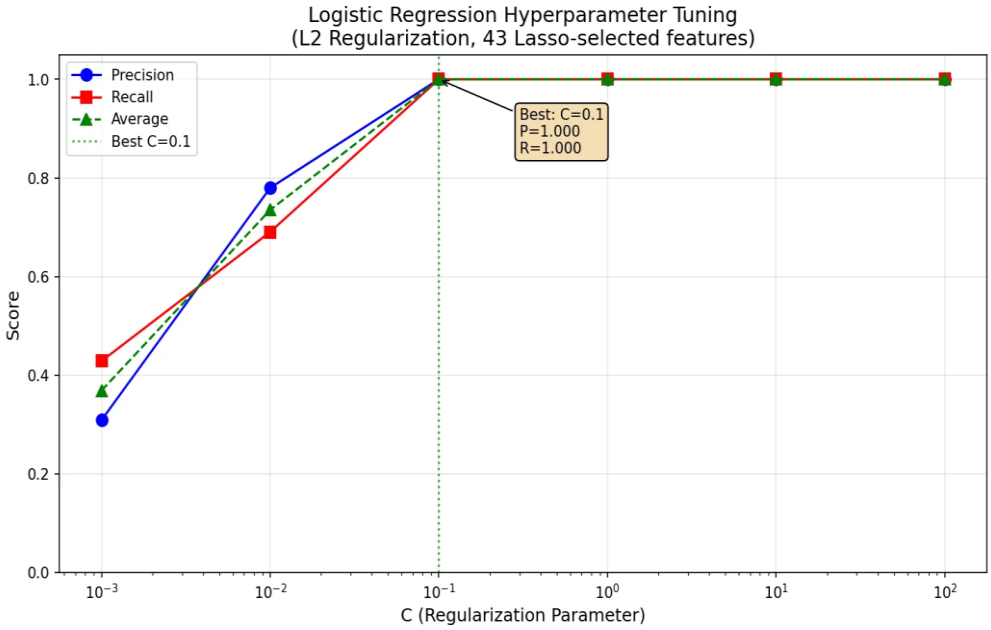


Figure 8: Hyperparameter tuning for Model 1. Both precision and recall reach 1.0 at C=0.1, which was selected as the optimal value.

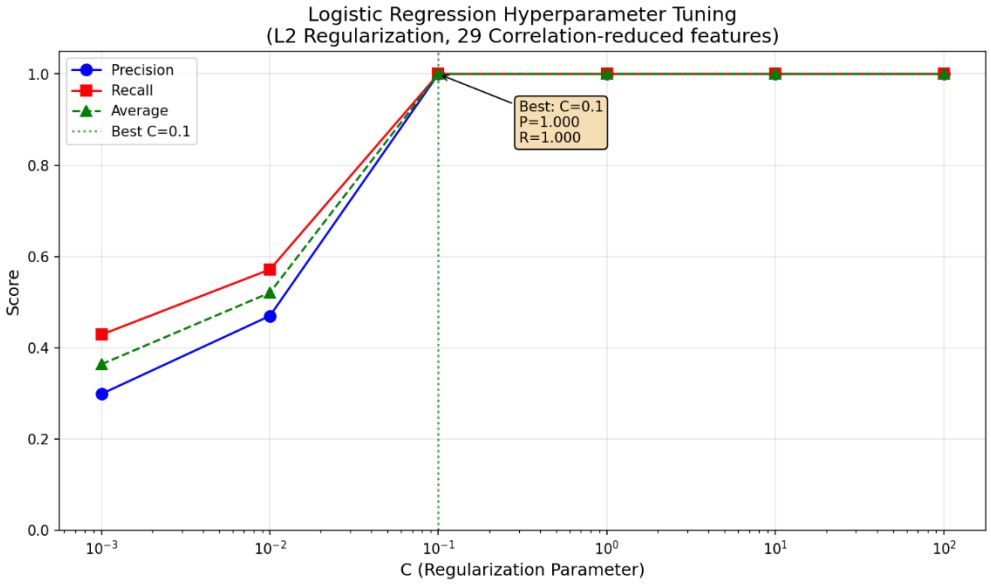


Figure 9: Hyperparameter tuning for Model 2. Both precision and recall reach 1.0 at C=0.1, which was selected as the optimal value.

Model 1 (43 features): Selected C=0.1 (first value achieving perfect validation performance)

Model 2 (29 features): Selected C=0.1 (same optimal point)

4.3 Tuning Observations

- Both models achieve 100% validation accuracy at C≥0.1
- Very low C (high regularization) causes underfitting - the model cannot capture class differences
- At C=0.01, Model 1 already reaches ~78% precision while Model 2 is at ~47%, suggesting Lasso-selected features are more immediately discriminative
- Performance plateaus at C≥0.1, so I selected the smallest C that achieves perfect validation performance to maintain regularization

5. Learning Curve Analysis

5.1 Methodology

Trained each model on progressively larger subsets of the training data (20% to 100%) and measured accuracy on both the training subset and the full validation set. This reveals how performance scales with data quantity.

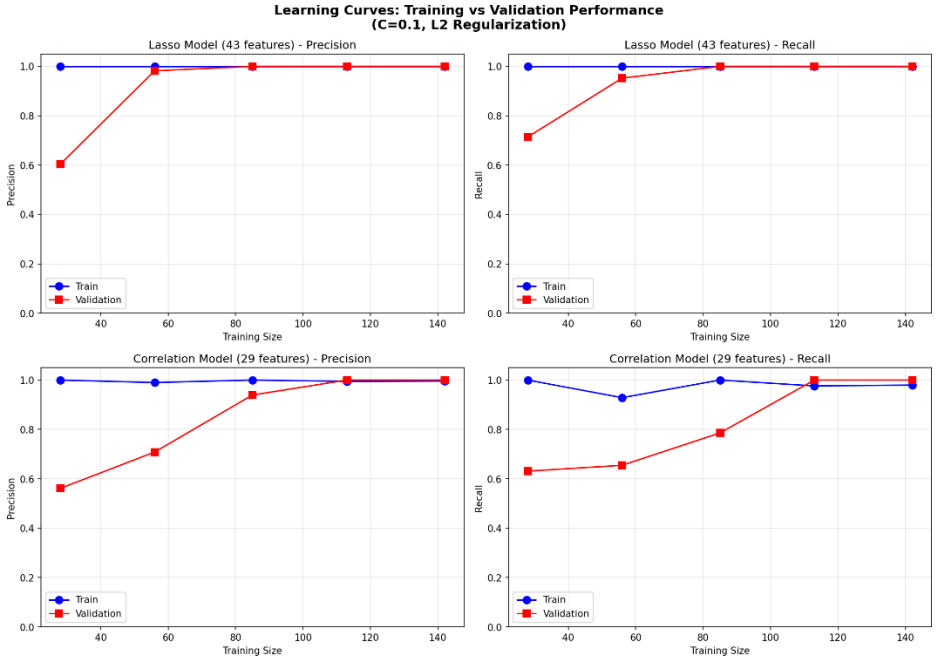


Figure 10: Learning curves for both models showing training and validation accuracy as training set size increases.

5.2 Interpretation

Model 1 (Lasso, 43 features): Training accuracy reaches 100% quickly and validation accuracy converges to ~100% by 85 training samples. The small gap between curves indicates good generalization with no overfitting.

Model 2 (Correlation, 29 features): Training accuracy falls below validation accuracy. This suggests the 29 features may not fully capture the patterns in certain training subsets, indicating the model is borderline underfitting.

Conclusion: Model 1 shows healthier learning curves. Model 2's pattern suggests the correlation-reduced feature set may have removed some useful information, which aligns with its slightly worse test performance.

6. Performance Analysis

6.1 Test Set Results

Final evaluation on the held-out test set (38 samples, including 5 Murad and 5 Teymur):

Metric	Lasso Model (43)	Correlation Model (29)
Accuracy	100%	97.4%
Precision (macro)	100%	98.4%
Recall (macro)	100%	97.1%
Total Errors	0	1

6.2 Brother Classification Performance

The core research question: Can we distinguish between the brothers? The confusion matrices (Figure 11) show both models correctly classified all Teymur samples. The Lasso model got all Murad samples correct, while the correlation model misclassified one Murad as Giorgi.

Critical finding: Neither model ever confused one brother for the other. The single error was Murad→Giorgi.

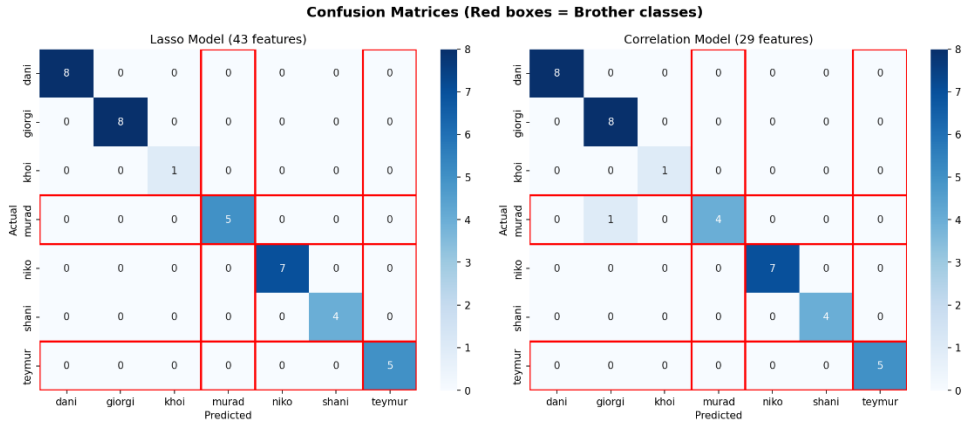


Figure 11: Confusion matrices for both models. Lasso model shows perfect diagonal. Correlation model shows one off entry: Murad→Giorgi.

6.3 ROC Analysis

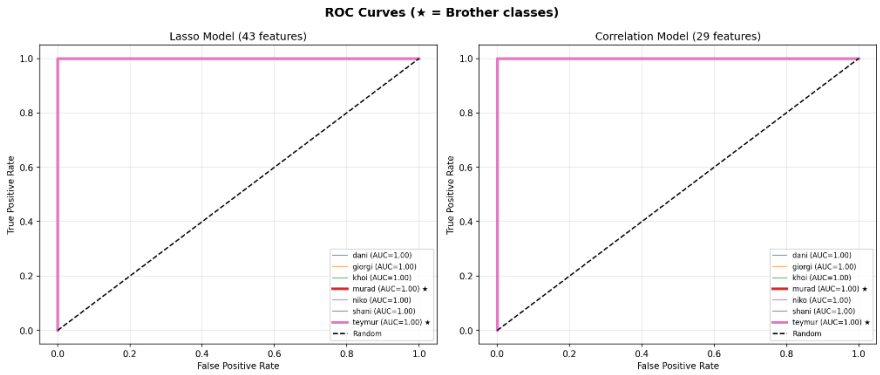
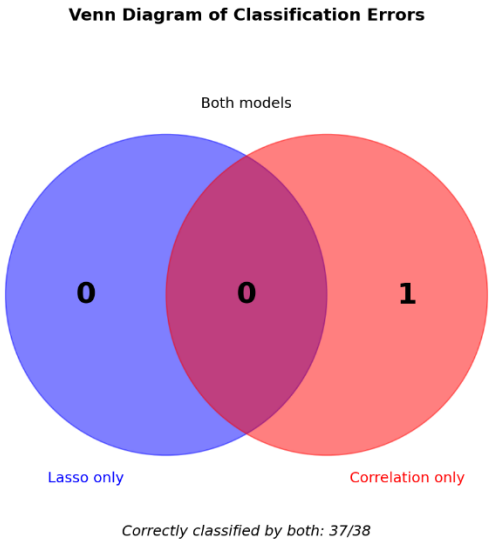


Figure 12: ROC curves for all speaker classes. Brother classes are marked with a star.

All classes achieved AUC = 1.00 for both models. Note that AUC measures ranking ability (how well probability scores separate classes), not hard classification accuracy. The correlation model's single

error (Murad→Giorgi) was likely a close decision where the correct class still had a high probability score.

6.5 Error Analysis



The Lasso model made no errors. The correlation model made one error (Murad→Giorgi) that the Lasso model classified correctly. This suggests the features removed during correlation reduction contained information that distinguishes Murad from Giorgi. Importantly, the error was not a brother confusion - the model did not mistake Murad for Teymur.