RAVDESS Emotion Recognition

Preliminary Model Output - Baseline SVM

Nico Estreba

Introduction: Why Emotion Recognition?

- **Human Communication**: Emotions convey meaning beyond words
- **Educational Applications**:
  - Detect student engagement levels
  - Provide personalized feedback
  - Support student well-being
- **Technical Challenge**:
  - Subjective and context-dependent
  - Requires reliable automated systems

Project Objectives

Immediate Goal: Establish baseline performance

- Build **traditional ML models** (SVM, Logistic Regression)
- Identify lower bound for model accuracy
- Understand dataset challenges and limitations

Ultimate Goal: Deploy deep learning solution

- Develop CNN/RNN models for improved accuracy
- Create interactive application for high school students
- Students record themselves reading a statement
- System processes audio and predicts emotion in real-time

Success Criteria: 70-80% macro-F1 score, < 3s inference latency

Project Overview

RAVDESS Dataset: Ryerson Audio-Visual Database of Emotional Speech and Song

- **Goal**: Classify emotions from audio features
- **Target Variable**: 8 emotion categories
  - angry, calm, disgust, fearful, happy, neutral, sad, surprised
- **Dataset Size**: 1,440 samples
- **Actors**: 24 professional actors (12 male, 12 female)
- **Features**: 90,003 audio signal features (y_0 to y_89995)

Data Pipeline

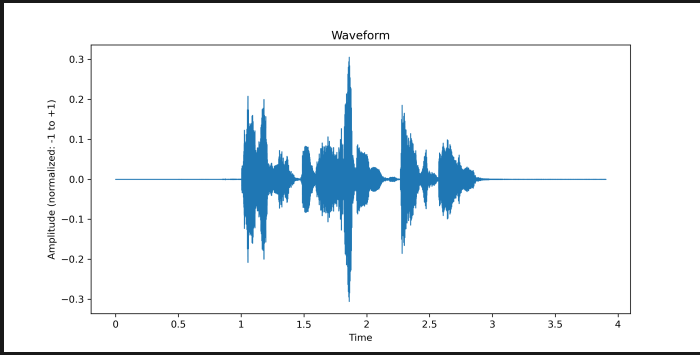
Raw Audio Files (1,440 .wav files)
↓
[1] Metadata Extraction (filename parsing)
↓
[2] EDA & Analysis (distributions, waveforms)
↓
[3] Preprocessing (silence trimming, standardization)
↓
[4] Analytics Base Table (1,440 × 90,005)
↓
[5] Baseline Modeling (SVM with raw features)
↓
[Future] Feature Engineering & Deep Learning

Dataset Structure

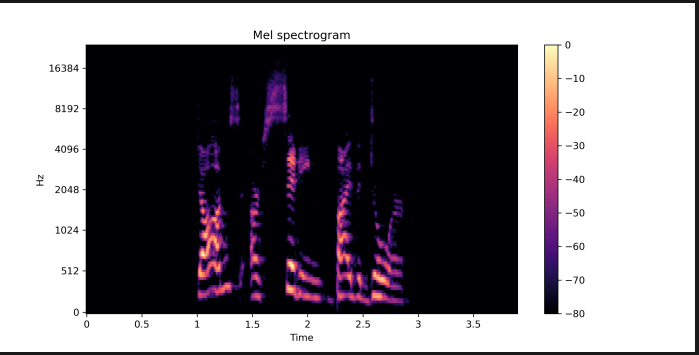
- **Audio Features**: Raw audio signal data points
- **Metadata**:
  - modality, vocal_channel, emotion_code
  - intensity, statement, repetition, actor
- **Target**: emotion (categorical)

```
1 dataset.shape  # (1440, 90005)
```

Sample Audio Visualization



Waveform



Mel Spectrogram

Listen to Sample Emotions

Sample Audio Files (Actor 16, Female, Angry emotion):

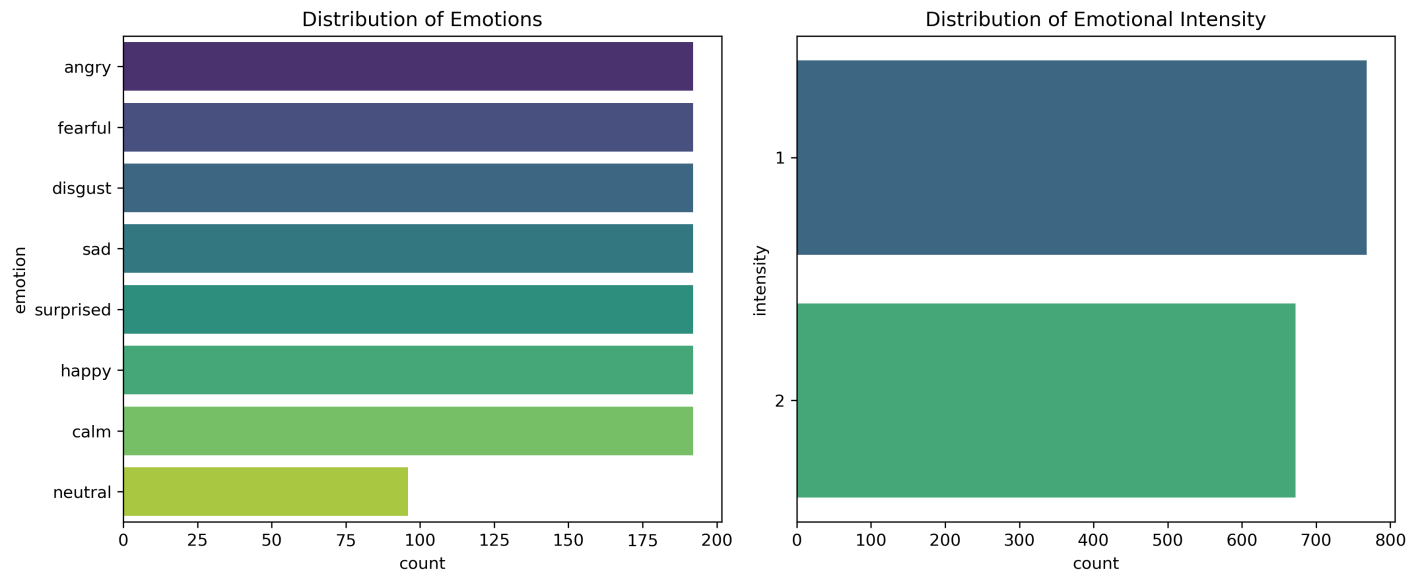
Original with silence: 

Dataset Information:

* File: `03-01-05-01-02-01-16.wav`
* Emotion: Angry (code 5)
* Intensity: Normal (1)
* Statement: “Dogs are sitting by the door” (statement 2)
* Duration: ~3.9 seconds

Note: All audio files undergo silence trimming and standardization to 89,996 samples (~4.08s) for model training.

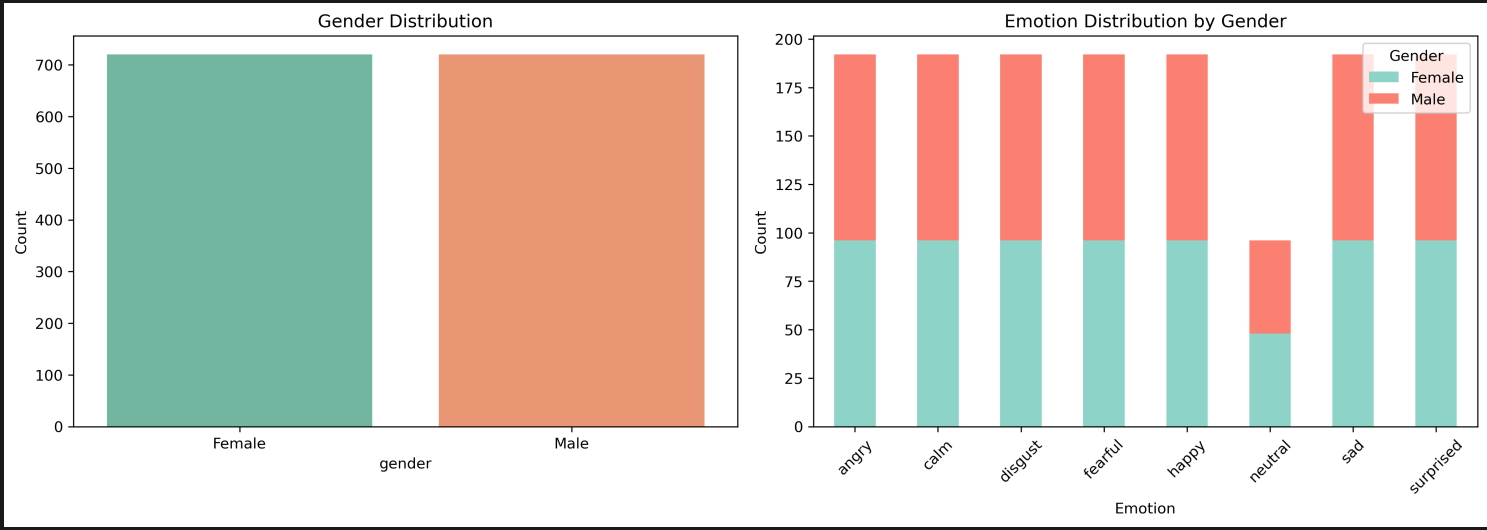
Emotion & Intensity Distribution



Distribution Analysis

- **Balanced dataset**: ~180 samples per emotion class
- **Equal intensity levels**: Normal vs Strong emotional expression

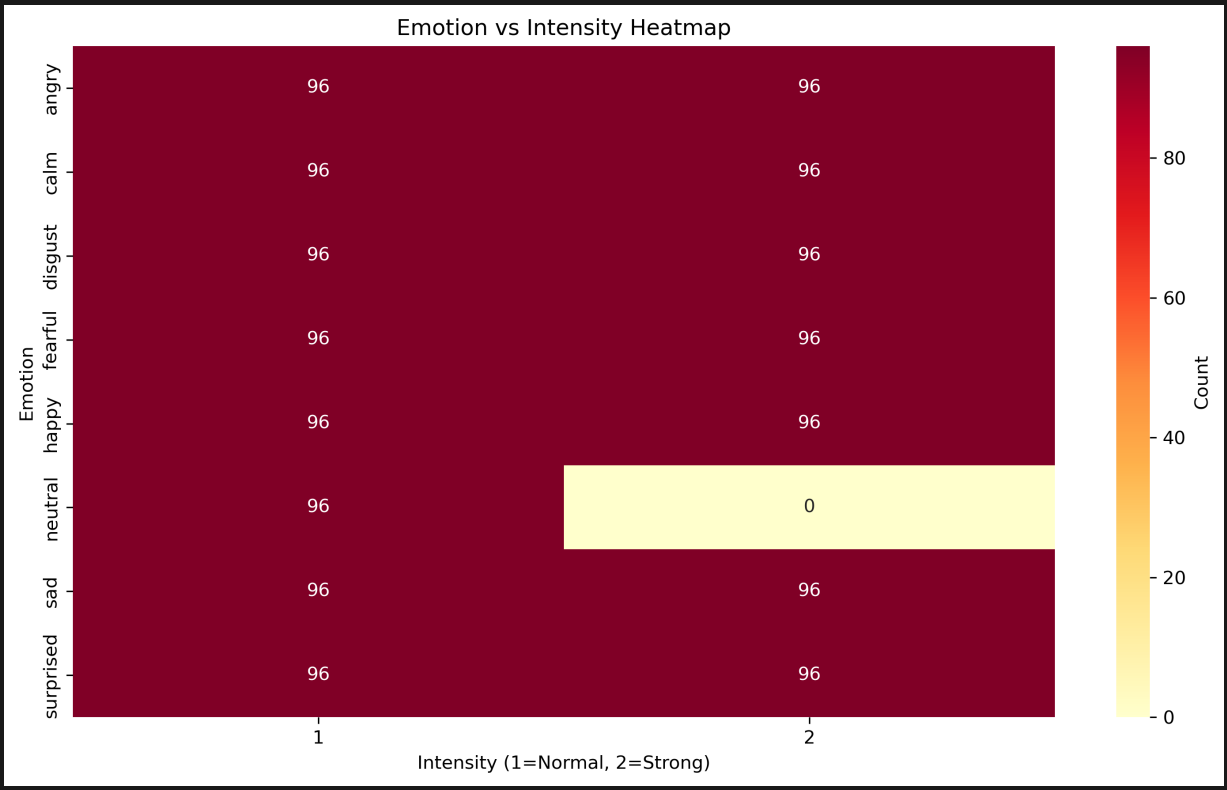
Gender Analysis



Gender Distribution

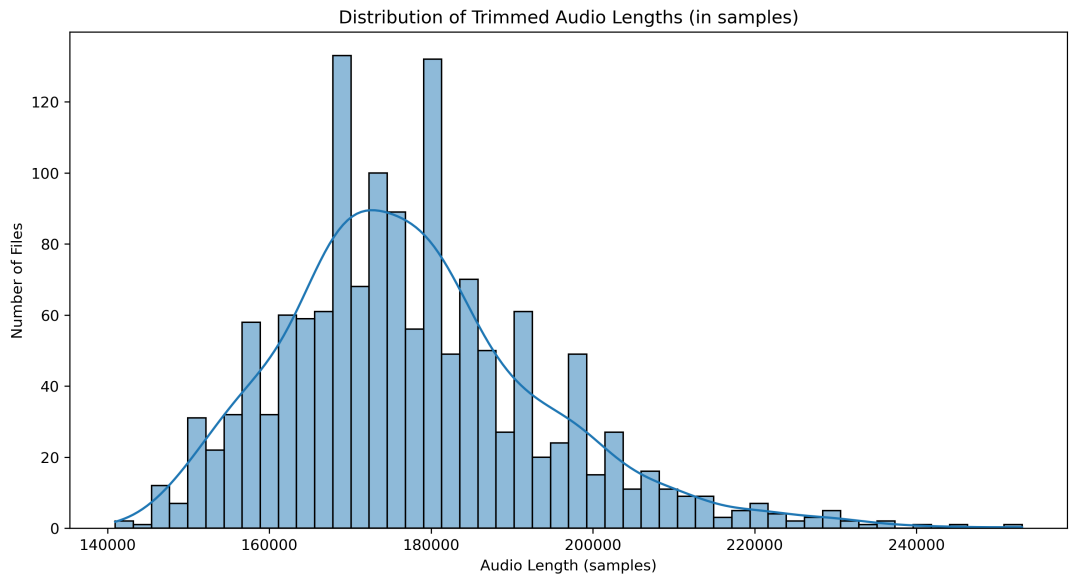
- **Balanced gender representation**: 12 male, 12 female actors
- **Consistent across emotions**: Even distribution per emotion class

Emotion vs Intensity Relationship



Heatmap Analysis

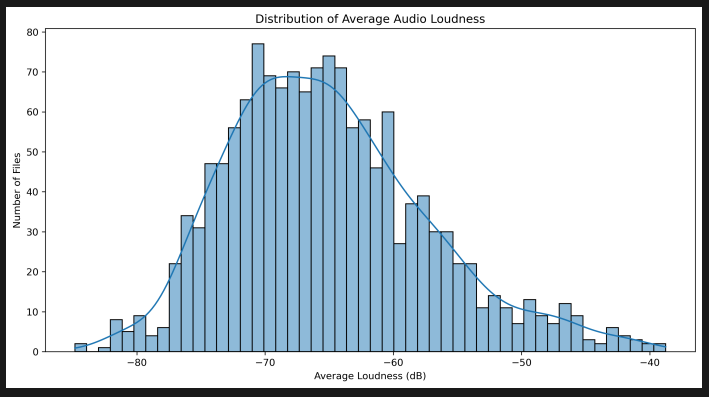
Audio Length Distribution



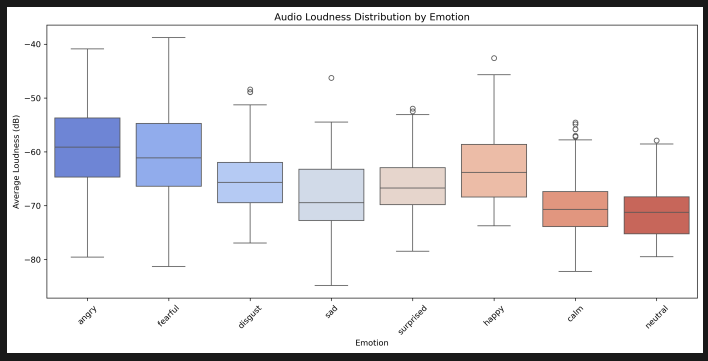
Audio Length

- **Variable lengths**: Range from ~150K to 250K samples
- **Need for standardization**: Critical for ML model input

Loudness Analysis

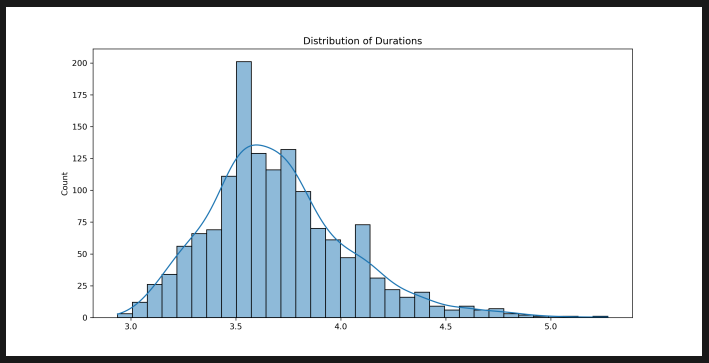


Overall Distribution

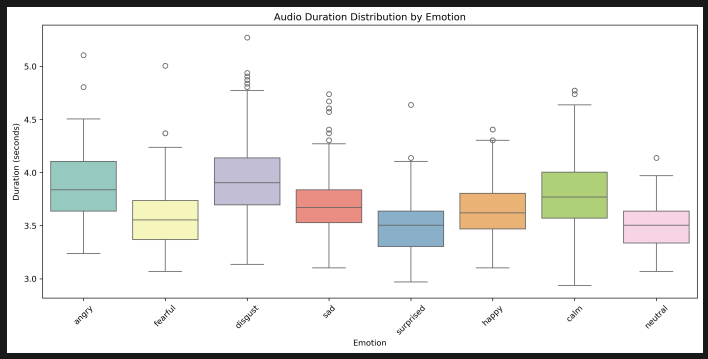


By Emotion

Duration Analysis



Overall Distribution



By Emotion

- **Mean duration**: ~3.5-4 seconds
- **Consistent across emotions**: Minimal variation

Preprocessing: Silence Removal

Challenge: Variable audio lengths with leading/trailing silence

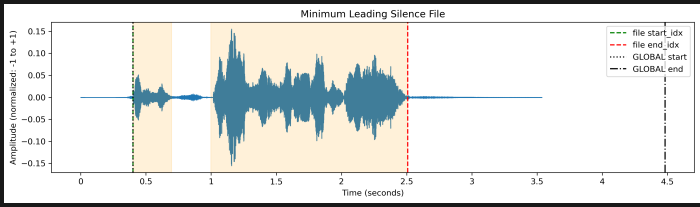
Solution: Global standardization approach

1. Detect non-silent regions using energy thresholds (top_db=25)
2. Find global start/end indices across all files
3. Trim all audio to common length (89,996 samples ≈ 4.08s)
4. Pad shorter files with zeros

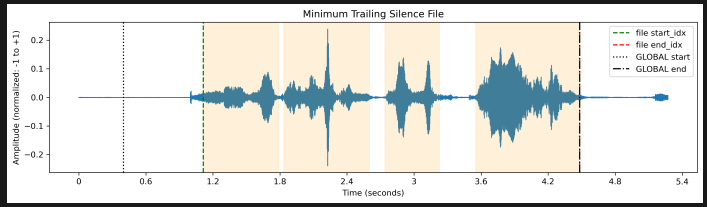
Result: Uniform input dimensions for machine learning

Silence Detection Visualization

Minimum Leading Silence



Minimum Trailing Silence



- **Orange shaded regions**: Detected speech
- **Green/Red lines**: File-specific boundaries
- **Black lines**: Global trim boundaries

Audio Comparison: Original vs Trimmed

Minimum Leading Silence Example

Original (with silence):

Trimmed (silence removed):

Minimum Trailing Silence Example

Original (with silence):

Trimmed (silence removed):

Result: Both trimmed to exactly 89,996 samples (4.08s)

Train-Test Split

Split Configuration * Training: 70% (1,008 samples) * Testing: 30% (432 samples) * Random state: 42

Preprocessing * Removed non-feature columns: `path`, `emotion` * Applied StandardScaler * Used sklearn pipeline

Why SVM for Baseline?

Advantages:

- Well-established for high-dimensional data
- Effective with limited training samples
- Good generalization with proper kernel selection

Baseline Strategy:

- Use **raw audio samples** (no feature engineering yet)
- RBF kernel for non-linear patterns
- StandardScaler for normalization

Expectation:

- Performance will be limited due to:
  - No domain-specific features (MFCCs, spectrograms)
  - High dimensionality (90K features vs 1.4K samples)
- Establishes lower bound for improvement

Baseline SVM Model

Model Configuration:

```
svm_pipeline = make_pipeline(
    StandardScaler(),
    SVC(kernel='rbf', random_state=42)
)
```

- **Kernel**: RBF (Radial Basis Function)
- **Scaling**: StandardScaler for feature normalization
- **Training Time**: ~79.51 seconds

Model Performance

Accuracy Scores

Metric	Score
Training Accuracy	**40.67%**
Testing Accuracy	**22.69%**

Observations * Low overall performance * Signs of overfitting * Testing accuracy < Training accuracy

Classification Report

Emotion	Precision	Recall	F1-Score	Support
angry	0.40	0.50	0.45	62
calm	0.19	**0.95**	0.32	57
disgust	0.13	0.11	0.12	56
fearful	**0.00**	**0.00**	**0.00**	55
happy	0.29	0.03	0.06	61
neutral	**0.00**	**0.00**	**0.00**	29
sad	0.22	0.09	0.13	53
surprised	**0.00**	**0.00**	**0.00**	59

Overall Accuracy: 23%

Key Findings

1. **Model Bias**: Strong bias towards predicting “calm” emotion
   - Calm recall: 95% (predicts calm too often)
   - Calm precision: 19% (many false positives)
2. **Failed Classes**: Some emotions never predicted
   - fearful, neutral, surprised: 0% precision
3. **Dimensionality Challenge**: 90K features vs 1.4K samples
   - High-dimensional curse of dimensionality

Challenges Identified

1. **High Dimensionality**
   - 90,003 features >> 1,440 samples
   - Ratio: ~62.5 features per sample
2. **Potential Class Imbalance**
   - Uneven distribution across emotions
3. **Feature Quality**
   - Raw audio signals may need feature engineering

Next Steps & Improvements

Feature Engineering:

- Apply PCA for dimensionality reduction
- Extract MFCC features
- Use feature selection techniques

Model Optimization:

- Hyperparameter tuning (GridSearch/RandomSearch)
- Try different kernels (linear, polynomial)
- Adjust C and gamma parameters

Logistic Regression Model

Model Configuration:

```
lr_pipeline = make_pipeline(
    StandardScaler(),
    LogisticRegression(max_iter=1000, random_state=42)
)
```

- **Algorithm**: Multinomial Logistic Regression
- **Scaling**: StandardScaler for feature normalization
- **Training Time**: ~12.83 seconds (6x faster than SVM)

Logistic Regression: Performance

Accuracy Scores

Metric	Score
Training Accuracy	**100.00%**
Testing Accuracy	**16.90%**

Observations * Severe overfitting * Perfect training fit, poor generalization * Worse than SVM baseline

Logistic Regression: Classification Report

Emotion	Precision	Recall	F1-Score	Support
angry	0.40	0.03	0.06	62
calm	0.23	**0.68**	0.34	57
disgust	0.19	0.12	0.15	56
fearful	0.09	0.04	0.05	55
happy	0.13	0.03	0.05	61
neutral	**0.03**	**0.03**	**0.03**	29
sad	0.13	0.30	0.18	53
surprised	0.19	0.07	0.10	59

Overall Accuracy: 17%

Model Comparison: SVM vs Logistic Regression

Metric	SVM (RBF)	Logistic Regression	Winner
Training Accuracy	40.77%	**100.00%**	LR (overfitting)
Testing Accuracy	**22.69%**	16.90%	**SVM**
Training Time	72.82s	**12.83s**	**LR**
Overfitting Gap	**18.08%**	83.10%	**SVM**
Best Class Recall	calm (**95%**)	calm (68%)	**SVM**
Macro F1-Score	**0.13**	0.12	**SVM**

Key Takeaway: SVM provides better baseline despite being slower. Both models severely underperform due to high dimensionality (90K features, 1.4K samples).

Random Forest Model

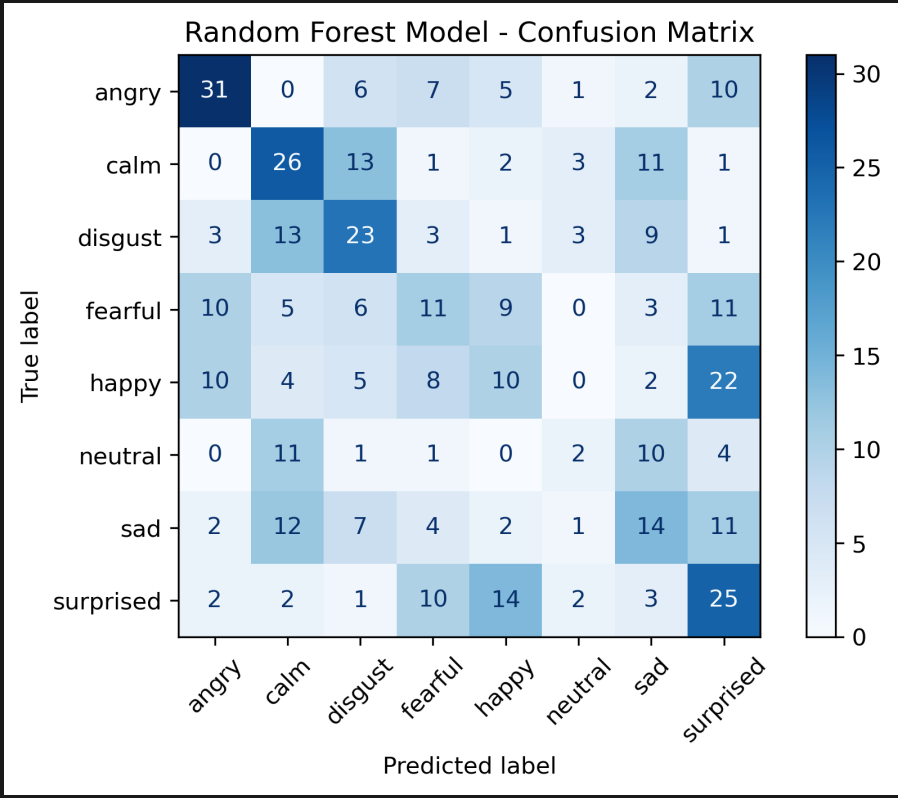
Model Configuration:

- Baseline with `max_features='sqrt'` for speed
- Training Accuracy: 95.44%
- Testing Accuracy: 34.03%
- Overfitting Gap: 61.41%

Observations:

- Better than SVM/LR (+11% test accuracy)
- Still severe overfitting
- Handles high dimensionality better than linear models

Random Forest: Confusion Matrix



Gradient Boosting Model

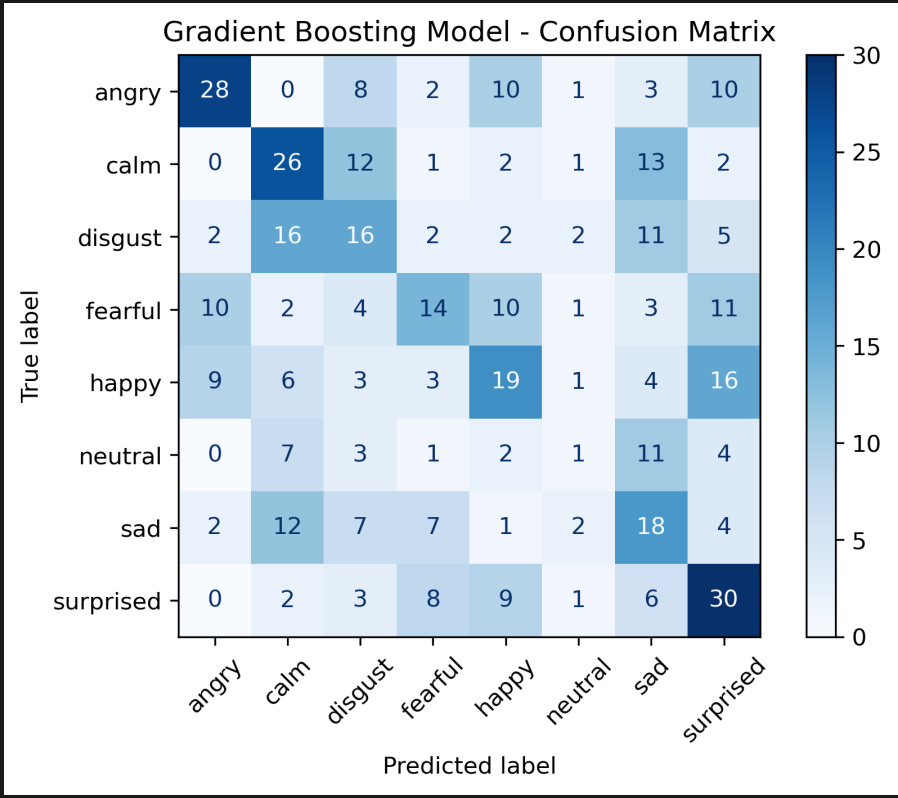
Model Configuration:

- Baseline with shallow trees (`max_depth=3`)
- Training Accuracy: 100.00%
- Testing Accuracy: 33.56%
- Overfitting Gap: 66.44%

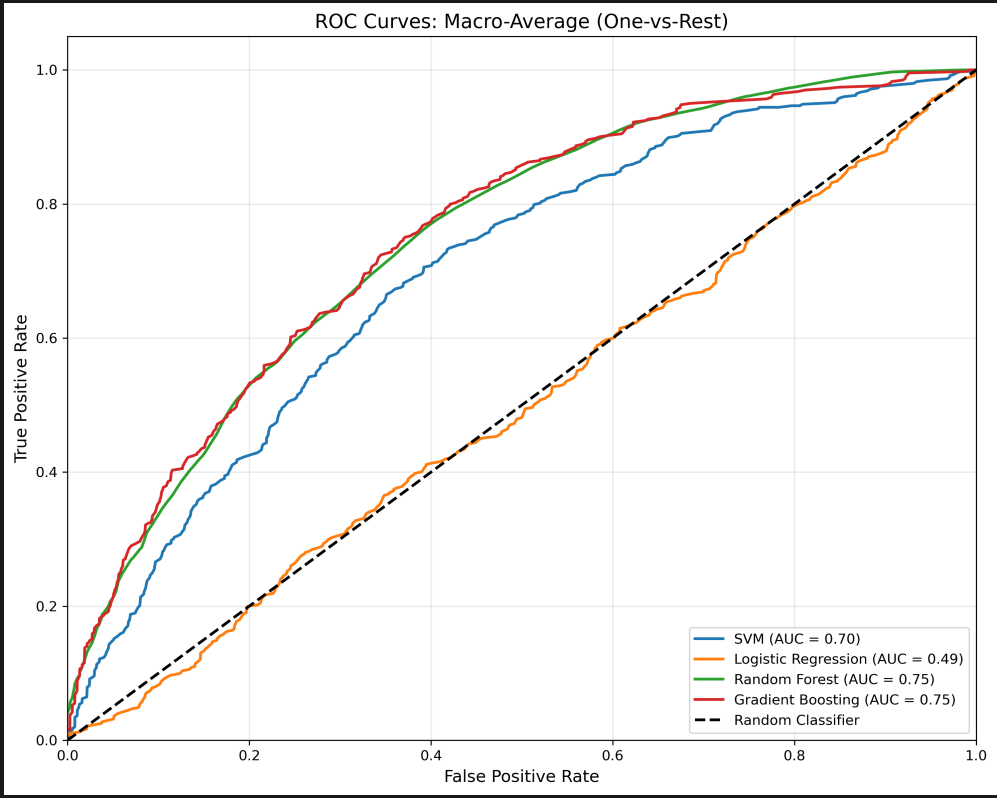
Observations:

- Similar performance to Random Forest
- Extreme overfitting (100% train accuracy)
- Benefits from `max_features='sqrt'` optimization

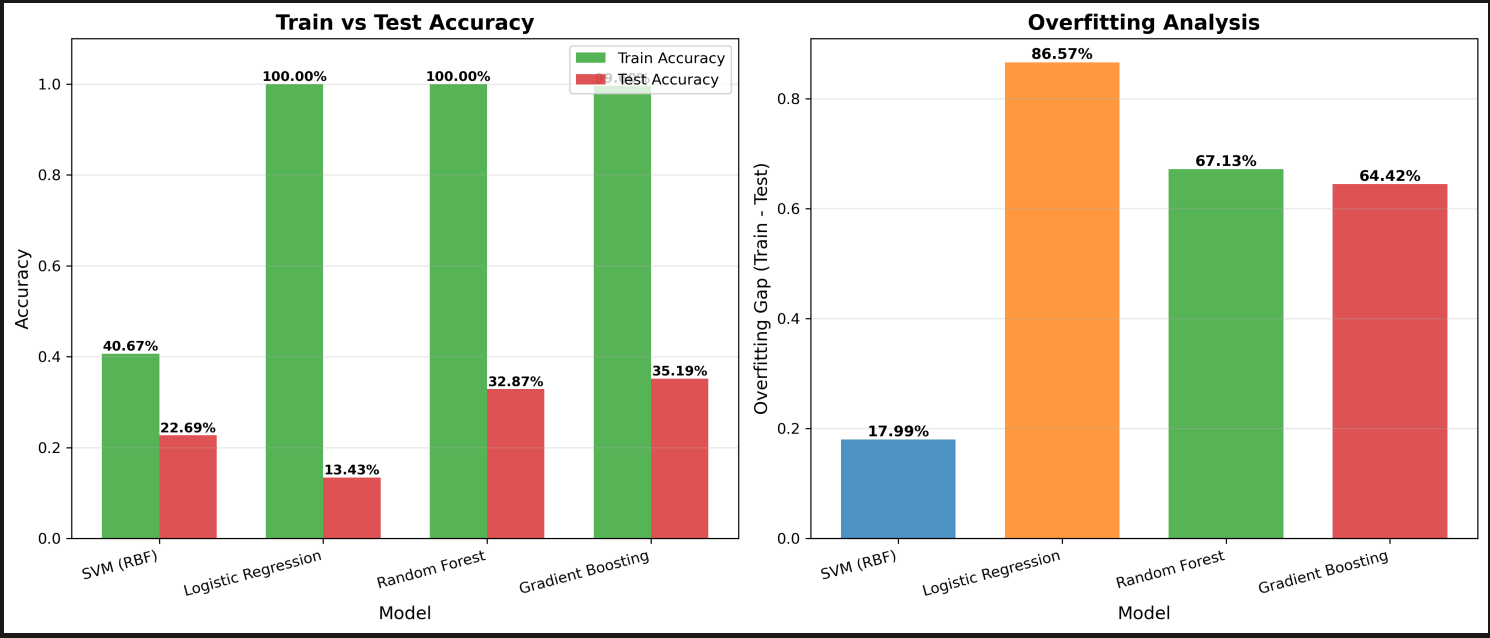
Gradient Boosting: Confusion Matrix



ROC Curves: All Preliminary Models



Preliminary Models Comparison



Key Insight: Random Forest and Gradient Boosting show promise but need hyperparameter tuning to reduce overfitting.

Hyperparameter Tuning: Methodology

Two Approaches Compared:

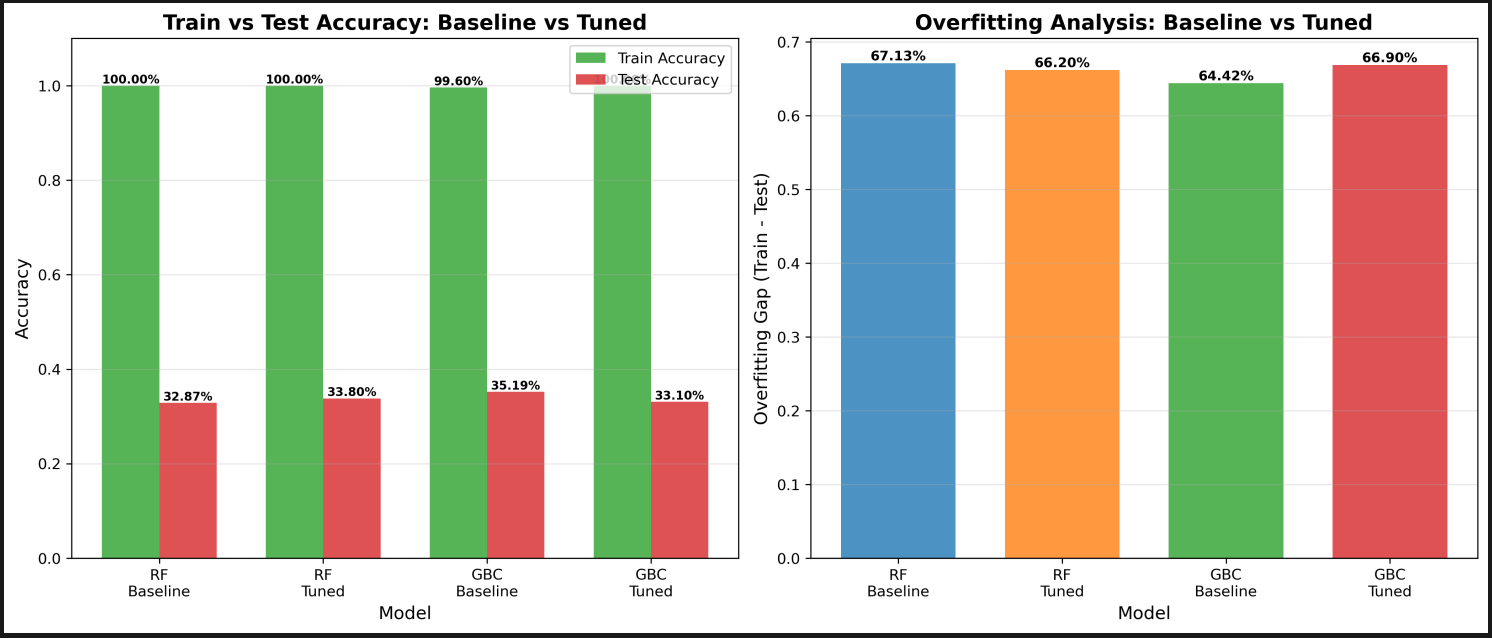
- **RandomizedSearchCV**: Grid-based exploration (8 trials, 5-fold CV)
- **Optuna**: Bayesian optimization (30 trials, TPE sampler)

Hyperparameters Tuned:

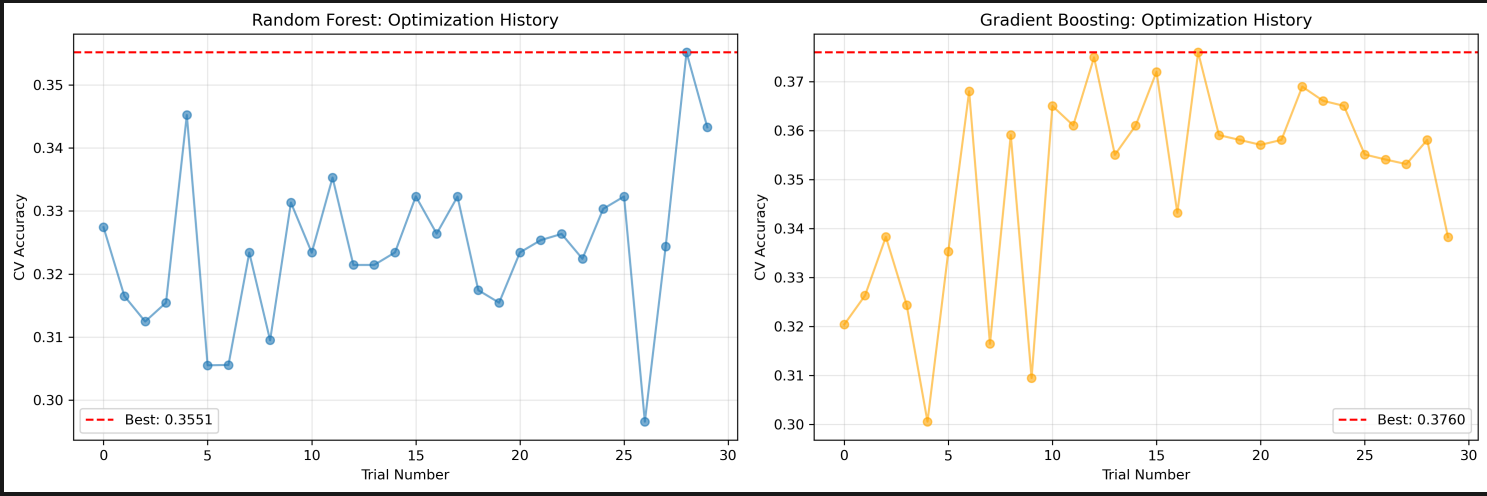
- n_estimators: 50-200
- max_depth: 3-15
- learning_rate: 0.01-0.3 (GBC only)
- max_features: sqrt vs log2
- subsample: 0.6-1.0 (GBC only)

Goal: Find optimal parameters to reduce overfitting while maintaining test accuracy

Baseline vs RandomizedSearch

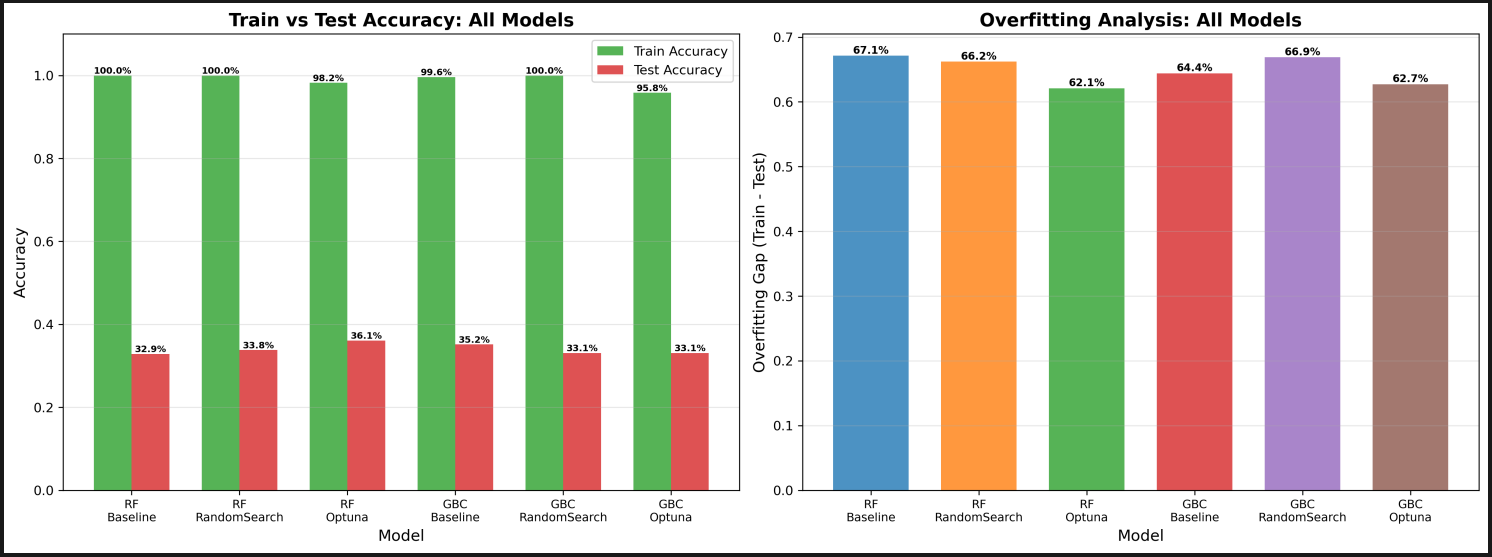


Optuna Optimization History

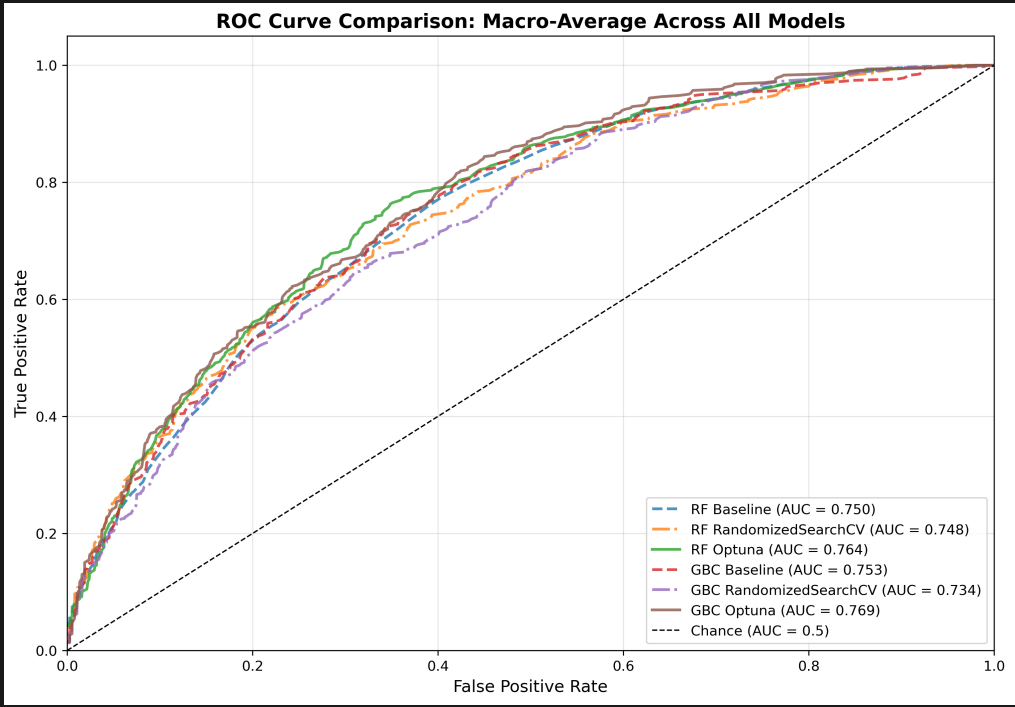


Observation: Optuna converges to optimal hyperparameters within 20-25 trials using Bayesian optimization.

Final Comparison: 6 Tuned Models



ROC Curves: All Tuned Models



Best Performance: GBC with Optuna (37.60% test accuracy, 76.87% macro-avg AUC)

Dimensionality Reduction: Motivation

The Problem:

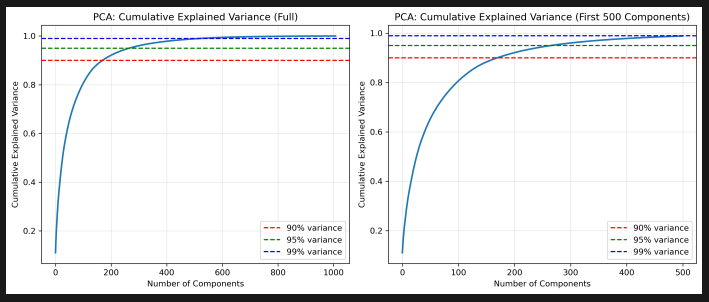
- 90,004 features vs 1,008 training samples
- Ratio: ~89 features per sample (curse of dimensionality)
- Models struggle to generalize

Approaches Tested:

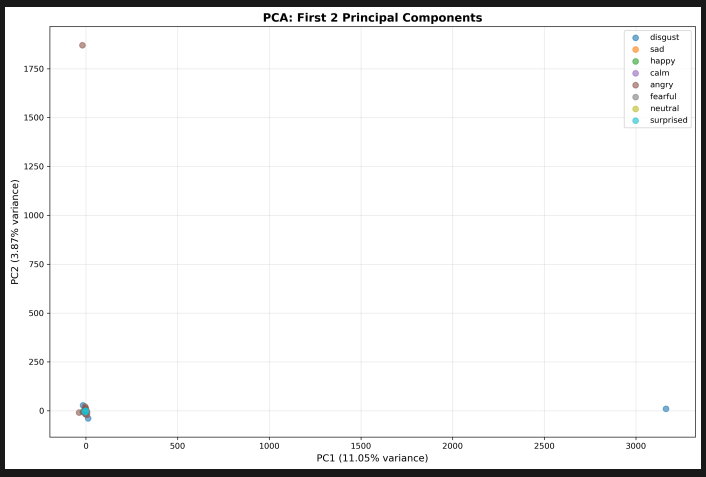
- **PCA**: Unsupervised, variance-based (95% variance → 266 components)
- **LDA**: Supervised, class-separability-based (7 components for 8 classes)
- **PCA+LDA**: Hybrid approach (300 → 7 components)

Method: Optuna tuning for fair comparison across all feature spaces

PCA Analysis



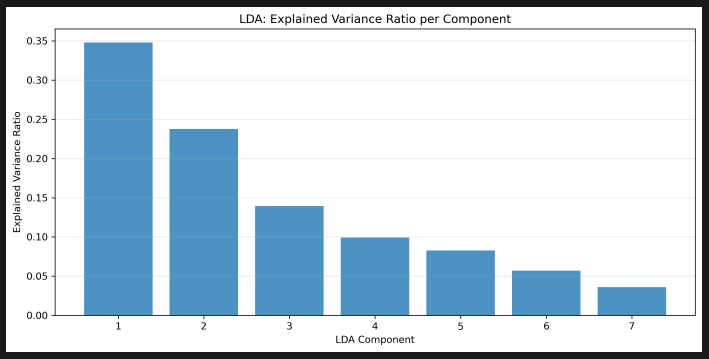
Cumulative Variance



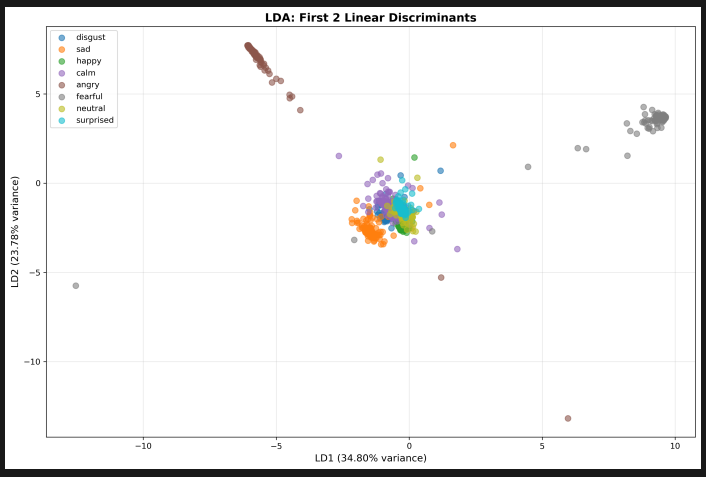
2D Visualization

Results: 95% variance retained with 266 components (99.70% reduction)

LDA Analysis



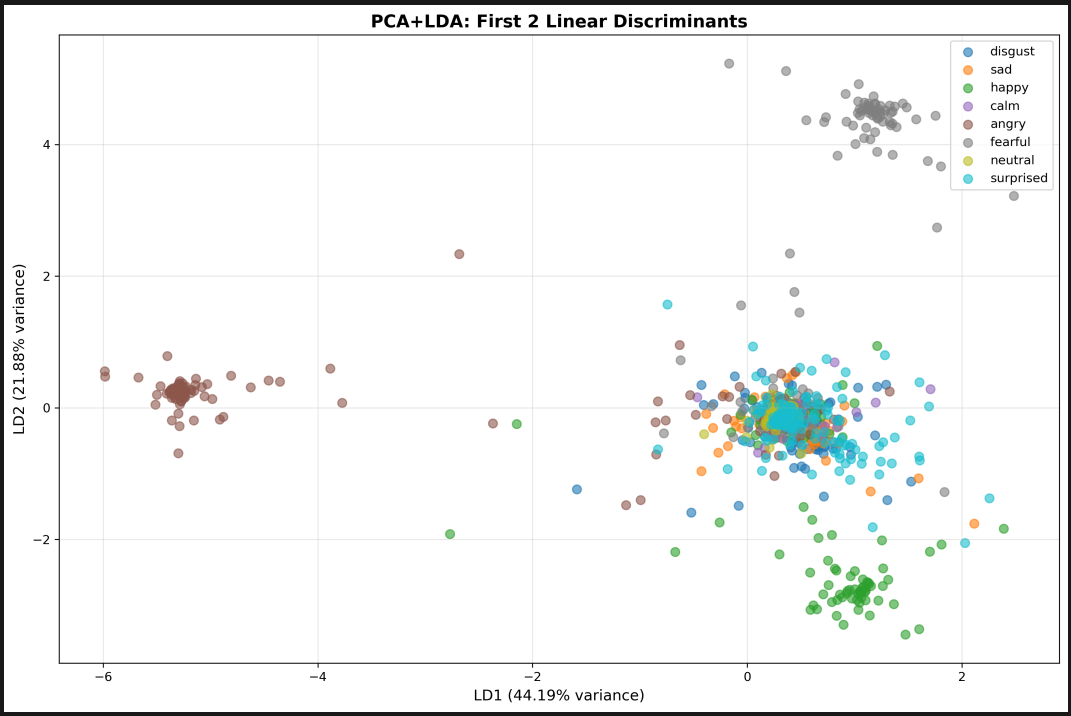
Variance per Component



2D Visualization

Results: 7 components for 8-class problem (99.99% reduction)

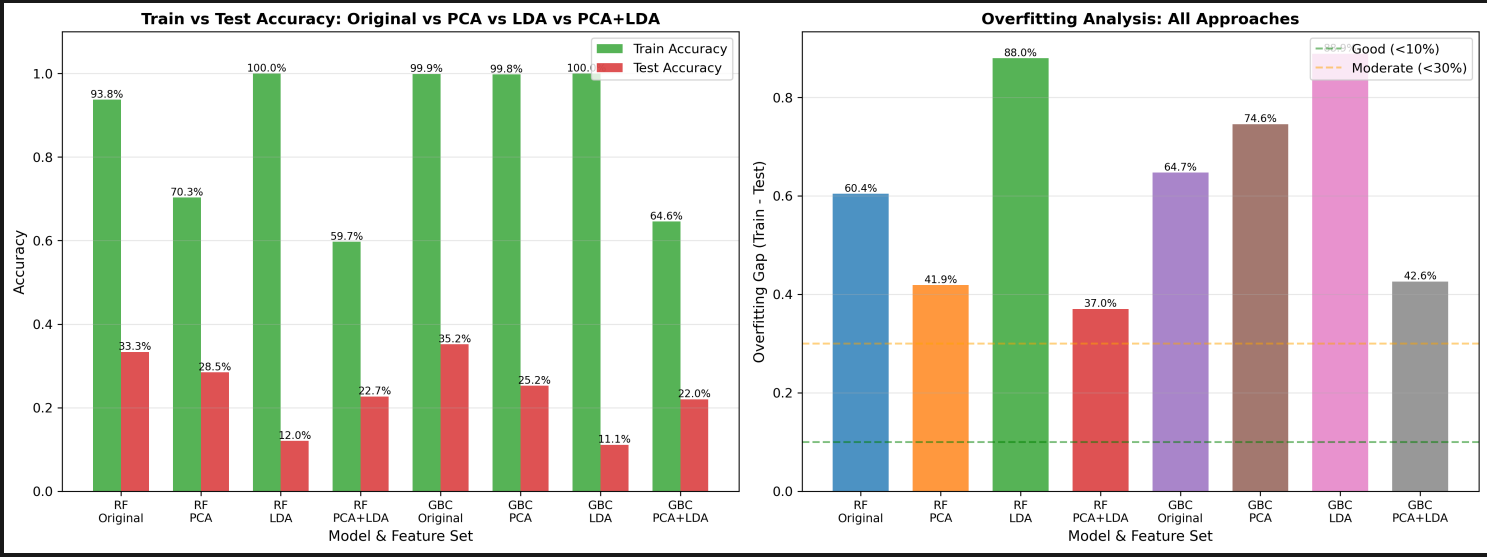
PCA+LDA Hybrid Approach



2D Visualization

Pipeline: 90,004 features → PCA (300 components) → LDA (7 components)

Dimensionality Reduction Results



Surprising Finding: Original features outperform all dimensionality reduction methods! Aggressive reduction loses critical temporal information.

Conclusion

✅ **Comprehensive Baseline Established**:

- **Traditional ML**: SVM (22.69%), LR (16.90%), **RF (34.03%)**, **GBC (33.56%)**
- **Hyperparameter Tuning**: GBC+Optuna achieved **37.60%** (best performance)
- **Dimensionality Reduction**: Original features outperform PCA/LDA (temporal information loss)

📊 **Key Findings**:

- **Tree-based models** outperform linear models by +11-15% test accuracy
- **Optuna tuning** provides +4% improvement over baseline (GBC: 33.56% → 37.60%)
- **Dimensionality reduction fails**: 90K features needed to preserve audio patterns
- All models suffer **severe overfitting** (60-88% gaps) due to limited training data

🎯 **Recommendations**:

- **Feature Engineering**: Extract MFCCs, mel-spectrograms (domain-specific features)
- **Deep Learning**: CNN/LSTM architectures for automatic temporal pattern learning
- **Data Augmentation**: Increase training samples to combat overfitting
- **Current Best**: GBC with Optuna (37.60% accuracy, 76.87% macro-AUC) - **baseline to beat**

Questions?

Thank you!