

Speech Emotion Recognition - Week 5

Integration, Documentation & Demo

Krishna Kukreja & Garvit Meena

November 2025

Week 5: Resources & Guide

1. Learning Objectives

- Integrate all modules into end-to-end pipeline.
- Create production-quality code.
- Build demo script for new audio samples.
- Prepare presentation and documentation.
- Buffer time for catching up on previous weeks.

2. Tasks Overview

1. **Integration:** Combine feature extraction + model prediction.
2. **Documentation:** Write comprehensive README and code comments.
3. **Demo:** Create working demo script.
4. **Presentation:** Prepare slides and demo video.
5. **Catch-up:** Complete any pending tasks from Weeks 1-4.

3. Reference Materials

- **Python Documentation Standards:** [PEP 257 - Docstring Conventions](#)
- **README Template:** [Best README Template on GitHub](#)
- **Code Comments:** [RealPython - Documenting Python Code](#)

4. Python Code Templates

End-to-End Prediction Pipeline

```
import pickle
import librosa
import numpy as np

class EmotionPredictor:
    def __init__(self, model_path='models/svm_model.pkl',
                  scaler_path='models/scaler.pkl'):
        with open(model_path, 'rb') as f:
            self.model = pickle.load(f)
        with open(scaler_path, 'rb') as f:
            self.scaler = pickle.load(f)

        self.emotion_map = {
            1: 'Neutral', 2: 'Calm', 3: 'Happy',
            4: 'Sad', 5: 'Angry', 6: 'Fearful',
            7: 'Disgust', 8: 'Surprised'
        }

    def predict(self, audio_path):
        # Load audio
        y, sr = librosa.load(audio_path, duration=3)

        # Extract features
        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
        mfcc_mean = np.mean(mfcc, axis=1)
        mfcc_std = np.std(mfcc, axis=1)

        features = np.hstack([mfcc_mean, mfcc_std])
        features = features.reshape(1, -1)

        # Normalize and predict
        features_scaled = self.scaler.transform(features)
        emotion_id = self.model.predict(features_scaled)[0]

        if hasattr(self.model, 'predict_proba'):
            probs = self.model.predict_proba(features_scaled)[0]
            confidence = probs[emotion_id - 1]
        else:
            confidence = None

        return self.emotion_map[emotion_id], confidence

# Usage
predictor = EmotionPredictor()
emotion, confidence = predictor.predict('test_audio.wav')
```

```
print(f"Emotion: {emotion}, Confidence: {confidence:.2%}")
```

Simple Demo Script

```
# demo.py
from emotion_predictor import EmotionPredictor

def main():
    print("=" * 60)
    print("SPEECH EMOTION RECOGNITION - DEMO")
    print("=" * 60)

    predictor = EmotionPredictor()

    test_files = [
        'samples/sample_happy.wav',
        'samples/sample_sad.wav',
        'samples/sample_angry.wav'
    ]

    for audio_file in test_files:
        emotion, confidence = predictor.predict(audio_file)
        print(f"\nFile: {audio_file}")
        print(f"Predicted: {emotion} ({confidence:.1%})")

if __name__ == '__main__':
    main()
```

5. Documentation Checklist

README.md with setup instructions.

Docstrings for all functions and classes.

requirements.txt file.

Usage examples in documentation.

API reference guide.

Troubleshooting section.

6. Assignments (Light)

1. Create end-to-end prediction pipeline script.
2. Write comprehensive README.md.
3. Add docstrings to all previous code.
4. Test pipeline on 5-10 new audio samples.

5. Prepare 10-slide presentation.
6. (Optional) Create 3-5 minute demo video.

7. Buffer Activities

If you finish early or need to catch up:

- Complete any pending assignments from Weeks 1-4.
- Improve model accuracy by feature engineering.
- Add more visualizations to EDA.
- Implement additional models (Neural Network, KNN).
- Create unit tests for code modules.
- Polish presentation and practice demo.

Final Deliverables Summary

Data Files

1. `data/processed/features.csv` - Extracted features ($7,356 \times 100+$)
2. `models/svm_model.pkl` - Trained SVM model
3. `models/rf_model.pkl` - Trained Random Forest model
4. `models/scaler.pkl` - Fitted StandardScaler

Documentation

1. `README.md` - Project overview and setup guide
2. Week 2 EDA report (2-3 pages)
3. Week 3-4 model comparison report (2-3 pages)
4. Final project report (5-10 pages)

Project Success Criteria

Technical Milestones

- All 7,356 features extracted successfully
- SVM validation accuracy $\geq 60\%$
- Random Forest validation accuracy $\geq 60\%$
- End-to-end pipeline working on new samples
- All code well-documented and tested

Learning Outcomes Achieved

- Understand audio signal processing theory
- Can extract MFCC and spectral features
- Can train and evaluate ML classifiers
- Can interpret confusion matrices and metrics
- Can build complete ML pipeline
- Can document and present technical work

Important Notes

- **Week 5 is intentionally light** - use it as buffer time.
- If behind schedule, focus on completing Weeks 1-4 first.
- Quality over speed - better to do 3 weeks well than 5 weeks poorly.
- Ask for help early if stuck on any concept.
- Document your learnings as you go, not at the end.

Contact & Support

- **Stack Overflow:** [Questions tagged \[librosa\]](#)
- **Reddit:** [r/MachineLearning](#)
- **Discord:** Search for "Machine Learning Discord servers"

Good luck with your project!

Remember: Learning is the goal, not perfection.