CM2013: Sleep Scoring Project

Biomedical Signal Processing

Automated Sleep Stage Classification

Project Overview

- Goal: Develop an automatic sleep scoring system
  - Multi-signal biosignal processing (EEG, EOG, EMG)
  - Machine learning classification of 5 sleep stages
- Duration: 10 weeks with 4 iterations
- Team-based development (3 members per group)
- Deliverables:
  - Working code (Python/MATLAB)
  - Technical report (15 pages max)
  - Project management documentation

Learning Objectives

- Apply signal processing techniques to real biomedical data
  - Filtering, artifact removal, feature extraction
- Implement machine learning classifiers for pattern recognition
  - k-NN, SVM, Random Forest
- Practice agile software development in teams
  - Sprints, iterative development, task management
- Develop professional documentation skills
  - Code documentation, technical reports, presentations

Assessment Criteria

- Methodology and Code Quality - 50%
  - Modular design, correct pipeline, documentation, testing
- Team Collaboration - 30%
  - Regular updates, integration, ClickUp usage
- Report & Documentation - 20%
  - Clear technical writing, comprehensive analysis

- Note: No fixed accuracy target - focus on process and learning!

DEVELOPMENT METHODOLOGY: AGILE VS. WATERFALL

Waterfall vs. Agile Methodology

Waterfall ❌

- Sequential phases
- No feedback until end
- High risk (all-or-nothing)
- Rigid - difficult to adapt
- Late discovery of issues
- Clear structure upfront

Agile ✓ (Our Approach)

✓ Iterative cycles (sprints)

✓ Feedback after each iteration

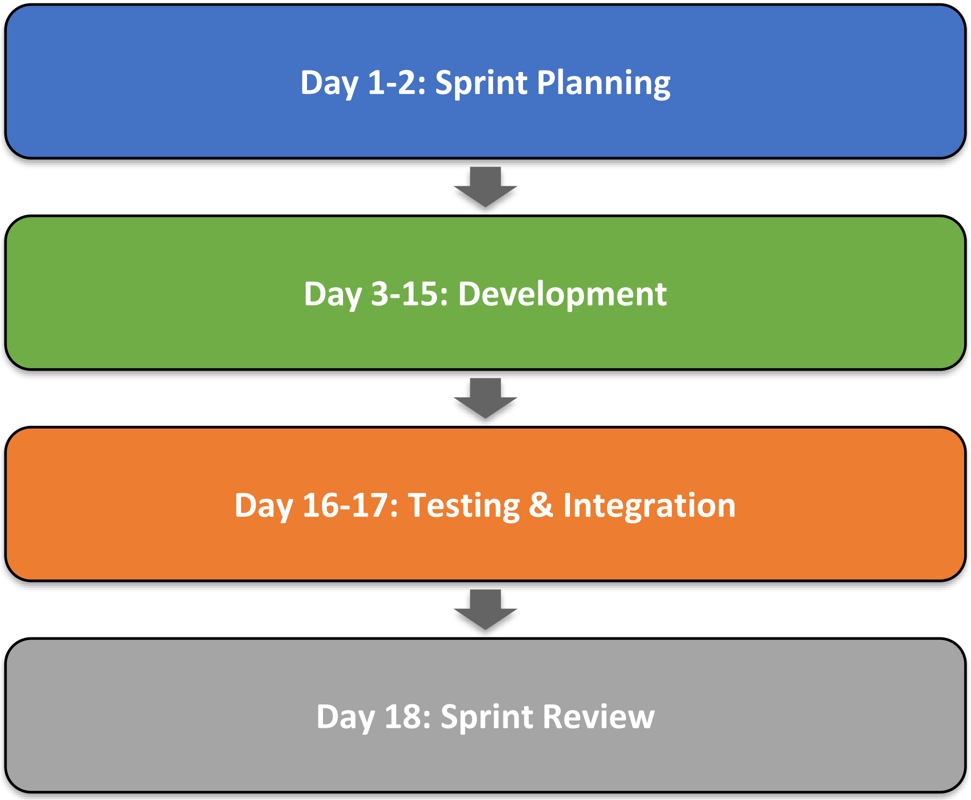
✓ Low risk (incremental)

✓ Flexible - adapt to results

✓ Early problem detection

✓ Perfect for research projects

Sprint Lifecycle (2.5 weeks each)



Day 1-2: Sprint Planning

Day 3-15: Development

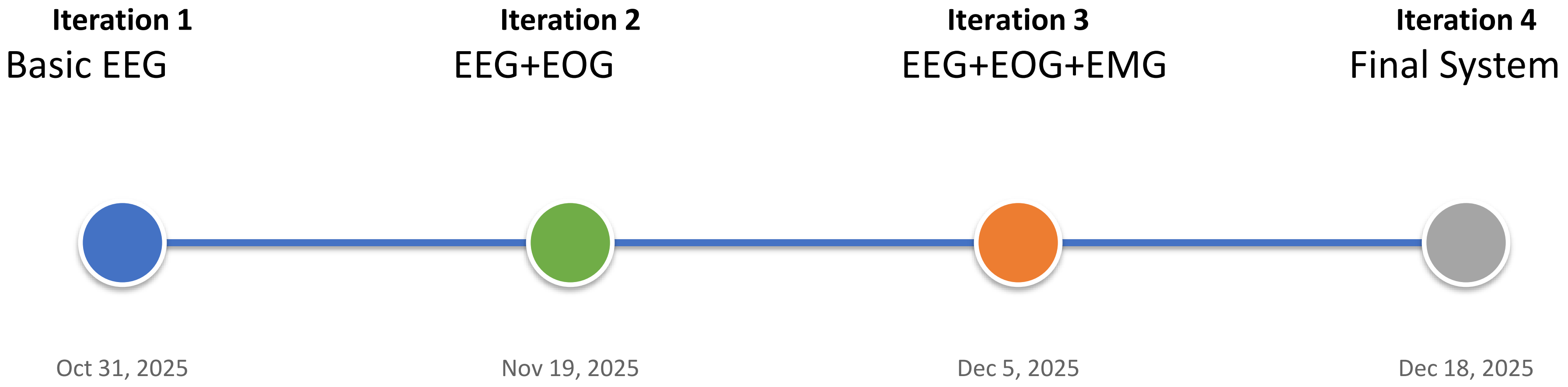
Day 16-17: Testing & Integration

Day 18: Sprint Review

Repeat for each of 4 iterations

ITERATION PLANNING:
4 SPRINTS OVER 10 WEEKS

Project Timeline - Key Milestones



Iteration 1
Basic EEG
Oct 31, 2025

Iteration 2
EEG+EOG
Nov 19, 2025

Iteration 3
EEG+EOG+EMG
Dec 5, 2025

Iteration 4
Final System
Dec 18, 2025

Iteration Timeline & Milestones

Iteration	Deadline	Signals	Features	Classifier
1	Oct 31, 2025	EEG	Time (16)	k-NN
2	Nov 19, 2025	EEG+EOG	Time+Freq (31)	SVM
3	Dec 5, 2025	EEG+EOG+EMG	Selected (30)	Random Forest
4	Dec 18, 2025	All signals	Optimized	RF-optimized

Iteration 1: Basic Pipeline (Oct 31)

- Focus: Get something working end-to-end
- Signals: EEG only (single or dual channel)
- Features: 16 time-domain features per channel
  - Mean, median, std, variance, RMS, Hjorth parameters
- Classifier: k-Nearest Neighbors (k-NN)
  - Simple, interpretable baseline
- Deliverables:
  - Complete data loading pipeline
  - Basic preprocessing (filtering, epoching)
  - Working end-to-end classification

Iteration 2: Enhanced Processing (Nov 19)

- Focus: Add EOG and frequency-domain features
- Signals: EEG + EOG (eye movement detection)
- Features: ~31 features (time + frequency domain)
  - Add frequency features: band powers, spectral entropy
  - EOG-specific: eye movement characteristics
- Classifier: Support Vector Machine (SVM)
  - Better handling of high-dimensional data
- Deliverables:
  - Multi-signal processing capability
  - Frequency-domain feature extraction

Iteration 3: Multi-Signal (Dec 5)

- Focus: Add EMG and implement feature selection
- Signals: EEG + EOG + EMG (muscle tone)
- Features: ~30 selected features (down from 50+)
  - Feature selection: statistical tests, mutual information
  - EMG features: muscle activity indicators
- Classifier: Random Forest
  - Handles non-linear relationships, provides feature importance
- Deliverables:
  - Complete multi-signal processing
  - Intelligent feature selection

Iteration 4: Full System (Dec 18)

- Focus: Optimization and finalization
- Signals: All available channels optimally combined
- Features: Optimized feature set
  - Cross-validation for robustness
- Classifier: Optimized Random Forest
  - Hyperparameter tuning, ensemble methods
- Deliverables:
  - Final competition submission
  - Complete technical report
  - Project documentation and presentation

PROJECT MANAGEMENT: USING CLICKUP

Why Project Management Tools?

- Essential for team collaboration and coordination
  - Track who is doing what and when
  - Visualize progress and identify blockers
  - Maintain accountability and transparency
- Professional skill development:
  - Industry standard practice
  - Critical for remote/distributed teams
- Required for assessment (grading checkpoints)
  - Instructor reviews your ClickUp at each milestone

ClickUp Setup (Project Manager Task)

- 1. Designate one team member as Project Manager
  - Role can rotate between iterations
- 2. Create workspace: CM2013_Sleep_Scoring_Group[X]
- 3. Add all team members with edit access
- 4. ⚠ MANDATORY: Add instructor as viewer
- 5. Create sprint folders:
  - Iteration 1: Basic EEG (Due: Oct 31, 2025)
  - Iteration 2: EEG+EOG (Due: Nov 19, 2025)
  - Iteration 3: EEG+EOG+EMG (Due: Dec 5, 2025)
  - Iteration 4: Full System (Due: Dec 18, 2025)

Task Organization in ClickUp

- Create tags for organization (free version):
  - Priority: 🔴 HIGH, 🟡 MEDIUM, 🟢 LOW
  - Signals: #EEG, #EOG, #EMG
  - Components: #preprocessing, #features, #classification
  - Status: #BLOCKED, #NEEDS-REVIEW, #BUG
- Task workflow:
  - To Do → In Progress → Review → Testing → Complete
- Each task must have:
  - Clear title: [Component] Specific action
  - Assignee, due date, priority, description

Daily Standups (via ClickUp)

- Each team member posts daily update as task comment:
- Format:
  - "Today: [what I did]
  - Tomorrow: [what I'll do]
  - Blockers: [any issues]"
- Benefits:
  - Keep everyone informed without meetings
  - Identify problems early
  - Build accountability and momentum
  - Tag PM if blocked: @mention

ClickUp Grading Checkpoints

- Instructor will review your ClickUp at:

- ✓ October 31, 2025 - Iteration 1 complete
- ✓ November 19, 2025 - Iteration 2 complete
- ✓ December 5, 2025 - Iteration 3 complete
- ✓ December 18, 2025 - Final delivery

- What is evaluated:
  - Task organization and clarity
  - Regular updates and progress
  - Team communication and collaboration
  - Problem-solving and adaptability

PYTHON JUMPSTART:
PROJECT STRUCTURE & USAGE

Python Jumpstart: What's Provided

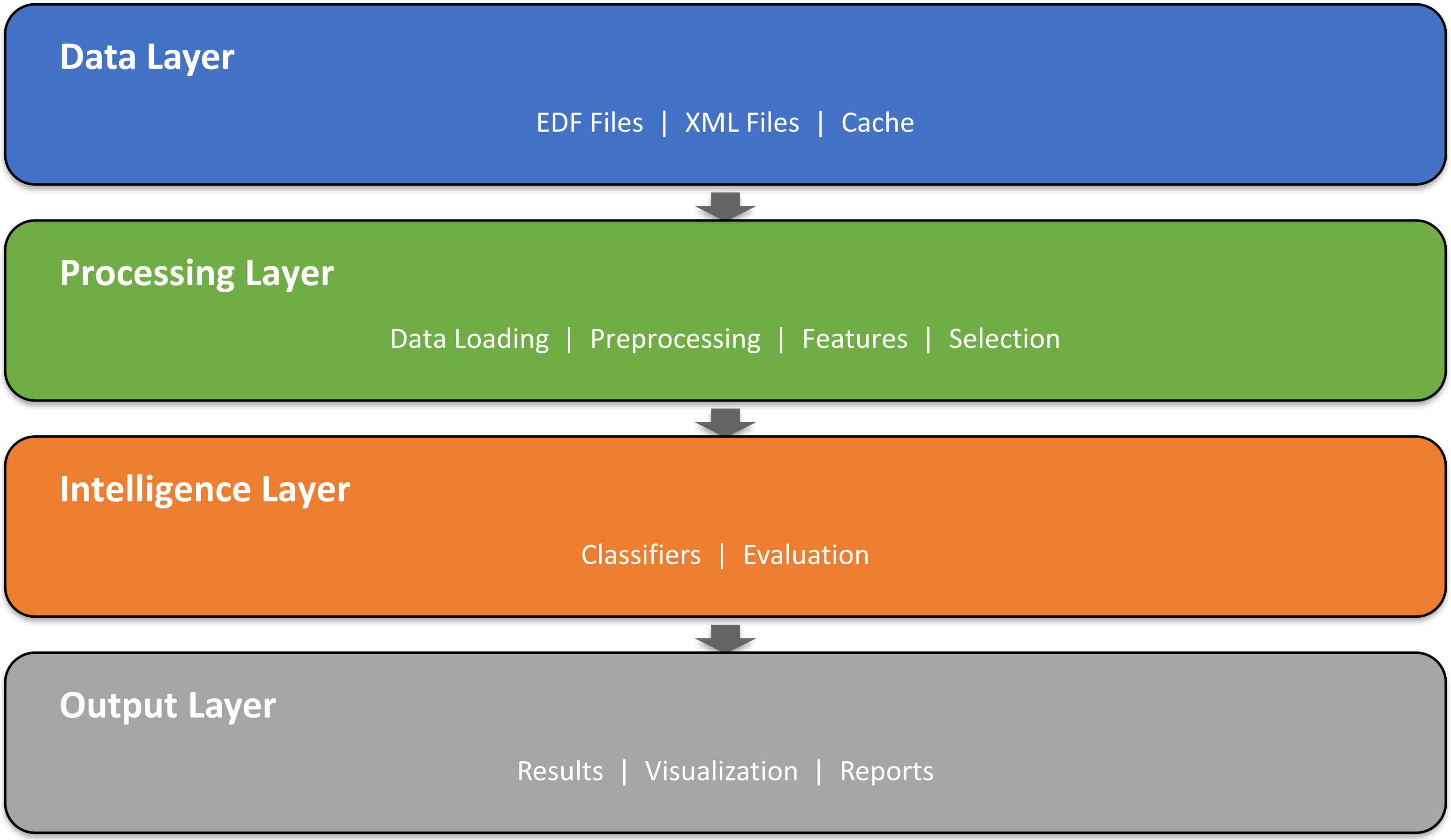
- ⚠ Structure and examples ONLY - not complete solution!

- Provided:
  - Modular project structure (src/ directory)
  - Configuration system (config.py)
  - Basic filter example (lowpass at 40Hz)
  - 3 simple features (mean, median, std)
  - Basic k-NN classifier with train/test split
  - Caching system for efficiency
  - Testing framework (pytest)
  - Google Colab notebook

Python Project Structure

- Python/
  - src/ - Core modules:
    - data_loader.py - Load EDF/XML files
    - preprocessing.py - Signal filtering
    - feature_extraction.py - Extract features
    - feature_selection.py - Select best features
    - classification.py - ML classifiers
    - visualization.py - Plot results
    - report.py - Generate reports
  - main.py - Training pipeline orchestration
  - run_inference.py - Generate predictions
  - config.py - Central configuration
  - colab_notebook.ipynb - Run in Google Colab

System Architecture - Data Flow



Data Layer

EDF Files | XML Files | Cache

Processing Layer

Data Loading | Preprocessing | Features | Selection

Intelligence Layer

Classifiers | Evaluation

Output Layer

Results | Visualization | Reports

Configuration System (config.py)

- Central control for the entire pipeline
- Key settings:
  - CURRENT_ITERATION (1-4) - Controls which features/algorithms
  - USE_CACHE (True/False) - Speed up development
  - File paths - Data directories
  - Preprocessing parameters - Filter frequencies
  - Model hyperparameters - Classifier settings
- Benefits:
  - Easy to switch between iterations
  - Consistent settings across team

Running the Training Pipeline

- 1. Setup (first time only):
  - pip install -r requirements.txt

- 2. Verify setup:
  - python -m pytest tests/ -v

- 3. Run training pipeline:
  - python main.py

- 4. Run inference (generate submission):
  - python run_inference.py
  - Creates submission.csv in data/ directory

Caching System for Efficiency

- Why caching?
  - Preprocessing is slow (minutes per file)
  - Feature extraction takes time
  - Don't repeat work during development
- How it works:
  - Results saved to cache/ directory
  - Automatically reused on next run
  - Control with USE_CACHE in config.py
- When to clear cache:
  - Changed preprocessing parameters
  - Modified feature extraction code

Google Colab Notebook

- Alternative to local development
- File: colab_notebook.ipynb

- Features:
  - No local setup required
  - Free GPU access (if needed)
  - Load from GitHub or Google Drive
  - Run complete pipeline in browser

- Usage:
  - Upload to Google Colab
  - Follow cell-by-cell instructions
  - View results inline

What You Must Implement

- Iteration 1:
  - Real EDF/XML file parsing
  - Bandpass filtering (0.5-40 Hz)
  - 13+ additional time-domain features
- Iteration 2:
  - Multi-channel processing (EEG+EOG)
  - Frequency-domain features (band powers)
  - SVM hyperparameter tuning
- Iteration 3:
  - EMG signal processing
  - Feature selection algorithms
- Iteration 4:
  - Cross-validation, optimization, final report

FINAL REPORT:
STRUCTURE & REQUIREMENTS

Technical Report (15 pages max)

- 1. Introduction (1 page)
  - Problem statement, objectives
- 2. Methods (3 pages)
  - Signal processing, features, classification
- 3. Results (4-5 pages)
  - Performance metrics, confusion matrices, analysis
- 4. Discussion (3-4 pages)
  - Interpretation, challenges, improvements
- 5. Conclusion (1 page)
  - Summary, future work
- 6. References (1 page)

Report Content Focus

- Focus on the process, not just results

- Methods section should explain:
  - WHY you chose specific approaches
  - HOW you implemented them
  - What alternatives you considered
- Results section should show:
  - Progression through iterations
  - Impact of different decisions
  - Statistical analysis of performance
- Discussion should reflect:
  - Critical thinking about your approach
  - Learning from what didn't work

DATA & FILE FORMATS

Data Organization

- data/training/ - EDF + XML files with labels
  - Use for training and validation
  - Both EDF and XML required for each recording
- data/holdout/ - EDF files only (no labels)
  - Use for final predictions/competition
- data/sample/ - Small test dataset
  - Quick testing during development

- Key signals in EDF files:
  - EEG: C3-A2, C4-A1 (125 Hz, hardware high-pass 0.15 Hz)
  - EOG: Left/Right (50 Hz, hardware high-pass 0.15 Hz)
  - EMG: (125 Hz, hardware high-pass 0.15 Hz)
  - ECG: (125 Hz), Respiration: Thor/Abdo (10 Hz), SpO2/HR (1 Hz)

Available Signals - Detailed Specifications

Signal	EDF Label	Sample Rate	Hardware Filter
EEG C3-A2	EEG (sec)	125 Hz	HP 0.15 Hz
EEG C4-A1	EEG	125 Hz	HP 0.15 Hz
EOG Left	EOG(L)	50 Hz	HP 0.15 Hz
EOG Right	EOG(R)	50 Hz	HP 0.15 Hz
EMG	EMG	125 Hz	HP 0.15 Hz
ECG	ECG	125 Hz	HP 0.15 Hz
Thorax RES	Thor RES	10 Hz	HP 0.05 Hz
Abdomen RES	Abdo RES	10 Hz	HP 0.05 Hz
SpO2	SaO2	1 Hz	-
Heart Rate	H.R.	1 Hz	-

EDF and XML File Formats

- EDF (European Data Format):
  - Standard for biosignals (EEG, EOG, EMG)
  - Multiple channels with metadata
  - Python: Use MNE library (mne.io.read_raw_edf)

- XML (Compumedics Annotation Format):
  - Sleep stage labels for each 30-second epoch
  - Stages: Wake, N1, N2, N3, REM
  - Python: Use xml.etree.ElementTree

- 📚 Reference: github.com/nsrr/edf-editor-translator/wiki

Signal Processing Considerations

- Hardware Filtering Already Applied:
  - EEG/EOG/EMG/ECG: High-pass 0.15 Hz
  - Respiration: High-pass 0.05 Hz
  - Design additional filters accordingly

- Different Sampling Rates:
  - Primary signals (EEG/EMG/ECG): 125 Hz
  - EOG signals: 50 Hz
  - Respiration/Airflow: 10 Hz
  - SpO2/Heart Rate: 1 Hz

- 30-second epochs = different sample counts per signal

TIPS FOR SUCCESS

Tips for Success

- Start early and work consistently
  - Don't wait until deadlines
- Code first, optimize later
  - Get something working, then improve
- Test often - run pipeline after changes
- Use caching to speed up development
- Document as you go - not at the end
- Communicate proactively with your team
  - Over-communication is better than under-communication
- Ask for help early when stuck
- Celebrate wins and learn from failures

Team Organization (3 members)

- Project Manager:
  - Coordination, ClickUp, integration, documentation
- Preprocessing Lead:
  - Signal cleaning, filtering, artifact removal
- Feature Engineer:
  - Feature extraction, selection, analysis

- Note: With 3 members, roles overlap!
  - Everyone should contribute to multiple areas
  - Cross-train and help each other
  - ML/classification can be shared responsibility

Resources & Documentation

- Project documentation:
  - PROJECT_GUIDE.md - Complete project guide
  - CLAUDE.md - Codebase overview
  - Python/README.md - Python jumpstart guide
- Key libraries:
  - MNE - EEG/biosignal processing
  - scikit-learn - Machine learning
  - NumPy/SciPy - Signal processing
- Support:
  - Office hours, course forum, team members

QUESTIONS?

Summary - Key Takeaways

- ✓ Iterative agile development over 10 weeks
- ✓ 4 sprints with clear milestones and deadlines
- ✓ Assessment based on process, not just accuracy
- ✓ Use ClickUp for project management and collaboration
- ✓ Python jumpstart provides structure, you implement algorithms
- ✓ Focus on learning and continuous improvement

- Start with Iteration 1: Get basic pipeline working!

- Good luck! 🚀