A 2-Phase PROJECT ASSIGNMENT for CO3117 - Machine Learning Course

COMPREHENSIVE MACHINE LEARNING MODELS: *IMPLEMENTATION AND COMPARISON*

This project aims to provide hands-on experience in implementing and comparing different machine learning models in one (preferred) or two particular use cases. The focus is on engineering practices, model implementation, and comparative analysis rather than achieving state-of-the-art performance.

I. PROJECT OBJECTIVES:

1. Implement all machine learning models covered in the syllabus
2. Compare models' characteristics and behaviors
3. Understand the strengths and limitations of each model
4. Practice engineering skills in ML projects
5. Develop model evaluation and comparison skills

II. GROUP PROJECT REQUIREMENTS

1. Project Organization:

- Form groups of 3-5 students
- Each group should designate one member as the repository owner
- Other members will fork from the main repository
- Establish clear roles and responsibilities within the team

2. GitHub Repository Structure: I suggest that each group should work on implementing and ***comparing ALL machine learning models from the syllabus using ONE unified use case and ONE dataset.*** However, you may choose ***ONE unified use case and ONE dataset*** for the 1st stage (Chapters 2-6) before the midterm, and ***ANOTHER unified use case and ANOTHER dataset*** for the 2nd stage after the midterm to the final (Chapters 6-10).

The repository should be organized to support multiple problem formulations and model implementations while maintaining clean code structure and documentation.

3. Repository Setup:

- Main group repository created by designated team member
- ***Other members fork and contribute through pull requests***
- Use branching strategy for different models/features
- Maintain comprehensive documentation

A suggested repository structure is available at https://lms.hcmut.edu.vn/mod/forum/discuss.php?d=24962#p59547.

4. **Key Requirements:**

- Clear documentation of problem statements and their variations
- Consistent interface across all model implementations
- Comprehensive testing for each component
- Detailed comparative analysis of all models
- Regular code reviews and pull requests
- Version control best practices

5. **Each team member should:**

- Actively contribute to the repository
- Document their work thoroughly
- Review other members' code
- Participate in group discussions
- Help maintain code quality

6. **The final repository should demonstrate:**

- Clean, organized code structure
- Comprehensive documentation
- Thorough testing
- Clear analysis and comparison of different models
- Professional software engineering practices

Remember to maintain a collaborative and professional approach to development, using GitHub's features effectively for project management and version control.

IV. EVALUATION RUBRIC: Each model mentioned in the syllabus will be evaluated individually; the total number of points per model is 100, for which a criterion carries 10 points, each sub-criterion is worth 2 points, the evaluation will be consistent across all models, points can be awarded partially (0, 0.5, 1, 1.5, 2). For example, the decision tree model will be evaluated as follows.

Criterion	Sub-criteria
1. Implementation Quality	Code organization; Documentation; Error handling; Modularity; Style compliance
2. Data Preprocessing	Data cleaning; Feature engineering; Handling missing values; Feature scaling; Feature selection
3. Model Implementation	Algorithm correctness; Parameter handling; Training implementation; Prediction implementation; Model persistence
4. Model Tuning	Hyperparameter selection; Cross-validation; Parameter search; Optimization strategy; Results analysis
5. Performance Analysis	Metrics implementation; Results visualization; Error analysis; Performance comparison; Statistical testing
6. Code Efficiency	Time complexity; Space complexity; Resource usage; Optimization attempts; Scalability
7. Documentation	Code comments; API documentation; Usage examples; Results interpretation; Implementation details
8. Model Analysis	Feature importance; Model behavior analysis; Limitation discussion; Comparison with theory; Use case fit analysis
9. Reproducibility	Environment setup; Data versioning; Random seed handling; Result reproducibility; Pipeline automation
10. Project Management	Git usage; Issue tracking; Project organization; Collaboration; Timeline management

V. AN EXAMPLE OF USE CASE AND DATASET SELECTION WITH ALL SPECIFIC MODEL REQUIREMENTS: ***SENTIMENT ANALYSIS***

Your team must discuss and choose your own use case and dataset (one (preferred) or two particular use cases) to study all models and algorithms in the syllabus similarly to the following sentiment analysis example.

1. Dataset: IMDb Movie Reviews Dataset Source: https://huggingface.co/datasets/imdb
2. Size: 50,000 reviews (25,000 training, 25,000 testing)
3. Classes: Positive/Negative
4. Characteristics: Long-form text, rich in features, well-labeled

MODEL-SPECIFIC REQUIREMENTS:

1. **Decision Trees:**

- Feature transformation for text data
- Handle high dimensionality
- Implement pruning strategies

2. **Neural Networks:**

- Text embeddings
- Network architecture design
- Handle text sequence length

3. **Naive Bayes:**

- Handle sparse features
- Implement smoothing
- Feature independence analysis
- Genetic Algorithms:
- Feature encoding for evolution
- Design appropriate fitness function
- Implement mutation and crossover operators
- Selection strategy for text features
- Population size and generation management

5. **Graphical Models (Bayesian Networks, HMM):**

- Model dependencies between words/phrases
- Design state transitions for text sequences
- Implement probability tables
- Handle temporal dependencies
- Structure learning from text data

6. **Support Vector Machines:**

- Implement kernel functions for text
- Handle high-dimensional text features
- Memory-efficient implementation
- Soft margin optimization
- Multi-class extension for sentiment levels

7. **Dimension Reduction (PCA/LDA):**

- Handle sparse text matrices
- Feature selection strategies
- Variance threshold selection
- Topic modeling implementation
- Dimensionality selection criteria

8. Ensemble Methods:

- Design diverse base models
- Implement bagging for text data
- Boosting implementation
- Voting/weighting strategies
- Model combination techniques

9. Discriminative Models:

- Feature-based linear classifiers
- Implement logistic regression
- Maximum entropy modeling
- CRF for sequence labeling
- Parameter optimization strategies

Each model should also include:

- **Technical Requirements:**
    - Clear input/output interfaces
    - Efficient data preprocessing pipeline
    - Model persistence functionality
    - Performance metrics implementation
    - Cross-validation support

- **Documentation Requirements:**
    - Model architecture explanation
    - Parameter descriptions
    - Usage examples
    - Performance analysis
    - Limitations discussionư