Final: Implementing Your Domain-Specific AI Assistant

Start Assignment

* **Due** Jul 31 by 11:59pm
* **Points** 100
* **Submitting** a file upload

FINAL: Implementing Your Domain-Specific AI Assistant

Overview

In this final group assignment, your team will implement key components of the domain-specific AI assistant that you designed in your midterm project plan. Your group will transform the theoretical framework into a functional prototype, focusing on data preparation, feature engineering, and basic functionality implementation.

Learning Objectives

* Apply data preprocessing techniques to real-world datasets
* Implement feature engineering strategies for your specific domain
* Create a functional AI assistant prototype that demonstrates core capabilities
* Evaluate the performance of your implementation against defined metrics
* Document the complete development process from planning to implementation
* Collaborate effectively to deliver a cohesive technical solution

Assignment Details

Deliverable Format

* A comprehensive group PDF report named according to the course convention: FN\_Submitername\_GroupName\_ITAI2377.pdf
* A Google Colab notebook containing all code, well-documented with markdown cells
* Any supplementary materials (datasets, diagrams, etc.) referenced in your report

Components to Implement

Based on your group's midterm project plan, you will now implement and document the following components:

1. Data Collection & Preprocessing (40%)

**Requirements:**

* Acquire the datasets identified in your midterm plan
* Implement a complete preprocessing pipeline that addresses:
  + Handling missing values
  + Normalizing or standardizing numerical features
  + Encoding categorical variables appropriately
  + Cleaning text data (if applicable)
  + Resizing or normalizing image data (if applicable)
* Document each preprocessing step with clear explanations of:
  + What issue it addresses
  + Why your group chose this particular approach
  + How it improves data quality
* Provide visualizations comparing raw data to processed data
* Include data quality metrics before and after preprocessing

**Deliverables:**

* Documented code for your preprocessing pipeline
* Visualizations of data distributions before and after preprocessing
* Quality metrics showing the improvement from raw to processed data
* Sample of processed data that will feed into your assistant

2. Feature Engineering & Implementation (40%)

**Requirements:**

* Create at least 3-5 domain-specific engineered features based on your midterm plan
* Perform feature importance analysis to identify the most valuable features
* Implement at least one core functionality of your assistant using:
  + The preprocessed data
  + Your engineered features
  + The technical approach your group specified (retrieval-based, few-shot, or rule-based)
* Develop a simple interface for interacting with your assistant
* Include at least 5 representative examples of your assistant in action

**Deliverables:**

* Documented code for your feature engineering process
* Analysis of feature importance with visualizations
* Implementation code for your assistant's core functionality
* Interface code (can be as simple as text input/output)
* Examples of assistant interactions with explanations

3. Evaluation & Reflection (20%)

**Requirements:**

* Test your assistant using the evaluation framework from your midterm plan
* Calculate and report performance metrics
* Analyze where your implementation succeeds and where it falls short
* Compare your actual implementation to your original plan:
  + What changed during implementation?
  + Why did your group make these changes?
  + How did these changes affect the final result?
* Reflect on the challenges your team encountered and how you addressed them
* Propose concrete next steps for improving your assistant

**Deliverables:**

* Evaluation results with metrics and analysis
* Comparative analysis of plan versus implementation
* Reflection on challenges and solutions
* Proposal for future improvements with specific steps

Implementation Requirements

Technical Constraints

* All code must run in Google Colab with reasonable resource requirements
* Use only publicly available datasets or synthetic data your group creates
* Focus on data science techniques rather than complex AI architectures
* Your assistant must implement at least one complete functional capability

Documentation Requirements

* Code must be thoroughly commented
* Include markdown cells explaining each major section
* Provide clear instructions for running your code
* Document any external dependencies or resources
* Clearly indicate which team members contributed to each section

Performance Expectations

* Your assistant should demonstrate basic functionality in your chosen domain
* Responses should be relevant and useful for the intended purpose
* Performance metrics should show improvement over baseline approaches
* The implementation should align with the goals stated in your midterm plan

Evaluation Criteria

Your group's final project will be evaluated based on:

* **Technical Implementation (50%)**: Quality of preprocessing, feature engineering, and assistant functionality
* **Documentation & Clarity (20%)**: Thoroughness of documentation, code comments, and explanations
* **Evaluation & Analysis (15%)**: Rigor of testing, quality of metrics, and depth of analysis
* **Innovation & Creativity (10%)**: Novel approaches to solving domain-specific problems
* **Team Collaboration & Contribution (5%)**: Evidence of equitable participation and effective teamwork

Submission Instructions

1. Submit your completed group PDF report via Canvas by **May 1, 2025, 11:59 PM**
2. Include a link to your group's Google Colab notebook in your report
3. Ensure your Colab notebook is set to "Anyone with the link can view"
4. Include a clear statement of contribution detailing what each team member worked on

Group Work Guidelines

* Each team member should have clear responsibilities in the implementation phase
* Regular team meetings are recommended to coordinate efforts and troubleshoot issues
* Maintain a shared project management document to track progress
* All team members should contribute to code development, documentation, or both
* If significant issues arise within your group, contact the instructor immediately

Implementation Guidance

Data Collection Strategies

* Use public datasets from repositories like Kaggle, UCI, or Hugging Face
* Consider using APIs that provide domain-specific information
* For text data, web scraping of public information may be appropriate (with proper attribution)
* Create synthetic data if needed for specialized domains

Feature Engineering Approaches

* For text data:
  + TF-IDF or word embeddings to capture semantic meaning
  + Entity extraction for domain-specific concepts
  + Sentiment analysis for opinion-based features
* For numerical data:
  + Statistical aggregations (mean, median, etc.)
  + Temporal features (time of day, day of week, etc.)
  + Interaction terms between related variables
* For categorical data:
  + Binary encodings for important categories
  + Frequency-based features
  + Target encoding (with proper validation)

Implementation Approaches

Retrieval-Augmented System

1. Create embeddings for your knowledge base documents
2. Implement vector similarity search
3. Retrieve relevant content based on user queries
4. Format responses using retrieved information

Few-Shot Learning

1. Design effective examples for your domain
2. Create a template-based system for generating responses
3. Select appropriate examples based on user input
4. Generate responses following the patterns in your examples

Rule-Based System

1. Define clear decision trees for your domain
2. Implement pattern matching for user inputs
3. Create response templates for different scenarios
4. Build a workflow that guides users through a structured process

Example Project Structure

Copy

1. Introduction - Project overview - Summary of midterm plan - Adjustments to original plan 2. Data Collection & Preprocessing - Data sources and acquisition process - Data exploration and quality assessment - Preprocessing pipeline implementation - Before/after comparisons 3. Feature Engineering - Feature creation process - Feature importance analysis - Feature selection decisions - Final feature set characteristics 4. Assistant Implementation - Technical approach details - Core functionality implementation - Interface design - Example interactions 5. Evaluation - Testing methodology - Performance metrics - Comparison to baseline - Strengths and limitations 6. Reflection & Future Work - Implementation challenges - Lessons learned - Proposed improvements - Potential applications 7. Team Contributions - Detailed breakdown of each member's contributions - Collaborative problem-solving examples 8. References 9. Appendices - Code snippets - Additional visualizations - Detailed test results

Computational Resource Management

Given the limitations of free Google Colab, your group should:

* Limit dataset sizes to what can be processed within Colab's constraints
* Consider preprocessing data in batches if working with larger datasets
* Implement checkpointing to save progress in case of session timeouts
* Use efficient algorithms that minimize computational requirements
* Test resource-intensive operations with small samples before scaling up

Resources & Support

* Use the discussion forum to ask questions about implementation challenges
* Office hours are available for technical guidance
* Reference the course materials on data preprocessing, feature engineering, and model evaluation
* Consider lightweight alternatives for complex techniques if you encounter resource limitations

Important Notes

* Start early to allow time for troubleshooting technical issues
* Coordinate among team members to avoid conflicts or duplication of effort
* Focus on getting one functionality working well rather than multiple partial implementations
* Document your process throughout, including challenges and decision points
* Keep your implementation aligned with the strengths of the data science techniques we've covered
* Remember that a well-implemented simple solution is better than a poorly implemented complex one