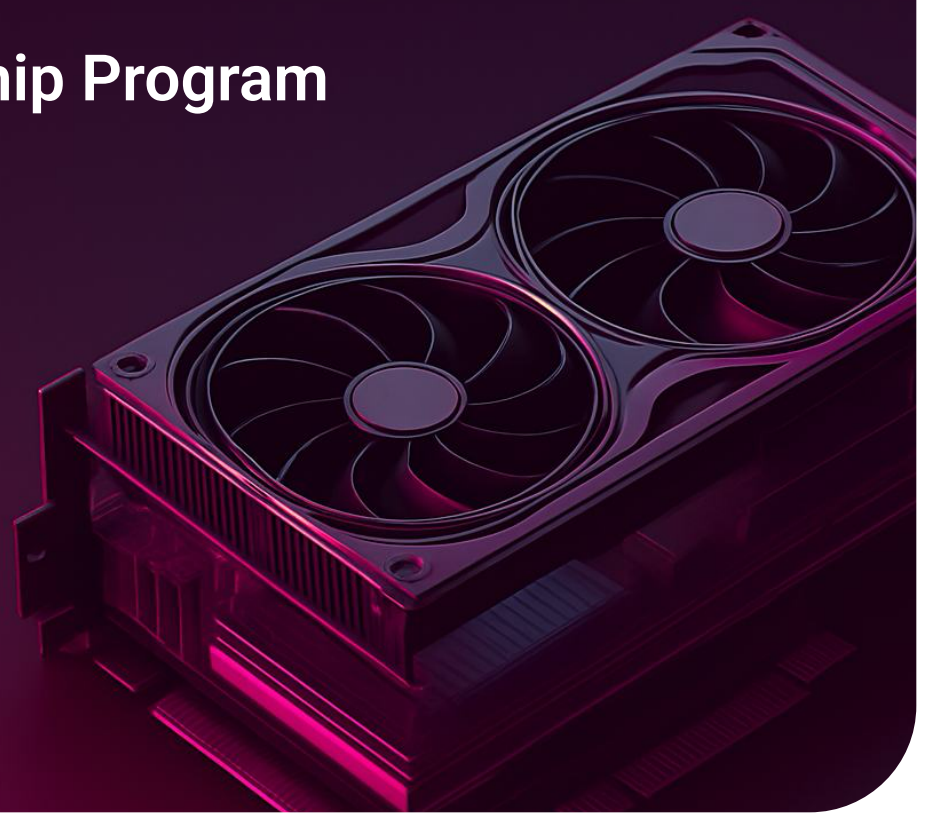
Inference.ai

ML Engineering Mentorship Program

Assignments and Projects Guide July 2025



This guide outlines the core weekly assignments and project deliverables that make up the backbone of your ML Engineering Mentorship Program experience. Each week, you'll build a real-world system that reflects what AI/ML engineers actually do in production — from deploying local LLM agents to developing voice-based assistants and fine-tuning your own models.

The projects are intentionally scoped to reinforce the engineering skills that matter in today's AI landscape: data processing, tool integration, evaluation, and iterative improvement.

You'll start by building modular components (like OCR pipelines, voice interfaces, and retrievers), and by the end of the program and demo-day, you'll combine them into a complete, portfolio presentation-ready research assistant powered by your own workflows.

Note: All code notebooks, starter templates, and datasets are provided once the program begins. What you see here is a preview of the hands-on projects you'll tackle — a glimpse into the practical, portfolio-ready work that will define your experience.

1 Agent Workflows & Local LLM Implementation

Intro: Kick off the program by learning how to design and orchestrate AI agents using real-world tools. You'll explore the MCP (Modular Command Pipeline) architecture and build workflows using a plugin ecosystem that includes GitHub, Puppeteer, Notion, and more. You'll also deploy a local LLM with Ollama, using OpenAI-compatible endpoints, and learn the fundamentals of LangChain, including LCEL syntax and how to chain components into runnable pipelines.

Assignment:

1. Implement an MCP-style agent capable of executing six plugin-driven operations: Brave Search → GitHub → Puppeteer → Filesystem → Sequential Thinking → Notion
2. (Advanced Optional) Extend your agent to automatically scrape web content and generate structured documentation in Notion.

Deliverables:

1. Functional agent script demonstrating plugin orchestration
2. Optional advanced web-to-Notion documentation pipeline

2 Data Extraction & Corpus Construction

Intro: This week is all about transforming unstructured data into clean, usable corpora. You'll optimize OCR pipelines using Tesseract (including layout tuning and multilingual settings), extract text from PDFs and HTML, and convert them into embedding-ready formats. You'll also learn how to assess and improve dataset quality through deduplication (via MinHash), PII removal, and metadata validation.

Project:

1. arXiv Paper Processing:
    a. Scrape 200 NLP research papers from arXiv (cs.CL category) using Trafilatura
    b. Run dual extraction pipelines:
        i. Clean HTML to arxiv_clean.json
        ii. OCR-extracted PDF text to pdf_ocr/ folder
- Submit a compressed, cleaned dataset (≤1MB) and an OCR quality report
2. Multimodal Transcription:
    a. Download 10 NLP-related YouTube talks using yt-dlp
    b. Transcribe and timestamp the audio into a structured talks_transcripts.jsonl file
3. Corpus Refinement:
    a. Merge outputs from previous steps into a single dataset
    b. Apply a cleaning and validation pipeline to ensure data quality and consistency

Deliverables:

1. Combined processing scripts
2. Final cleaned dataset
3. Validation report with quality metrics

3 Voice Agent Development

Intro: This week, you'll build a voice-enabled AI agent capable of real-time conversation. You'll design the full stack: automatic speech recognition (ASR), a local LLM-based dialogue engine, and text-to-speech (TTS) synthesis. You'll also learn how to manage conversational state, maintain context across multiple turns, and optimize for low latency.

Project:

1. Set up a FastAPI server to handle requests
2. Integrate ASR using whisper.cpp or Google Speech-to-Text
3. Generate responses using the Llama 3 8B model (via HuggingFace textgeneration pipeline)
4. Synthesize responses with Cozyvoice or other TTS system
5. Submit an end-to-end demo capable of 5+ continuous back-and-forth turns

Deliverables:

1. Code repository
2. 2-minute demo video showing full voice interaction loop

4 RAG for arXiv Paper Processing

Intro: Dive into RAG systems by designing a document retriever and reader pipeline for academic papers. You'll implement chunking strategies, handle LaTeX-heavy formats, and extract relevant metadata. This module focuses on understanding document structure and deploying performant retrieval pipelines.

Project:

1. Scrape 50 recent arXiv papers from the cs.CL category (including PDF and metadata)
2. Use PyMuPDF to extract and section text (≤512 tokens per chunk)
3. Build a FAISS index using all-MiniLM-L6-v2 sentence embeddings
4. Implement a simple API to query the index and return the top 3 most relevant passages

Deliverables:

1. Jupyter Notebook containing code and retrieval logic
2. A test report with 5 example queries and output passages

5 Embedding Database Optimization

Intro: This week focuses on improving the performance and versatility of your retrieval system. You'll enhance embedding quality through normalization and optional dimensionality reduction, and explore hybrid search—combining keyword and vector-based retrieval. You'll also design a lightweight but powerful database schema to support multimodal querying.

Project:

1. Upgrade your RAG system to a multimodal database:
    a. Encode text chunks with text-embedding-3-small
    b. Optionally generate image embeddings with CLIP for figures

1. Implement hybrid retrieval that blends BM25 (keyword) and vector search
2. Evaluate search quality improvement over pure vector search

Deliverables:

1. Database schema diagram
2. Retrieval performance comparison showing ≥10% recall gain

6 LLM Function Calling for Agents

Intro: Learn how to extend your voice agent with tool-augmented reasoning. This module introduces OpenAI-style function calling and dynamic tool selection based on LLM intent parsing. You'll build logic that allows your agent to trigger different tools in response to natural queries.

Project:

1. Integrate two new tools into your voice agent:
    a. Search_arxiv(query) — taps into your Week 5 database
    b. Calculate(expression) — evaluates math expressions
2. Use Llama 3's function-calling abilities to detect user intents and auto-route to the correct tools
3. Build an execution chain that returns voice responses

Deliverables:

1. Extended voice agent codebase
2. Logs of at least 3 function call examples

7 Fine-tuning with Synthetic Data

Intro: This week, you'll explore how to generate high-quality synthetic training data and fine-tune open-source models for specialized use cases. You'll learn prompt engineering techniques using GPT-4 (including role-play and counterfactual strategies), structure data for instruction tuning, and run efficient fine-tuning workflows using QLoRA and the PEFT library.

Assignment:

1. Select 100 arXiv papers and generate 5 question-answer pairs per abstract using GPT-4 Turbo.

1. Include "misinterpretation" examples and corrections to simulate real-world user misunderstandings.
2. Fine-tune the Llama 3 8B model using Unsloth on Colab with QLoRA (4-bit) for resource efficiency.
3. Evaluate your model's performance on academic Q&A using human-labeled accuracy comparisons (pre/post tuning).

Deliverables:

1. Synthetic dataset in JSONL format
2. Fine-tuning script
3. Evaluation report comparing QA accuracy

8 Summarization with Expert Models

Intro: Explore how to generate and evaluate high-quality technical summaries using state-of-the-art models. This week covers multimodal inputs, advanced summarization models, and preference-based reward training. You'll also experiment with model routing based on input types.

Project:

1. Build a summarization system:
   a. Use DeepSeek-VL to process text + figures
   b. Generate summaries using Mixtral 8x22B
2. Collect human preferences (Summary A vs. B) and fine-tune a DeBERTa-v3 reward model
3. Output quality-scored summaries across 10 papers

Deliverables:

1. Summarization pipeline code
2. Trained reward model
3. Summary results with quality ratings

9&10 Integrated Research Assistant System

Intro: In this final stretch, you'll combine everything you've built into a complete voice-powered research assistant. It will be capable of live conversation, document retrieval, summarization, and automated Notion sync—creating a seamless research interface.

Project:

1. Build a full pipeline:
    a. Take voice input → identify relevant paper fragments → generate spoken answer
    b. Automatically summarize conversation + paper content after session ends
    c. Sync final output to Notion via your Week 1 integration
2. Use a realistic demo case (e.g. new paper → voice Q&A → synced summary)

Deliverables:

1. Fully runnable local assistant
2. System demo video (60 seconds)