

AI Engineering & Machine Learning

Duration	Live classes	Schedule	Batch size
12 weeks	120 hours	Mon to Fri, 2 hrs/day	10 students max

Tools you will use

 Python + scikit-learn
  PyTorch + Transformers
  LangChain + LangGraph

 RAG + Pinecone
  Agents + Mem0 + MCP
  FastAPI + Docker

10 students a batch. Exceptional mentors who guide you personally, and a **syllabus built from real job posts**, cuz we never skip the "hard parts" 😊

The class

10 per batch

Ten students. Never more.

A hard cap, not a target. Your mentor knows your name and your work.

The mentor

4–5 yrs in the job

Someone who does the job.

Working professionals, not full-time teachers. They read your work line by line.

The syllabus

1/3 is fundamentals

The hard parts, not just the basics.

Teaches everything the job needs and never skips the hard parts most courses skip, built from real job descriptions. Check the syllabus before you enroll.

Built into every course

Gate assessment

A short test before Day 1 finds your strengths and pace, then sets your plan.

A real, live project

Every course ends with a deployed project you can show employers.

Revision every 5th class

Every fifth class goes back over the week, so nothing piles up.

1:1 mock interview

A real interview practice with your mentor in the final week.

Certification exam

Pass the exam at the end of the course to earn your certificate.

CV and LinkedIn help

We help you rewrite your CV and LinkedIn before you apply.

AI Engineering & Machine Learning

Why This Course

This course trains you to become a junior AI engineer who builds real AI apps that people use. Companies in Nepal and remote teams abroad are hiring for these roles.

- **Employers want these skills.** You learn embeddings (turning text into numbers that hold meaning) and RAG (letting AI answer from your own documents). You also learn agents (AI that uses tools to finish a task) and how to put them online.
- **You start with the basics.** Weeks 1 to 5 cover Python from zero, data tools, machine learning (ML) and deep learning.
- **Weeks 6 to 12 focus on LLMs.** LLMs are large language models like ChatGPT. You build RAG and agents, test their answers, and fine-tune models with LoRA and QLoRA (cheap ways to retrain a model). You also run models on your own laptop with Ollama.
- **Your final project goes live.** It runs online, logs every step and tracks what each request costs. It is not just a practice notebook.
- **You use AI tools like Cursor and Claude Artifacts from Week 1.**
- **We do not promise a salary.** Your pay depends on your skills and your portfolio.

Who Is This Course For

- **Complete beginners** who want a real AI engineering career and will put in the work on the basics.
- **People who know some Python** and want to go past tutorials to build AI apps that run online.
- **Data scientists** who want to move from analysis notebooks to building and launching real systems.
- **Full stack or backend developers** who want to specialise in AI.
- **ML learners** who want to learn how AI is built and run properly at real companies.
- **Builders** who want to make real AI products instead of demos.

Course Snapshot

PARAMETER	DETAILS
Course Code	TR-10

PARAMETER	DETAILS
Duration	12 Weeks (3 Months)
Schedule	Monday to Friday, 2 Hours/Day
Total Hours	120 Hours
Batch Size	Maximum 10 Students
Course Fee	NPR 41,150
Level	Beginner-friendly, taught from the fundamentals → AI Engineering Apprentice
Outcome	AI Engineering Apprentice / Junior LLM-App Developer, with a realistic 3–6 month runway into junior AI engineer roles

Prerequisites

- You need no coding experience. We teach Python from zero in Weeks 1 and 2.
- You need a laptop with at least 8GB RAM. 16GB is better. You do not need a graphics card (GPU).
- You use a computer every day and know how files and folders work.
- You can read and write basic English.
- You take the Saarathi Gate Assessment before Day 1. It only checks your starting level, so there is no pass or fail.

You do not need to install anything first. In the first class we set up VS Code, Python, and your OpenAI, Anthropic and HuggingFace accounts.

Optional warm-up. This is not required, because Week 1 starts from zero:

- Watch chapter 1 of the 3Blue1Brown Neural Networks series on YouTube.
- Do the first 3 sections of freeCodeCamp's Scientific Computing with Python.
- Pick one AI product you use and guess how it works. We come back to it in Week 6.

Assessments at a glance

- **Gate Assessment before Day 1.** It has no pass or fail.
- **Certification Exam.** It lasts 2 hours and has about 30 questions of different types. You need 70% to pass and get your certificate.

- **Exam rules.** The exam runs in fullscreen. Switching tabs and pasting are recorded, and three of these end your exam. Answer every question.
- **Projects.** Your mentor gives you 2 projects during the course. They need to be approved before you get your certificate and before the Career Launchpad opens.
- **Recommendation letters are not automatic.** Mentors give them only to learners who pass, and they are never guaranteed.

Your Learning Week

DAY	ACTIVITY
Mon–Fri	2-hour live class with your mentor, where you build real projects
Mon–Fri	2 hours of study at home for projects, practice and fixing errors
Sunday	Open classroom if you want to build, or rest
Day 5 (every week)	The last hour of class is for revision, a review of how your project is built, and clearing doubts

Week-by-Week Curriculum

Phase 1 · Python, Machine Learning & Deep Learning Foundations (Weeks 1–5, 50 Hours)

WEEK	FOCUS AREA	TOPICS
Week 1	Python Foundations I: Core Language & Tooling	– Workspace setup: VS Code (your code editor), the terminal and Git basics (version control) – Values, variables, data types (text, numbers, true/false), operators and truthiness – Strings and f-strings (the format behind every prompt later) – Control flow: conditionals (if/elif/else) – Loops: for/while, range, break/continue, the accumulate pattern – Collections: lists and tuples (ordered, zero-based indexing, the #1 beginner bug) plus dictionaries and sets (fast labelled lookup, no duplicates) – Cursor autocomplete from the start (reading and judging the code comes in Week 2)
Week 2	Python Foundations II:	– Functions and comprehensions: inputs, outputs, defaults, <code>*args</code> / <code>**kwargs</code> , scope (taught after a week of concrete

WEEK	FOCUS AREA	TOPICS
	Engineering for AI	code so the abstraction clicks), then comprehensions read as sugar over a loop you already understand - Type hints: labelling values to catch mistakes early (a contract for Pydantic and your editor) - Error handling: try/except, a custom exception, never a bare except - File and JSON I/O with pathlib and context managers - Modules, imports and the main-guard, then object-oriented programming, lean: classes, methods, one inheritance example (class vs dict) - numpy and pandas first look: the on-ramp into the data toolkit - Disciplined AI pair-coding (Cursor, Copilot): verify every line; async first look, awareness only, drilled in Week 6
Week 3	Data Toolkit & Supervised Machine Learning	- Data toolkit finished plus SQL: numpy, pandas DataFrames (filter, groupby, merge), venv/pip, Jupyter, then SELECT/WHERE/JOIN/GROUP BY (SQLite/Postgres) read straight into pandas - HuggingFace datasets and reproducible train/val/test splits - First models: regression and classification: linear regression (predict a number) and logistic regression (predict yes/no), judged only on unseen data, scored with RMSE, MAE, R², confusion matrix, precision, recall, F1, ROC-AUC (and why accuracy lies on imbalanced data) - Overfitting vs underfitting (memorising vs never studying), one cross-validation pass - Data leakage: one visceral cheating-model story, cured with a scikit-learn Pipeline - Why this powers your LLM work: the same metrics and no-leak habit become your RAG and LLM-as-judge evals - Copilot and Claude Artifacts for fast, verified ML exploration
Week 4	Tree Models, Clustering & Feature Engineering	- Decision trees and random forests: feature importance and the few hyperparameters that matter (XGBoost/LightGBM awareness) - k-means clustering demo: finding natural groups (elbow, silhouette, scale first) - Feature engineering in scikit-learn Pipelines with ColumnTransformer (ratios, bins, date parts), leakage-safe - The features→chunking bridge: the same judgement powers your RAG retrieval choices

WEEK	FOCUS AREA	TOPICS
		- Bake-off: tuned random forest vs the Week 3 logistic baseline, interpretability vs accuracy - When a simple model beats an LLM (cost, latency, reliability), an employer-tested judgement - Copilot and Claude Artifacts for verified tree/clustering/feature code
Week 5	Deep Learning & Transformer Intuition	- Neural network intuition: neurons, layers, activation functions (pictures first, no math) - PyTorch: tensors, autograd, a guided line-by-line training loop (SGD vs Adam) - Attention scaffolded: "each word decides who to listen to", then queries/keys/values (diagram only) - Transformer architecture, the design behind ChatGPT and Claude: encoder/decoder, tokenisation and embeddings at intuition level - HuggingFace Transformers: pipeline , AutoModel , AutoTokenizer - Fine-tune walkthrough as a Week-11 preview - CNNs and RNNs awareness; research awareness (backprop math, arXiv)

Phase 2 · LLMs, RAG & Vector Databases (Weeks 6–8, 30 Hours)

WEEK	FOCUS AREA	TOPICS
Week 6	Prompt Engineering & LLM APIs	- How LLMs are made: pretraining, RLHF, alignment (overview) - Prompting: zero-shot, few-shot, chain-of-thought and reasoning models (ChatGPT Thinking, Claude Extended Thinking, Gemini thinking, DeepSeek R1 / V-series) - Checking AI output by hand: a small test set of prompts and expected answers, marked right or wrong yourself before trusting a response in code - Context engineering: curating every token the model sees (system prompt, tools, retrieved data, memory, history) - Reliable outputs: JSON mode, OpenAI Structured Outputs strict mode, Anthropic forced tool use, function calling - OpenAI & Anthropic SDKs: chat completions, Messages API, streaming, message roles, system prompts, prompt caching (cache_control) - Provider choice and cost: OpenAI (ChatGPT) vs Claude vs Google Gemini vs open-weight models, with Groq for fast

WEEK	FOCUS AREA	TOPICS
		inference as a fallback tier, model-routing, context-window length and token-cost math - AI Gateway awareness (OpenRouter, Portkey, Helicone): routing, retries, unified spend - Streaming into a simple chat interface with async/await (the Week-2 async intro becomes hands-on)
Week 7	Embeddings & Vector Databases	- Embeddings: text as numbers that capture meaning; cosine similarity - Embedding models: OpenAI text-embedding-3 , Voyage voyage-4 , Cohere Embed 4 (embed-v4.0 , dense+sparse, multimodal), Google Gemini Embedding, BAAI bge-m3 (multilingual), open-weight Qwen3-Embedding , Matryoshka, sentence-transformers, chosen against a labelled retrieval set you build yourself, not an AI's own claim about which is best - Vector databases: Pinecone setup, indexing, metadata filtering - ChromaDB locally, pgvector in Postgres (Qdrant, Weaviate, Milvus awareness) - Chunking strategies with reasoning: fixed-size + overlap, semantic, parent-document - Hybrid search: dense + sparse (BM25), with reranking (Cohere Rerank, bge-reranker), scored by recall@k, MRR and nDCG - Multi-modal inputs: images to OpenAI vision (ChatGPT), Claude Vision and Gemini vision (base64 and URL) - Realtime / voice awareness (OpenAI Realtime, ElevenLabs)
Week 8	RAG Pipeline Development	- RAG (Retrieval-Augmented Generation): look up your documents first, answer from what was found - LangChain chains and LCEL (LangChain 1.0; LangGraph is the stateful path, AgentExecutor now legacy) - Basic RAG pipeline end-to-end: load, chunk, embed, store, retrieve, rerank, answer with citations - Better retrieval: multi-query and parent-document retrieval, scored with recall@k / MRR / nDCG - LlamaIndex as the alternative - RAG evaluation with RAGAS: are answers faithful to the sources (ship the eval set before the bot) - Beyond basic RAG (awareness): agentic RAG (agent decides when/what to retrieve) and GraphRAG (Microsoft GraphRAG, LightRAG, LazyGraphRAG, Neo4j) for multi-hop, cost-routed vs vector RAG - OpenAI Responses + Conversations APIs vs custom RAG trade-

WEEK	FOCUS AREA	TOPICS
		offs (Assistants API removed 26 August 2026) - Production RAG: latency, caching, cost; capstone scoped with your mentor this week

Phase 3 · Agents, Evaluation & Model Adaptation (Weeks 9–11, 30 Hours)

WEEK	FOCUS AREA	TOPICS
Week 9	Tool-Using Agents & Harnesses	- Agent harness: the loop-and-plumbing layer around the model (the model decides, the harness does); hand-build a minimal harness (gather → act → verify → repeat) over <code>read_file</code> / <code>write_file</code> / <code>run_shell</code>, then the Claude Agent SDK as that same harness as a library; harness vs framework vs runtime - AI agents: LLMs that use tools and act in steps; the ReAct pattern, and when NOT to use an agent (prefer a deterministic pipeline unless control flow is truly open-ended) - Tool schemas and LangChain's <code>create_agent</code> (the LangChain 1.x standard on the LangGraph engine; AgentExecutor and <code>create_react_agent</code> now legacy/maintenance) - Agent memory across chats: Mem0 hands-on (Zep + Graphiti, Letta/MemGPT, LangMem awareness) - Agent context engineering: compaction, note-taking, sub-agent isolation, just-in-time retrieval, tool-result clearing - Deep Agents (<code>deepagents</code>) hands-on: the four pillars (planning todos, sub-agents, virtual file system, detailed system prompt) for long-horizon work, built on LangGraph - LangSmith tracing and observability - MCP (Model Context Protocol), now under the Linux Foundation) hands-on: build a one-tool server, plug into Claude Desktop / Cursor; the protocol stack MCP (<code>agenttools</code>) / A2A (<code>agentagent</code>) / AG-UI (<code>agentuser</code>) awareness - Agent runtimes: LangGraph vs OpenAI Agents SDK vs Claude Agent SDK (CrewAI, AutoGen, Computer Use / browser agents awareness) - Agent cost & latency budgets: a max step count, token/cost ceiling and latency target set before scaling up tool use, measured per completed task, not per model call - Hugging Face Agents Course in self-study; certificate milestone due end of Week 10

WEEK	FOCUS AREA	TOPICS
Week 10	LangGraph Orchestration, Evaluation & Safety	- LangGraph orchestration: StateGraph (nodes, edges, typed state), branching, supervisor pattern, human-in-the-loop, checkpointing - LLM eval pipelines and LLM-as-judge (a strong model grades your outputs against a disclosed rubric), with cost and p95 latency tracked alongside correctness so a regression fails the gate too - Agent evaluation: success rate, tool-call correctness, trajectory quality - Prompt injection prevention, the OWASP LLM Top 10 (esp. LLM01 Prompt Injection, LLM06 Excessive Agency) and the new OWASP Agentic Top 10 (2026) (goal hijacking, tool misuse, memory poisoning): input guards, output filters (Guardrails AI / NeMo Guardrails awareness) - Eval + observability stack: RAGAS + Promptfoo + DeepEval; LangSmith + Langfuse over OpenTelemetry GenAI; trace vs span (every LLM call traced) - Safety writeup: bias, model cards, red teaming - Capstone eval + safety layer wired in; deploy gated on eval scores
Week 11	Fine-Tuning & Local Inference	- Fine-tuning vs prompt engineering vs RAG: the decision ladder, and the more useful skill of knowing when NOT to fine-tune; dataset prep, synthetic data via distillation (licence implications) - Graded path: hosted OpenAI fine-tuning API (previewed in Week 5), measured against the base model - Graded path: LoRA / QLoRA / HuggingFace PEFT adapter on a free Colab T4, measured against the base model (two shipped fine-tuned artifacts per student) - Graded path: Ollama local serving + Q4 GGUF quantisation ("shrink the model to fit your RAM"), Q4/Q5/Q8 trade-offs - Fully local RAG pipeline (optional stretch) - Self-host awareness: vLLM, SGLang (Hugging Face TGI now legacy), with hosted-API cost-per-request compared against a self-hosted quantised model's cost-per-request on the same task; reinforcement learning awareness (RLHF, DPO)

Phase 4 · Productionize, Capstone & Career (Week 12, 10 Hours)

WEEK	FOCUS AREA	TOPICS
Week 12	Productionize, Ship the Capstone & Career	- Ship the capstone as a live service: FastAPI (with Pydantic), async endpoints, healthcheck/version routes - Multi-stage Docker build and deploy to a public URL (small cloud host, or a guided GPU-cloud deploy to RunPod / Lambda Labs as time allows) - Per-request tracing (latency, tokens, cost), alerts on error rate and drift, p50/p95/p99, and a real cost-per-request figure in the README - As time allows: MLflow experiment tracking, CI evals (RAGAS / Promptfoo / LLM-as-judge on PRs), an AI Gateway cost dashboard with semantic / TTL caching - Capstone finalisation: failure modes, architecture diagram, eval scorecard, README + demo video - AI-engineer resume and LinkedIn (quantified bullets, never an unmeasured "100% accuracy"), plus async-English written-update practice for remote teams - ML system design interview practice and mock interviews (coding, ML/LLM concepts, capstone defence), AI tools allowed and your screen shared like a real onsite, explaining every choice and failure mode out loud, plus a timed take-home drill before the mock loop, with a timed DSA drill closing the light Weeks 9–12 strand and feeding the mock coding round - Demo day and career planning

Skills You Will Gain

CATEGORY	TECHNOLOGIES
Language	Python (from basics to async, type hints, OOP)
Classical ML	scikit-learn (regression, classification, decision trees, random forests, k-means), evaluation metrics, cross-validation, Pipelines, NumPy, pandas, SQL (SELECT/JOIN/GROUP BY for data prep)
Deep Learning	PyTorch, HuggingFace Transformers
LLM APIs	OpenAI (incl. Structured Outputs strict mode), Anthropic Claude (Messages API + caching, Extended Thinking), Google Gemini, Groq, reasoning models

CATEGORY	TECHNOLOGIES
	(ChatGPT Thinking, Gemini thinking, DeepSeek R1 / V-series), streaming, token-cost math + model routing
AI Gateway	OpenRouter, Portkey, Helicone for multi-provider routing and spend control
Multi-modal AI	ChatGPT, Claude Vision, Gemini vision, image inputs, Realtime / voice API awareness
RAG	LangChain (LCEL), LlamaIndex, Pinecone, ChromaDB, pgvector (Qdrant/Weaviate/Milvus awareness), modern embeddings (text-embedding-3, voyage-4, Cohere Embed 4, Gemini Embedding, bge-m3, Qwen3-Embedding, Matryoshka), reranking (Cohere Rerank, bge-reranker), retrieval metrics (recall@k, MRR, nDCG), RAGAS, agentic RAG + GraphRAG/LightRAG awareness
Context Engineering	Context-window curation, compaction, note-taking memory, sub-agent isolation, just-in-time retrieval, tool-result clearing
Agents	Agent harness (hand-built minimal loop; harness vs framework vs runtime), LangChain <code>create_agent</code> on the LangGraph engine, LangSmith, memory layers (Mem0 hands-on; Zep + Graphiti, Letta/MemGPT, LangMem awareness), Deep Agents (<code>deepagents</code>) hands-on, MCP server build + consume, OpenAI Agents SDK / Claude Agent SDK (compared), A2A / AG-UI awareness
Eval & Observability	LLM-as-judge, RAGAS, DeepEval, agent evaluation, prompt-injection + guardrails; Promptfoo / Braintrust / OpenAI Evals + LangSmith / Langfuse / Helicone / Arize Phoenix over OpenTelemetry GenAI
Fine-Tuning	hosted OpenAI fine-tune (graded), LoRA / QLoRA / HuggingFace PEFT on a Colab T4 (graded), synthetic data generation
Local & Self-Host Inference	Ollama, llama.cpp, quantisation; vLLM and SGLang awareness for production multi-tenant serving (HuggingFace TGI now legacy)
ML	MLflow, GitHub Actions (eval on PR), FastAPI, Docker, GPU cloud, cost monitoring, AI Gateway-based spend dashboards
AI Productivity	Cursor, GitHub Copilot, Claude Artifacts

Topic Depth and Awareness

Depth: practised until you can apply it. **Awareness:** introduced so you can read and compare it later.

FOCUS AREA	LEARNING FOCUS
Python fundamentals (OOP, type hints, async, file I/O)	Depth
Python for AI (numpy, pandas, HuggingFace datasets)	Depth
SQL for data prep (SELECT/JOIN/GROUP BY, SQLite/Postgres into pandas)	Depth
Classical ML (sklearn: regression, classification, trees/forests, k-means, evaluation, cross-validation, Pipelines)	Depth
Deep learning intuition (PyTorch, transformer architecture)	Depth
LLM APIs, OpenAI, Anthropic Claude, Google Gemini, Groq, reasoning models (ChatGPT Thinking, Extended Thinking, Gemini thinking, DeepSeek R1 / V-series)	Depth
Context engineering (window curation, compaction, note-taking, sub-agent isolation, just-in-time retrieval, tool-result clearing)	Depth
Structured Outputs strict mode + AI Gateway (OpenRouter, Portkey)	Depth
Multi-modal AI (vision inputs)	Depth
Modern embeddings + reranking (Cohere Rerank, bge-reranker)	Depth
RAG (embeddings, Pinecone, ChromaDB, RAGAS, hybrid retrieval)	Depth
Agents (hand-built harness, LangChain <code>create_agent</code> on LangGraph, Deep Agents <code>deepagents</code> , Mem0 memory, LangSmith tracing)	Depth
MCP (Model Context Protocol) server build + consume, and runtime comparison (LangGraph vs OpenAI Agents SDK vs Claude Agent SDK)	Depth
LLM evaluation (LLM-as-judge, RAGAS, agent evaluation, prompt-injection guard, OWASP LLM + Agentic Top 10)	Depth
Cost optimisation (token counting, caching, batch APIs)	Depth

FOCUS AREA	LEARNING FOCUS
Fine-tuning (hosted OpenAI fine-tune + LoRA/QLoRA PEFT on Colab T4, both graded; dataset prep, synthetic data generation)	Depth
Local inference (Ollama, quantisation / Q4 GGUF serving)	Depth
ML (MLflow, GitHub Actions evals on PR, FastAPI, Docker, cost monitoring)	Depth
Memory-layer landscape (Zep + Graphiti temporal KG, Letta / MemGPT, LangMem)	Awareness
Agent protocols (A2A agent-to-agent, AG-UI) and CrewAI / AutoGen runtimes	Awareness
Eval-framework + observability survey (Promptfoo / DeepEval / Braintrust / OpenAI Evals; LangSmith / Langfuse / Helicone / Arize Phoenix over OpenTelemetry GenAI)	Awareness
Guardrail libraries (Guardrails AI, NeMo Guardrails)	Awareness
Agentic RAG and GraphRAG / LightRAG / knowledge-graph augmented retrieval	Awareness
Vector-DB landscape (Qdrant, Weaviate, Milvus)	Awareness
Realtime / voice API (OpenAI Realtime, ElevenLabs)	Awareness
Computer Use / browser agents (Anthropic CUA, Browserbase)	Awareness
Production self-host inference (vLLM, SGLang; TGI legacy)	Awareness
CNNs, RNNs	Awareness
Deep learning from scratch / research	Awareness
Reinforcement learning	Awareness

Project Pool

In Week 8, you pick **one** final project (capstone) with your mentor. The list changes from batch to batch:

#	PROJECT	STACK FOCUS
1	Production RAG over private docs (evaluated with RAGAS)	Embeddings, Pinecone, retrieval evaluation
2	Multi-modal AI assistant (text + image inputs via Claude Vision / OpenAI vision (ChatGPT) / Gemini vision)	Multi-modal APIs, prompt engineering
3	Domain-tuned agent with tool use and LangGraph workflows	LangGraph, LangSmith, eval
4	Cost-optimised LLM service with caching + batch processing	Token optimisation, Anthropic/OpenAI Batch APIs
5	Fine-tuned local model deployed with Ollama + FastAPI	LoRA, quantisation, serving
6	ML pipeline with cost monitoring + rollback	MLflow, FastAPI, Docker, spend tracking

Career Paths & Trajectory

ROLE	TIMELINE	FOCUS
AI Engineering Apprentice / Junior LLM-App Developer	First 3–6 months	Build LLM, RAG and agent features with a senior checking your work, and use your portfolio to land a first junior job
Junior AI Engineer / Applied ML Engineer	0–2 yrs	Build LLM features, RAG, agents
AI Engineer / ML Engineer	2–4 yrs	Own AI systems in production
Senior AI Engineer / ML Platform Engineer	4–7 yrs	Architecture, platform, cost/reliability
AI Solutions Architect / Applied AI Lead	7+ yrs	Strategy, org-wide impact

FAQ

What does AI Engineering & Machine Learning cost and how do I pay?

NPR 41,150. Pay by cash at our Old Baneshwor campus, bank transfer, or Fonepay. Email info@saarathiacademy.com.np for the payment plan.

Do I need prior experience?

No. The Gate Assessment before Day 1 shows your starting level. If you need more basics first, we suggest a short foundation course.

How long is the course?

12 weeks (3 months). You build your final project during the course. In the final week you do a 1:1 mock job interview.

What tools will I use?

You use **PyTorch, HuggingFace, LangChain, FastAPI, Pinecone and Docker**. Each part of the course ends with a project that runs online and goes in your portfolio.

What roles does this prepare me for?

See Career Paths above. We do not promise jobs.

- We help you build your portfolio, practise interviews, and improve your CV and LinkedIn.
- We refer you to companies in our hiring network.

How does the batch and mentorship work?

- Each batch has at most 10 students, so you get direct time with your mentor and feedback on your projects.
- Classes are in person at Old Baneshwor. Recordings and extra help hours are online.
- The Sunday Open Classroom is free for enrolled students.

Enrollment

- **Next Batch:** see the live course page
 - **Location:** Old Baneshwor Chowk, Kathmandu, Nepal
 - **Mode:** Online + Offline
 - **Contact:** 9744442469, 9761095364
 - Limited to 10 seats per batch
-

AI Engineering & Machine Learning — full details, fees and next batch dates:
saarathiacademy.com.np/ai-engineering-and-machine-learning-course-in-nepal

Version 2.0 · Updated 11 Ashwin 2083

