TR-10

AI Engineering & Machine Learning

COURSE BRIEF

SKILLS & TOOLS YOU WILL MASTER

Python + scikit-learn | PyTorch + Transformers | LangChain + LangGraph | RAG + Pinecone | Agents + Mem0 + MCP | FastAPI + Docker

10 Reasons This Works.

Why students choose Saarathi - and how each card helps you ship a real career.

01

10 Max / Batch

Not 50. Not 30. Ten. Every student seen, heard, and unblocked - every class.

02

Sunday Open Classroom

Every Sunday the classroom is open. Come in, pair-program, get unstuck, or just focus.

03

Weekly Detailed Syllabus

Full syllabus from day one. Every week: what to read, what to build, what the outcome is.

04

Revision Every 5th Day

Every fifth class is a revision session. Nothing piles up or slips through.

05

Week 1: Mindset

The first week is about getting into the right headspace before touching a line of code.

06

3 Ways of Learning

Self-learning + collaboration + mentorship. All three built into every course.

07

1:1 Mock Interview

Real mock interview with your mentor in the final week. Actual hiring prep, not fluff.

08

CV + LinkedIn

Resume writing and LinkedIn optimisation. Leave with a profile that gets replies.

09

Real Projects

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

10

Gate Assessment

Aptitude test before Day 1. Maps your strengths, weaknesses, learning style, and pace - then builds your personal plan.

saarathiacademy.com.np/ai-engineering-and-machine-learning-course-in-nepal

+977-9744442469 · +977-9761095364

AI Engineering & Machine Learning

Why This Course

For people who want to become junior AI engineers shipping real LLM, RAG and agent systems, not ChatGPT wrappers. Employers want engineers who understand what an embedding is, can build an evaluated RAG pipeline, design tool-using agents and deploy models with production discipline. Those roles are in demand in Nepal and on remote international teams. Pay scales with your skills and portfolio; we don't promise a salary.

The 12 weeks are foundations-first: five weeks of base-building before any LLM work. Python from scratch in Weeks 1–2, then the numpy/pandas/SQL data toolkit and a right-sized classical machine-learning block, then deep learning and transformer intuition. The metrics, train/test discipline and leakage cures from that block become your LLM evaluation discipline later.

Weeks 6–12 are LLMs, RAG, agents, evaluation, fine-tuning (LoRA/QLoRA) and local inference (Ollama), closing with a deployed, traced, cost-tracked capstone. You leave with that capstone and the eval, cost and deployment discipline employers screen for, not demo notebooks.

AI productivity tools (Cursor for coding, Claude Artifacts for prototyping) are woven in from Week 1.

Who Is This Course For

- **Complete beginners** who want a real AI engineering career and are willing to put in the foundations work
- **Python-comfortable learners** who want to move beyond tutorials into deployed AI systems
- **Data scientists** transitioning from notebook analysis into engineering and deployment
- **Full stack or backend developers** specialising into AI systems
- **ML learners** who want production discipline around what they already know
- **Builders** who want to ship real LLM + RAG + agent products instead of demos

Course Snapshot

PARAMETER	DETAILS
Course Code	TR-10
Duration	12 Weeks (3 Months)

PARAMETER	DETAILS
Schedule	Monday to Friday, 2 Hours/Day
Total Hours	120 Hours
Batch Size	Maximum 10 Students
Course Fee	NPR 39,750
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

- No prior coding required, Python is taught from scratch in Weeks 1 and 2
- A laptop with 8GB+ RAM, 16GB is better, no GPU needed
- Comfortable using a computer daily, files and folders
- Basic English reading and writing, you will write short READMEs and progress updates
- Saarathi Gate Assessment before Day 1, diagnostic, no pass or fail

Nothing to install beforehand. VS Code, Python and the OpenAI, Anthropic and HuggingFace accounts are set up together in the first class.

Optional warm-up. None of this is required and Week 1 assumes zero, but arriving with it makes the first two weeks easier:

- 3Blue1Brown Neural Networks chapter 1 on YouTube, visual intuition only, no code
- The first 3 sections of freeCodeCamp Scientific Computing with Python at freecodecamp.org/learn, variables, loops and functions
- Skim one AI product you already use and write down what you think it does under the hood, we come back to it in Week 6

Assessments at a glance

Saarathi Gate Assessment, before Day 1. Diagnostic, no pass or fail; it maps your learning style and pace.

Saarathi Certification Exam, at the end. 2 hours, around 30 questions specific to this course, in mixed formats (single choice, multi-select, true/false, numeric, ordering, matching, fill-in-the-blank, short and long answer, code, short case studies). Pass mark 60 percent, and a pass is

required for the Saarathi Certificate of Completion. Proctored: it runs in fullscreen, switching tab or app and pasting are logged, and three strikes end it. Answer in any order, but every question needs an answer before you submit.

Recommendation letters are mentor-issued, only to learners who pass, and never guaranteed.

Your Learning Week

DAY	ACTIVITY
Mon–Fri	2-hour live class session (hands-on, project-driven) with mentor
Mon–Fri	2-hour self-study at home (assignments, practice, debugging)
Sunday	Open classroom for focused build time or rest
Day 5 (every week)	Within the 2-hour session, the last 1 hour is a built-in revision block plus architecture review for consolidation and doubt-clearing

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 - Engineering for AI	- **Functions** and **comprehensions**: inputs, outputs, defaults, `*args` / `**kwargs`, scope (taught after a week of concrete code so the abstraction clicks), then comprehensions read as sugar over a loop you already understand

WEEK	FOCUS AREA	TOPICS
		- **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**, **R^2**, **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 - Bake-off: tuned random forest vs the Week 3 logistic baseline, interpretability vs accuracy

WEEK	FOCUS AREA	TOPICS
		- 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) - **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 inference as a fallback tier, **model-routing**, context-window length and **token-cost math** - **AI Gateway** awareness (**OpenRouter**, **Portkey**, **Helicone**): routing, retries, unified spend

WEEK	FOCUS AREA	TOPICS
		- 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 - **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-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 & LangGraph Orchestration	- **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 `read_file` / `write_file` / `run_shell`, 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 `create_agent` (the LangChain 1.x standard on the **LangGraph** engine; **AgentExecutor** and `create_react_agent` 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** (`deepagents`) 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 (agenttools) / **A2A** (agentagent) / **AG-UI** (agentuser) awareness - Agent runtimes: **LangGraph** vs **OpenAI Agents SDK** vs **Claude Agent SDK** (**CrewAI**, **AutoGen**, Computer Use / browser agents awareness) - LangGraph orchestration: **StateGraph** (nodes, edges, typed state), branching, supervisor pattern, **human-in-the-loop**, checkpointing - **Hugging Face Agents Course** in self-study; certificate milestone due end of Week 10
Week 10	Evaluation & Safety	- **LLM eval** pipelines and **LLM-as-judge** (a strong model grades your outputs against a disclosed rubric) - **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,

WEEK	FOCUS AREA	TOPICS
		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); 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

WEEK	FOCUS AREA	TOPICS
		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), 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 (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 `create_agent` on the LangGraph engine, LangSmith, memory

CATEGORY	TECHNOLOGIES
	layers (Mem0 hands-on; Zep + Graphiti, Letta/MemGPT, LangMem awareness), Deep Agents (`deepagents`) 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** means repeated practice until you can apply the concept confidently. **Awareness** means you are introduced clearly enough to 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

FOCUS AREA	LEARNING FOCUS
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 `create_agent` on LangGraph, Deep Agents `deepagents`, 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
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

FOCUS AREA	LEARNING FOCUS
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

Each student picks **one** capstone with their mentor in the Week 8 architecture-review slot. Pool rotates per 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

#	PROJECT	STACK FOCUS
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	Ship LLM, RAG and agent features under review; convert the portfolio into a first junior role
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 39,750. 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?

Prerequisites are listed above. The Saarathi Gate Assessment runs before Day 1 to map your starting point, diagnostic, not pass-or-fail. Below prerequisites? We suggest a short foundation track first.

How long is the course?

12 Weeks (3 Months). The capstone is scoped in Week 8 and built inside the course, with a 1:1 mock interview in the final week.

What tools will I use?

PyTorch, HuggingFace, LangChain, FastAPI, Pinecone, Docker. Every module ships a public, deployed artefact for your portfolio.

What roles does this prepare me for?

Roles are listed in the Career Paths section. We do not promise jobs. We build your portfolio, run mock interviews, polish your CV and LinkedIn, and refer you into our hiring network.

How does the batch and mentorship work?

Maximum 10 students per batch, every student gets direct mentor time and project review. Hybrid by default, in-person at Old Baneshwor with online recordings and office hours. Sunday Open Classroom is free for enrolled students: pair-program, get unstuck, or just focus.

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 1.0 · Updated July 2, 2026