AI Engineer Bootcamp

A Twelve Week Journey from Foundations to Capstone

Week 01 What is AI Engineering & Environment Setup

- Understand the differences between an AI Engineer, ML Engineer, and Data Scientist, and review the career roadmap and salary landscape globally and in India.
- Gain an overview of key tools including Python, FastAPI, LangChain, OpenAI, and HuggingFace, and explore how modern AI apps are built using end-to-end architecture.
- Review real-world AI product demos such as ChatGPT, Copilot, and Perplexity.
- Set up your development environment by installing Python 3.11+, pip, virtual environments (venv/conda), Docker Desktop, and VS Code with essential extensions (Python, Pylance, GitLens, REST Client).
- Configure a Git and GitHub account with SSH keys, and learn bash terminal fundamentals including navigation, chmod, and environment variables.
- Write your first "Hello AI World" Python script, create and push to a GitHub repository, and understand the structure of .gitignore, README.md, LICENSE, and requirements.txt files.
- Review course expectations and grading, form teams, and access essential resources like Roadmap.sh, HuggingFace, and OpenAI documentation.

Week 02 Python Mastery for AI Engineers

- Refresh core Python fundamentals, covering data types, comprehensions, unpacking, the walrus operator, and functions (args, kwargs, decorators, closures).
- Implement exception handling patterns specifically used in APIs.
- Work with JSON, CSV, and YAML files, use the requests library to call REST APIs, and apply regular expressions for text preprocessing.
- Master Async Python by understanding asyncio, async/await, event loops, and aiohttp for async HTTP calls, and learn why async matters in AI for streaming responses.
- Build a Python CLI tool that calls the OpenAI API and saves the output to a file.

Week 03 Data Structures, OOP & Clean Code

- Learn Object-Oriented Programming (OOP) in Python, focusing on classes, inheritance, and dunder methods.
- Understand data structuring using dataclasses and Pydantic models.
- Study core data structures including lists, dicts, sets, queues, and stacks, alongside basic algorithms for sorting and searching.
- Explore basic concepts such as recursion, generators, and itertools.
- Apply clean code practices utilizing PEP8, type hints, and mypy, and learn how to write unit tests using pytest.
- Participate in code review labs using a structured rubric and collaboratively refactor poorly written scripts.

Week 04 APIs & FastAPI Introduction

- Understand core REST API concepts, including HTTP methods (GET, POST, PUT, DELETE, PATCH), status codes, headers, and the request/response lifecycle.
- Differentiate between JSON, Form data, and Multipart formats, and test APIs using tools like Postman, curl, and the VS Code REST Client.
- Dive into FastAPI project architecture, exploring routers, models, schemas, path parameters, query parameters, and request bodies using Pydantic.
- Learn dependency injection, the appropriate use of async endpoints, and how to utilize autogenerated documentation via Swagger UI and ReDoc.
- Build a Task Manager REST API with full CRUD operations, integrating validation, error handling, and documentation.

Week 05 GitHub, CI/CD & Developer Workflows

- Master Git branching strategies (GitFlow, trunk-based development), understand merge versus rebase, and learn how to resolve merge conflicts.
- Utilize advanced Git commands such as stash, cherry-pick, and bisect.
- Collaborate effectively by writing and reviewing Pull Requests, and managing work via GitHub Issues, Projects, and Milestones.
- Set up CI/CD pipelines using GitHub Actions and safely manage secrets.

Week 06 FastAPI Advanced + Database Integration

- Containerize Python FastAPI applications using Dockerfiles, configure multi-service setups with docker-compose, and push images to Docker Hub or GitHub Container Registry.
- Deploy FastAPI applications to cloud platforms (Railway or Render) and manage production configuration using environment variables.
- Integrate databases using SQLAlchemy ORM (models, sessions, relationships) with PostgreSQL, and manage database migrations using Alembic.
- Implement asynchronous database operations with asyncpg.
- Apply FastAPI production patterns including JWT Authentication (OAuth2PasswordBearer), background tasks, WebSockets, rate limiting, CORS, and middleware.
- Enhance performance by caching API responses with Redis, implementing connection pooling, optimizing queries, and profiling applications with py-spy.
- Upgrade the Task Manager API project by adding authentication and database persistence.

Week 07 Machine Learning Fundamentals

- Understand foundational ML concepts, differentiating between supervised, unsupervised, and reinforcement learning.
- Learn crucial training principles such as train/test/validation splits, overfitting, underfitting, loss functions, gradient descent, and learning rate.
- Apply Scikit-learn for practical ML, working with linear and logistic regression, decision trees, and random forests.
- Perform feature engineering, normalization, and pipeline creation, and evaluate models using precision, recall, F1, and ROC-AUC.
- Serve machine learning models by saving/loading with joblib or pickle, wrapping them in FastAPI endpoints, and managing model versions.
- Train and serve a sentiment classifier via a custom API.

Week 08 Deep Learning & Neural Networks

- Explore neural network foundations, covering perceptrons, activation functions, backpropagation, and conceptual overviews of CNNs (image) and RNNs (text).
- Work with PyTorch basics, including tensors, autograd, and building training loops.
- Dive into Transformer architectures by understanding the math and intuition behind attention mechanisms, and compare models like BERT, GPT, and T5.
- Study tokenization strategies including BPE, WordPiece, and SentencePiece.
- Implement embeddings and semantic search using Word2Vec, GloVe, Sentence Transformers, cosine similarity, and dot products.
- Use the HuggingFace Transformers pipeline API to build a semantic search system on a custom dataset.

Week 09 LLMs & Prompt Engineering

- Learn how state-of-the-art LLMs (GPT-4, Claude, Gemini, Llama) work, and understand parameters like context window, temperature, top-p, and max_tokens.
- Gain hands-on experience using the OpenAI API, Anthropic API, and Groq API.
- Master prompt engineering techniques, including zero-shot, few-shot, and chain-of-thought prompting, alongside system prompts, role prompting, and guardrails.
- Force structured output from LLMs using JSON mode and function calling.
- Understand when to fine-tune versus when to prompt engineer, and explore Parameter-Efficient Fine-Tuning (PEFT) methods like LoRA and QLoRA, along with a demo of the HuggingFace Trainer API.
- Build a domain-specific chatbot that delivers structured output using the OpenAI API and FastAPI.

Week 10 RAG & Vector Databases

- Understand Retrieval-Augmented Generation (RAG) architecture and why it outperforms fine-tuning for specific knowledge tasks.
- Embed documents at scale and implement chunking strategies (fixed, semantic, recursive).
- Compare vector databases like Pinecone, ChromaDB, Weaviate, and pgvector, and understand HNSW indexes and approximate nearest neighbour search.
- Apply metadata filtering to optimize vector search results.
- Build systems using LangChain (chains, prompts, memory, tools) and LlamaIndex (document ingestion, query engines), and implement hybrid search utilizing BM25 and vector data.
- Develop a Q&A bot that queries custom PDF documents using RAG and FastAPI.

Week 11 Tools & Production Deployment

- Ensure observability and evaluation using LangSmith, Weights & Biases, and Langfuse for tracing, and evaluate outputs using RAGAS and LLM-as-judge.
- Deploy AI applications to production environments (AWS / GCP / Azure overview), utilize GPU serverless platforms (Modal, Replicate), and configure monitoring, cost tracking, and rate limiting.
- Finalize teams and conduct architecture design sessions for the upcoming Capstone project.

Week 12 Capstone Project

- Participate in a final build sprint to complete and polish a full-stack AI application.
- Conduct code reviews with mentors to ensure performance and security, and deploy the project to production via a CI/CD pipeline.
- Create comprehensive project documentation including a README, API docs, and a demo video.
- Present the finished project on Demo Day (pitch and live demo) to guest evaluators and participate in peer voting.
- Prepare for career advancement by setting up portfolios (GitHub, LinkedIn, personal site), optimizing resumes for AI Engineer roles, and learning how to contribute to open source.