# Artificial Intelligence & Machine Learning

***Comprehensive Professional Training Syllabus***

**Issued by: Ally Tech Services**

**Website: https://allytechservices.in/**

*This document presents the official training syllabus for the AI/ML program offered by Ally Tech Services. It covers a structured journey from foundational programming and mathematics to advanced agentic AI systems, with hands-on practice on industry-standard tools and frameworks.*

## Program Overview

The Ally Tech Services AI/ML program is an industry-aligned, hands-on training curriculum designed to build strong proficiency in Artificial Intelligence and Machine Learning. The program walks learners through Python programming, essential mathematics, data engineering, cloud fundamentals, classical machine learning, deep learning, computer vision, modern NLP with Transformers, Generative AI with LLMs & RAG, MLOps, model deployment, and the latest Agentic AI patterns and frameworks. The curriculum is delivered through guided lectures, code-along sessions, labs and capstone-style mini-projects.

### Key Highlights

- Online Instructor-led learning.
- 11 modules organized in 4 phases: Foundations, Core AI/ML, Specialization, and Advanced AI.
- Hands-on labs, code walkthroughs and capstone-style projects in every module.
- Industry tooling: Python, NumPy, Pandas, Polars, TensorFlow, PyTorch, Hugging Face, LangChain, CrewAI, MLFlow, FastAPI, Docker.
- Cloud platforms: AWS (SageMaker, S3, EC2), Azure ML, and Google Cloud Vertex AI.

- Modern Generative AI coverage: LLMs, Prompt Engineering, Vector Databases, RAG, Multimodal AI and Agentic Systems.

## Phase-Wise Module Summary

| Phase | Module | Topics Covered |
|---|---|---|
| Foundations | Python Programming | For Beginners: Core concepts from scratch (Variables, Data Types, Control Flow, Functions, File Handling, OOP). For Data Science: NumPy, Pandas, Matplotlib, Seaborn. |
| Foundations | Math for AI/ML | Essential Linear Algebra (Vectors, Matrices), Probability fundamentals, and key Statistical concepts (Descriptive & Inferential Statistics). |
| Foundations | Data Engineering Foundations | Advanced SQL for complex data retrieval, joining multiple tables, window functions. Introduction to ETL (Extract, Transform, Load) principles using Pandas. |
| Foundations | Cloud Fundamentals | Core concepts of AWS, Azure, & GCP. Focus on key services for AI/ML: Object Storage (S3), Compute (EC2), and Managed ML Platforms (SageMaker/Vertex AI). |
| Core AI/ML | Machine Learning | Supervised Learning: Classification (Logistic Regression, SVM, Decision Trees), Ensemble Methods (Random Forests, Gradient Boosting). Unsupervised Learning: Clustering (K-Means), Dimensionality Reduction (PCA). |
| Core AI/ML | Deep Learning & Computer Vision | Intro to Neural Networks (ANNs), Convolutional Neural Networks (CNNs) for image data, Transfer Learning with pre-trained models (e.g., VGG16, ResNet). |
| Specialization | NLP with Hugging Face Transformers | Word Embeddings, Transformer Architecture. Hands-on with Hugging Face: Using the transformers library for pipelines, tokenization, and fine-tuning models for tasks like sentiment analysis. |
| Specialization | LLMs, Vector Databases & RAG | Working with LLM APIs (OpenAI), Prompt Engineering, LangChain framework, Vector Databases (Chroma/FAISS), building Retrieval-Augmented Generation (RAG) pipelines. |
| Specialization | MLOps using MLFlow | MLOps principles, experiment tracking with MLFlow, model registry for versioning, packaging models for deployment. |
| Specialization | Model Serving & Deployment | Building REST APIs with FastAPI, data validation with Pydantic, containerizing applications with Docker for reproducible environments. |
| Advanced AI | Agentic AI & Frameworks | Core concepts of AI Agents (Reasoning, Memory, Tool Use). Building autonomous agents with frameworks like LangChain Agents or CrewAI. |
| | | |

# Detailed Curriculum

*The detailed module-wise curriculum below outlines each section, topic, sub-topics, and an estimated duration in minutes.*

## Module 1: Python Programming

| Section | Topic | Sub-Topics |
|---|---|---|
| Python Introduction & Setup | Installing Python & Anaconda | History of Python; Why Python is popular for AI/ML; Installing Python on Windows/Mac/Linux; Setting up Anaconda Distribution |
| Python Introduction & Setup | Introduction to Jupyter Notebooks & VS Code | Launching Jupyter Notebook; Basic Notebook workflow; Setting up VS Code for Python |
| Python Introduction & Setup | Writing Your First Program | Printing 'Hello, World!'; Running Python scripts from terminal |
| Python Introduction & Setup | Variables & Data Types | int, float, str, bool; Declaring variables; Dynamic typing |
| Python Introduction & Setup | Type Casting | Converting between int, float, str; Implicit vs Explicit casting |
| Operators & Strings | Arithmetic, Comparison & Logical Operators | +, -, *, /, %, //, **; ==, !=, >, <, >=, <=; and, or, not |
| Operators & Strings | String Indexing & Slicing | Positive & Negative Indexing; Slice notation |
| Operators & Strings | Common String Methods | .upper(), .lower(), .split(), .strip() |
| Operators & Strings | Formatted Strings | f-strings; .format() method |
| Core Data Structures (Part 1) | Lists in Python | Defining lists; Nested lists |
| Core Data Structures (Part 1) | List Operations, Slicing & Methods | Indexing; append(), insert(), remove(), pop(); list slicing; sort(), reverse(), count() |
| Core Data Structures (Part 1) | Tuples in Python | Defining tuples; Tuple packing/unpacking; Why tuples are immutable; Benefits and trade-offs; Tuple vs List |
| Core Data Structures (Part 2) | Sets in Python | Set Operations (union, intersection, difference); set() function; Uniqueness of elements |
| Core Data Structures (Part 2) | Dictionaries in Python | Creating key-value pairs; Accessing, adding, modifying values |
| Core Data Structures (Part 2) | Control Flow: if/elif/else | Conditional syntax; Nested conditionals |
| Core Data Structures (Part 2) | Control Flow: For Loops | Looping over lists, tuples, dicts; Using range() |
| Core Data Structures (Part 2) | Control Flow: While Loops | Loop structure; break & continue; Avoiding infinite loops |
| Functions | Functions vs Methods in Python | def keyword; Calling functions |
| Functions | Parameters, Arguments, Scope, Return | Positional, keyword, default arguments; Local vs Global variables; Returning values |
| Functions | Lambda Functions | Syntax and usage; Single-line functions |
| File Handling & OOP Intro | Reading & Writing Files | open(), read(), write(); Reading CSV files |
| File Handling & OOP Intro | Using with Statement | Context managers for safe file handling |
| OOPs in Python | OOPs Concepts | Classes, Objects, Encapsulation, Abstraction, Polymorphism, Methods |
| NumPy for Numerical Data | Why NumPy? | Performance comparison with Python lists |

| | | |
|---|---|---|
| NumPy for Numerical Data | Creating NumPy Arrays | np.array(), np.zeros(), np.ones() |
| NumPy for Numerical Data | Array Indexing & Slicing | Accessing and slicing multi-dimensional arrays |
| NumPy for Numerical Data | Array Shape & Reshape | shape, reshape, flatten |
| NumPy for Numerical Data | Mathematical Operations | element-wise addition, multiplication |
| NumPy for Numerical Data | Statistical Operations | mean(), median(), std() |
| NumPy for Numerical Data | Broadcasting | Broadcasting rules and examples |
| Pandas for Data Manipulation | Introduction to Series & DataFrames | Creating Series & DataFrames manually |
| Pandas for Data Manipulation | Reading Data | pd.read_csv, pd.read_excel |
| Pandas for Data Manipulation | Inspecting DataFrames | head(), tail(), info(), describe() |
| Pandas for Data Manipulation | Selecting Data | loc, iloc, column selection |
| Pandas for Data Manipulation | Handling Missing Values | isnull(), dropna(), fillna() |
| Data Visualization | Intro to Visualization | Why visualization matters in ML/AI |
| Data Visualization | Basic Matplotlib Plots | Line, Bar, Histogram |
| Data Visualization | Intro to Seaborn | Boxplot, Heatmap |

## Module 2: Math for AI/ML

| Section | Topic | Sub-Topics |
|---|---|---|
| Linear Algebra Essentials | Understanding Vectors | Definition of a Vector; Scalar & Vector Quantities |
| Linear Algebra Essentials | Vector Operations | Vector Addition & Subtraction; Scalar Multiplication |
| Linear Algebra Essentials | Understanding Matrices | Definition of a Matrix; Matrix Notation & Dimensions |
| Linear Algebra Essentials | Matrix Operations | Matrix Addition; Matrix Multiplication (Row × Column rule) |
| Linear Algebra Essentials | Introduction to Tensors | What is a Tensor?; Tensor as multi-dimensional array |
| Linear Algebra Essentials | Matrix Transpose & Inverse | Transpose of a Matrix; Concept of Matrix Inverse |
| Linear Algebra Essentials | Dot Product | Intuition & Formula; Applications in ML |
| Linear Algebra Essentials | Eigenvectors & Eigenvalues | Conceptual Understanding; Use in PCA |
| Probability Fundamentals | Sample Spaces & Events | Defining Sample Space; Events & Outcomes |
| Probability Fundamentals | Basic Probability Rules | Addition Rule; Multiplication Rule |
| Probability Fundamentals | Conditional Probability | P(A\|B) |
| Probability Fundamentals | The Chain Rule | Extending conditional probability to multiple events |

| | | |
|---|---|---|
| Probability Fundamentals | Bayes' Theorem | Formula & Intuition; Naive Bayes use case |
| Probability Fundamentals | Random Variables | Discrete vs Continuous Random Variables |
| Probability Fundamentals | Common Distributions | Normal/Gaussian Distribution; Binomial Distribution |
| Statistics for ML | Measures of Central Tendency | Mean; Median; Mode |
| Statistics for ML | Measures of Dispersion | Variance; Standard Deviation; Range |
| Statistics for ML | Correlation vs Causation | Correlation Coefficient; Why correlation ≠ causation |
| Statistics for ML | Population vs Sample | Parameters vs Statistics; Sampling Techniques |
| Statistics for ML | Introduction to Hypothesis Testing | Null & Alternative Hypotheses; Type I & Type II Errors |
| Statistics for ML | Understanding P-Value | Definition; Statistical Significance & Confidence Level |

## Module 3: Data Engineering Foundations

| Section | Topic | Sub-Topics |
|---|---|---|
| SQL Refresher | SQL Fundamentals | Basic SQL Commands - DDL, DML, DQL, TCL |
| SQL Refresher | Advanced SQL | Joins, Window Functions |
| ETL Principles & Practice | Introduction to Data Engineering for AI/ML | Role of data engineering in ML lifecycle; Typical data flow from raw → cleaned → feature store |
| ETL Principles & Practice | ETL Process Overview | Extract, Transform, Load explained with examples; Why ETL is critical for ML data readiness |
| ETL with Pandas | Extract Step | Reading from CSV, JSON, SQL Databases; Handling large datasets with chunks |
| ETL with Pandas | Transform Step | Cleaning data (missing values, duplicates); Feature engineering using .apply() and .groupby() |
| ETL with Pandas | Load Step | Writing cleaned datasets to CSV, Parquet, or Database |
| ETL with Polars | Introduction to Polars | Why Polars is faster than Pandas (lazy evaluation); Installation and basic syntax |
| ETL with Polars | Transformations with Polars | DataFrames, LazyFrame operations; GroupBy and aggregation with high performance |
| Designing Data Pipelines | Batch Data Pipelines | Building reproducible batch ETL pipelines; Using Python scripts or orchestration tools |
| Designing Data Pipelines | Orchestration Basics | Intro to Apache Airflow / Prefect; DAGs (Directed Acyclic Graphs) for ML data flows |
| Designing Data Pipelines | Automating ML Data Workflows | Automating daily data refresh; Feature store population |
| Performance & Scalability | Handling Large Datasets | Memory optimization tips in Pandas/Polars; Using chunking, dtypes optimization |
| Performance & Scalability | Parallel & Distributed Data Processing | Intro to Dask / Spark for scaling beyond a single machine |
| Data Quality & Validation | Data Validation Techniques | Schema validation (Great Expectations, Pydantic); Detecting anomalies/outliers |

## Module 4: Cloud Fundamentals

| Section | Topic | Sub-Topics |
|---|---|---|
| Introduction to Cloud Computing | What is Cloud Computing? | Definition & Key Characteristics; Service Models: IaaS, PaaS, SaaS |
| Introduction to Cloud Computing | Benefits of Cloud | Scalability; Elasticity; Cost-Effectiveness |
| Introduction to Cloud Computing | Overview of Cloud Providers | AWS, Azure, GCP: Market share, strengths, use cases |
| Introduction to Cloud Computing | Global Infrastructure | Regions, Availability Zones, Edge Locations |
| Core Services: Compute & Storage | Introduction to Compute Services | AWS EC2, Azure VMs, Google Compute Engine |
| Core Services: Compute & Storage | Compute Use Cases | When to use VMs; On-demand vs Reserved vs Spot Instances |
| Core Services: Compute & Storage | Introduction to Storage Services | Object Storage Basics: AWS S3, Azure Blob, GCS |
| Core Services: Compute & Storage | Storage Use Cases | Dataset storage; Model artifact storage |
| Core Services: Databases & Networking | Managed Relational Databases | RDS (AWS), Cloud SQL (GCP), Azure SQL |
| Core Services: Databases & Networking | NoSQL Databases | DynamoDB, Firestore basics |
| Core Services: Databases & Networking | VPC Concepts | CIDR ranges; Public vs Private Subnets |
| Core Services: Databases & Networking | Security Groups & Firewalls | Inbound & Outbound rules; IAM and Policy Manager |
| AI/ML on the Cloud | AWS SageMaker Overview | Pre-built Jupyter environments; Training & Deploying Models |
| AI/ML on the Cloud | Azure ML Overview | Workspaces, Pipelines, Endpoints |
| AI/ML on the Cloud | Google Vertex AI Overview | Managed pipelines, AutoML, Model Registry |
| AI/ML on the Cloud | Benefits of Managed Platforms | Pre-configured environments; Auto-scaling; MLOps integration |

## Module 5: Machine Learning

| Section | Topic | Sub-Topics |
|---|---|---|
| Introduction to Machine Learning | ML Fundamentals | Definition, Core Terminology; Types of ML: Supervised, Unsupervised, Reinforcement; CRISP-DM Workflow |
| Introduction to Machine Learning | Data Preprocessing & Feature Engineering | Importance of clean data; Train-Test Split; Feature Scaling; Encoding Categorical Variables |
| Supervised Learning: Regression | Linear Regression | Intuition (predicting continuous values); Simple & Multiple Linear Regression |
| Supervised Learning: Regression | Cost Function & Gradient Descent | MSE as cost function; Gradient Descent (conceptual overview) |
| Supervised Learning: Regression | Model Evaluation (Regression) | MAE, MSE, RMSE, $R^2$; Interpreting coefficients |
| Supervised Learning: Regression | Advanced Regression | Polynomial Regression; Overfitting & Underfitting (Bias-Variance Tradeoff); Regularization (L1/Lasso, L2/Ridge) |

| | | |
|---|---|---|
| Supervised Learning: Classification | Classification Basics | What is Classification?; Binary vs Multi-class |
| Supervised Learning: Classification | Core Classification Algorithms (Part 1) | Logistic Regression; KNN (intuition & tuning k) |
| Supervised Learning: Classification | Core Classification Algorithms (Part 2) | SVM (hyperplanes, kernels); Naive Bayes |
| Supervised Learning: Classification | Decision Trees | Nodes, splits, Gini/Entropy; Visualizing Trees |
| Model Evaluation for Classification | Classification Metrics | Accuracy & its Limitations; Confusion Matrix; Precision, Recall, F1-Score; ROC Curve & AUC |
| Model Evaluation for Classification | Handling Imbalanced Data | Undersampling, Oversampling, SMOTE |
| Advanced ML: Ensemble Learning | Ensemble Overview | Wisdom of the crowd; Bagging vs Boosting |
| Advanced ML: Ensemble Learning | Bagging & Random Forests | Bootstrap Aggregating; How Random Forests improve Decision Trees |
| Advanced ML: Ensemble Learning | Boosting & XGBoost | Gradient Boosting concept; XGBoost overview & why it dominates competitions |
| Unsupervised Learning | Clustering | What is Clustering?; K-Means Algorithm; Choosing k (Elbow Method); Silhouette Score |
| Unsupervised Learning | Dimensionality Reduction | Curse of Dimensionality; PCA intuition and use cases |
| Model Optimization & Advanced Concepts | Hyperparameter Tuning & CV | Model Parameters vs Hyperparameters; K-Fold Cross-Validation; Grid Search vs Random Search |
| Model Optimization & Advanced Concepts | Federated Learning | Concept & Benefits; High-level Architecture & Use Cases |

## Module 6: Deep Learning & Computer Vision

| Section | Topic | Sub-Topics |
|---|---|---|
| Fundamentals | Limitations of Traditional ML | Why ML struggles with unstructured data like images/text/audio |
| Fundamentals | Biological Inspiration: Neurons | The Brain → Neurons, Synapses, Activations |
| Fundamentals | Artificial Neuron (Perceptron) | Inputs, Weights, Weighted Sum, Activation Function |
| Fundamentals | Artificial Neural Networks (ANN) Overview | What is an ANN?; Stacking layers of perceptrons |
| Fundamentals | ANN Architecture | Input, Hidden, and Output Layers explained |
| Fundamentals | Forward Propagation | How signals flow through the network |
| Fundamentals | Backpropagation (Conceptual) | How networks adjust weights to minimize error |
| Fundamentals | Activation Functions | Sigmoid, Tanh, ReLU and why non-linearity is crucial |
| Fundamentals | Loss Functions | Categorical Cross-Entropy, MSE (when to use each) |
| Fundamentals | Optimizers | Gradient Descent intuition, Adam optimizer |
| Fundamentals | Intro to Frameworks | Overview of TensorFlow, Keras, PyTorch |
| Fundamentals | Building a Sequential Model | Adding Dense layers step by step |
| Fundamentals | Model Compilation | Choosing optimizer, loss, metrics |
| Fundamentals | Training a Model | Using .fit(), epochs, batch size |
| Fundamentals | Model Evaluation | Using .evaluate() on test data |
| Deep Learning for CV | Image Representation in Computers | Pixels, Channels (RGB/Grayscale), Image Dimensions |
| Deep Learning for CV | Challenges for ANN with Image Data | Flattening destroys spatial relationships; parameter explosion |

| | | |
|---|---|---|
| Deep Learning for CV | CNN Core Idea | Filters detect edges, shapes, textures |
| Deep Learning for CV | Convolutional Layer | Filters, Feature Maps, Padding, Stride |
| Deep Learning for CV | Pooling Layer | Max Pooling, Average Pooling, dimensionality reduction |
| Deep Learning for CV | CNN Architecture | Typical architecture (Conv → Pool → Flatten → Dense → Output) |
| Deep Learning for CV | Training a CNN | Fitting model on Fashion-MNIST/CIFAR-10 |
| Deep Learning for CV | Data Augmentation | Flipping, Rotating, Zooming images |
| Deep Learning for CV | Transfer Learning Basics | Concept & Benefits of Transfer Learning |
| Deep Learning for CV | Using Pre-trained Models | VGG16, ResNet as feature extractors |
| Deep Learning for CV | Fine-Tuning Pre-trained Models | Unfreezing top layers and retraining |
| Deep Learning for NLP | RNNs and LSTM Networks - Theory | Basics of RNN and LSTM; Architecture; Advantages and Disadvantages |
| Deep Learning for NLP | RNNs and LSTM Networks - Practice | Text processing with RNN and LSTM |

## Module 7: NLP with Hugging Face Transformers

| Section | Topic | Sub-Topics |
|---|---|---|
| Foundations of Modern NLP | From Traditional NLP to Word Embeddings | Bag-of-Words & TF-IDF limitations; Word2Vec, GloVe; contextual embeddings |
| Foundations of Modern NLP | Text Preprocessing for Transformers | Tokenization (BPE, WordPiece); Special Tokens (CLS, SEP, PAD); Attention Masks |
| The Transformer Architecture | Limitations of RNNs/LSTMs & Rise of Transformers | Vanishing gradient issues; Encoder-Decoder overview; why Transformers solved these problems |
| The Transformer Architecture | Attention Mechanism | Basic attention formula; weighted sum of values; intuition |
| The Transformer Architecture | Understanding Self-Attention | Query, Key, Value vectors; Multi-Head Attention; positional encoding |
| The Transformer Architecture | Transformer Model Families | BERT (Encoder), GPT (Decoder), T5 (Encoder-Decoder); key use cases |
| The Hugging Face Ecosystem | Introduction to Hugging Face Hub & Pipelines | Exploring Hub (Models/Datasets/Spaces); Using pipeline() for zero-shot tasks |
| The Hugging Face Ecosystem | Loading Models & Tokenizers | AutoModel, AutoTokenizer; Text→Tokens→Input IDs→Logits→Predictions |
| Fine-Tuning a Transformer Model | Dataset Preparation | Loading with load_dataset; preprocessing with .map(); format for PyTorch/TF |
| Fine-Tuning a Transformer Model | Training Setup with Trainer API | TrainingArguments (epochs, LR, batch size); Trainer setup with metrics |
| Fine-Tuning a Transformer Model | Model Training & Evaluation | Running .train(), .evaluate(); saving model & tokenizer |
| Model Deployment & Optimization | Model Inference & Deployment | Converting to pipeline; running inference; exposing via FastAPI |
| Model Deployment & Optimization | Performance Optimization | Mixed precision inference; quantization basics; batching & latency tips |
| Model Deployment & Optimization | Error Analysis & Evaluation | Confusion matrix; Precision/Recall/F1; error breakdown for NLP tasks |

# Module 8: LLMs, Vector Databases & RAG (Gen AI)

| Section | Topic | Sub-Topics |
|---|---|---|
| The Gen AI Landscape | Introduction to Generative AI | Discriminative vs Generative; key applications (text, image, audio, code) |
| The Gen AI Landscape | Generative Model Families | LLMs; Diffusion, GANs, VAEs; where each shines |
| The Gen AI Landscape | LLM Fundamentals & Scaling | Transformer refresher; context windows; scaling laws & limits |
| The Gen AI Landscape | Model Selection & Access | Open vs closed; model sizes (7B–70B); capability vs cost |
| Working with LLMs | API Fundamentals & First Calls | Keys/env; chat/completions; hello-world flows |
| Working with LLMs | Tokens, Pricing, Latency & Streaming | Token accounting; batching; response streaming |
| Working with LLMs | Function Calling & Tool Use | JSON/function calling; schema design; tool routing |
| Working with LLMs | Structured Outputs & Validation | Forcing JSON; Pydantic/JSONSchema validation; retries |
| Prompt Engineering | Prompting Fundamentals | Zero/one/few-shot; roles/personas; system prompts |
| Prompt Engineering | Advanced Prompting & Patterns | CoT, self-consistency; delimiters; output formatting |
| Prompt Engineering | Prompt Optimization & Evaluation | A/B prompts; rubrics; automatic critique; guardrails basics |
| Fine-Tuning & Adaptation | Fine-Tuning vs PEFT Overview | Full FT vs adapters vs PEFT—when to use what |
| Fine-Tuning & Adaptation | LoRA / QLoRA Concepts | Low-rank adapters; quantization; small demo plan |
| Fine-Tuning & Adaptation | In-Context vs Fine-Tune vs RAG | Decision framework; cost/latency/quality trade-offs |
| Multimodal Gen AI | Multimodal Model Overview | Text-image-audio ecosystems; CLIP; vision-language |
| Multimodal Gen AI | Image Generation Basics | Diffusion pipeline intuition; prompts & safety |
| Embeddings & Search | Embeddings Fundamentals | Sentence embeddings; cosine/dot; model choices |
| Embeddings & Search | Chunking Strategies & Query Rewriting | Chunk size/overlap; metadata; query expansion |
| Vector Databases | Vector DB Fundamentals & Indexes | ANN vs SQL; FAISS/Chroma; HNSW/IVF basics |
| Vector Databases | Hybrid Search & Reranking | BM25 + dense; cross-encoder rerankers; filters |
| RAG | RAG Pipeline End-to-End | Ingest → chunk → embed → store → retrieve → generate |
| RAG | Retrieval Optimization | k, similarity thresholds; metadata, namespaces |
| RAG | RAG Evaluation | Retrieval (P@k/R@k/MRR); Answer-level (faithfulness, citations) |
| RAG | Advanced RAG Patterns | Multi-hop; routers; citations; grounding strategies |
| Productionizing Gen AI | Serving Patterns | FastAPI; async; streaming endpoints; inference gateways |
| Productionizing Gen AI | Caching, Rate Limits, Cost Control | Response caching; KV stores; budget guards |

## Module 9: Model Serving & Deployment

| Section | Topic | Sub-Topics |
|---|---|---|
| Introduction to Model Deployment | Why Deploy Models? | Moving from notebook to production; Batch vs Real-time inference; Importance of scalability, reliability, accessibility |
| Introduction to Model Deployment | REST API Fundamentals | What is an API?; Client-Server Model; HTTP Methods (GET, POST); Endpoints, Request/Response Cycle |
| Building APIs with FastAPI | Getting Started with FastAPI | Why FastAPI (speed, auto-docs); Installing FastAPI & Uvicorn; Creating a minimal 'Hello World' app |
| Building APIs with FastAPI | Creating API Endpoints | Path Operations (@app.get, @app.post); Handling Request Data (Path Params, Query Params, Body); Returning JSON responses |
| Building APIs with FastAPI | Testing & Documentation | Interactive docs (Swagger, ReDoc); Writing unit tests with pytest; API testing with Postman |
| Data Validation with Pydantic | Introduction to Pydantic | Why data validation matters; Settings management in APIs |
| Data Validation with Pydantic | Implementing Schemas | Defining Pydantic models; Request body validation; Response models for consistent output |
| Containerization with Docker | Docker Fundamentals | Containerization vs VMs; Docker architecture & core terms (Image, Container, Dockerfile) |
| Containerization with Docker | Writing & Running a Dockerfile | FROM, WORKDIR, COPY, RUN, CMD instructions; Building Docker image; Running the containerized API |
| Containerization with Docker | Docker Compose & Multi-Container Setup | Creating docker-compose.yml; Running API + DB in one network; Environment variables & volumes |
| End-to-End ML Model Deployment | Packaging & Loading a Model | Saving/loading model with joblib/pickle; Project structure for deployment |
| End-to-End ML Model Deployment | Creating Prediction Service | Creating FastAPI endpoint for prediction; Loading model, making predictions; Returning JSON response |

## Module 10: MLOps using MLFlow

| Section | Topic | Sub-Topics |
|---|---|---|
| Introduction to MLOps | Why MLOps? & ML Lifecycle | Why most models never reach production; Challenges beyond Jupyter Notebooks; Full ML lifecycle: data → training → deployment → monitoring |
| Introduction to MLOps | Core MLOps Principles | DevOps vs MLOps; Key pillars: Automation, Reproducibility, Collaboration; CI/CD for ML, monitoring, and rollback concepts |
| Getting Started with MLFlow | Introduction to MLFlow & Components | What MLFlow is & why it's used; Four components: Tracking, Projects, Models, Registry |
| Getting Started with MLFlow | MLFlow Installation & Setup | Installing MLFlow; Launching MLFlow UI; Understanding mlruns directory structure |
| Practical Experiment Tracking | Logging Experiments | Using mlflow.start_run(); Logging parameters (log_param) & metrics (log_metric) |
| Practical Experiment Tracking | Logging Artifacts & Models | Logging artifacts (plots, CSVs); Logging models (mlflow.sklearn.log_model, mlflow.pytorch.log_model); Concept of model flavors |
| Practical Experiment Tracking | MLFlow UI for Experiment Comparison | Filtering, comparing runs; Analyzing metrics & artifacts |
| Model Management & Registry | Understanding the Model Registry | Difference between experiment artifacts vs registered models; Purpose of registry in production workflows |
| Model Management & Registry | Working with Registered Models | Registering models via UI & code; Versioning (v1, v2…); Stage transitions (Staging → Production); Archiving & rollback |

| | | |
|---|---|---|
| Packaging & Serving | MLFlow Projects for Reproducibility | MLproject file structure; Dependency specification; Packaging code for reproducible training |
| Packaging & Serving | Serving Models | Loading a registered model for inference (mlflow.pyfunc.load_model); Serving model as REST API locally (mlflow models serve); Testing with cURL / Postman |
| Advanced Concepts | Model Monitoring & Drift Detection | Why monitoring matters post-deployment; Concept drift & data drift; Simple metrics/logging for drift detection |

## Module 11: Agentic AI & Frameworks

| Section | Topic | Sub-Topics |
|---|---|---|
| Introduction to AI Agents | Core Concepts of AI Agents | Moving beyond single LLM calls; Core Components: LLM (brain), Memory, Tools, Planning |
| Introduction to AI Agents | The Agentic Loop | The cycle: Observe → Think → Act; How agents perceive environment and make decisions |
| Core Component: Reasoning & Planning | Reasoning Frameworks (ReAct) | The ReAct framework (Reason + Act); How agents reason about intermediate steps |
| Core Component: Reasoning & Planning | Task Decomposition & Planning | Breaking down a complex goal into smaller steps; Sequencing and prioritizing tasks |
| Core Component: Tool Use | Introduction to Tool Use | What are Tools? Extending LLM's abilities (Search, Code Execution, APIs) |
| Core Component: Tool Use | Implementing Tools | Writing a custom tool in Python; How the agent decides which tool to call |
| Core Component: Memory | Memory Fundamentals | Why agents need memory (multi-turn context); Short-term vs long-term memory |
| Core Component: Memory | Types of Memory | Conversation Buffer; Summary Memory; Vector Store-backed Memory (semantic recall) |
| Building Single-Agent Systems | Introduction to LangChain Agents | What LangChain Agents are; Initializing agent with LLM, tools, prompt template |
| Building Single-Agent Systems | Building a Research Agent (Hands-on) | Creating a simple research agent with search tool; Observing intermediate reasoning (thought/action traces) |
| Introduction to Multi-Agent Systems | Multi-Agent Motivation | Why use multiple agents; Specialized roles & collaboration for complex problems |
| Introduction to Multi-Agent Systems | Agent-to-Agent Communication (A2A) | Communication channels between agents; Collaboration strategies (hierarchical, conversational) |
| Introduction to Multi-Agent Systems | Model Context Protocol | Enabling External Data access to LLMs; Exploring MCP options |
| Introduction to Multi-Agent Systems | Building MCP Example | Hands-on MCP Example |
| Building Multi-Agent Systems with CrewAI | Introduction to CrewAI | What is CrewAI and its components (Agents, Tasks, Crew) |
| Building Multi-Agent Systems with CrewAI | Practical Multi-Agent Build | Defining roles, goals, backstories; Creating tasks & assigning to agents; Executing tasks as a Crew |
| Advanced Agentic Patterns | Agentic RAG | Difference from vanilla RAG; Agent decides when/what to retrieve, then reasons on context |

# About Ally Tech Services

Ally Tech Services is committed to delivering industry-grade technology training that bridges the gap between academic concepts and real-world enterprise applications. Our AI/ML program is designed and continuously updated by practitioners and architects working on production-grade AI systems. We thank every learner for choosing us as a partner in their learning journey.

## Contact Information

**Website: https://allytechservices.in/**

**Email: info@allytechservices.in**

**Phone: +91-7411011500**

**Office: Ally Tech Services B, 1, Bannerghatta Slip Rd, KEB Colony, New Gurappana Palya, 1st Stage, BTM 1st Stage, Bengaluru, Karnataka 560029, India**

*© Ally Tech Services. All Rights Reserved.*