

Complete Guide for .NET Developers Transitioning to AI Engineering

Core AI & ML Foundations

Fundamental Concepts

- **Artificial Intelligence (AI)** - Broad field of creating intelligent systems
- **Machine Learning (ML)** - Subset of AI that learns from data
- **Deep Learning (DL)** - ML using neural networks with multiple layers
- **Generative AI (GenAI)** - AI that creates new content (text, images, code)

Learning Paradigms

- **Supervised Learning** - Learning with labeled training data
- **Unsupervised Learning** - Finding patterns in unlabeled data
- **Reinforcement Learning** - Learning through rewards and penalties
- **Fine-tuning** - Adapting pre-trained models to specific tasks
- **Transfer Learning** - Leveraging knowledge from one domain to another

Neural Network Architectures

- **Feedforward Neural Networks** - Basic neural network structure
- **Convolutional Neural Networks (CNN)** - For image processing
- **Recurrent Neural Networks (RNN)** - For sequential data
- **Long Short-Term Memory (LSTM)** - Advanced RNN for long sequences
- **Transformers** - Modern architecture powering LLMs
- **Attention Mechanisms** - Core component of transformers

Language AI Technologies

- **Natural Language Processing (NLP)** - Text understanding and generation
- **Large Language Models (LLMs)** - GPT-4, Claude, Gemini, etc.
- **Small Language Models (SLMs)** - Lightweight models for specific tasks
- **Tokenization** - Converting text to numerical tokens
- **Embeddings** - Vector representations of text/data
- **Semantic Search** - Search based on meaning, not keywords
- **Prompt Engineering** - Crafting effective AI prompts
- **Chain-of-Thought Prompting** - Step-by-step reasoning techniques
- **Few-shot Learning** - Learning from minimal examples

Advanced AI Patterns

- **Retrieval-Augmented Generation (RAG)** - Combining retrieval with generation
- **Multi-Modal AI** - Processing text, images, audio together
- **Agent-based AI** - AI systems that can take actions
- **Tool Calling/Function Calling** - AI invoking external functions
- **Multi-Agent Systems** - Coordinating multiple AI agents

Computer Vision Essentials

- **Image Classification** - Categorizing images
- **Object Detection** - Finding and locating objects
- **Optical Character Recognition (OCR)** - Text extraction from images
- **Document AI** - Understanding structured documents
- **Face Recognition** - Identity verification through faces

.NET & Microsoft AI Ecosystem

Azure AI Services

- **Azure OpenAI Service** - Hosted GPT-4, ChatGPT, DALL-E
- **Azure AI Studio** - Unified AI development platform
- **Azure AI Foundry** - Enterprise AI deployment platform
- **Azure Cognitive Services** - Pre-built AI APIs
 - Language Services (Text Analytics, Translator)
 - Speech Services (Speech-to-Text, Text-to-Speech)
 - Vision Services (Computer Vision, Face API)
 - Decision Services (Personalizer, Anomaly Detector)

Microsoft AI Development Tools

- **Semantic Kernel SDK** - AI orchestration framework for .NET
- **Microsoft.Extensions.AI (MEAI)** - Unified AI abstractions for .NET
- **ML.NET** - Machine learning framework for .NET
- **Bot Framework SDK** - Building conversational AI
- **Copilot Studio** - No-code/low-code AI assistant creation
- **Power Platform AI Builder** - Business process automation with AI

Cross-Platform & Integration

- **ONNX Runtime for .NET** - Cross-platform ML inference
- **TensorFlow.NET** - TensorFlow integration for .NET
- **SciSharp Stack** - Scientific computing libraries

- **Hugging Face .NET** - Access to open-source models
- **LangChain.NET** - Building LLM applications

Specialized Services

- **Azure Form Recognizer (Document Intelligence)** - Extract data from documents
- **Azure Translator** - Real-time text translation
- **Azure Speech Services** - Advanced speech processing
- **Azure Video Indexer** - Video content analysis
- **Azure Immersive Reader** - Accessibility-focused reading tools

Integration & Architecture Patterns

Model Hosting & Deployment

- **Embedded Models** - Running models directly in applications
- **API-based Models** - Consuming models through REST APIs
- **Edge Deployment** - Running AI on local devices
- **Hybrid Architecture** - Combining cloud and edge deployment
- **Model Serving Patterns** - A/B testing, canary deployments
- **Container Orchestration** - Docker, Kubernetes for AI services

API Orchestration & Composition

- **Multi-Service Orchestration** - Combining multiple AI services
- **RAG Architecture Patterns** - Document retrieval + generation
- **Multi-Agent Coordination** - Orchestrating multiple AI agents
- **Workflow Automation** - Azure Logic Apps, Durable Functions
- **Event-Driven Architecture** - Processing AI workloads asynchronously
- **MCP** - Provides standardized ways to connect AI models to tools and data sources

Data Engineering for AI

- **Data Pipelines** - ETL processes for AI data
- **Data Preprocessing** - Cleaning and preparing data
- **Data Labeling** - Creating training datasets
- **Azure Data Factory** - Cloud ETL/ELT service
- **Azure Synapse** - Analytics platform
- **Azure Event Hubs** - Real-time data streaming
- **Data Versioning** - Managing datasets over time

Vector Storage & Search

- **Vector Databases** - Pinecone, Weaviate, Qdrant
- **Azure AI Search** - Cognitive search with vector capabilities
- **Redis Vector Similarity** - Vector operations in Redis
- **Embedding Stores** - Storing and retrieving embeddings
- **Hybrid Search** - Combining keyword and vector search

Performance & Scalability

- **Caching Strategies** - Caching AI responses and embeddings
- **Batch Processing** - Processing multiple requests efficiently
- **Load Balancing** - Distributing AI workloads
- **Auto-scaling** - Dynamic resource allocation
- **Rate Limiting** - Managing API usage and costs
- **Connection Pooling** - Efficient API connection management

Security, Compliance & Operations

Authentication & Authorization

- **Azure Active Directory (Entra ID)** - Identity management
- **API Key Management** - Secure key storage and rotation
- **Role-Based Access Control (RBAC)** - Granular permissions
- **Managed Identity** - Passwordless authentication
- **OAuth 2.0 & OpenID Connect** - Standard authentication protocols
- **Zero Trust Architecture** - Security model for AI systems

Data Privacy & Compliance

- **GDPR Compliance** - European data protection regulations
- **HIPAA Compliance** - Healthcare data protection
- **SOC 2 Compliance** - Security and availability standards
- **Data Residency** - Geographic data storage requirements
- **PII Detection & Handling** - Personally identifiable information
- **Data Encryption** - At rest and in transit

Content Safety & Moderation

- **Azure Content Safety** - Detecting harmful content
- **Content Filtering** - Blocking inappropriate outputs
- **Bias Detection** - Identifying unfair AI outputs
- **Toxicity Screening** - Preventing harmful content generation

- **Custom Safety Policies** - Organization-specific content rules

Monitoring & Observability

- **Application Insights** - AI application monitoring
- **Azure Monitor** - Infrastructure monitoring
- **Custom Telemetry** - Tracking AI-specific metrics
- **Performance Metrics** - Latency, throughput, accuracy
- **Usage Analytics** - Understanding AI feature adoption
- **Cost Monitoring** - Tracking AI service expenses
- **Model Performance Tracking** - Monitoring model drift

Strategic & Ethical AI Principles

- **Responsible AI Principles** (Microsoft, OECD, NIST frameworks)
- **Explainability & Transparency**
- **Data Privacy & Compliance** (GDPR, HIPAA, regional laws)
- **AI Governance Frameworks** – MLOps, ModelOps
- **Social Impact & Ethical AI Use Cases**

Artificial Intelligence (AI)

— মানুষের মতো চিন্তা ও শেখার ক্ষমতা সম্পন্ন প্রযুক্তি

আর্টিফিশিয়াল ইন্টেলিজেন্স (AI) বা কৃত্রিম বুদ্ধিমত্তা হলো এমন প্রযুক্তি যা কম্পিউটার বা যন্ত্রকে মানুষের মতো চিন্তা করতে, সিদ্ধান্ত নিতে এবং শেখার ক্ষমতা দেয়।

□ সহজ ভাষায় AI কী?

AI এমন একটি প্রযুক্তি যা মানুষের বুদ্ধিমত্তা অনুকরণ করে। এটি ডেটা থেকে শেখে, প্যাটার্ন চিনে, এবং সেই অনুযায়ী কাজ করে। যেমন:

- মানুষ যেমন ছবি দেখে বিড়াল চিনতে পারে,
- AI-ও হাজার হাজার বিড়ালের ছবি দেখে শিখে ফেলে কোনটা বিড়াল আর কোনটা নয়।

🔗 সহজ উদাহরণ

আপনি Netflix-এ সিনেমা দেখছেন।

Netflix আপনার দেখা সিনেমাগুলোর ধরন বিশ্লেষণ করে এবং বলে:

🗣️ “আপনি অ্যাকশন পছন্দ করেন, তাই এই অ্যাকশন মুভিটি আপনার ভালো লাগতে পারে।”

এখানে Netflix-এর AI আপনার পছন্দ বুঝে আপনাকে সাজেশন দিচ্ছে—একদম মানুষের মতো চিন্তা করে।

★ সংক্ষেপে:

📌 সংক্ষেপে:	
বিষয়	ব্যাখ্যা
সংজ্ঞা	মানুষের মতো চিন্তা ও শেখার ক্ষমতা সম্পন্ন প্রযুক্তি
কাজ	ডেটা বিশ্লেষণ, সিদ্ধান্ত গ্রহণ, প্যাটার্ন চিনে কাজ করা
উদাহরণ	Netflix সাজেশন, Google Translate, Chatbot, Self-driving car

Machine Learning (ML)

— মেশিন শেখে ডেটা থেকে

সংজ্ঞা:

Machine Learning হলো এমন একটি প্রযুক্তি যেখানে কম্পিউটার নিজে নিজে ডেটা থেকে শেখে এবং ভবিষ্যতের জন্য সিদ্ধান্ত নিতে পারে।

উদাহরণ:

আপনি যদি হাজারটা আমের ছবি দেখান, ML নিজে বুঝে ফেলবে কোনটা পাকা, কোনটা কাঁচা।

☞ যেমন: **বিকাশ** বা **নগদ** অ্যাপে সন্দেহজনক লেনদেন ধরার জন্য ML ব্যবহার হয়।

প্রধান ধরণ:

- Supervised Learning (লেবেলসহ ডেটা)
- Unsupervised Learning (লেবেল ছাড়া)
- Reinforcement Learning (ট্রায়াল-এন্ড-এরর শেখা)

Deep Learning (DL)

— মানুষের মস্তিষ্ক অনুকরণ

সংজ্ঞা:

Deep Learning হলো ML-এর একটি শাখা, যেখানে **Neural Network** ব্যবহার করে জটিল সমস্যার সমাধান করা হয়।

উদাহরণ:

আপনার ফোনে **Face Unlock**—এটি DL ব্যবহার করে মুখ চিনে ফেলে।

☞ যেমন: **YouTube** আপনার পছন্দের ভিডিও সাজেস্ট করে DL-এর মাধ্যমে।

মূল বৈশিষ্ট্য:

- অনেকগুলো লেয়ার থাকে (Input → Hidden → Output)
- GPU ব্যবহার করে বিশাল ডেটা প্রসেস করে
- Image, Voice, Text—সবকিছু বুঝতে পারে

Deep Learning Process — সংক্ষিপ্ত ধাপে

❶ Data Preparation

আপনি কী শেখাতে চান, সেই ডেটা প্রস্তুত করুন

- CSV, JSON, Image, Text—যেকোনো format
- .NET-এ `System.Text.Json`, `ML.NET`, বা Azure Blob Storage ব্যবহার করে ডেটা লোড করা যায়
- Example: SDG রিপোর্ট থেকে poverty score prediction

❷ Model Selection

কোন pre-trained model বা architecture ব্যবহার করবেন তা ঠিক করুন

- Image → CNN
- Text → BERT, GPT
- Time Series → LSTM
- .NET-এ ONNX format model ব্যবহার করে ASP.NET Core API-তে deploy করা যায়

❸ Training (optional for .NET dev)

মডেলকে ডেটা দিয়ে শেখানো হয়

- Backpropagation + Gradient Descent
- Dropout, Batch Normalization ইত্যাদি regularization
- .NET dev হিসেবে আপনি Azure AutoML বা Hugging Face API ব্যবহার করে training skip করতে পারেন

❹ Evaluation

মডেল কতটা ভালো কাজ করছে তা যাচাই করুন

- Accuracy, Precision, Recall, F1-score
- Azure ML Studio বা ML.NET দিয়ে validation করা যায়
- Example: SDG classifier কতটা সঠিকভাবে “Health” vs “Education” ট্যাগ করছে

❺ Inference

নতুন ডেটা দিয়ে prediction বা output তৈরি করুন

- .NET-এ `HttpClient` দিয়ে Hugging Face API কল
- Azure Function দিয়ে real-time inference

- Example: UNDP dashboard-এ live poverty score prediction

⑥ Deployment

মডেলকে production-ready API বা UI-তে যুক্ত করুন

- ASP.NET Core Web API
- Azure App Service বা Azure Container Instance
- Blazor UI বা Angular SPA থেকে API consume

✓ Summary (এক লাইনে)

ডেটা → মডেল নির্বাচন → (training) → মূল্যায়ন → inference → deployment

Natural Language Processing (NLP)

— ভাষা বোঝার ক্ষমতা

সংজ্ঞা:

NLP হলো AI-এর এমন একটি অংশ যা মানুষের ভাষা বুঝতে, বিশ্লেষণ করতে এবং উত্তর দিতে পারে।

উদাহরণ:

☞ আপনি যদি বলেন “আজকের আবহাওয়া কেমন?”—Google Assistant বা Siri সেটা বুঝে উত্তর দেয়।

☞ বাংলা চ্যাটবট বা অনুবাদ অ্যাপ NLP ব্যবহার করে।

মূল টেকনিক:

- Tokenization, Lemmatization
- Named Entity Recognition (NER)
- Sentiment Analysis
- Machine Translation

🔧 NLP Process — সংক্ষিপ্ত ধাপে

① Text Input

ব্যবহারকারীর ভাষাভিত্তিক ইনপুট সংগ্রহ করুন

- Text, Sentence, Document

- .NET-এ `HttpClient`, `System.Text.Json`, বা Blazor Form থেকে ইনপুট নেওয়া যায়
- Example: SDG রিপোর্টের সারাংশ তৈরি করতে paragraph ইনপুট

❷ Text Preprocessing

ইনপুট টেক্সটকে বিশ্লেষণের জন্য প্রস্তুত করুন

- Tokenization → শব্দ ভাগ করা
- Stopword Removal → “the”, “is” ইত্যাদি বাদ
- Lemmatization/Stemming → শব্দের মূল রূপে রূপান্তর
- .NET-এ `Microsoft.ML.Transforms.Text` বা Python API call করে করা যায়

❸ Model Selection

কোন pre-trained NLP model ব্যবহার করবেন তা ঠিক করুন

- Sentiment → BERT, DistilBERT
- Summarization → BART, T5
- Chat → GPT
- .NET-এ Hugging Face Inference API বা Azure OpenAI API ব্যবহার করে সহজে যুক্ত করা যায়

❹ Inference

মডেল দিয়ে টেক্সট বিশ্লেষণ বা প্রেডিকশন করুন

- Text Classification, Named Entity Recognition, Translation
- .NET-এ `HttpClient` দিয়ে API কল করে JSON result পাওয়া যায়
- Example: “বাংলাদেশে দারিদ্র্য হ্রাস” → SDG category: “Poverty”

❺ Postprocessing

মডেলের আউটপুটকে ব্যবহারযোগ্য করে তুলুন

- JSON → Text, Table, Tag
- Blazor বা Razor View-এ result দেখানো
- Example: সারাংশ, ট্যাগ, বা recommendation UI-তে দেখানো

❻ Deployment

NLP ফিচারকে production-ready অ্যাপে যুক্ত করুন

- ASP.NET Core Web API

- Azure App Service বা Azure Function
- Blazor SPA বা Angular UI থেকে consume

✓ Summary (এক লাইনে)

Text → Preprocessing → Model → Inference → Postprocessing → Deployment

Generative AI (GenAI)

— নতুন কিছু তৈরি করার ক্ষমতা

সংজ্ঞা:

Generative AI এমন AI যা নতুন ছবি, লেখা, কোড, বা মিউজিক তৈরি করতে পারে—পুরোপুরি নতুন কনটেন্ট।

উদাহরণ:

- ☞ আপনি যদি বলেন “একটা গ্রামে সূর্যাস্তের ছবি আঁকো”—DALL·E সেটা তৈরি করে।
- ☞ ChatGPT আপনার জন্য ব্লগ লিখে দিতে পারে।

প্রধান প্রযুক্তি:

- Large Language Models (LLM) যেমন GPT, PaLM
- Generative Adversarial Networks (GAN)
- Text-to-Image, Text-to-Code, Text-to-Music

☐ Generative AI Process — সংক্ষিপ্ত ধাপে

① Prompt Input

ব্যবহারকারীর কাছ থেকে নির্দেশনা বা প্রশ্ন সংগ্রহ করুন

- Text prompt: “বাংলাদেশে দারিদ্র্য হ্রাস নিয়ে একটি সারাংশ লিখো”
- .NET-এ Blazor Form, Razor Page, বা API endpoint থেকে ইনপুট নেওয়া যায়

② Model Selection

কোন pre-trained GEN AI model ব্যবহার করবেন তা ঠিক করুন

- Text → GPT, Claude, Mistral

- Image → DALL·E, Stable Diffusion
- Code → Codex, Codestral
- .NET-এ Azure OpenAI API বা Hugging Face Inference API ব্যবহার করে সহজে যুক্ত করা যায়

③ Prompt Engineering

Prompt-কে এমনভাবে সাজান যাতে মডেল সঠিকভাবে উত্তর দিতে পারে

- Clear instruction + context
- Example: “Write a 3-line summary in Bangla about SDG 1 (No Poverty)”
- .NET-এ `StringBuilder` বা `PromptTemplate` ব্যবহার করে dynamic prompt তৈরি করা যায়

④ Inference

মডেল থেকে output সংগ্রহ করুন

- Text generation, summarization, translation, code suggestion
- .NET-এ `HttpClient` দিয়ে API কল করে JSON result পাওয়া যায়
- Example: GPT থেকে SDG summary নিয়ে Blazor UI-তে দেখানো

⑤ Postprocessing

মডেলের আউটপুটকে ব্যবহারযোগ্য করে তুলুন

- JSON → Text, HTML, Markdown
- .NET-এ Razor View বা Blazor Component-এ bind করা যায়
- Example: “বাংলাদেশে দারিদ্র্য হ্রাসের ৩টি কারণ” → bullet list আকারে দেখানো

⑥ Deployment

GEN AI ফিচারকে production-ready অ্যাপে যুক্ত করুন

- ASP.NET Core Web API
- Azure App Service, Azure Function
- Blazor SPA বা Angular UI থেকে consume

✓ Summary (এক লাইনে)

Prompt → Model → Engineering → Inference → Postprocessing → Deployment

Supervised Learning

— শিক্ষক সহ শেখা

সংজ্ঞা:

এখানে মডেলকে এমন ডেটা দিয়ে শেখানো হয় যেখানে ইনপুট ও সঠিক আউটপুট (লেবেল) আগে থেকেই জানা থাকে।

উদাহরণ:

ইমেইল স্প্যাম ডিটেকশন—ইমেইলগুলো “স্প্যাম” বা “নন-স্প্যাম” হিসেবে লেবেল করা থাকে, মডেল সেই অনুযায়ী শেখে।

মূল ব্যবহার:

- ক্লাসিফিকেশন (যেমন: রোগ নির্ণয়)
- রিগ্রেশন (যেমন: বাড়ির দাম প্রেডিকশন)

Unsupervised Learning

— শিক্ষক ছাড়া শেখা

সংজ্ঞা:

এখানে ডেটা লেবেল ছাড়া থাকে। মডেল নিজে নিজে প্যাটার্ন বা গ্রুপ খুঁজে বের করে।

উদাহরণ:

একটি শপিং মল কাস্টমারদের কেনাকাটার ডেটা বিশ্লেষণ করে গ্রুপ করে—কে বেশি খরচ করে, কে কম।

মূল টেকনিক:

- Clustering (যেমন: K-Means)
- Dimensionality Reduction (যেমন: PCA)

Reinforcement Learning

— পুরস্কার ভিত্তিক শেখা

সংজ্ঞা:

এখানে একটি “Agent” পরিবেশে কাজ করে এবং প্রতিটি কাজের জন্য পুরস্কার বা শাস্তি পায়। এই অভিজ্ঞতা থেকে সে শেখে।

উদাহরণ:

একটি রোবট যদি সঠিক পথে চলে, সে পুরস্কার পায়; ভুল পথে গেলে শাস্তি। ধীরে ধীরে সে সঠিক পথ শিখে ফেলে।

মূল উপাদান:

- Agent, Environment, Reward, Policy, Value Function

🎮 Reinforcement Learning Process — সংক্ষিপ্ত ধাপে

❶ Environment Setup

Agent কোন পরিবেশে কাজ করবে তা নির্ধারণ করুন

- Game, Simulation, Workflow, বা SDG Policy Engine
- .NET-এ আপনি Unity ML-Agents, Azure Simulation API, বা custom C# logic দিয়ে environment তৈরি করতে পারেন
- Example: SDG recommendation engine যেখানে agent নীতিগত পদক্ষেপ নিয়ে reward পায়

❷ Agent Initialization

Agent কে এমনভাবে তৈরি করুন যাতে সে সিদ্ধান্ত নিতে পারে

- State, Action, Policy, Reward
- .NET-এ আপনি class structure দিয়ে agent behavior define করতে পারেন
- Example: `class SDGAgent { public Action Decide(State s) { ... } }`

❸ Reward Function

Agent কী করলে পুরস্কার (reward) পাবে তা নির্ধারণ করুন

- Positive reward → সঠিক সিদ্ধান্ত
- Negative reward → ভুল সিদ্ধান্ত

- .NET-এ আপনি `RewardCalculator` class দিয়ে reward logic সাজাতে পারেন
- Example: “দারিদ্র্য হ্রাস” হলে +10, “বাজেট অপচয়” হলে -5

④ Policy Learning

Agent কীভাবে সিদ্ধান্ত নেবে তা শেখানো হয়

- Q-learning, Deep Q-Networks (DQN), Policy Gradient
- .NET dev হিসেবে আপনি pre-trained RL model ব্যবহার করতে পারেন—training না করেও Azure API বা ONNX model deploy করা যায়
- Example: Hugging Face থেকে SDG policy agent model নিয়ে ASP.NET API-তে ব্যবহার

⑤ Exploration vs Exploitation

Agent নতুন কিছু চেষ্টা করবে নাকি পুরনো অভিজ্ঞতা ব্যবহার করবে?

- ϵ -Greedy strategy
- .NET-এ আপনি config বা runtime parameter দিয়ে exploration ratio নিয়ন্ত্রণ করতে পারেন
- Example: নতুন SDG policy test করার জন্য ২০% random action

⑥ Deployment

RL logic কে production-ready অ্যাপে যুক্ত করুন

- ASP.NET Core Web API
- Azure Container Instance বা Azure ML Endpoint
- Blazor SPA থেকে SDG recommendation UI-তে RL agent যুক্ত করা যায়

✓ Summary (এক লাইনে)

Environment → Agent → Reward → Policy → Decision → Deployment

Fine-Tuning

— পূর্ব শেখা মডেলকে নতুন কাজে মানিয়ে নেওয়া

সংজ্ঞা:

একটি প্রি-ট্রেন্ড মডেলকে নতুন, নির্দিষ্ট কাজে ব্যবহারযোগ্য করে তোলা—কম ডেটা ও কম রিসোর্সে।

উদাহরণ:

GPT মডেলকে বাংলা প্রশ্নোত্তরের জন্য ফাইন-টিউন করা, যাতে সে স্থানীয় প্রসঙ্গ বুঝতে পারে।

মূল ধাপ:

- প্রি-ট্রেন্ড মডেল নির্বাচন
- কিছু লেয়ার ফ্রিজ করা
- নতুন ডেটা দিয়ে ট্রেন্ড করা

✂ Fine-Tuning Process — সংক্ষিপ্ত ধাপে

❶ Pre-trained Model নির্বাচন

যে মডেলটি আগে থেকেই বিশাল ডেটায় ট্রেন্ড করা হয়েছে, সেটি বেছে নিন

- GPT, BERT, T5, Whisper ইত্যাদি
- .NET-এ আপনি Azure OpenAI বা Hugging Face API দিয়ে pre-trained model access করতে পারেন
- Example: GPT-3.5 কে SDG-specific প্রশ্নের উত্তর দিতে ফাইন-টিউন করা

❷ Custom Dataset প্রস্তুত

নিজের ডেটা তৈরি করুন যাতে মডেল আপনার প্রসঙ্গ বুঝতে পারে

- Format: JSONL, CSV, YAML
- Structure: prompt → completion
- .NET-এ `System.Text.Json` বা Azure Blob Storage দিয়ে ডেটা প্রস্তুত করা যায়
- Example: “প্রশ্ন: দারিদ্র্য হ্রাসের উপায়?” → “উত্তর: কর্মসংস্থান, শিক্ষা, সামাজিক নিরাপত্তা”

❸ Training Configuration

কতবার শেখাবে, কীভাবে শেখাবে তা নির্ধারণ করুন

- Epochs, Batch Size, Learning Rate
- .NET dev হিসেবে আপনি Azure ML Studio বা Hugging Face CLI দিয়ে config করতে পারেন
- Example: 3 epochs, 0.001 learning rate, 1000 sample

❹ Model Training

মডেলকে আপনার ডেটা দিয়ে পুনরায় শেখানো হয়

- Transfer Learning + Backpropagation
- .NET dev হিসেবে আপনি training নিজে না করে API বা AutoML ব্যবহার করতে পারেন
- Example: Azure OpenAI-এর /fine-tune endpoint-এ dataset upload করে training trigger

⑤ Evaluation

ফাইন-টিউনড মডেল কতটা ভালো কাজ করছে তা যাচাই করুন

- Accuracy, BLEU Score, Confusion Matrix
- .NET-এ আপনি API response log করে validation করতে পারেন
- Example: SDG প্রশ্নের উত্তর কতটা প্রাসঙ্গিক তা UI-তে দেখানো

⑥ Deployment

ফাইন-টিউনড মডেলকে production-ready অ্যাপে যুক্ত করুন

- ASP.NET Core Web API
- Azure App Service, Azure Function
- Blazor SPA বা Angular UI থেকে consume

✓ Summary (এক লাইনে)

Pre-trained Model → Custom Dataset → Training → Evaluation → Deployment

Transfer Learning

— পুরাতন শেখা নতুন কাজে ব্যবহার

সংজ্ঞা:

একটি টাস্কে শেখা জ্ঞানকে অন্য, সম্পর্কিত টাস্কে ব্যবহার করা—যেমন গিটার বাজানো শিখে ইউকুলেলে বাজানো সহজ হয়।

উদাহরণ:

ImageNet-এ ট্রেন করা মডেলকে মেডিকেল ইমেজ ক্লাসিফিকেশনে ব্যবহার করা।

লাভ:

- কম ডেটা লাগে
- দ্রুত ফলাফল পাওয়া যায়
- কম খরচে ভালো পারফরম্যান্স

Feedforward Neural Networks (FNN)

— একমুখী প্রবাহ

সংজ্ঞা:

এটি সবচেয়ে সাধারণ Neural Network যেখানে তথ্য ইনপুট থেকে আউটপুট পর্যন্ত একমুখীভাবে প্রবাহিত হয়—কোনো লুপ বা পুনরাবৃত্তি নেই।

উদাহরণ:

একটি ব্যাংক যদি ক্রেডিট স্কোর নির্ধারণ করে ব্যবহারকারীর ইনকাম, খরচ, ও ইতিহাস দেখে—FNN সেই তথ্য বিশ্লেষণ করে স্কোর দেয়।

মূল বৈশিষ্ট্য:

- Input → Hidden → Output লেয়ার
- Backpropagation দিয়ে শেখে
- Classification ও Regression কাজে ব্যবহৃত

Convolutional Neural Networks (CNN)

— ছবি ও প্যাটার্ন চিনতে দক্ষ

সংজ্ঞা:

CNN হলো এমন একটি Neural Network যা ছবির মতো গ্রিড-ভিত্তিক ডেটা থেকে ফিচার বের করতে পারে—যেমন চোখ, নাক, বা টেক্সচার।

উদাহরণ:

আপনার ফোনের Face Unlock বা Google Photos-এ মুখ চিনে ফেলা—এগুলো CNN ব্যবহার করে।

মূল উপাদান:

- Convolutional Layer (ফিল্টার দিয়ে ফিচার বের করে)
- Pooling Layer (ডেটা কম্প্রেস করে)
- Fully Connected Layer (চূড়ান্ত সিদ্ধান্ত)

Recurrent Neural Networks (RNN)

— ধারাবাহিক তথ্যের জন্য

সংজ্ঞা:

RNN এমন একটি Network যা পূর্ববর্তী ইনপুট মনে রাখতে পারে এবং বর্তমান সিদ্ধান্তে তা ব্যবহার করে।

উদাহরণ:

আপনি যদি টাইপ করেন “আমি বাংলাদেশ...”—RNN ভবিষ্যদ্বাণী করতে পারে “থেকে এসেছি”।

মূল বৈশিষ্ট্য:

- Hidden State থাকে
- Sequence Data যেমন টেক্সট, অডিও, টাইম সিরিজে ব্যবহৃত
- Backpropagation Through Time (BPTT) ব্যবহার করে শেখে

Long Short-Term Memory (LSTM)

— দীর্ঘমেয়াদি স্মৃতির জন্য

সংজ্ঞা:

LSTM হলো RNN-এর উন্নত সংস্করণ, যা দীর্ঘ সময়ের তথ্য মনে রাখতে পারে—Forget Gate, Input Gate, Output Gate ব্যবহার করে।

উদাহরণ:

ভাষা অনুবাদে “আমি ইতালি থেকে এসেছি”—LSTM বুঝতে পারে “ইতালি” ও “ইতালিয়ান” এর সম্পর্ক।

মূল বৈশিষ্ট্য:

- Cell State ও Hidden State
- Vanishing Gradient সমস্যা সমাধান করে
- Speech, Translation, Time Series-এ ব্যবহৃত

Transformers

— একসাথে পুরো তথ্য বিশ্লেষণ

সংজ্ঞা:

Transformers এমন একটি Network যা পুরো ইনপুট একসাথে বিশ্লেষণ করে—Self-Attention ব্যবহার করে কোন অংশ গুরুত্বপূর্ণ তা নির্ধারণ করে।

উদাহরণ:

ChatGPT বা Google Translate—Transformers ব্যবহার করে পুরো বাক্য একসাথে বিশ্লেষণ করে।

মূল উপাদান:

- Encoder-Decoder Architecture
- Multi-Head Attention
- Positional Encoding

Attention Mechanism

— কোন অংশে বেশি মনোযোগ

সংজ্ঞা:

Attention Mechanism মডেলকে শেখায় কোন অংশ বেশি গুরুত্বপূর্ণ—যেমন বাক্যের কোন শব্দটি পরবর্তী শব্দ নির্ধারণে বেশি প্রভাব ফেলবে।

উদাহরণ:

“সে বইটি পড়ছে”—Attention বুঝতে পারে “পড়ছে” শব্দটি “বই”-এর সাথে বেশি সম্পর্কিত।

মূল ধাপ:

- Query, Key, Value
- Similarity Calculation
- Softmax দিয়ে Attention Weight নির্ধারণ

Large Language Models (LLMs)

— বিশাল ভাষা শেখা মডেল

❶ সংজ্ঞা

LLM হলো এমন একটি **Transformer-based neural network** যা **বিলিয়ন বা ট্রিলিয়ন প্যারামিটার** নিয়ে ভাষা বুঝতে ও তৈরি করতে পারে।

এটি টোকেনের সম্ভাব্যতা গণনা করে—যেমন: “বাংলাদেশে দারিদ্র্য হ্রাসের জন্য ____” → “শিক্ষা”, “কর্মসংস্থান” ইত্যাদি।

❷ জনপ্রিয় LLM উদাহরণ

Model	কাজ	.NET-এ ব্যবহার
GPT-4	Text generation, reasoning	Azure OpenAI API
BERT	Text classification, NER	Hugging Face API
BART	Summarization, translation	ONNX via ML.NET
Mistral	Lightweight, open-source	LangChain.NET integration
Claude	Human-aligned responses	API via HTTP client

❏ [GeeksforGeeks-এর LLM তালিকা](#)

❏ [Google Developers-এর LLM পরিচিতি](#)

③ LLM কীভাবে কাজ করে

- **Self-attention:** প্রতিটি শব্দের প্রসঙ্গ বুঝে
- **Tokenization:** টেক্সটকে ছোট অংশে ভাগ করে
- **Probability Prediction:** পরবর্তী শব্দের সম্ভাবনা গণনা
- .NET-এ আপনি API call করে prompt পাঠান → JSON আউটপুট পান

④ .NET ভিত্তিক বাস্তব প্রয়োগ

- SDG Recommendation Engine → GPT-4 দিয়ে policy suggestion
- Bengali Text Classifier → BERT দিয়ে SDG tagging
- Chatbot → Azure OpenAI API দিয়ে Blazor UI-তে যুক্ত
- Summarizer → BART model Hugging Face থেকে call করে ASP.NET API-তে deploy

⑤ Deployment Flow (.NET dev perspective)

Prompt → API Call → JSON Result → Razor/Blazor UI

- `HttpClient` দিয়ে API call
- `System.Text.Json` দিয়ে response parse
- Azure App Service বা Azure Function দিয়ে host
- Blazor SPA বা Angular UI থেকে consume

🔗 বিস্তারিত: [LLM vs NLP Explained](#), [Google's LLM Guide](#)

Small Language Models (SLMs)

— ছোট কিন্তু দক্ষ

সংজ্ঞা:

SLM হলো ছোট আকারের ভাষা মডেল যা কম রিসোর্সে কাজ করে, যেমন মোবাইল, IoT, বা লোকাল সার্ভারে।

উদাহরণ:

- **Phi-3 Mini:** Microsoft-এর ছোট মডেল
- **Gemma:** Google-এর লাইটওয়েট মডেল
- **Bayer's Crop Assistant:** কৃষকদের প্রশ্নের উত্তর দেয়

লাভ:

- দ্রুত ইনফারেন্স
- কম খরচে ট্রেনিং
- নির্দিষ্ট কাজে ফাইন-টিউন করা সহজ

☞ বিস্তারিত: [SLM Explained with Examples](#), [HBR on SLMs](#)

❶ সংজ্ঞা

SLM হলো এমন ভাষাভিত্তিক মডেল যার **প্যারামিটার সংখ্যা কম** (সাধারণত ১M–১০B এর মধ্যে), এবং যা **কম রিসোর্সে দ্রুত কাজ করতে পারে**।

এটি LLM-এর তুলনায় হালকা, দ্রুত, এবং সহজে কাস্টমাইজযোগ্য।

❷ SLM vs LLM

বৈশিষ্ট্য	LLM (GPT-4, Claude)	SLM (Phi-3, SmoLM)
Parameters	100B+	1B–10B
Speed	তুলনামূলক ধীর	দ্রুত inference
Resource Need	GPU-heavy	CPU বা edge device-এ চলে
Customization	কঠিন ও ব্যয়বহুল	সহজ ও সাশ্রয়ী
Deployment	Cloud preferred	Local বা on-premise সম্ভব

☐ বিস্তারিত: [DataCamp-এর SLM গাইড](#)

❸ SLM কীভাবে তৈরি হয়

- **Knowledge Distillation:** বড় মডেল থেকে ছোট মডেল শেখে
- **Pruning:** অপ্রয়োজনীয় নোড বাদ দেওয়া
- **Quantization:** কম precision ব্যবহার করে computation হালকা করা
- .NET-এ ONNX format ব্যবহার করে SLM deploy করা যায় Azure App Service-এ

❹ জনপ্রিয় SLM উদাহরণ

Model	Parameters	ব্যবহার
Phi-3-Mini	3.8B	Microsoft-এর multilingual SLM
Llama3.2-1B	1B	Edge device-এ deploy উপযোগী
Qwen2.5-1.5B	1.5B	Multilingual chatbot
SmolLM2-1.7B	1.7B	Stack-Edu, SmolTalk dataset
DeepSeek-R1-1.5B	1.5B	Reasoning-focused SLM

❏ বিস্তারিত: [Hugging Face-এর SLM বিশ্লেষণ](#)

🔗 .NET ভিত্তিক বাস্তব প্রয়োগ

- SDG Chatbot → Phi-3-Mini API call করে ASP.NET Core Web API
- Bengali Summarizer → SmolLM2 Hugging Face থেকে call করে Razor UI-তে দেখানো
- Offline Deployment → ONNX format SLM Blazor Desktop App-এ যুক্ত
- Edge AI → Llama3.2-1B Raspberry Pi বা IoT device-এ deploy

✓ Summary (এক লাইনে)

Lightweight model → Faster inference → Easy customization → .NET API integration

Tokenization

— টেক্সট ভেঙে ছোট অংশে ভাগ করা

সংজ্ঞা:

Tokenization হলো টেক্সটকে ছোট ছোট অংশে ভাগ করা—যেমন শব্দ, অক্ষর, বা সাব-ওয়ার্ড।

উদাহরণ:

“আমি বাংলাদেশে থাকি” → ["আমি", "বাংলাদেশে", "থাকি"]

বা → ["আ", "মি", " ", "বা", "ং", "লা", ...]

ধরন:

- Word Tokenization
- Character Tokenization
- Subword Tokenization (BPE, WordPiece)

☞ বিস্তারিত: [Tokenization Explained](#), [GeeksforGeeks Tokenization](#)

❶ সংজ্ঞা

Tokenization হলো টেক্সটকে ছোট ছোট অংশে ভাগ করার প্রক্রিয়া, যাতে মেশিন সহজে বিশ্লেষণ করতে পারে।

এই অংশগুলোকে বলা হয় **token**—যেমন শব্দ, অক্ষর, বা sub-word।

❷ Tokenization কেন দরকার

- মডেল raw text বুঝতে পারে না
- Tokenization ছাড়া NLP pipeline শুরুই করা যায় না
- এটি text classification, translation, summarization-এর ভিত্তি

☐ বিস্তারিত: [GeeksforGeeks-এর Tokenization গাইড](#)

❸ Tokenization-এর ধরন

ধরন	উদাহরণ	ব্যবহার
Word Tokenization	"AI is powerful" → ["AI", "is", "powerful"]	সাধারণ NLP task
Character Tokenization	"AI" → ["A", "I"]	Spell correction, low-resource language
Sub-word Tokenization	"unhappiness" → ["un", "happi", "ness"]	Rare word handling, BPE
Sentence Tokenization	Paragraph → ["Sentence 1", "Sentence 2"]	Summarization, translation
Byte-level Tokenization	Text → byte chunks	GPT-style model input

❹ .NET ভিত্তিক বাস্তব প্রয়োগ

- `Microsoft.ML.Transforms.Text.TokenizeIntoWords()` → ML.NET pipeline-এ
- Azure OpenAI API → Token count limit বুঝতে
- Hugging Face API → Tokenized input পাঠিয়ে GPT বা BERT model call
- Example: SDG রিপোর্টের paragraph → tokenized → classification

⑤ Deployment Flow (.NET dev perspective)

Raw Text → Tokenization → Model Input → Inference → UI Display

- Blazor বা Razor Page থেকে ইনপুট
- Tokenization via API বা local ML.NET
- JSON result Razor View-এ bind

✓ Summary (এক লাইনে)

Tokenization হলো NLP-এর প্রথম ধাপ, যা টেক্সটকে token-এ ভাগ করে মেশিনের জন্য অর্থবোধক করে তোলে

Embeddings

— শব্দকে সংখ্যায় রূপান্তর

সংজ্ঞা:

Embeddings হলো এমন ভেক্টর যা শব্দের অর্থ ও সম্পর্ককে সংখ্যায় প্রকাশ করে।

উদাহরণ:

“রাজা” ও “রানী” → কাছাকাছি ভেক্টর

“বিড়াল” ও “কুকুর” → কাছাকাছি ভেক্টর

“বই” ও “পাখি” → দূরের ভেক্টর

প্রযুক্তি:

- Word2Vec, GloVe
- BERT Embeddings
- Sentence Embeddings

🔗 বিস্তারিত: [Word Embeddings Explained](#), [How Embeddings Work](#)

❶ সংজ্ঞা

Embedding হলো **টেক্সটের সংখ্যাগত রূপ**—যেখানে প্রতিটি শব্দ, বাক্য, বা ডকুমেন্টকে একটি **vector** আকারে প্রকাশ করা হয়।

এই vector semantic অর্থ বহন করে—যেমন “রাজা” এবং “রানী” embedding space-এ কাছাকাছি থাকে।

❏ বিস্তারিত: [GeeksforGeeks-এর Embedding গাইড](#)

❷ Embedding কেন দরকার

- Raw text → মেশিন বুঝতে পারে না
- Embedding → semantic অর্থ বোঝে
- Similar meaning → similar vector
- .NET-এ Embedding API ব্যবহার করে similarity, classification, বা clustering করা যায়

❸ Embedding-এর ধরন

ধরন	উদাহরণ	ব্যবহার
Word Embedding	Word2Vec, GloVe	SDG tagging, sentiment
Contextual Embedding	BERT, GPT	Chatbot, summarization
Sentence Embedding	Sentence-BERT, T5	Semantic search
Document Embedding	Doc2Vec, TF-IDF	SDG report classification
Image Embedding	CLIP, DINO	Vision-Language tasks

❹ .NET ভিত্তিক বাস্তব প্রয়োগ

- Azure OpenAI /embeddings endpoint → Bengali SDG sentence → vector
- Hugging Face API → BERT embedding → cosine similarity
- ML.NET → TF-IDF vectorizer → classification pipeline
- Example: “বাংলাদেশে দারিদ্র্য হ্রাস” → embedding → SDG category: Poverty

❺ Deployment Flow (.NET dev perspective)

Text → Tokenization → Embedding → Model Input → Inference → UI Display

- Blazor বা Razor Page থেকে ইনপুট

- API call দিয়ে embedding vector সংগ্রহ
- Razor View-এ similarity বা classification result দেখানো

✓ Summary (এক লাইনে)

Embedding হলো টেক্সটের সংখ্যাগত রূপ, যা semantic অর্থ বহন করে এবং .NET অ্যাপে NLP ফিচার যুক্ত করতে সাহায্য করে

Semantic Search

— অর্থ বুঝে খোঁজ করা

সংজ্ঞা:

Semantic Search হলো এমন সার্চ প্রযুক্তি যা শুধু শব্দ নয়, তার অর্থ ও প্রসঙ্গ বুঝে ফলাফল দেয়।

উদাহরণ:

আপনি লিখলেন: “ভালো ল্যাপটপ ডিজাইন স্টুডেন্টদের জন্য”

→ সার্চ ইঞ্জিন বুঝবে আপনি গ্রাফিক্স, RAM, ডিসপ্লে নিয়ে ভাবছেন।

প্রযুক্তি:

- Embeddings
- Vector Search
- Knowledge Graph

☞ বিস্তারিত: [Google Cloud Semantic Search](#), [Elastic Semantic Guide](#)

❶ সংজ্ঞা

Semantic Search হলো এমন একটি তথ্য অনুসন্ধান পদ্ধতি, যেখানে keyword match নয়, বরং query-এর অর্থ ও প্রসঙ্গ বোঝা হয়।

এটি Natural Language Processing (NLP) এবং Embedding ব্যবহার করে intent অনুযায়ী relevant result দেয়।

❷ Traditional vs Semantic Search

বৈশিষ্ট্য	Traditional Search	Semantic Search
Keyword Matching	"poverty reduction" → exact match	"reduce poverty" → similar meaning match
Context Handling	নেই	আছে (intent + meaning বোঝে)
Personalization	সাধারণ	ইউজার প্রসঙ্গ অনুযায়ী ফলাফল
Relevance Logic	frequency-based	semantic similarity-based

❏ বিস্তারিত: [GeeksforGeeks-এর Semantic Search গাইড](#)

❸ Semantic Search কীভাবে কাজ করে

1. **Query Understanding** → NLP দিয়ে query-এর intent ও entity বোঝা
2. **Embedding Generation** → query ও document কে vector-এ রূপান্তর
3. **Similarity Matching** → cosine similarity বা dot product দিয়ে মিল খোঁজা
4. **Ranking** → semantic relevance অনুযায়ী result সাজানো

❹ .NET ভিত্তিক বাস্তব প্রয়োগ

- Azure Cognitive Search → semantic configuration + embedding index
- Hugging Face API → BERT embedding → cosine similarity
- LangChain.NET → SDG document থেকে semantic answer retrieval
- Example: "বাংলাদেশে দারিদ্র্য হ্রাসের উপায়" → SDG policy docs থেকে semantic match

❺ Deployment Flow (.NET dev perspective)

Query → Embedding → Similarity Match → Ranked Result → Razor/Blazor UI

- Blazor বা Razor Page থেকে query ইনপুট
- API call দিয়ে embedding ও similarity match
- Razor View-এ ranked result দেখানো

✓ Summary (এক লাইনে)

Semantic Search keyword নয়, query-এর অর্থ ও প্রসঙ্গ বুঝে relevant result দেয়—.NET অ্যাপে NLP ও embedding API দিয়ে সহজে যুক্ত করা যায়

Prompt Engineering

— AI-কে সঠিকভাবে নির্দেশ দেওয়া

সংজ্ঞা:

Prompt Engineering হলো এমন কৌশল যেখানে আপনি AI-কে কীভাবে প্রশ্ন করবেন, সেটা ঠিক করে ফলাফল নিয়ন্ত্রণ করেন।

উদাহরণ:

✗ “বাংলাদেশ সম্পর্কে বলো”

✓ “বাংলাদেশের ইতিহাস, সংস্কৃতি ও অর্থনীতি সম্পর্কে ১০ লাইনের সারাংশ দাও”

ব্যবহার:

- কনটেন্ট জেনারেশন
- কোডিং সহায়তা
- কাস্টমার সার্ভিস
- এডুকেশন

☞ বিস্তারিত: [Prompt Engineering Explained, Coursera Guide](#)

□ Prompt Engineering — সংক্ষিপ্তভাবে

❶ সংজ্ঞা

Prompt Engineering হলো AI model-এর জন্য সঠিক নির্দেশনা তৈরি করার কৌশল, যাতে output হয় প্রাসঙ্গিক, নির্ভুল, এবং context-aware।

এটি LLM-এর সাথে যোগাযোগের gateway—যেখানে আপনি কীভাবে প্রশ্ন করবেন, সেটাই নির্ধারণ করে AI কীভাবে উত্তর দেবে।

□ বিস্তারিত: [Couchbase-এর Prompt Engineering গাইড](#)

❷ Prompt Engineering কেন দরকার

- vague prompt → ভুল বা অস্পষ্ট output
- clear, structured prompt → নির্ভুল ও কার্যকর output
- .NET dev হিসেবে আপনি prompt design করে GPT-ভিত্তিক API response নিয়ন্ত্রণ করতে পারেন

❸ Prompt-এর ধরন

❹ .NET ভিত্তিক বাস্তব প্রয়োগ

ধরন	উদাহরণ	ব্যবহার
Instructional	"Summarize this in Bangla"	Content generation
Conversational	"You are an SDG expert..."	Chatbot
Few-shot	"Q: ... A: ... Q: ... A: ..."	Pattern-based response
Chain-of-thought	"Let's think step by step..."	Reasoning
Role-based	"Act as a .NET architect..."	Persona-driven output

- Azure OpenAI → `system, user, assistant` role-based prompt
- LangChain.NET → `PromptTemplate` দিয়ে dynamic prompt তৈরি
- Hugging Face API → Bengali SDG prompt পাঠিয়ে summarization
- Example: "বাংলাদেশে দারিদ্র্য হ্রাসের উপায় ৩ লাইনে বলো"—GPT-4 থেকে concise Bangla output

🔗 Deployment Flow (.NET dev perspective)

Prompt → API Call → JSON Output → Razor/Blazor UI

- Blazor বা Razor Page থেকে prompt ইনপুট
- `HttpClient` দিয়ে API call
- Razor View-এ output দেখানো

✓ Summary (এক লাইনে)

Prompt Engineering হলো AI-এর সাথে সঠিকভাবে কথা বলার কৌশল—.NET অ্যাপে GPT বা Hugging Face API ব্যবহার করে এটি নিয়ন্ত্রণ করা যায়

□ Prompt Engineering Techniques — .NET dev perspective

① Role-based Prompting

AI-কে নির্দিষ্ট ভূমিকা দিন

□ উদাহরণ:

You are a senior .NET architect. Explain the difference between microservices and monolith in Bangla.

✦ ব্যবহার: Persona-driven output, SDG advisor, chatbot

② Few-shot Prompting

কয়েকটি উদাহরণ দিয়ে AI-কে pattern শেখান

□ উদাহরণ:

Q: SDG 1 কী? A: দারিদ্র্য হ্রাস

Q: SDG 3 কী? A: স্বাস্থ্য ও কল্যাণ

Q: SDG 4 কী? A: ...

★ ব্যবহার: Classification, translation, Bangla glossary builder

③ Chain-of-Thought Prompting (CoT)

AI-কে ধাপে ধাপে চিন্তা করতে বলুন

□ উদাহরণ:

Let's think step by step. First, identify the SDG goal. Then, list 3 policy actions. Finally, summarize in Bangla.

★ ব্যবহার: Reasoning, policy recommendation, SDG planning

④ Self-Consistency Prompting

একই প্রশ্নে একাধিক উত্তর চেয়ে সবচেয়ে যুক্তিসঙ্গতটি বেছে নিন

□ উদাহরণ:

Generate 3 possible summaries of this SDG report. Pick the most relevant one.

★ ব্যবহার: Summarization, content validation

⑤ Retrieval-Augmented Generation (RAG)

Prompt-এর সাথে external knowledge যুক্ত করুন

□ উদাহরণ:

Using the following document, answer in Bangla: [UNDP SDG PDF] → What are the key poverty reduction strategies?

★ ব্যবহার: Semantic search + generation, SDG document QA

⑥ Prompt Chaining

একাধিক prompt ধাপে ধাপে যুক্ত করুন

□ উদাহরণ:

1 Summarize the SDG report

2 Translate the summary into Bangla

3 Generate 3 hashtags for social media

★ ব্যবহার: Workflow automation, content pipeline

⑦ Format-specific Prompting

AI-কে নির্দিষ্ট format-এ output দিতে বলুন

□ উদাহরণ:

Give the output in JSON format with keys: goal, strategy, region.

✦ ব্যবহার: API integration, structured output for .NET backend

⑧ Meta Prompting

AI-কে নিজেই prompt তৈরি করতে বলুন

□ উদাহরণ:

Generate a prompt that helps summarize SDG reports in Bangla.

✦ ব্যবহার: Dynamic prompt builder, user-driven customization

⑨ ReAct (Reasoning + Acting)

AI reasoning করে এবং external tool ব্যবহার করে

□ উদাহরণ:

Search SDG 1 definition, then summarize it in Bangla.

✦ ব্যবহার: LangChain.NET workflow, API chaining, SDG knowledge base integration

⑩ Tree-of-Thought Prompting

AI-কে একাধিক চিন্তার শাখা তৈরি করতে বলুন

□ উদাহরণ:

List 3 possible strategies for poverty reduction. For each, list pros and cons. Then choose the best.

✦ ব্যবহার: Decision support, SDG policy analysis

⑪ Multimodal Prompting

Text + Image + Audio একসাথে ব্যবহার

□ উদাহরণ:

Given this image of a flood-affected area, generate a Bangla caption and suggest SDG response.

✦ ব্যবহার: Azure Vision + OpenAI integration, Blazor UI-based input

⑫ Program-Aided Prompting (PAL)

Prompt-এর সাথে code যুক্ত করে reasoning বাড়ানো

□ উদাহরণ:

Use C# logic to calculate poverty index, then explain the result in Bangla.

✦ ব্যবহার: .NET backend + GPT explanation combo

13 Directional Stimulus Prompting

AI-কে নির্দিষ্ট দৃষ্টিভঙ্গি বা bias দিয়ে চালিত করা

□ উদাহরণ:

Respond from the perspective of a Bangladeshi youth leader working on SDG 4.

★ ব্যবহার: Persona-driven chatbot, advocacy simulation

14 Automatic Prompt Engineering

AI নিজেই trial-and-error করে সেরা prompt খুঁজে বের করে

□ উদাহরণ:

Try 5 variations of this prompt and pick the one that gives the clearest Bangla summary.

★ ব্যবহার: LangChain.NET loop, Azure Function orchestration

15 Graph Prompting

AI-কে causal বা logical graph structure অনুসরণ করতে বলুন

□ উদাহরণ:

Draw a cause-effect chain for poverty in Bangladesh using SDG indicators.

★ ব্যবহার: SDG dashboard, Blazor diagram integration

Chain-of-Thought Prompting

— ধাপে ধাপে চিন্তা করানো

সংজ্ঞা:

এটি এমন একটি কৌশল যেখানে AI-কে ধাপে ধাপে চিন্তা করতে শেখানো হয়—একবারে উত্তর না দিয়ে, যুক্তি সহ ব্যাখ্যা করে।

উদাহরণ:

প্রশ্ন: “ $৫ + ৭ + ৯ - ১২ = ?$ ”

→ AI বলবে:

“ $৫ + ৭ = ১২$, $১২ + ৯ = ২১$, $২১ - ১২ = ৯$ ”

লাভ:

- যুক্তিসম্মত উত্তর
- ভুল কম হয়

- জটিল সমস্যায় ভালো পারফরম্যান্স

☞ বিস্তারিত: [CoT Prompting Guide](#), [Codecademy Explanation](#)

Few-shot Learning

— অল্প উদাহরণে শেখা

সংজ্ঞা:

Few-shot Learning হলো এমন শেখার পদ্ধতি যেখানে AI অল্প কিছু উদাহরণ দেখে নতুন টাস্কে ভালো পারফর্ম করে।

উদাহরণ:

আপনি AI-কে ২টা ছবি দেখালেন—একটা ক্যান্সারাস, একটা নয়। এরপর সে নতুন ছবি দেখে নিজে বুঝে ফেলে কোনটা ক্যান্সারাস।

ব্যবহার:

- মেডিকেল ডায়াগনোসিস
- রেয়ার স্পিসি চিনে ফেলা
- কাস্টমার সাপোর্টে নতুন প্রশ্ন বোঝা

☞ বিস্তারিত: [Few-shot Learning Explained](#), [GeeksforGeeks FSL](#)

Retrieval-Augmented Generation (RAG)

— তথ্য খুঁজে এনে উত্তর তৈরি

সংজ্ঞা:

RAG হলো এমন একটি AI কৌশল যেখানে মডেল শুধু নিজের শেখা তথ্য নয়, বাইরের ডেটাবেস বা ডকুমেন্ট থেকেও তথ্য খুঁজে এনে উত্তর তৈরি করে।

উদাহরণ:

UNDP-এর SDG রিপোর্ট বিশ্লেষণ করতে চাইলে, RAG প্রথমে রিপোর্ট থেকে প্রাসঙ্গিক তথ্য খুঁজে বের করে, তারপর সেই তথ্য দিয়ে উত্তর তৈরি করে।

মূল ধাপ:

1. 🔍 প্রশ্নকে Embedding করে Vector DB-তে খোঁজ
2. 📄 প্রাসঙ্গিক ডেটা রিট্রিভ করে
3. 📄 সেই তথ্য Prompt-এ যুক্ত করে
4. 🗨️ LLM উত্তর তৈরি করে

ব্যবহার:

- নলেজ বেস চ্যাটবট
- কাস্টমার সাপোর্ট
- নীতিমালা বিশ্লেষণ
- মেডিকেল রিপোর্ট সারাংশ

[আরও জানতে দেখুন AWS-এর RAG গাইড](#), [শিল্পভিত্তিক RAG উদাহরণ](#)

❶ সংজ্ঞা

RAG হলো এমন একটি AI কৌশল যেখানে LLM-এর সাথে বাইরের তথ্যসূত্র যুক্ত করে output তৈরি করা হয়।

এটি traditional LLM-এর সীমাবদ্ধতা কাটিয়ে **real-time, context-aware, factually accurate** উত্তর দিতে সক্ষম।

📄 বিস্তারিত: [GeeksforGeeks-এর RAG গাইড](#)

❷ RAG কীভাবে কাজ করে

1. **Query → Embedding**
 - ইউজারের প্রশ্নকে vector-এ রূপান্তর করা হয়
2. **Vector Search → External Knowledge Base**
 - SDG docs, PDFs, API data থেকে relevant chunk খোঁজা হয়
3. **Prompt Augmentation**
 - retrieved data + original query → final prompt
4. **LLM Generation**
 - GPT বা অন্য model augmented prompt থেকে উত্তর তৈরি করে

❸ .NET ভিত্তিক বাস্তব প্রয়োগ

- Azure Cognitive Search → SDG document indexing
- LangChain.NET → Retriever + PromptTemplate + LLMChain
- Hugging Face → Bengali SDG FAQ chatbot
- Example: “বাংলাদেশে দারিদ্র্য হ্রাসের উপায়” → SDG docs থেকে relevant অংশ খুঁজে → GPT summary

❹ RAG-এর সুবিধা

সুবিধা	Traditional LLM	RAG
Real-time knowledge	✗	✓
Domain-specific accuracy	✗	✓
Explainability	✗	✓
Cost-effective (no retrain)	✗	✓

❑ বাস্তব উদাহরণ: [Merge.dev-এর RAG use cases](#) দেখুন—lead recommendation, medical Q&A, internal search ইত্যাদি


❺ Deployment Flow (.NET dev perspective)

Query → Embedding → Retrieval → Augmented Prompt → LLM Output → Razor/Blazor UI

- Blazor SPA থেকে query ইনপুট
- Azure Search বা LangChain.NET দিয়ে retrieval
- GPT API call করে Razor View-এ output দেখানো

✓ Summary (এক লাইনে)

RAG হলো AI-এর জন্য external knowledge যুক্ত করার কৌশল—.NET অ্যাপে SDG chatbot, Bengali Q&A, বা document summarizer তৈরিতে এটি অত্যন্ত কার্যকর

 RAG Use Cases for .NET Developers	
Use Case	সংক্ষিপ্ত বর্ণনা
❶ SDG Chatbot with Document Retrieval	ইউজারের প্রশ্ন অনুযায়ী SDG রিপোর্ট থেকে তথ্য খুঁজে GPT দিয়ে উত্তর তৈরি
❷ Bengali FAQ Assistant for UNDP	বাংলা প্রশ্নের জন্য Hugging Face থেকে relevant SDG নথি খুঁজে উত্তর তৈরি
❸ Policy Recommendation Engine	Poverty বা Climate নিয়ে query দিলে SDG policy docs থেকে strategy সাজিয়ে দেয়
❹ Semantic Search + Summarizer	Azure Cognitive Search দিয়ে SDG ডেটা খুঁজে GPT দিয়ে summary তৈরি
❺ Medical Q&A Assistant	Local health guideline PDF থেকে তথ্য খুঁজে GPT দিয়ে উত্তর তৈরি (Bangla + English)
❻ Legal Document Explainer	আইনগত নথি থেকে clause খুঁজে GPT দিয়ে সহজ ভাষায় ব্যাখ্যা
❼ Internal Knowledge Base Assistant	UNDP বা NGO internal docs থেকে তথ্য খুঁজে GPT দিয়ে উত্তর তৈরি
❽ Multilingual SDG Translator	SDG নথি খুঁজে GPT দিয়ে Bengali-English translation
❾ Lead Recommendation System	NGO impact report থেকে relevant partner বা donor খুঁজে সাজেশন দেয়
❿ Academic Research Assistant	SDG-related গবেষণা পত্র থেকে তথ্য খুঁজে GPT দিয়ে summary বা citation তৈরি


📖 Open-Source Tools for RAG Implementatio

<div><div>12</div><div>34</div></div>	Tool Name	কাজের ধরন	GitHub/Docs
❶	LangChain	RAG pipeline orchestration	LangChain GitHub
❷	LlamaIndex	Document indexing + retrieval	LlamaIndex GitHub
❸	Haystack	Search + LLM integration	Haystack GitHub
❹	R2R (Retrieve-to-Respond)	Hybrid search + knowledge graph	R2R GitHub
❺	Cognita	Modular RAG framework (scalable)	Cognita GitHub
❻	Firecrawl	Web scraping for LLM ingestion	Firecrawl Docs
❼	Weaviate	Vector DB for semantic search	Weaviate GitHub
❽	Qdrant	Fast vector similarity search	Qdrant GitHub
❾	Milvus	Scalable vector database	Milvus GitHub
❿	FAISS	Facebook’s vector search lib	FAISS GitHub
⓫	OpenAI Embedding API	Text → vector conversion	OpenAI Docs
⓬	Hugging Face Transformers	Pretrained LLMs + Embedding	HF GitHub

🔗 Typical RAG Stack (.NET Perspective)

User Query → Embedding (OpenAI/HF) → Vector DB (Qdrant/Weaviate) → Retrieval (LangChain/LlamaIndex) → Prompt Augmentation → LLM Response → ASP.NET UI

🔗 Azure Tools for RAG Implementatio

	Tool Name	কাজের ধরন
❶	Azure AI Search	Vector + Hybrid Search (semantic + keyword)
❷	Azure OpenAI Service	LLM-based generation (GPT-4, GPT-3.5)
❸	Azure AI Studio	Prompt orchestration, chaining, evaluation
❹	Azure AI Document Intelligence	PDF, scanned docs থেকে chunk extraction
❺	Azure Blob Storage	SDG বা UNDP ডেটা সংরক্ষণ ও ingestion
❻	Azure Functions	Orchestration logic, chaining, embedding calls
❼	Azure Cosmos DB / Table Storage	Metadata বা chunked docs সংরক্ষণ
❽	Azure Machine Learning	Custom embedding model training (optional)
❾	Azure Semantic Kernel	LangChain-style orchestration in .NET
❿	Azure Key Vault	API keys ও credentials নিরাপদে সংরক্ষণ

□ Typical Azure RAG Workflow (.NET Perspective)

User Query → Azure OpenAI Embedding → Azure AI Search → Top-k Chunks → Prompt Augmentation → GPT Response → ASP.NET UI

Multi-Modal AI

— একাধিক ধরনের ডেটা একসাথে বোঝা

সংজ্ঞা:

Multi-Modal AI এমন AI যা একসাথে টেক্সট, ছবি, অডিও, ভিডিও ইত্যাদি ডেটা বিশ্লেষণ করতে পারে।

উদাহরণ:

একটি অ্যাপ যদি ছবি দেখে রোগ শনাক্ত করে, এবং সেই রোগের বর্ণনা বাংলায় বলে দেয়— তাহলে এটি Multi-Modal AI।

ব্যবহার:

- Vision + Text: ছবি দেখে বর্ণনা তৈরি
- Audio + Text: স্পিচ শুনে উত্তর তৈরি
- Video + Sensor: ট্রাফিক বিশ্লেষণ

প্রযুক্তি:

- Transformer + CNN + RNN
- Attention Mechanism
- Tensor Fusion

[IBM-এর Multi-Modal AI গাইড](#), [GeeksforGeeks বিশ্লেষণ](#)

Agent-based AI

— কাজের জন্য স্বাধীন AI কর্মী

সংজ্ঞা:

Agent-based AI হলো এমন AI যা নিজে সিদ্ধান্ত নিতে পারে, কাজ করতে পারে, এবং পরিবেশ অনুযায়ী নিজেকে মানিয়ে নিতে পারে।

উদাহরণ:

একটি AI অ্যাসিস্ট্যান্ট যদি আপনার ক্যালেন্ডার দেখে মিটিং শিডিউল করে, ইমেইল পাঠায়, এবং রিপোর্ট তৈরি করে—তাহলে এটি একটি Agent।

ধরন:

- Simple Reactive Agent
- Goal-Based Agent
- Learning Agent
- Utility-Based Agent

ব্যবহার:

- কাস্টমার সাপোর্ট
- প্রজেক্ট ম্যানেজমেন্ট
- এজেন্টিক চ্যাটবট

[Agent টাইপ ও উদাহরণ দেখুন](#), [Google Cloud-এর Agent গাইড](#)

Tool Calling / Function Calling

— AI নিজে কাজ চালায়





সংজ্ঞা:

Tool Calling হলো এমন ক্ষমতা যেখানে AI বাইরের API, ফাংশন বা টুল ব্যবহার করে কাজ সম্পন্ন করে—যেমন ডেটা রিট্রিভ, রিপোর্ট তৈরি, বা ট্রান্সাকশন চালানো।

উদাহরণ:

আপনি AI-কে বলেন “১০০ USD কে EUR-এ কনভার্ট করো”—AI একটি Currency API কল করে রিয়েল-টাইম রেট নিয়ে ফলাফল দেয়।

ধাপ:

1.  Intent বুঝে
2.  টুল নির্বাচন করে
3.  API কল করে
4.  ফলাফল নিয়ে উত্তর দেয়

ব্যবহার:

- রিয়েল-টাইম ডেটা
- অটোমেটেড ওয়ার্কফ্লো
- কাস্টম ফাংশন এক্সিকিউশন

[Tool Calling Explained, IBM-এর টেকনিক্যাল গাইড](#)

Multi-Agent Systems

— AI টিম একসাথে কাজ করে

সংজ্ঞা:

Multi-Agent Systems (MAS) হলো এমন AI সিস্টেম যেখানে একাধিক Agent একসাথে কাজ করে—প্রতিটি Agent স্বাধীনভাবে সিদ্ধান্ত নিতে পারে, কিন্তু সম্মিলিতভাবে একটি বড় কাজ সম্পন্ন করে।

উদাহরণ:

একটি স্মার্ট শহরে ট্রাফিক নিয়ন্ত্রণ করতে—এক Agent সিগনাল দেখে, আরেকটি Agent গাড়ির গতি বিশ্লেষণ করে, আরেকটি Agent রুট সাজেস্ট করে।

ধরন:

- Decentralized
- Local View
- Autonomous Decision Making

ব্যবহার:

- Robotics (Swarm Robots)
- Smart Grid
- Supply Chain Optimization
- Collaborative Chatbots

[MAS গাইড ও বাস্তব উদাহরণ, 12 MAS বাস্তব প্রয়োগ](#)

Image Classification

— ছবি দেখে শ্রেণিবিন্যাস

সংজ্ঞা:

Image Classification হলো এমন একটি প্রযুক্তি যেখানে একটি ছবি দেখে AI বলে দেয় সেটি কোন ক্যাটাগরির মধ্যে পড়ে।

উদাহরণ:

- একটি ছবি দেখে বলে দেওয়া: “এটি একটি বিড়াল”
- কৃষি ড্রোন ছবি দেখে বলে দেয়: “এই গাছটি অসুস্থ”
- মেডিকেল স্ক্যান দেখে বলে: “এই টিউমারটি ম্যালিগন্যান্ট”

প্রযুক্তি:

- CNN (Convolutional Neural Network)
- Transfer Learning (ResNet, MobileNet)
- Softmax Classifier

ব্যবহার:

- স্বাস্থ্যসেবা (রোগ শনাক্তকরণ)
- কৃষি (ফসলের স্বাস্থ্য বিশ্লেষণ)
- রিটেইল (পণ্যের ছবি শ্রেণিবিন্যাস)

Object Detection

— ছবিতে বস্তু খুঁজে বের করা

সংজ্ঞা:

Object Detection শুধু বলে না ছবিতে কী আছে, বরং সেটি কোথায় আছে তা দেখায়—Bounding Box সহ।

উদাহরণ:

- একটি সিসিটিভি ফুটেজে AI বলে দেয়: “এই জায়গায় একজন মানুষ দাঁড়িয়ে আছে”
- ট্রাফিক ক্যামেরা বলে: “এই ছবিতে ৩টি গাড়ি আছে”

প্রযুক্তি:

- YOLO (You Only Look Once)
- SSD (Single Shot Detector)
- Faster R-CNN

ব্যবহার:

- নিরাপত্তা (অনুপ্রবেশকারী শনাক্ত)
- স্বয়ংচালিত গাড়ি (পথে মানুষ, সিগনাল চিনে)
- রিটেইল (পণ্য গননা)

Optical Character Recognition (OCR)

— ছবি থেকে লেখা পড়া

সংজ্ঞা:

OCR হলো এমন প্রযুক্তি যা ছবি বা স্ক্যান করা ডকুমেন্ট থেকে লেখা পড়ে এবং তা ডিজিটাল টেক্সটে রূপান্তর করে।

উদাহরণ:

- একটি স্ক্যান করা ইনভয়েস থেকে নাম, তারিখ, পরিমাণ বের করে
- মোবাইল অ্যাপে চেক স্ক্যান করে ব্যাংক ডিপোজিট করে
- রাস্তার সাইনবোর্ড থেকে লেখা পড়ে অনুবাদ করে

প্রযুক্তি:

- Tesseract OCR
- Google Vision API
- AWS Textract

ব্যবহার:

- ব্যাংকিং (চেক প্রসেসিং)
- স্বাস্থ্যসেবা (রোগীর ফর্ম ডিজিটাইজ)
- শিক্ষা (হাতের লেখা থেকে নোট তৈরি)

Document AI

— ডকুমেন্ট বুঝে তথ্য বের করা

সংজ্ঞা:

Document AI হলো এমন প্রযুক্তি যা OCR-এর পরে ডকুমেন্টের কাঠামো, প্রসঙ্গ, এবং অর্থ বিশ্লেষণ করে—যেমন ফর্ম, ইনভয়েস, রিপোর্ট।

উদাহরণ:

- UNDP-এর SDG রিপোর্ট থেকে “বাংলাদেশে দারিদ্র্য হ্রাস” সম্পর্কিত তথ্য বের করা
- একটি পিডিএফ থেকে নাম, ঠিকানা, তারিখ, এবং মোট পরিমাণ আলাদা করে ফেলা

প্রযুক্তি:

- Google Document AI
- Azure Form Recognizer
- NLP + OCR + Layout Analysis

ব্যবহার:

- ফাইন্যান্স (ট্যাক্স ফর্ম প্রসেসিং)
- স্বাস্থ্যসেবা (ইন্টেক ফর্ম বিশ্লেষণ)
- সরকারি নীতিমালা বিশ্লেষণ

Face Recognition

— মুখ দেখে মানুষ শনাক্ত

সংজ্ঞা:

Face Recognition হলো এমন প্রযুক্তি যা মুখের ছবি দেখে ব্যক্তিকে শনাক্ত বা যাচাই করে।

উদাহরণ:

- ফোন আনলক করতে মুখ চিনে
- এয়ারপোর্টে পাসপোর্ট যাচাই
- সিসিটিভি ফুটেজে অপরাধী শনাক্ত

প্রযুক্তি:

- FaceNet, DeepFace
- Haar Cascade, Dlib
- Embedding + Similarity Matching

ব্যবহার:

- নিরাপত্তা (অ্যাক্সেস কন্ট্রোল)
- উপস্থিতি যাচাই (স্কুল, অফিস)
- ফরেনসিক (ক্রিমিনাল ট্র্যাকিং)

Google Transformer Architecture

— “Attention is All You Need”

সংজ্ঞা:

Transformer হলো Google-এর তৈরি একটি নিউরাল নেটওয়ার্ক আর্কিটেকচার যা **Self-Attention Mechanism** ব্যবহার করে টেক্সটের প্রসঙ্গ বুঝে কাজ করে। এটি RNN বা LSTM-এর সীমাবদ্ধতা দূর করে।

মূল উপাদান:

- Encoder: ইনপুট বিশ্লেষণ করে
- Decoder: আউটপুট তৈরি করে
- Multi-Head Attention: একসাথে বিভিন্ন প্রসঙ্গ বিশ্লেষণ
- Positional Encoding: শব্দের অবস্থান বোঝে

উদাহরণ:

- Google Translate
- BERT (Bidirectional Encoder Representations from Transformers)
- GPT, T5, PaLM—all inspired by Transformer architecture

MCP (Model Context Protocol)

— AI-এর USB-C কানেক্টর

সংজ্ঞা:

MCP হলো একটি ওপেন-স্ট্যান্ডার্ড প্রোটোকল যা LLM-কে বাইরের টুল, API, বা ডেটাবেসের সাথে সংযুক্ত করে। এটি AI-কে শুধু উত্তর দেওয়া নয়, বরং কাজ করতেও সক্ষম করে।

মূল কাজ:

- AI → MCP Client → MCP Server → External Tool/API
- OAuth দিয়ে নিরাপদ সংযোগ
- Gemini CLI বা Azure OpenAI-এর সাথে একত্রে কাজ করে

ব্যবহার:

- রিয়েল-টাইম ডেটা ফেচ
- অটোমেটেড টাস্ক
- এজেন্টিক AI অ্যাপ্লিকেশন

Hugging Face

— Machine Learning-এর GitHub

সংজ্ঞা:

Hugging Face হলো একটি ওপেন সোর্স প্ল্যাটফর্ম যেখানে আপনি NLP, CV, এবং অন্যান্য AI মডেল খুঁজে পেতে, শেয়ার করতে, এবং ব্যবহার করতে পারেন।

মূল ফিচার:

- Transformers লাইব্রেরি
- Pretrained Models (BERT, RoBERTa, BLOOM)
- Datasets, Spaces, Inference API
- Gradio Integration

ব্যবহার:

- Sentiment Analysis
- Text Classification
- Chatbot Development
- Bengali NLP Projects

ONNX (Open Neural Network Exchange)

— AI মডেল ট্রান্সফার ফরম্যাট

সংজ্ঞা:

ONNX হলো একটি ওপেন ফরম্যাট যা বিভিন্ন ML ফ্রেমওয়ার্কের মধ্যে মডেল ট্রান্সফার সহজ করে—যেমন PyTorch থেকে TensorFlow-এ।

মূল সুবিধা:

- Framework Interoperability
- Hardware Optimization (CPU, GPU, FPGA)
- Production Deployment সহজ হয়

ব্যবহার:

- PyTorch-এ ট্রেন করে ONNX-এ এক্সপোর্ট করে Azure-এ ডিপ্লয়
- Mobile বা Embedded Device-এ ইনফারেন্স চালানো

Semantic Kernel

— Microsoft-এর Agent Framework

সংজ্ঞা:

Semantic Kernel হলো Microsoft-এর তৈরি একটি SDK যা LLM, প্লাগইন, মেমোরি, এবং ফাংশন কলিং একত্রে ব্যবহার করে Agent তৈরি করে।

মূল ফিচার:

- Plugins (Skills)
- Function Calling
- Planning & Memory
- Multi-Agent Workflow
- Integration with Hugging Face, Azure, ONNX, MCP

ব্যবহার:

- AI অ্যাসিস্ট্যান্ট তৈরি
- এন্টারপ্রাইজ অটোমেশন
- কাস্টম চ্যাটবট
- Workflow Orchestration

TensorFlow.NET

— .NET দিয়ে TensorFlow ব্যবহার

সংজ্ঞা:

TensorFlow.NET হলো Google-এর TensorFlow লাইব্রেরির জন্য C# ও .NET ভিত্তিক বাইন্ডিং, যা .NET ডেভেলপারদের জন্য AI ও Deep Learning মডেল তৈরি সহজ করে।

ব্যবহার:

- C# দিয়ে Neural Network তৈরি
- .NET অ্যাপে Image Classification বা NLP যুক্ত করা
- UNDP-এর মতো সংস্থায় .NET ভিত্তিক AI সমাধান তৈরি

উদাহরণ:

```
var tensor = tf.constant(new float[] { 1, 2, 3 }); var result = tf.math.reduce_sum(tensor);
```

🔗 [TensorFlow.NET ডকুমেন্টেশন](#)

Ollama

— লোকাল মেশিনে LLM চালানোর প্ল্যাটফর্ম

সংজ্ঞা:

Ollama হলো একটি ওপেন-সোর্স টুল যা আপনাকে লোকাল মেশিনে LLM (Large Language Model) চালাতে দেয়—ক্লাউড ছাড়াই, সম্পূর্ণ ডেটা নিয়ন্ত্রণসহ।

মূল বৈশিষ্ট্য:

- লোকাল ইনফারেন্স (Privacy-friendly)
- Llama 3 সহ বিভিন্ন মডেল সাপোর্ট
- GPU-সক্ষমতা থাকলে দ্রুত পারফরম্যান্স

ব্যবহার:

- অফলাইন চ্যাটবট
- সংবেদনশীল ডেটা বিশ্লেষণ
- গবেষণায় কাস্টম মডেল ট্রায়াল

🔗 [Ollama Explained on GeeksforGeeks](#)

LLM Orchestration

— LLM-ভিত্তিক অ্যাপের Workflow পরিচালনা

সংজ্ঞা:

LLM Orchestration হলো এমন একটি কাঠামো যা LLM, API, Prompt, Memory ইত্যাদি একত্রে পরিচালনা করে একটি Agent বা অ্যাপ্লিকেশন তৈরি করে।

মূল ফ্রেমওয়ার্ক:

- LangChain: Prompt chaining ও Tool calling
- LlamaIndex: RAG ও ডেটা রিট্রিভ
- Semantic Kernel: Microsoft-এর Agent SDK

ব্যবহার:

- মাল্টি-স্টেপ প্রশ্নোত্তর
- কাস্টম চ্যাটবট
- এজেন্টিক Decision Making

🔗 [IBM-এর LLM Orchestration গাইড](#)

Hallucination

— ভুল বা বানানো তথ্য তৈরি

সংজ্ঞা:

Hallucination হলো এমন একটি সমস্যা যেখানে LLM মডেল ভুল, বানানো বা বাস্তবতাবিহীন তথ্য তৈরি করে—যদিও তা বিশ্বাসযোগ্য মনে হয়।

উদাহরণ:

- “বাংলাদেশে ১৯৮২ সালে AI ইনস্টিটিউট প্রতিষ্ঠিত হয়” — অথচ এটি সত্য নয়
- বানানো রেফারেন্স বা গবেষণা পেপার উল্লেখ

কারণ:

- Probabilistic Text Generation
- Training Data-র সীমাবদ্ধতা
- Context retention-এর অভাব

সমাধান:

- Retrieval-Augmented Generation (RAG)
- Human-in-the-loop Verification
- Uncertainty-aware Training

🔗 [LLM Hallucination Explained](#)

Telemetry

— AI সিস্টেমের আচরণ পর্যবেক্ষণ

সংজ্ঞা:

Telemetry হলো এমন একটি প্রযুক্তি যা AI সিস্টেমের পারফরম্যান্স, ব্যবহার, এবং আচরণ পর্যবেক্ষণ করে—যেমন latency, token usage, error rate।

ধরন:

- Metrics: latency, token count
- Logs: event tracking
- Traces: workflow path

ব্যবহার:

- AI debugging
- Model optimization
- User behavior analysis (Semantic Telemetry)

☞ [Microsoft Semantic Telemetry Project](#)

LangChain.NET

LangChain.NET হলো LangChain-এর একটি C#/.NET ভিত্তিক সংস্করণ, যা .NET ডেভেলপারদের জন্য LLM (Large Language Model) ভিত্তিক অ্যাপ্লিকেশন তৈরি সহজ করে তোলে। Python-ভিত্তিক মূল LangChain যেমন LLM, টুল, মেমোরি, এবং চেইন পরিচালনা করে, LangChain.NET সেই একই ধারণা .NET পরিবেশে নিয়ে আসে।

□ LangChain.NET কী করে?

✓ মূল বৈশিষ্ট্য:

- **Prompt Management:** ইনপুট ও আউটপুট কাস্টমাইজ করে, PromptTemplate ব্যবহার করে
- **Chains:** একাধিক ধাপে LLM ও টুল একত্রে ব্যবহার করে
- **Agents:** AI নিজে সিদ্ধান্ত নিয়ে API কল বা ডেটা রিট্রিভ করতে পারে
- **Memory:** আগের কথোপকথন মনে রাখে, ধারাবাহিকতা বজায় রাখে
- **Vector DB Integration:** Embedding ব্যবহার করে similarity search করে

□ বাস্তব উদাহরণ

১. কাস্টম চ্যাটবট:

UNDP-এর জন্য একটি চ্যাটবট তৈরি করা যায় যা SDG রিপোর্ট থেকে তথ্য রিট্রিভ করে এবং প্রশ্নের উত্তর দেয়।

২. ডকুমেন্ট বিশ্লেষণ:

LangChain.NET ব্যবহার করে OCR + LLM যুক্ত করে একটি Document AI তৈরি করা যায় যা ইনভয়েস বা ফর্ম থেকে তথ্য বের করে।

৩. Tool Calling:

আপনি চাইলে LangChain.NET দিয়ে Currency API, Weather API, বা Azure Function কল করতে পারেন—LLM নিজে সিদ্ধান্ত নিয়ে।

☞ রেফারেন্স ও রিসোর্স

- [LangChain.NET GitHub Repository](#) — C# ডেভেলপারদের জন্য ওপেন সোর্স প্রকল্প
- [LangChain.NET Quickstart Guide](#) — NuGet দিয়ে শুরু করার ধাপ ও উদাহরণ কোড
- [LangChain Overview \(Python ভিত্তিক\)](#) — মূল ধারণা ও কাঠামো ব্যাখ্যা করেছে

Neural Network

— মানুষের মস্তিষ্ক অনুকরণকারী AI

সংজ্ঞা:

Neural Network (বা Artificial Neural Network) হলো এমন একটি AI কাঠামো যা মানুষের মস্তিষ্কের নিউরনের মতো কাজ করে—ডেটা থেকে শেখে, প্যাটার্ন চিনে, এবং সিদ্ধান্ত নেয়।

মূল গঠন:

- **Input Layer:** ডেটা গ্রহণ করে
- **Hidden Layers:** তথ্য বিশ্লেষণ করে
- **Output Layer:** ফলাফল দেয়
- প্রতিটি লেয়ারে থাকে **Weights, Biases**, এবং **Activation Function**

উদাহরণ:

- ছবি দেখে বলে দেওয়া: “এটি একটি কুকুর”
- ব্যাংকে লেনদেন বিশ্লেষণ করে: “এটি সন্দেহজনক”
- UNDP-এর রিপোর্ট থেকে SDG অগ্রগতি প্রেডিক্ট করা

🔗 বিস্তারিত: [SkillCamper-এর Neural Network গাইড](#)

Multimodal AI

— একাধিক ধরনের ডেটা একসাথে বিশ্লেষণ

সংজ্ঞা:

Multimodal AI এমন AI যা একসাথে টেক্সট, ছবি, অডিও, ভিডিও ইত্যাদি ডেটা বিশ্লেষণ করতে পারে—এক modality-তে সীমাবদ্ধ নয়।

উদাহরণ:

- আপনি একটি পাখির ছবি ও তার ডাকের অডিও দেন → AI বলে দেয়: “এটি একটি দোয়েল”
- একটি চ্যাটবট আপনার মুখের ছবি দেখে চশমার সাইজ সাজেস্ট করে

প্রযুক্তি:

- CNN (ছবি), RNN (অডিও), Transformer (টেক্সট)
- Attention Mechanism
- Tensor Fusion

🔗 বিস্তারিত: [IBM-এর Multimodal AI বিশ্লেষণ](#)

Context Engineering

— AI-কে সঠিকভাবে তথ্য পরিবেশন

সংজ্ঞা:

Context Engineering হলো এমন একটি কৌশল যেখানে AI-কে শুধু প্রশ্ন নয়, বরং প্রাসঙ্গিক তথ্য, ইতিহাস, এবং নির্দেশনা একত্রে দেওয়া হয়—যাতে সে সঠিক ও অর্থপূর্ণ উত্তর দিতে পারে।

মূল উপাদান:

- **Memory Management:** আগের কথোপকথন সংরক্ষণ
- **External Inputs:** ডকুমেন্ট, API, ডেটাবেস
- **Prompt Structuring:** AI কীভাবে উত্তর দেবে তা নির্ধারণ
- **Relevance Filtering:** কোন তথ্য সবচেয়ে গুরুত্বপূর্ণ

উদাহরণ:

UNDP-এর রিপোর্ট বিশ্লেষণ করতে চাইলে, AI-কে শুধু প্রশ্ন নয়, রিপোর্টের সারাংশ, টাইমলাইন, এবং SDG টার্গেটও দিতে হয়।

☞ বিস্তারিত: [Geeky Gadgets-এর Context Engineering গাইড](#)

Backpropagation

— ভুল থেকে শেখার কৌশল

সংজ্ঞা:

Backpropagation হলো এমন একটি অ্যালগরিদম যা Neural Network-এর ভুল (Error) বিশ্লেষণ করে ওজন (Weight) ও বায়াস (Bias) আপডেট করে—যাতে ভবিষ্যতে সঠিক সিদ্ধান্ত নিতে পারে।

ধাপ:

1. **Forward Pass:** ইনপুট → আউটপুট
2. **Error Calculation:** আসল ও প্রেডিক্টেড ফলাফলের পার্থক্য
3. **Backward Pass:** Gradient Descent দিয়ে ওজন পরিবর্তন

উদাহরণ:

একটি মডেল যদি বলে “ছবিটি বিড়াল”, অথচ এটি কুকুর—তাহলে Backpropagation সেই ভুল বিশ্লেষণ করে ওজন ঠিক করে।

☞ বিস্তারিত: [GeeksforGeeks-এর ব্যাকপ্রপাগেশন গাইড](#)

Anomaly Detection

— অস্বাভাবিকতা শনাক্তকরণ

সংজ্ঞা:

Anomaly Detection হলো এমন একটি প্রযুক্তি যা ডেটার মধ্যে অস্বাভাবিক বা ব্যতিক্রমী আচরণ শনাক্ত করে—যা সাধারণ প্যাটার্নের বাইরে।

উদাহরণ:

- ব্যাংকে হঠাৎ ১০ লাখ টাকার লেনদেন → সম্ভাব্য জালিয়াতি
- সার্ভারে হঠাৎ CPU স্পাইক → সাইবার আক্রমণ
- UNDP ডেটাতে হঠাৎ SDG সূচকে পতন → নীতিগত সমস্যা

প্রযুক্তি:

- Supervised, Unsupervised, Semi-supervised Learning
- Statistical Methods, Isolation Forest, Autoencoders

□ বিস্তারিত: [GeeksforGeeks-এর বিশ্লেষণ](#) এবং [IBM-এর বাস্তব উদাহরণ](#)

AlphaGo

— AI যা মানুষকে হারিয়েছে Go খেলায়

সংজ্ঞা:

AlphaGo হলো Google DeepMind-এর তৈরি একটি AI যা Reinforcement Learning ও Deep Neural Network ব্যবহার করে Go খেলায় বিশ্বচ্যাম্পিয়নকে হারিয়েছিল।

মূল প্রযুক্তি:

- Policy Network: কোন চাল খেলবে
- Value Network: কে জিতবে তা অনুমান
- Monte Carlo Tree Search: সম্ভাব্য চাল বিশ্লেষণ

উদাহরণ:

২০১৬ সালে AlphaGo দক্ষিণ কোরিয়ায় Lee Sedol-কে ৪-১ ব্যবধানে হারায়—যা AI ইতিহাসে যুগান্তকারী ঘটনা।

□ বিস্তারিত: [DeepMind-এর অফিসিয়াল পৃষ্ঠা](#) এবং [GeeksforGeeks বিশ্লেষণ](#)

MLOps

— ML-এর DevOps সংস্করণ

সংজ্ঞা:

MLOps হলো এমন একটি পদ্ধতি যা ML মডেল তৈরি, ডিপ্লয়, মনিটরিং এবং আপডেট করার পুরো লাইফসাইকেলকে অটোমেট করে—DevOps-এর মতো।

উদাহরণ:

UNDP যদি একটি Poverty Prediction মডেল চালু করে, MLOps নিশ্চিত করে যে সেটি সঠিকভাবে ডিপ্লয় হয়েছে, পারফর্ম করছে, এবং ডেটা পরিবর্তনের সাথে আপডেট হচ্ছে।

মূল ধাপ:

- Version Control
- Continuous Integration & Deployment
- Monitoring & Retraining

□ বিস্তারিত: [GeeksforGeeks-এর MLOps গাইড](#) এবং [AWS-এর বিশ্লেষণ](#)

AutoML

— মেশিন লার্নিং অটোমেটিকভাবে শেখানো

সংজ্ঞা:

AutoML হলো এমন একটি প্রযুক্তি যা ML মডেল তৈরির জটিল ধাপগুলো—যেমন ডেটা প্রিপ্রসেসিং, মডেল সিলেকশন, হাইপারপ্যারামিটার টিউনিং—স্বয়ংক্রিয়ভাবে করে।

উদাহরণ:

Azure AutoML ব্যবহার করে UNDP-এর SDG রিপোর্ট বিশ্লেষণ করা যায়—কোন মডেল ভালো কাজ করবে তা AutoML নিজেই ঠিক করে।

প্রযুক্তি:

- Google AutoML, Azure AutoML, H2O.ai
- No-code/Low-code ML
- Hyperparameter Optimization

□ বিস্তারিত: [CodeConductor-এর AutoML গাইড](#) এবং [Google Developers-এর স্টার্টার](#)

AGI (Artificial General Intelligence)

— মানুষের মতো বুদ্ধিমত্তা

সংজ্ঞা:

AGI হলো এমন একটি AI যা মানুষের মতো চিন্তা করতে পারে—যেকোনো টাস্কে শেখা, সিদ্ধান্ত নেওয়া, এবং নতুন সমস্যার সমাধান করতে পারে।

উদাহরণ:

একটি AGI যদি UNDP-এর রিপোর্ট পড়ে, নিজে বিশ্লেষণ করে, নীতিগত সুপারিশ দেয়—তাহলে সেটি Narrow AI নয়, AGI।

বৈশিষ্ট্য:

- Adaptability
- Self-improvement
- General Understanding
- Multitask Capability

□ বিস্তারিত: [IBM-এর AGI উদাহরণ](#) এবং [GeeksforGeeks-এর সংজ্ঞা](#)

LangChain

— LLM ভিত্তিক অ্যাপ তৈরির ফ্রেমওয়ার্ক

সংজ্ঞা:

LangChain হলো একটি ওপেন-সোর্স ফ্রেমওয়ার্ক যা LLM (Large Language Model) ব্যবহার করে অ্যাপ্লিকেশন তৈরি সহজ করে। এটি Prompt, Memory, Tool, এবং Chain পরিচালনা করে।

মূল উপাদান:

- **Chains:** একাধিক ধাপে LLM ও টুল একত্রে ব্যবহার
- **Agents:** AI নিজে সিদ্ধান্ত নিয়ে API বা টুল কল করে
- **Memory:** আগের কথোপকথন মনে রাখে
- **Vector Store:** Embedding দিয়ে similarity search করে

- **Model Integration:** GPT, Claude, Hugging Face ইত্যাদি

ব্যবহার:

- কাস্টম চ্যাটবট
- ডকুমেন্ট বিশ্লেষণ
- SDG রিপোর্ট থেকে তথ্য রিট্রিভ
- Azure বা Hugging Face API-এর সাথে সংযুক্তি

□ বিস্তারিত: [LangChain অফিসিয়াল ডকুমেন্টেশন](#), [GeeksforGeeks বিশ্লেষণ](#)

LangGraph

— গ্রাফ-ভিত্তিক এজেন্টিক ওয়ার্কফ্লো

সংজ্ঞা:

LangGraph হলো LangChain-এর উপর ভিত্তি করে তৈরি একটি ফ্রেমওয়ার্ক যা **গ্রাফ আর্কিটেকচার** ব্যবহার করে জটিল AI ওয়ার্কফ্লো পরিচালনা করে—প্রতিটি ধাপ একটি “নোড” হিসেবে কাজ করে।

মূল বৈশিষ্ট্য:

- **Stateful Graphs:** প্রতিটি ধাপে তথ্য সংরক্ষণ
- **Cyclical Execution:** লুপ বা পুনরাবৃত্তি সমর্থন
- **Human-in-the-Loop:** মানুষের ইনপুট যুক্ত করা যায়
- **ToolNode, DecisionNode:** কাস্টম একশন নোড

ব্যবহার:

- মাল্টি-স্টেপ এজেন্টিক অ্যাপ
- কাস্টম Decision Tree
- SDG বিশ্লেষণ ও রিপোর্ট জেনারেশন
- Azure Function ও API কলিং

□ বিস্তারিত: [IBM-এর LangGraph বিশ্লেষণ](#), [GeeksforGeeks গাইড](#)

CrewAI

— মাল্টি-এজেন্ট সহযোগিতামূলক ফ্রেমওয়ার্ক

সংজ্ঞা:

CrewAI হলো একটি ওপেন-সোর্স ফ্রেমওয়ার্ক যেখানে একাধিক AI Agent একসাথে কাজ করে একটি বড় টাস্ক সম্পন্ন করে—প্রতিটি Agent-এর নির্দিষ্ট ভূমিকা থাকে।

মূল উপাদান:

- **Agent:** ভূমিকা, লক্ষ্য, এবং দক্ষতা সহ সংজ্ঞায়িত
- **Task:** প্রতিটি Agent-এর জন্য নির্দিষ্ট কাজ
- **Crew:** Agent ও Task একত্রে পরিচালনা
- **Delegation & Memory:** Agent একে অপরকে সাহায্য করতে পারে

ব্যবহার:

- কনটেন্ট জেনারেশন
- সফটওয়্যার ডেভেলপমেন্ট
- গবেষণা ও রিপোর্টিং
- SDG ভিত্তিক টিমওয়ার্ক (যেমন: Poverty Analyst, Policy Planner, Data Retriever)

□ বিস্তারিত: [CrewAI Overview](#), [GeeksforGeeks বিশ্লেষণ](#), [IBM-এর CrewAI গাইড](#)

Scikit-learn

— Python ভিত্তিক ML লাইব্রেরি

সংজ্ঞা:

Scikit-learn (sklearn) হলো Python-এর একটি জনপ্রিয় Machine Learning লাইব্রেরি যা Classification, Regression, Clustering, এবং Preprocessing-এর জন্য ব্যবহৃত হয়।

উদাহরণ:

- Decision Tree দিয়ে ছাত্রদের রেজাল্ট প্রেডিকশন
- K-Means দিয়ে UNDP ডেটা ক্লাস্টারিং
- Linear Regression দিয়ে বাজেট পূর্বাভাস

☞ বিস্তারিত: [Scikit-learn টিউটোরিয়াল](#)

Regression

— সংখ্যাগত মান প্রেডিকশন

সংজ্ঞা:

Regression হলো Supervised Learning-এর একটি ধরণ যেখানে লক্ষ্য থাকে একটি **continuous value** প্রেডিক্ট করা।

উদাহরণ:

- বাড়ির দাম প্রেডিকশন
- UNDP-এর Poverty Index পূর্বাভাস
- বিজ্ঞাপন খরচ থেকে বিক্রয় অনুমান

ধরন:

- Linear Regression
- Polynomial Regression
- Ridge & Lasso Regression
- Decision Tree Regression

☞ বিস্তারিত: [Regression মডেল ও উদাহরণ](#)

Classification

— শ্রেণিবিন্যাস ভিত্তিক প্রেডিকশন

সংজ্ঞা:

Classification হলো এমন একটি টেকনিক যেখানে AI ডেটাকে নির্দিষ্ট শ্রেণিতে ভাগ করে—যেমন “স্প্যাম” বা “নন-স্প্যাম”।

উদাহরণ:

- ইমেইল স্প্যাম ডিটেকশন
- রোগ শনাক্তকরণ (ডায়াবেটিস, ক্যান্সার)
- SDG রিপোর্টে দেশকে “উন্নত”, “মধ্যম”, “পিছিয়ে” শ্রেণিতে ভাগ

ধরন:

- Binary Classification
- Multiclass Classification
- Multi-label Classification

☞ বিস্তারিত: [Classification গাইড](#)

Keras

— সহজ Deep Learning API

সংজ্ঞা:

Keras হলো TensorFlow-এর উপর ভিত্তি করে তৈরি একটি High-Level API যা Deep Learning মডেল তৈরি সহজ করে।

উদাহরণ:

- MNIST হ্যান্ডরাইটিং ক্লাসিফিকেশন
- SDG রিপোর্ট থেকে টেক্সট বিশ্লেষণ
- Sequential Model দিয়ে দ্রুত প্রোটোটাইপ তৈরি

ফিচার:

- Sequential ও Functional API
- GPU/TPU সাপোর্ট
- Callback, Early Stopping, Model Checkpoint

☞ বিস্তারিত: [Keras গাইড ও উদাহরণ](#)

PyTorch

— Dynamic Deep Learning Framework

সংজ্ঞা:

PyTorch হলো একটি ওপেন-সোর্স Deep Learning ফ্রেমওয়ার্ক যা **Dynamic Computation Graph** ব্যবহার করে—রিয়েল-টাইমে মডেল পরিবর্তন সম্ভব।

উদাহরণ:

- Image Classification
- NLP টাস্ক (Sentiment Analysis, Translation)
- SDG ডেটা থেকে টাইম সিরিজ প্রেডিকশন

ফিচার:

- Autograd
- GPU সাপোর্ট
- Torch.nn ও Optim API

∞ বিস্তারিত: [PyTorch টিউটোরিয়াল](#)

Stemming

— শব্দকে মূল রূপে নামানো (কর্তন)

সংজ্ঞা:

Stemming হলো এমন একটি NLP টেকনিক যা শব্দের শেষে থাকা suffix কেটে ফেলে মূল রূপে রূপান্তর করে—যদিও তা সবসময় সঠিক শব্দ না-ও হতে পারে।

উদাহরণ:

- “playing” → “play”
- “happiness” → “happi”
- “flies” → “fli”

অ্যালগরিদম:

- Porter Stemmer
- Snowball Stemmer
- Lancaster Stemmer

∞ বিস্তারিত: [Stemming গাইড](#)

Lemmatization

— শব্দকে অভিধানভিত্তিক মূল রূপে নামানো

সংজ্ঞা:

Lemmatization হলো এমন একটি NLP টেকনিক যা শব্দের অর্থ ও ব্যাকরণ বিশ্লেষণ করে সঠিক অভিধানভিত্তিক রূপে রূপান্তর করে।

উদাহরণ:

- “running” → “run”
- “was” → “be”
- “mice” → “mouse”

লাইব্রেরি:

- NLTK (WordNetLemmatizer)
- spaCy
- TextBlob

☞ বিস্তারিত: [Lemmatization উদাহরণসহ](#)

Big Data

— বিশাল ও বৈচিত্র্যময় ডেটার বিশ্লেষণ

সংজ্ঞা:

Big Data হলো এমন বিশাল পরিমাণ ডেটা যা এত দ্রুত, বৈচিত্র্যময় ও জটিলভাবে তৈরি হয় যে সাধারণ ডেটা প্রসেসিং টুল দিয়ে তা পরিচালনা করা যায় না।

৫টি V:

- **Volume:** YouTube-এ প্রতি মিনিটে ৫০০+ ঘণ্টার ভিডিও আপলোড হয়
- **Velocity:** রিয়েল-টাইমে ডেটা প্রবাহ (যেমন: সেন্সর, সোশ্যাল মিডিয়া)
- **Variety:** টেক্সট, ছবি, ভিডিও, অডিও, লগ
- **Veracity:** ডেটার নির্ভরযোগ্যতা
- **Value:** বিশ্লেষণ করে ব্যবসায়িক মূল্য তৈরি

ব্যবহার:

- UNDP-এর SDG রিপোর্ট বিশ্লেষণ
- স্বাস্থ্যসেবা, ট্রান্সপোর্ট, ফাইন্যান্সে ডেটা-চালিত সিদ্ধান্ত

☐ [GeeksforGeeks-এর বিশ্লেষণ](#)

Inference

— শেখা জ্ঞান ব্যবহার করে সিদ্ধান্ত নেওয়া

সংজ্ঞা:

Inference হলো AI মডেল যখন নতুন, অদেখা ডেটা দেখে পূর্ব শেখা জ্ঞান ব্যবহার করে সিদ্ধান্ত বা প্রেডিকশন দেয়।

উদাহরণ:

- একটি ট্রেন্ড মডেল দেখে বলে: “এই ছবিতে কুকুর আছে”
- UNDP-এর Poverty Model পূর্ববর্তী ডেটা দেখে নতুন বছরের পূর্বাভাস দেয়

ধাপ:

1. ইনপুট ডেটা প্রস্তুত
2. Forward Pass (মডেল বিশ্লেষণ করে)
3. আউটপুট তৈরি (যেমন: ক্লাস, স্কোর)

☐ [IBM-এর Inference গাইড](#)

Vector Database

— অর্থভিত্তিক সার্চের জন্য বিশেষ ডেটাবেস

সংজ্ঞা:

Vector Database হলো এমন ডেটাবেস যা Embedding (সংখ্যায় রূপান্তরিত অর্থপূর্ণ ডেটা) সংরক্ষণ করে এবং similarity search চালাতে পারে।

উদাহরণ:

- “বাংলাদেশে দারিদ্র্য হ্রাস” প্রশ্ন দিলে → SDG রিপোর্টের প্রাসঙ্গিক অংশ খুঁজে বের করে
- ChatGPT বা RAG সিস্টেমে semantic search চালাতে ব্যবহৃত

জনপ্রিয় ডেটাবেস:

- Pinecone, Weaviate, Milvus, Qdrant, ChromaDB

☐ [Devart-এর বিশ্লেষণ](#)

Pre-trained Models

— পূর্ব শেখা মডেল যা পুনরায় ব্যবহারযোগ্য

সংজ্ঞা:

Pre-trained Model হলো এমন একটি মডেল যা বিশাল ডেটাসেটে আগে থেকেই ট্রেন করা হয়েছে এবং পরে নতুন টাস্কে ফাইন-টিউন করা যায়।

উদাহরণ:

- BERT: NLP টাস্কে ব্যবহৃত
- GPT: টেক্সট জেনারেশন
- ResNet: ছবি ক্লাসিফিকেশন
- UNDP-এর রিপোর্ট বিশ্লেষণে GPT বা BERT ব্যবহার করে সারাংশ তৈরি

লাভ:

- কম ডেটা লাগে
- দ্রুত ফলাফল
- Transfer Learning সহজ হয়

📖 [SpotIntelligence-এর গাইড](#)

Mistral AI

— ইউরোপীয় ওপেন-সোর্স LLM নির্মাতা

সংজ্ঞা:

Mistral AI হলো ফ্রান্স-ভিত্তিক একটি AI কোম্পানি যা উচ্চ-ক্ষমতাসম্পন্ন ওপেন-সোর্স ও API-ভিত্তিক LLM তৈরি করে।

মূল মডেল:

- Mistral 7B: ছোট, দক্ষ, ওপেন-সোর্স

- **Mixtral 8x7B:** Sparse Mixture-of-Experts, দ্রুত ও সাশ্রয়ী
- **Mistral Large 2:** GPT-4-এর প্রতিদ্বন্দ্বী, 128K context
- **Devstral, Codestral:** কোড লেখার জন্য বিশেষায়িত

ব্যবহার:

- কনটেন্ট জেনারেশন
- কোড সহায়তা
- SDG রিপোর্ট বিশ্লেষণ
- Azure Bedrock ও Google Vertex AI-তে ডিপ্লয়যোগ্য

☐ easel.ai-এর বিশ্লেষণ

Pinecone

— শক্তিশালী Vector Database

সংজ্ঞা:

Pinecone হলো একটি উচ্চ-পারফরম্যান্স Vector Database যা Embedding ডেটা সংরক্ষণ ও similarity search চালাতে ব্যবহৃত হয়—বিশেষ করে RAG (Retrieval-Augmented Generation) ও Semantic Search অ্যাপ্লিকেশনে।

মূল বৈশিষ্ট্য:

- **Index তৈরি** করে Embedding ডেটা সংরক্ষণ
- **Namespace** দিয়ে ডেটা ভাগ করা যায়
- **Metadata Filtering ও Reranking** সমর্থন করে
- Hugging Face, OpenAI, LangChain-এর সাথে সহজ ইন্টিগ্রেশন

ব্যবহার:

- SDG রিপোর্ট থেকে প্রাসঙ্গিক তথ্য খুঁজে এনে উত্তর তৈরি
- কাস্টম চ্যাটবটের জন্য ডেটা রিট্রিভ
- Azure Function বা LangChain.NET-এর সাথে সংযুক্তি

☐ [Pinecone অফিসিয়াল ডকুমেন্টেশন](#)

Indexing

— Embedding ডেটা সংগঠিত করার কৌশল

সংজ্ঞা:

Vector Indexing হলো এমন একটি কৌশল যা Embedding ডেটাকে সংগঠিত করে যাতে দ্রুত similarity search চালানো যায়—যেমন “এই প্রশ্নের সাথে মিল আছে এমন ডেটা কোথায়?”

প্রযুক্তি:

- **IVF (Inverted File Index):** K-means clustering দিয়ে ডেটা ভাগ করে
- **Tree-based Indexing:** KD-Tree, Ball Tree
- **Hashing-based Indexing:** LSH, Deep Hashing
- **Graph-based Indexing:** HNSW (Hierarchical Navigable Small World)

উদাহরণ:

UNDP-এর রিপোর্টে “দারিদ্র্য হ্রাস” সম্পর্কিত অংশ খুঁজে বের করতে Embedding → Index → Query Matching করা হয়।

☐ [Vector Indexing Explained](#)

MCP Host

— AI Agent-এর কেন্দ্র

সংজ্ঞা:

MCP Host হলো সেই AI অ্যাপ্লিকেশন যা MCP Client তৈরি করে এবং MCP Server-এর সাথে সংযোগ স্থাপন করে—যেমন Claude Desktop, Visual Studio Code, বা Azure Copilot।

ভূমিকা:

- ব্যবহারকারীর ইনপুট গ্রহণ করে
- MCP Client-এর মাধ্যমে Server থেকে context আনে
- Agent Workflow পরিচালনা করে

☐ [MCP Architecture Overview](#)

MCP Client

— সংযোগকারী ও বার্তাবাহক

সংজ্ঞা:

MCP Client MCP Host-এর ভিতরে একটি কম্পোনেন্ট যা MCP Server-এর সাথে একক সংযোগ বজায় রাখে এবং context, tool, বা prompt সংগ্রহ করে।

মূল ফিচার:

- **Sampling:** Server চাইলে LLM থেকে Completion আনে
- **Elicitation:** ব্যবহারকারীর কাছ থেকে তথ্য সংগ্রহ করে
- **Roots:** কোন ফোল্ডার বা ফাইল অ্যাক্সেসযোগ্য তা নির্ধারণ করে

☐ [Understanding MCP Clients](#)

MCP Server

— context ও tool সরবরাহকারী

সংজ্ঞা:

MCP Server হলো এমন একটি lightweight প্রোগ্রাম যা MCP Client-কে context, tool, resource, এবং prompt সরবরাহ করে—AI Agent যেন কাজ চালাতে পারে।

ধরন:

- **Local Server:** STDIO transport দিয়ে লোকাল প্রসেস
- **Remote Server:** HTTP/SSE transport দিয়ে ক্লাউড বা API
- Azure API Management দিয়ে MCP Server গভার্ন করা যায়

ব্যবহার:

- SDG ডেটাবেস থেকে তথ্য রিট্রিভ
- REST API কে MCP Tool হিসেবে এক্সপোজ
- Workflow Automation ও Decision Support

☐ [MCP Server Explained](#)

Agent Memory

— AI এজেন্টের স্মৃতিশক্তি

সংজ্ঞা:

Agent Memory হলো AI এজেন্টের সেই ক্ষমতা যা তাকে আগের কথোপকথন, কাজ, বা সিদ্ধান্ত মনে রাখতে সাহায্য করে—যাতে সে ভবিষ্যতে আরও বুদ্ধিমত্তার সাথে কাজ করতে পারে।

ধরন:

- **Short-Term Memory:** বর্তমান সেশন বা কাজের জন্য সাময়িক স্মৃতি
- **Long-Term Memory:** বহু সেশনের তথ্য, ব্যবহারকারীর পছন্দ, অভিজ্ঞতা
- **Semantic Memory:** সাধারণ জ্ঞান (যেমন: “ঢাকা বাংলাদেশের রাজধানী”)
- **Procedural Memory:** কীভাবে কাজ করতে হয় তা মনে রাখা
- **Episodic Memory:** নির্দিষ্ট ঘটনা বা অভিজ্ঞতা মনে রাখা

ব্যবহার:

- চ্যাটবট আগের প্রশ্ন মনে রেখে উত্তর দেয়
- ভার্চুয়াল অ্যাসিস্ট্যান্ট আপনার রুটিন অনুযায়ী কাজ সাজায়
- SDG বিশ্লেষণে পূর্ববর্তী সিদ্ধান্ত মনে রেখে পরবর্তী সুপারিশ দেয়

📄 [GeeksforGeeks-এর Agent Memory বিশ্লেষণ](#)

Episodic Memory

— AI এজেন্টের অভিজ্ঞতা-ভিত্তিক স্মৃতি

সংজ্ঞা:

Episodic Memory হলো AI এজেন্টের সেই ক্ষমতা যা তাকে নির্দিষ্ট ঘটনা, সময়, এবং ফলাফল মনে রাখতে সাহায্য করে—যেমন “সোমবার ইউজার X প্রযুক্তি সহায়তা চেয়েছিল, এবং সমাধান Y পেয়েছিল।”

ধাপ:

1. **Encoding:** কোন ঘটনা সংরক্ষণ করবে তা নির্ধারণ
2. **Storage:** টাইম-স্ট্যাম্পসহ ডেটা সংরক্ষণ
3. **Retrieval:** প্রাসঙ্গিক ঘটনার স্মৃতি পুনরুদ্ধার
4. **Adaptation:** পূর্ব অভিজ্ঞতা থেকে শেখা

ব্যবহার:

- কাস্টমার সাপোর্ট এজেন্ট আগের অভিযোগ মনে রেখে দ্রুত সমাধান দেয়
- রোবট আগের পথচলা মনে রেখে নতুন পথ ঠিক করে
- SDG রিপোর্ট বিশ্লেষণে পূর্ববর্তী ব্যর্থতা থেকে শিক্ষা নিয়ে নতুন পরিকল্পনা সাজায়

❏ [GeeksforGeeks-এর Episodic Memory বিশ্লেষণ](#)

ReAct Prompting

— Reasoning + Acting একত্রে

সংজ্ঞা:

ReAct (Reasoning + Acting) হলো এমন একটি prompting কৌশল যেখানে AI প্রথমে চিন্তা করে (Thought), তারপর কাজ করে (Action), এবং পর্যবেক্ষণ করে (Observation)—এই চক্রে সমস্যার সমাধান করে।

ফরম্যাট:

Thought: সমস্যা নিয়ে চিন্তা Action: একটি কাজ সম্পাদন Observation: কাজের ফলাফল ...
Final Answer: চূড়ান্ত উত্তর

উদাহরণ:

প্রশ্ন: “ঢাকা থেকে চট্টগ্রাম যাওয়ার সস্তা উপায় কী?”

- Thought: ট্রেন, বাস, ফ্লাইট খুঁজতে হবে
- Action: Search[ঢাকা-চট্টগ্রাম ট্রেন ভাড়া]
- Observation: ট্রেন ভাড়া ৩৫০ টাকা
- Final Answer: ট্রেন সবচেয়ে সস্তা উপায়

❏ [Prompt Engineering Guide-এর ReAct বিশ্লেষণ](#)

AI Red Teaming

— AI নিরাপত্তা যাচাইয়ের জন্য আক্রমণ অনুকরণ

সংজ্ঞা:

AI Red Teaming হলো একটি কাঠামোগত পরীক্ষা যেখানে বিশেষজ্ঞরা AI সিস্টেমের দুর্বলতা খুঁজে বের করতে ইচ্ছাকৃতভাবে আক্রমণ বা ভুল ব্যবহার অনুকরণ করে।

পরীক্ষার ধরন:

- Prompt Injection
- Data Poisoning
- Model Hallucination

- Bias & Fairness Testing
- API Misuse

ব্যবহার:

- SDG বিশ্লেষণকারী এজেন্ট যদি ভুল তথ্য দেয়, Red Teaming তা ধরতে পারে
- নিরাপত্তা এজেন্ট যদি অপ্রত্যাশিত API কল করে, Red Teaming তা বিশ্লেষণ করে

☐ [Palo Alto Networks-এর AI Red Teaming গাইড](#)

Agentic AI Security

— স্বয়ংক্রিয় এজেন্টের নিরাপত্তা ঝুঁকি

সংজ্ঞা:

Agentic AI Security হলো এমন নিরাপত্তা কাঠামো যা Autonomous AI Agent-এর আচরণ, API ব্যবহার, এবং সিদ্ধান্ত গ্রহণের ঝুঁকি বিশ্লেষণ ও নিয়ন্ত্রণ করে।

ঝুঁকি:

- API Misuse (Agent নিজে API কল করে বিপদ ডেকে আনে)
- Autonomous Exploitation (Agent নিজে ভুল সিদ্ধান্ত নিয়ে ক্ষতি করে)
- Prompt Injection, Escalation, Data Leakage

ব্যবহার:

- SDG এজেন্ট যদি নিজে রিপোর্ট তৈরি করে, তা যাচাই করতে নিরাপত্তা নিয়ন্ত্রণ দরকার
- Malware Analyst Agent যদি নিজে কোড বিশ্লেষণ করে, তা sandbox-এ সীমাবদ্ধ রাখতে হয়

☐ [Lasso Security-এর Agentic AI বিশ্লেষণ](#)

Multi-Agent System (MAS)

— একাধিক AI এজেন্টের সমন্বিত কাজ

সংজ্ঞা:

Multi-Agent System হলো এমন একটি কাঠামো যেখানে একাধিক AI এজেন্ট একসাথে কাজ করে একটি জটিল টাস্ক সম্পন্ন করে। প্রতিটি এজেন্ট স্বাধীনভাবে সিদ্ধান্ত নিতে পারে, কিন্তু সম্মিলিতভাবে একটি বড় লক্ষ্য অর্জন করে।

উদাহরণ:

- UNDP-এর Climate Dashboard:
- “Data Collector” এজেন্ট API থেকে তথ্য আনে
- “Analyzer” এজেন্ট SDG সূচক বিশ্লেষণ করে
- “Planner” এজেন্ট নীতিগত সুপারিশ তৈরি করে

প্রয়োগ ক্ষেত্র:

- Supply Chain Optimization
- Smart Cities
- Customer Support Automation
- Autonomous Robotics

□ বিস্তারিত: [Kubiya.ai-এর Multi-Agent Systems গাইড](#)

Symbolic Reasoning

— নিয়মভিত্তিক যুক্তি ও সিদ্ধান্ত

সংজ্ঞা:

Symbolic Reasoning (বা Symbolic AI) হলো এমন একটি পদ্ধতি যেখানে AI যুক্তি করে **নিয়ম, লজিক, এবং প্রতীক** ব্যবহার করে—ডেটা নয়, জ্ঞানভিত্তিক সিদ্ধান্ত।

উদাহরণ:

- “সব মানুষ মরণশীল, রাশকিনুর একজন মানুষ → রাশকিনুর মরণশীল”
- Expert System: “IF রোগী জ্বর AND কাশি THEN সম্ভাব্য ইনফ্লুয়েঞ্জা”

প্রযুক্তি:

- Logic Programming (Prolog)
- Semantic Networks
- Rule Engines (IF-THEN)
- Knowledge Graphs

লাভ:

- Explainability (কেন AI এমন সিদ্ধান্ত নিল তা বোঝা যায়)
- Low-data domainে কার্যকর (যেমন: আইন, চিকিৎসা)

□ বিস্তারিত: [Pathmind-এর Symbolic Reasoning বিশ্লেষণ](#)

Keras Library

— সহজ ও শক্তিশালী Deep Learning API

সংজ্ঞা:

Keras হলো TensorFlow-এর উপর ভিত্তি করে তৈরি একটি High-Level API যা Deep Learning মডেল তৈরি সহজ করে তোলে—বিশেষ করে শিক্ষার্থীদের ও গবেষকদের জন্য।

উদাহরণ:

```
model = Sequential([ layers.Flatten(input_shape=(28, 28)), layers.Dense(128, activation='relu'),  
layers.Dense(10, activation='softmax') ])
```

ফিচার:

- Sequential ও Functional API
- GPU/TPU সাপোর্ট
- Vision, NLP, Time Series—সব ধরনের টাস্কে কার্যকর
- Rapid Prototyping ও Production Deployment সহজ

□ বিস্তারিত: [Keras অফিসিয়াল উদাহরণসমূহ](#)

Deep Q-Networks (DQNs)

— শেখা ভিত্তিক সিদ্ধান্ত গ্রহণ

সংজ্ঞা:

DQN হলো Reinforcement Learning-এর একটি উন্নত সংস্করণ যা Neural Network ব্যবহার করে Q-value অনুমান করে—যাতে Agent সর্বোচ্চ পুরস্কার পায়।

মূল উপাদান:

- Neural Network: $Q(s, a)$ অনুমান করে
- Experience Replay: পুরাতন অভিজ্ঞতা থেকে শেখে
- Target Network: স্থিতিশীলতা বজায় রাখে
- ϵ -Greedy Policy: Exploration vs Exploitation ব্যালেন্স করে

উদাহরণ:

- AI Agent ভিডিও গেম খেলে শেখে
- রোবট শেখে কীভাবে বস্তু ধরতে হয়
- SDG Policy Planner শেখে কোন পদক্ষেপে বেশি প্রভাব পড়ে

□ বিস্তারিত: [GeeksforGeeks-এর DQN গাইড](#)

GPT (Generative Pre-trained Transformer)

সংজ্ঞা:

GPT হলো OpenAI-এর তৈরি একটি **Generative AI মডেল** যা Transformer আর্কিটেকচারের উপর ভিত্তি করে। এটি বিশাল পরিমাণ টেক্সট ডেটা থেকে শেখে এবং মানুষের মতো ভাষা তৈরি করতে পারে।

মূল বৈশিষ্ট্য:

- **Pre-trained:** আগে থেকেই বিশাল ডেটাসেটে শেখানো
- **Generative:** নতুন টেক্সট তৈরি করতে পারে
- **Transformer-based:** Attention Mechanism ব্যবহার করে

উদাহরণ:

- ChatGPT: প্রশ্নের উত্তর, কোড লেখা, সারাংশ তৈরি
- GPT-5: reasoning, tool calling, multimodal input সমর্থন করে

□ [Coursera-র GPT বিশ্লেষণ](#)

BERT (Bidirectional Encoder Representations from Transformers)

সংজ্ঞা:

BERT হলো Google-এর তৈরি একটি NLP মডেল যা **বাক্যের দুই দিক থেকেই** ভাষা বিশ্লেষণ করে—এটি শব্দের প্রসঙ্গ বুঝতে অসাধারণ দক্ষ।

মূল বৈশিষ্ট্য:

- Bidirectional Encoding
- Context-aware Understanding
- Hugging Face-এ সহজে ব্যবহারযোগ্য

উদাহরণ:

- “bank” শব্দটি “river bank” vs “bank account”—BERT প্রসঙ্গ বুঝে আলাদা অর্থ নির্ধারণ করে
- Named Entity Recognition, Sentiment Analysis, Question Answering

📄 [MachineLearningMastery-র BERT গাইড](#)

BART (Bidirectional and Auto-Regressive Transformer)

সংজ্ঞা:

BART হলো Facebook AI-এর তৈরি একটি **Sequence-to-Sequence** মডেল যা BERT-এর bidirectional encoding এবং GPT-এর autoregressive decoding একত্রে ব্যবহার করে।

মূল বৈশিষ্ট্য:

- Text Denoising: ভুল বা অসম্পূর্ণ টেক্সট ঠিক করে
- Summarization, Translation, Question Answering
- Encoder-Decoder আর্কিটেকচার

উদাহরণ:

- “Plants create through photosynthesis” → “oxygen”
- SDG রিপোর্টের সারাংশ তৈরি

📄 [Analytics Vidhya-র BART বিশ্লেষণ](#)

NLTK (Natural Language Toolkit)

সংজ্ঞা:

NLTK হলো Python ভিত্তিক একটি NLP লাইব্রেরি যা ভাষা বিশ্লেষণের জন্য টুলস সরবরাহ করে—যেমন Tokenization, POS Tagging, Named Entity Recognition, এবং Sentiment Analysis।

মূল ফিচার:

- ৫০+ Corpora ও WordNet

- Stemming, Lemmatization
- Chunking, Parsing, Classification

উদাহরণ:

```
import nltk
text = "John works at Google in New York"
tokens = nltk.word_tokenize(text)
tagged = nltk.pos_tag(tokens)
entities = nltk.chunk.ne_chunk(tagged)
```

☐ [NLTK অফিসিয়াল ডকুমেন্টেশন](#)

Semantic Web

— অর্থভিত্তিক ও সংযুক্ত ওয়েব

সংজ্ঞা:

Semantic Web হলো এমন একটি ওয়েব কাঠামো যেখানে ডেটা শুধু মানুষের জন্য নয়, মেশিনের জন্যও অর্থপূর্ণ—যাতে AI এজেন্ট বা সার্চ ইঞ্জিন ডেটার অর্থ বুঝতে পারে।

প্রযুক্তি:

- RDF (Resource Description Framework)
- OWL (Web Ontology Language)
- Schema.org, JSON-LD

উদাহরণ:

- Google Knowledge Panel: “Paul Schuster was born in Dresden” → AI বুঝে “Paul” একজন ব্যক্তি, “Dresden” একটি স্থান
- Rich Snippets: রেটিং, ছবি, ঠিকানা সহ সার্চ রেজাল্ট

☐ [Wikipedia-র Semantic Web বিশ্লেষণ](#)

Scalable AI

— বড় পরিসরে কাজ করার ক্ষমতা

সংজ্ঞা:

Scalable AI এমন AI সিস্টেম যা বড় ডেটা, বেশি ব্যবহারকারী, এবং জটিল টাস্ক সামলাতে পারে—পারফরম্যান্স বা নির্ভরযোগ্যতা হারানো ছাড়াই।

মূল বৈশিষ্ট্য:

- **Cloud-native Architecture:** Azure, AWS, GCP
- **Model Portability:** OpenAI, Anthropic, Hugging Face
- **Data Pipeline Optimization:** RAG, AutoML, MLOps
- **Multi-modal Support:** টেক্সট, ছবি, অডিও একসাথে

ব্যবহার:

- UNDP-এর SDG রিপোর্ট বিশ্লেষণ: হাজারো ডেটাসেট থেকে realtime recommendation
- E-commerce: recommendation engine যা ১০০ জন থেকে ১০ লাখ ব্যবহারকারী পর্যন্ত স্কেল করে
- Healthcare: MRI স্ক্যান বিশ্লেষণ করা হাজারো রোগীর জন্য

☐ [Scalable AI Explained by Vercel](#), [Accredian-এর বিশ্লেষণ](#)

Human-Centered AI

— মানুষের প্রয়োজনে AI ডিজাইন

সংজ্ঞা:

Human-Centered AI (HCAI) হলো এমন AI যা **মানবিক মূল্যবোধ, ব্যবহারযোগ্যতা, এবং নৈতিকতা** মাথায় রেখে ডিজাইন করা হয়—মানুষকে বাদ দিয়ে নয়, মানুষকে কেন্দ্র করে।

মূল নীতিমালা:

- **Transparency:** AI কীভাবে সিদ্ধান্ত নিচ্ছে তা বোঝা যায়
- **Fairness:** Bias কমানো, বৈচিত্র্য রক্ষা
- **Empathy:** ব্যবহারকারীর আবেগ ও প্রেক্ষাপট বোঝা
- **Accessibility:** সকলের জন্য ব্যবহারযোগ্য

ব্যবহার:

- Mental Health Chatbot: রোগীর আবেগ বুঝে সহানুভূতিশীল উত্তর দেয়
- Personalized Learning: শিক্ষার্থীর শেখার ধরন অনুযায়ী কনটেন্ট সাজায়
- Inclusive Hiring: AI bias কমিয়ে ন্যায্য নিয়োগ নিশ্চিত করে

☐ [Interaction Design Foundation-এর HCAI গাইড](#), [SmartKarrot-এর বাস্তব উদাহরণ](#)

Robust & Secured AI

— নিরাপদ ও নির্ভরযোগ্য AI

সংজ্ঞা:

Robust & Secured AI হলো এমন AI সিস্টেম যা **ভুল, আক্রমণ, এবং অপব্যবহার** থেকে নিরাপদ—এবং যেকোনো পরিস্থিতিতে সঠিকভাবে কাজ করতে পারে।

নিরাপত্তা ঝুঁকি:

- **Prompt Injection:** ব্যবহারকারী সিস্টেম prompt override করে
- **Model Inversion:** মডেল থেকে গোপন তথ্য বের করে
- **Adversarial Attack:** ইচ্ছাকৃতভাবে বিভ্রান্তিকর ইনপুট দিয়ে ভুল করানো
- **Data Leakage:** AI output-এ সংবেদনশীল তথ্য প্রকাশ

নিরাপত্তা কৌশল:

- **Secure by Design:** শুরু থেকেই নিরাপত্তা অন্তর্ভুক্ত
- **Zero Trust Architecture:** প্রতিটি কম্পোনেন্ট যাচাইযোগ্য
- **Red Teaming:** ইচ্ছাকৃতভাবে আক্রমণ অনুকরণ করে দুর্বলতা খোঁজা
- **RBAC & DLP:** Role-based Access Control ও Data Loss Prevention

ব্যবহার:

- Azure AI Security Framework: AI asset inventory, API protection, insider risk management
- Healthcare AI: রোগীর তথ্য leakage রোধে encryption ও access control
- SDG Policy Agent: নিরাপদভাবে API ও ডেটা access করে

☐ [Microsoft Secure AI Framework](#), [Palo Alto Networks-এর Secure Infrastructure গাইড](#)

Fairness (ন্যায্যতা)

সংজ্ঞা:

AI যেন কোনো গোষ্ঠী, লিঙ্গ, জাতি বা ভাষার বিরুদ্ধে পক্ষপাত না করে—এটাই Fairness।

উদাহরণ:

একটি AI যদি চাকরির জন্য পুরুষ প্রার্থীদের বেশি স্কোর দেয়, সেটি Unfair। Responsible AI নিশ্চিত করে যে সকল প্রার্থী সমানভাবে মূল্যায়িত হবে।

২ বাস্তব সমস্যা: Apple-এর ক্রেডিট কার্ড একবার নারীদের কম ক্রেডিট লিমিট দিয়েছিল, যদিও তাদের আর্থিক প্রোফাইল সমান ছিল।

Privacy & Security (গোপনীয়তা ও নিরাপত্তা)

সংজ্ঞা:

ব্যবহারকারীর ব্যক্তিগত তথ্য যেন AI সিস্টেম সুরক্ষিতভাবে সংরক্ষণ করে এবং অপব্যবহার না করে।

উদাহরণ:

UNDP-এর স্বাস্থ্য রিপোর্ট বিশ্লেষণ করতে হলে রোগীর নাম, ঠিকানা, বা মেডিকেল ইতিহাস যেন লিক না হয়।

✓ সমাধান: Encryption, Access Control, Data Minimization

Reliability & Safety (নির্ভরযোগ্যতা ও নিরাপত্তা)

সংজ্ঞা:

AI যেন সব পরিস্থিতিতে সঠিকভাবে কাজ করে এবং ভুল সিদ্ধান্ত না নেয়—বিশেষ করে গুরুত্বপূর্ণ ক্ষেত্রে যেমন স্বাস্থ্য, আইন, বা অর্থনীতি।

উদাহরণ:

একটি মেডিকেল AI যদি ভুলভাবে ক্যান্সার শনাক্ত করে, সেটি বিপজ্জনক। Responsible AI নিশ্চিত করে যে মডেলটি যথাযথভাবে ট্রেন ও টেস্ট করা হয়েছে।

✂ সমাধান: Rigorous Testing, Monitoring, Fail-safe Design

Transparency (স্বচ্ছতা)

সংজ্ঞা:

AI কীভাবে সিদ্ধান্ত নিচ্ছে, কী ডেটা ব্যবহার করছে, এবং কে এটি তৈরি করেছে—এই তথ্য যেন ব্যবহারকারীর কাছে পরিষ্কার থাকে।

উদাহরণ:

একটি SDG Recommendation Agent যদি বলে “বাংলাদেশে পানি সংকট বাড়ছে”—তাহলে ব্যবহারকারী জানতে চাইতে পারে: “এই সিদ্ধান্তের ভিত্তি কী?”

□ সমাধান: Explainable AI, Model Cards, Transparency Notes

Accountability (দায়িত্ব)

সংজ্ঞা:

AI যদি ভুল করে, তাহলে কে দায়ী? Responsible AI নিশ্চিত করে যে মানুষের oversight থাকবে এবং AI-এর ব্যবহার নিয়ন্ত্রিত হবে।

উদাহরণ:

একটি এজেন্ট যদি ভুলভাবে বাজেট বরাদ্দ করে, তাহলে সেই সিদ্ধান্তের জন্য একজন মানব-পর্যবেক্ষক দায়িত্ব নেবেন।

★ সমাধান: Human-in-the-loop, Audit Trail, Governance Framework

Inclusiveness (সবার জন্য AI)

সংজ্ঞা:

AI যেন সকল ব্যবহারকারী—যে কোনো ভাষা, সংস্কৃতি, বা শারীরিক সক্ষমতার—জন্য ব্যবহারযোগ্য হয়।

উদাহরণ:

বাংলা ভাষায় SDG চ্যাটবট তৈরি করা যাতে বাংলাদেশের প্রত্যন্ত অঞ্চলের মানুষও ব্যবহার করতে পারে।

□ Microsoft-এর Responsible AI Framework এই নীতিগুলোকে বাস্তবায়নের জন্য উদাহরণস্বরূপ দেখানো হয়েছে।

সামাজিক প্রভাব: AI কীভাবে সমাজকে রূপান্তর করছে

✓ ইতিবাচক প্রভাব:

- **স্বাস্থ্যসেবা উন্নয়ন:** AI-ভিত্তিক রোগ নির্ণয় (যেমন: ক্যান্সার শনাক্তকরণ) গ্রামীণ এলাকাতেও দ্রুত ও সশ্রমী চিকিৎসা পৌঁছে দিতে সাহায্য করছে।
- **শিক্ষা প্রবেশযোগ্যতা:** ভাষাভিত্তিক AI (যেমন GPT বা BERT) বাংলা ভাষায় শিক্ষামূলক কনটেন্ট তৈরি করে, যা বাংলাদেশের শিক্ষার্থীদের জন্য বড় সহায়তা।
- **দারিদ্র্য বিশ্লেষণ:** SDG রিপোর্ট বিশ্লেষণে AI ব্যবহার করে কোন অঞ্চলে দারিদ্র্য বেশি, তা চিহ্নিত করে নীতিগত পদক্ষেপ নেওয়া সহজ হয়।

- **কৃষি সহায়তা:** AI ড্রোন ও সেন্সর ব্যবহার করে ফসলের স্বাস্থ্য বিশ্লেষণ করে, কৃষকদের সঠিক সময়ে সঠিক পদক্ষেপ নিতে সাহায্য করে।

⚠ নেতিবাচক প্রভাব:

- **বেকারত্ব:** স্বয়ংক্রিয়তা অনেক ক্ষেত্রে মানুষের চাকরি হ্রাস করতে পারে
- **ডেটা বৈষম্য:** উন্নত দেশগুলোর ডেটা বেশি থাকায় AI সিস্টেম তাদের পক্ষে পক্ষপাত করতে পারে
- **সামাজিক বিভাজন:** ভুল বা biased recommendation সমাজে বিভ্রান্তি বা বৈষম্য সৃষ্টি করতে পারে

নৈতিক AI ব্যবহার: বাস্তব উদাহরণ

⚕ স্বাস্থ্যসেবা:

- **AI Diagnosis:** Mayo Clinic ও Google Health AI ব্যবহার করে রোগ নির্ণয়ে bias কমাতে কাজ করছে
- **Ethical Safeguards:** রোগীর অনুমতি ছাড়া ডেটা ব্যবহার না করা, এবং Explainable AI ব্যবহার করে চিকিৎসকের সিদ্ধান্তে সহায়তা করা

⚖ বিচার ও আইন:

- **AI Sentencing Tools:** U.S.-এ COMPAS নামক AI টুল ব্যবহার করে বিচারিক সিদ্ধান্তে bias ধরা পড়েছে
- **Ethical Use:** এখন AI ব্যবহার করা হচ্ছে শুধুমাত্র সহায়ক হিসেবে, চূড়ান্ত সিদ্ধান্ত মানব বিচারকের

🎓 শিক্ষা:

- **Adaptive Learning:** AI শিক্ষার্থীর শেখার ধরন বুঝে কনটেন্ট সাজায়
- **Ethical Concern:** শিক্ষার্থীর ডেটা যেন নিরাপদ থাকে এবং bias-free recommendation দেয়

👤 চাকরি ও নিয়োগ:

- **AI Resume Screening:** Amazon-এর AI রিক্রুটিং টুল একসময় পুরুষ প্রার্থীদের বেশি প্রাধান্য দিত
- **Ethical Correction:** এখন bias audit ও fairness testing বাধ্যতামূলক করা হয়েছে

📋 নীতিমালা ও গাইডলাইন

- **Microsoft Responsible AI Principles:** Fairness, Reliability, Privacy, Transparency, Accountability
- **EU AI Act:** High-risk AI সিস্টেমের জন্য বিশেষ নিয়ন্ত্রণ

- UNESCO Recommendation on AI Ethics: Global নৈতিক AI ব্যবহারের জন্য গাইডলাইন

INTERVIEW QUESTION

❶ প্রশ্ন: Activation Function-এর কাজ কী?

উত্তর:

Activation Function নেটওয়ার্কে **non-linearity** যোগ করে, যাতে মডেল জটিল সম্পর্ক শিখতে

পারে। .NET অ্যাপ্লিকেশনে যদি আপনি ONNX বা Azure ML ব্যবহার করেন, তাহলে ReLU, Sigmoid ইত্যাদি ফাংশন মডেলের অংশ হিসেবে থাকে।

.NET প্রয়োগ:

Azure ML Studio-তে pre-trained model deploy করলে আপনি activation function দেখতে পাবেন Model Summary-তে।

❷ প্রশ্ন: Dropout কীভাবে মডেলকে overfitting থেকে রক্ষা করে?

উত্তর:

Dropout training-এর সময় কিছু neuron সাময়িকভাবে বন্ধ করে দেয়, যাতে মডেল নির্ভরশীল না হয় নির্দিষ্ট ফিচারের উপর। এটি generalization বাড়ায়।

.NET প্রয়োগ:

আপনি যদি Keras model কে ONNX-এ রূপান্তর করে ASP.NET API-তে ব্যবহার করেন, তাহলে Dropout লেয়ারটি inference-এ inactive থাকে—কিন্তু training-এ কার্যকর।

❸ প্রশ্ন: Backpropagation-এর মূল উদ্দেশ্য কী?

উত্তর:

Backpropagation gradient descent-এর মাধ্যমে প্রতিটি weight-এর ভুল (error) হিসাব করে এবং তা সংশোধন করে। এটি মডেলকে শেখার সুযোগ দেয়।

.NET প্রয়োগ:

যেহেতু .NET ডেভেলপাররা সাধারণত training করেন না, আপনি Azure AutoML বা Hugging Face API ব্যবহার করে pre-trained model নিতে পারেন—যেখানে backpropagation আগেই সম্পন্ন হয়েছে।

❹ প্রশ্ন: Transfer Learning কখন ব্যবহার করবেন?

উত্তর:

যখন আপনার কাছে কম ডেটা থাকে, তখন pre-trained model ব্যবহার করে নতুন টাস্কে ফাইন-টিউন করা হয়—এটাই Transfer Learning।

.NET প্রয়োগ:

Azure Custom Vision বা Hugging Face Inference API ব্যবহার করে আপনি Bengali Text Classification বা SDG Tagging করতে পারেন—pre-trained BERT বা GPT model ব্যবহার করে।

❺ প্রশ্ন: CNN কোন ধরনের ডেটার জন্য উপযুক্ত?

উত্তর:

CNN মূলত ছবি বা ভিজ্যুয়াল ডেটা বিশ্লেষণের জন্য উপযুক্ত। এটি spatial feature ধরতে পারে।

.NET প্রয়োগ:

আপনি Azure Computer Vision API বা ONNX-ভিত্তিক CNN model ব্যবহার করে ASP.NET Core অ্যাপে ছবি থেকে object বা emotion শনাক্ত করতে পারেন।