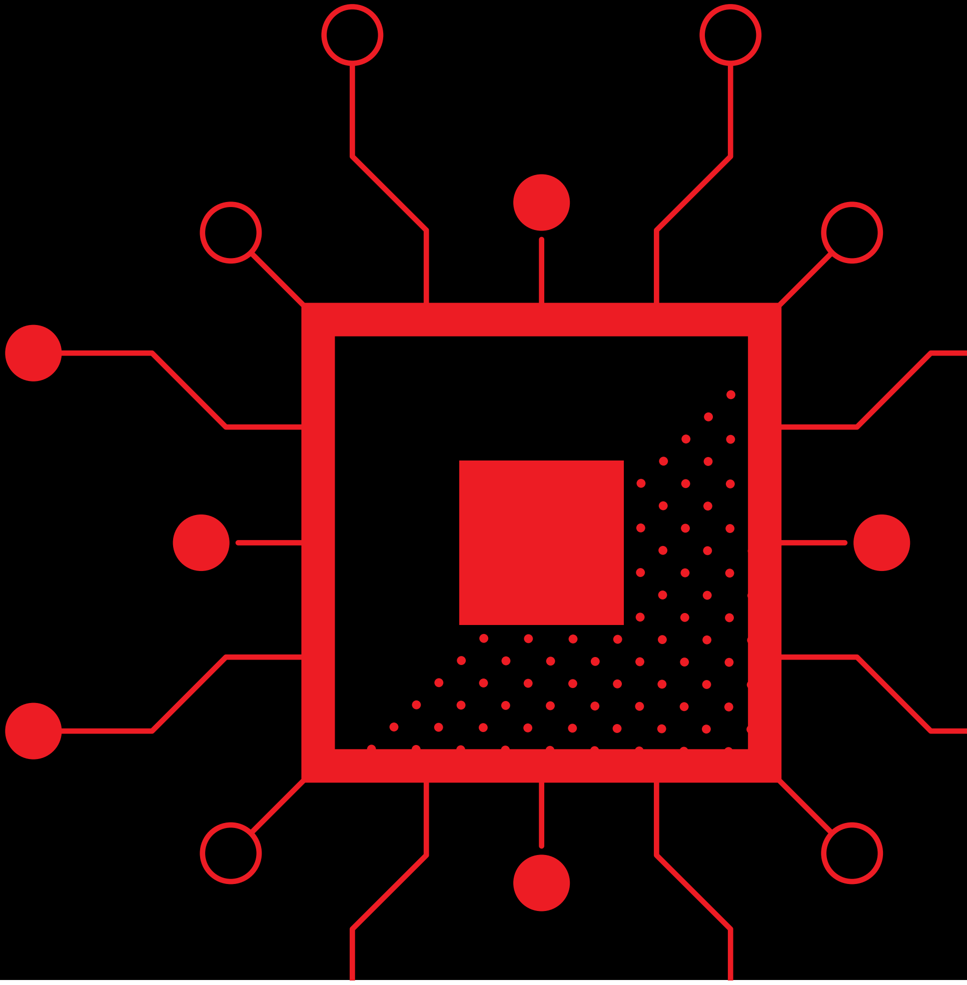
Open source AI for developers

A guide for planning and building AI-enabled applications



Red Hat

Contents

1 Open source and AI: A transformative combination

2 Plan your development strategy

3 Build innovative AI-based applications

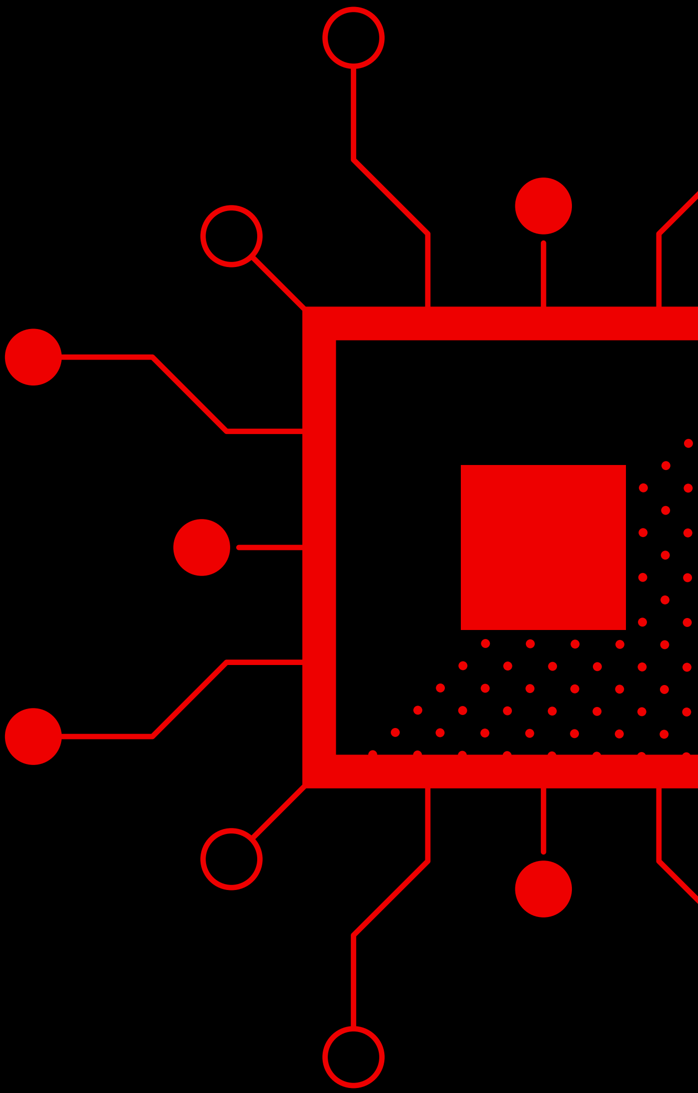
3.1 Predictive AI

3.2 Generative AI

3.3 AI models in action

4 Adopt advanced tools and technologies for AI

5 Ready, set, develop: Start building AI-enabled applications



Open source and AI: A transformative combination

Artificial intelligence (AI) and machine learning operations (MLOps) are transforming applications and development processes.

Innovative solutions based on AI models offer new possibilities for creating content, enhancing decision-making processes, and personalizing user experiences. Modern MLOps workflows streamline integration, deployment, and management of AI models into production environments, ensuring their reliability and performance. Together, AI and MLOps help developers become more agile and respond rapidly to evolving business needs with dynamic applications, efficient workflows, and shorter development cycles.

Open source AI tools offer significant benefits for development teams in terms of flexibility and customization. These tools help developers modify and adapt intelligent applications to meet business needs with tailored solutions. By encouraging collaboration within a broad community of users and contributors, open source projects support continuous improvements and new feature development in key AI technologies. This adaptability lets organizations customize AI tools to meet their requirements, making them an ideal choice for projects that demand specialized functionality.

Who should read this e-book

This e-book is intended for beginner and intermediate-level developers and data scientists who want to apply open source AI tools, platforms, and strategies in their projects. It emphasizes practical development aspects, guiding you through model selection, integration, refinement, and deployment to create innovative, AI-based applications.

Plan your development strategy

A structured approach to developing AI-based applications is needed to achieve meaningful outcomes. Each step in the process requires careful consideration to ensure that your application both meets intended goals and also operates responsibly and efficiently. From setting clear objectives to selecting a model and ensuring ethical practices, these steps help you create robust, reliable AI-based solutions.

1 **Identify AI objectives.** Begin by clearly outlining the problem your AI-based application aims to address. Determine the specific tasks that your application needs to perform. These may involve generation, classification, conversation, summarization, or code-related functions.

2 **Assess available data.** Thoroughly assess your data, as the quality and scope of your data can determine the success of your model. Generative AI models typically require large training datasets, while predictive AI models can be effective with smaller, labeled datasets. Ensure that your data is relevant, diverse, and representative of the scenarios your application will encounter. Address any gaps or biases in your data to avoid skewed or inaccurate outcomes.

3 **Analyze AI tasks.** Choose an AI approach based on your application's requirements. Generative AI models excel in nuanced language understanding, creative output, data augmentation, and real-time interaction, making them ideal for content creation, conversational AI, and text summarization. For structured tasks with well-defined outputs—including image segmentation and fraud detection—predictive AI models like convolutional neural networks (CNNs) and decision trees offer lower inference times for real-time applications. In some cases, combining both approaches can be effective. For example, customer support chatbots can use predictive AI models for intent classification and generative AI models for natural language processing.

4 **Select appropriate models.** Ready-to-use AI models from third-party application programming interfaces (APIs) or managed services let you efficiently add advanced AI features to applications. These models help you integrate complex functionalities like natural language processing, image recognition, and predictive analytics without extensive AI or machine learning expertise. By simplifying model training and maintenance processes, third-party APIs and managed services help you offer AI features that boost user engagement and application performance.

5 **Ensure ethical compliance.** AI-based applications often handle sensitive data and can significantly influence decisions. Consider the ethical implications of your application's actions and predictions. Prioritize transparency, fairness, and accountability to avoid unintended consequences and build trust with users.

Build innovative AI-based applications

AI encompasses a range of technologies that serve different purposes. Among these, predictive AI and generative AI are key technologies—each with distinct capabilities—for creating intelligent applications. Predictive AI focuses on analyzing existing data to forecast future outcomes or trends. It uses historical data to predict events, behaviors, or conditions, helping users make informed decisions based on likely scenarios. In contrast, generative AI creates new content or data based on patterns learned from existing information. It can generate text, images, or other media that resemble its training data, providing innovative solutions for content creation and personalization. While predictive AI provides insights into future possibilities, generative AI produces novel outputs from learned patterns.

Predictive AI

Building AI-based applications often begins with selecting the right model for the task. For predictive AI, this involves choosing a pretrained model or architecture tailored to your needs. Common models include ResNet for image classification, YOLO (You Only Look Once) for object detection, and Isolation Forest for anomaly detection. When selecting a model, it's essential to consider model size and complexity in relation to inference speed (how fast a trained model makes predictions based on new input data). If your organization has an established AI practice, you also have the option to build your own predictive AI model using your company's data. Open source libraries and deep learning frameworks—like OpenCV, scikit-learn, TensorFlow, and PyTorch—can help you efficiently integrate your internal and external models with AI-based applications.

Data preparation and model evaluation are critical next steps. By thoroughly analyzing your data, you can better understand its characteristics and address any potential issues. Experimentation with different model architectures or pretrained weights can help you find the optimal balance between performance and inference speed. Rigorous validation on a separate test dataset helps ensure that models generalize well.

After selecting and validating a model, refinement and deployment follow in the development process. You may need to preprocess your data using techniques like resizing and normalization. When using pretrained models, fine tuning on specific datasets is crucial. And in some cases, post-processing techniques can improve model output. By monitoring model performance in production—including inference response time and resource use—you can efficiently retrain and optimize your model to maintain effectiveness. Monitoring models for data drift can help ensure that the data seen during inferencing does not vary significantly from that data used to train or tune the model.

Because predictive AI models typically infer faster than generative AI models, they are often ideal for real-time applications. By combining predictive AI with generative AI, you can create more complete solutions. However, these solutions can result in increased model run time and complexity.

Generative AI

The first step in developing applications based on generative AI is to select the appropriate large language model (LLM). There are several open source options to choose from—including Bidirectional Encoder Representations from Transformers (BERT), Text-to-Text Transfer Transformer (T5), and Granite models—each offering unique strengths for different tasks. It's important to select an LLM that aligns to your application's objectives. For instance, Granite-7B-Starter can be fine-tuned for summarizing insurance-specific text that highlights risk factors, coverage, and liabilities, while BERT excels in sentiment analysis.

Evaluating model performance is crucial, as LLMs vary in accuracy, fluency, and overall efficacy for tasks relevant to your applications. Additionally, high-capability models like GPT-3 and some Granite variants can require significant computational resources, including expensive graphics processing unit (GPU) resources, so it's essential to balance these needs against your available infrastructure and budget. And with access to sufficient high-quality data for fine tuning, you can ensure optimal LLM performance that meets application requirements.

Frameworks like Langchain simplify the integration of LLMs into applications, allowing you to focus on the core application logic. These frameworks offer tools for prompt engineering and model chaining, while enhancing LLM-based components with memory or context.

After selecting the optimal LLM and frameworks, you are ready to add generative capabilities into your applications. This process involves refining the model's performance and crafting precise and effective prompts that guide the AI to deliver desired outcomes. Establishing robust feedback loops is crucial for continuous improvement, ensuring the model adapts and enhances its outputs over time.

How to evaluate and compare LLMs

1. **Generate prompts.**
   Create a variety of prompts to assess the creative and generative capabilities. Refine your prompts to elicit the best responses and outputs for various scenarios.
2. **Experiment with data.**
   Test models with proprietary data unique to your application. Adjust prompts and settings to optimize model performance for specific tasks.
3. **Benchmark performance.**
   Evaluate and compare model performance using the results of your experiments.

Prompts help you instruct the LLM to generate the desired output. By creating clear, concise prompts, using templates for structured instructions, and employing techniques like chaining to guide the LLM through complex tasks, you can significantly enhance the model's effectiveness. These strategies ensure that AI models produce consistent and relevant responses, even in multistep interactions.

The reinforcement learning from human feedback (RLHF) loop is crucial for fine tuning your LLM. After deploying your model, gather user interactions and use this feedback to refine the LLM performance. This iterative process helps your model learn from mistakes and continuously improve, increasing its ability to deliver accurate and relevant outputs as it adapts to real use cases.

Fine tuning further customizes pretrained LLMs to fit your specific domain or task. By training models on smaller, task-specific datasets, you can enhance performance and customize outputs to meet your application requirements. Tools like **Hugging Face Transformers** let you take advantage of the pretrained model's knowledge while refining it for your purposes. The model alignment method from **InstructLab** helps you align the model's outputs with your organizational values or user needs, ensuring responses are accurate and contextually appropriate.

Retrieval-augmented generation (RAG) combines LLMs with information retrieval systems, allowing models to access and incorporate relevant data from external sources during generation. This approach improves the factual accuracy and coherence of the outputs and is often used when augmenting LLM results with internal and corporate data. Langchain's built-in RAG capabilities streamline this process, especially when using Granite models to produce accurate and contextually relevant responses.

Agents are autonomous systems that operate within a defined environment to achieve specific goals. By incorporating interactive and adaptive behaviors, these systems can dynamically modify their operating context to respond to changing conditions. This allows them to handle complex tasks and make real-time decisions. Developing these agents involves constructing multicomponent systems that plan, execute, and evaluate actions based on AI model outputs. By orchestrating complex tasks—including real-time decision making and external API and data source integration—you can enhance your system's operational capabilities.

Model chaining connects multiple AI models or processes into a cohesive workflow, where each model builds on the outputs of the previous one. This approach allows you to develop applications capable of handling complex tasks with multistep interactions. By using the capabilities of different models in a coordinated sequence, you can build efficient systems tailored to your requirements.

By thoroughly evaluating your application's workflow with the integrated AI, you can ensure a user-friendly and efficient experience. Rigorous testing of the entire system helps you identify and address any issues or inefficiencies, allowing you to refine the application for improved functionality and usability. This iterative process not only enhances performance but also aligns the application more closely with user needs and expectations.

Example 1: Customer support chatbot

- **Problem:** Service organizations need to address customer questions and issues rapidly and efficiently—all day, every day.
- **Data:** Chat logs, product documentation, and knowledge base articles
- **Models:** Granite-7B-Starter adapted for conversations and intent classification, and enhanced with RAG for knowledge retrieval
- **Ethical considerations:** Ensuring data privacy, mitigating bias, and providing transparency to users
- **Deployment and iteration:** Deploy the chatbot, monitor interactions, gather feedback, retrain models, and update the knowledge base regularly.

Build a chatbot using third-party APIs

- **Model service:** OpenAI ChatGPT API via **Quarkus REST Client**
- **Additional considerations:** Intent classification and knowledge base integration

Example 2: Insurance underwriting risk assessment

- **Problem:** Organizations need to streamline underwriting processes by automatically summarizing complex insurance documents.
- **Data:** A complete collection of insurance documents, including policies, claims, and medical reports
- **Model:** Granite-7B-Starter fine-tuned for summarizing insurance-specific text focused on risk factors, coverage, and liabilities
- **Ethical considerations:** Prioritizing accuracy, ensuring legal compliance, and maintaining strict data privacy
- **Deployment and iteration:** Integrate models into the underwriting workflow, gather feedback, and refine models to improve risk assessment.

Assess insurance risk using third-party APIs

- **Model service:** Google Gemini API via Quarkus REST Client
- **Additional considerations:** Fine tuning and data preprocessing

Adopt advanced tools and technologies for AI

An understanding of prompt engineering, fine tuning, RAG, and agents is essential for developing innovative AI-based applications. Each method provides distinct tools and strategies for addressing complex challenges and improving the interactive features of AI-based applications. Successfully applying these techniques helps you create more intelligent, responsive, and effective AI-based systems that meet critical business goals.

Red Hat® OpenShift® AI builds on Red Hat OpenShift to provide a comprehensive platform for building, training, fine tuning, deploying, and monitoring models and applications, while meeting the workload and performance demands of modern AI solutions. It includes tools and environments that boost AI development productivity:

- **Jupyter Notebooks** and **PyTorch** ease experimentation and collaborative development, streamlining the transition from prototype to production.
- **Red Hat OpenShift Pipelines** automates continuous integration/continuous deployment (CI/CD) workflows to ensure smooth and efficient model delivery.
- **Enhanced monitoring and observability tools** track model performance and health in real time and monitor for data drift and bias, supporting proactive adjustments and maintenance.

Red Hat OpenShift AI also streamlines development and deployment of AI-based applications across hybrid cloud environments. Enhanced model serving capabilities—including support for model servers and runtimes like KServe, vLLM, and Text Generation Inference Server (TGIS)—help you deploy AI models simply and flexibly. Red Hat OpenShift AI extends model serving capabilities to edge locations, so you can deliver AI-based solutions in resource-constrained environments. Self-service access to hardware accelerators lets you rapidly iterate and optimize applications.

Red Hat Enterprise Linux® AI is a foundation model platform to seamlessly develop, test, and run Granite family LLMs to power enterprise applications. It allows portability across hybrid cloud environments, and makes it possible to then scale your AI workflows with Red Hat OpenShift AI.

Finally, the **Podman Desktop AI Lab extension** provides a streamlined setup for developing and testing AI-based applications locally, so you can simulate production environments accurately and efficiently.

Ready, set, develop

Start building AI-enabled applications today with Red Hat OpenShift AI.

Red Hat OpenShift AI simplifies complex workflows, speeds development cycles, and expands deployment options. By reducing the time and effort required to build and deploy AI models, you can focus on innovation and create compelling solutions in less time.

Get started with Red Hat OpenShift AI for free

Access Red Hat OpenShift AI in the free Developer Sandbox.

Try it for free

Learn how to use Red Hat OpenShift AI

Access interactive learning paths for a variety of AI use cases.

Learn now

Red Hat